Intel® Iris® Xe and UHD Graphics Open Source

Programmer's Reference Manual

For the 2020-2021 11th Generation Intel Xeon®, Core™, Celeron®, Pentium® Gold Processors based on the "Tiger Lake" Platform

Volume 1: Preface

December 2021, Revision 1.0

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Table of Contents

About This Programmer's Reference Manual 1

A Public View of the PRM 1

In Multiple PDF Files 1

Without Cross-book Links 1

Searching Across Multiple PDF Files 1

Organization of the Programmer's Reference Manual 7

Reserved Bits and Software Compatibility 10

Terminology 11

About This Programmer's Reference Manual

This Preface serves as an Introduction to the Programmer's Reference Manual (PRM) for the Graphics Processing Unit (GPU) included on the Tiger Lake series of chips. The PRM includes both narrative content that explains conceptually how the different modules of the GPU work, and a comprehensive Command Reference that describes named memory addresses/registers, commands, structures and enumerations, at the bit-level.

A Public View of the PRM

For each major chip series, Intel publicly releases a version of the PRM for the Open Source development community and developers at large. For those releases, we release the PRM in Portable Document Format (PDF).

In Multiple PDF Files

Due to file size constraints in generating PDF output (the Tiger Lake "Open Source" version of the PRM is roughly 5,000 pages long), the PRM is generated in multiple PDF files. The organization of those files reflects the major components of the Tiger Lake GPU. Both the narrative content and the Command Reference had to be partitioned into multiple PDF files.

Without Cross-book Links

Cross-book hyperlinks could not reasonably be included for the PDF. Instead, those links have been disabled and colored dark red and display in a serif font for easy recognition. This is an example:

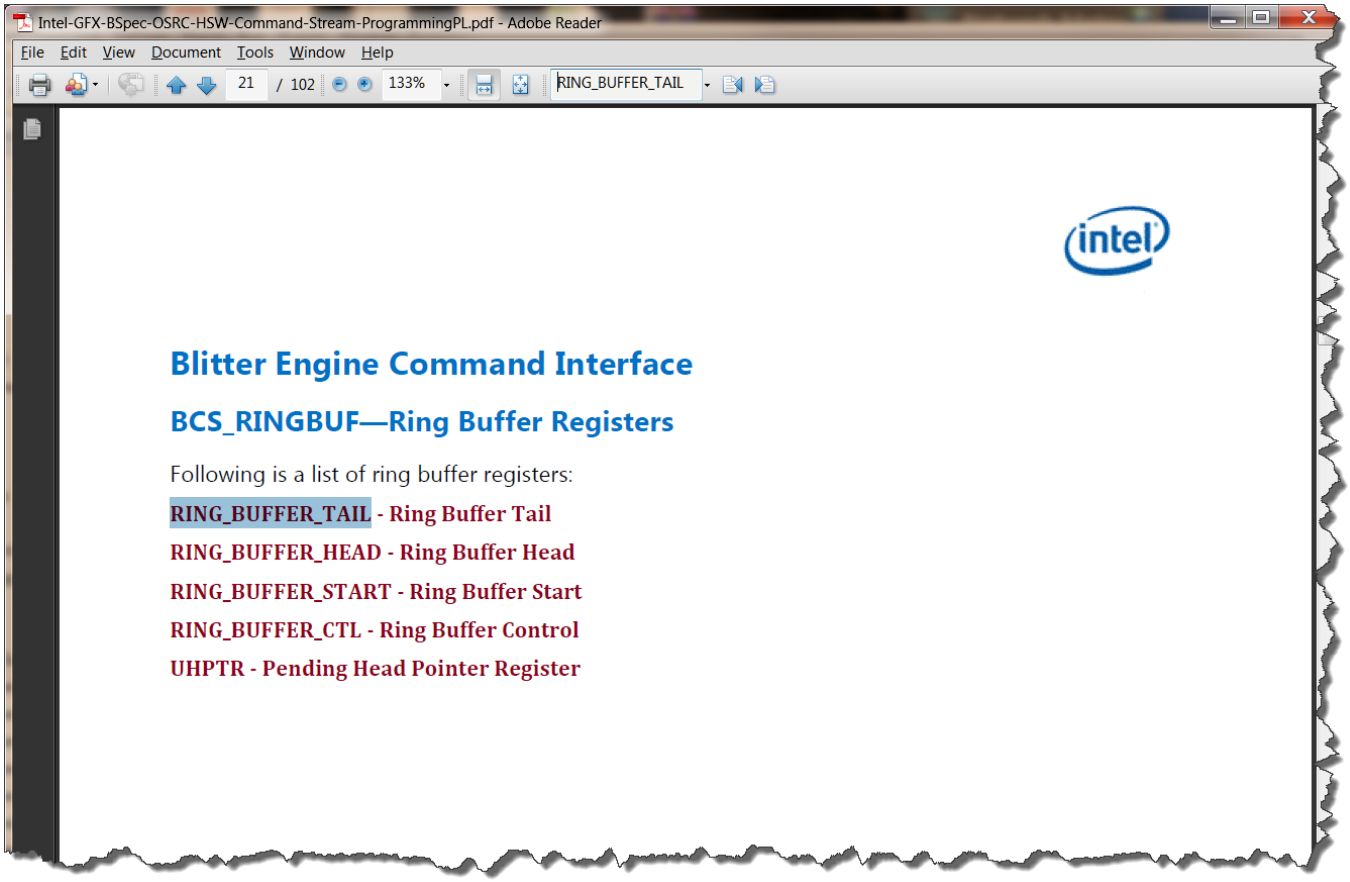
RING_BUFFER_TAIL – Ring Buffer Tail
RING_BUFFER_HEAD – Ring Buffer Head

Searching Across Multiple PDF Files

You can quickly access the detailed explanation for each named memory address, command, structure or enumerator utilizing the Advanced Search capabilities in Acrobat Reader.

To maximize the efficiency of the search, make sure to place all PRM PDFs in a single folder, or at least the four Command Reference PDFs in a single folder.

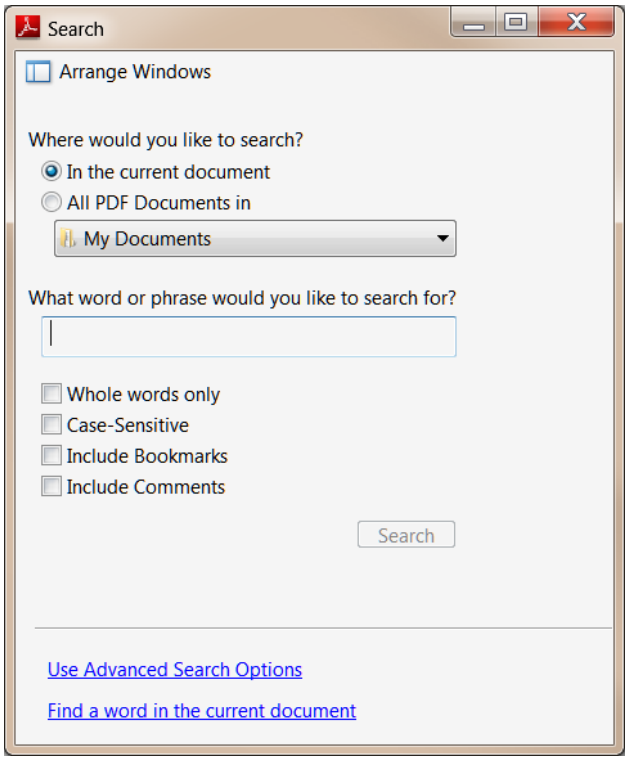
Let's walk through an example PDF search, you might be reading the Command Stream Programming module PDF file and encounter the page illustrated below.



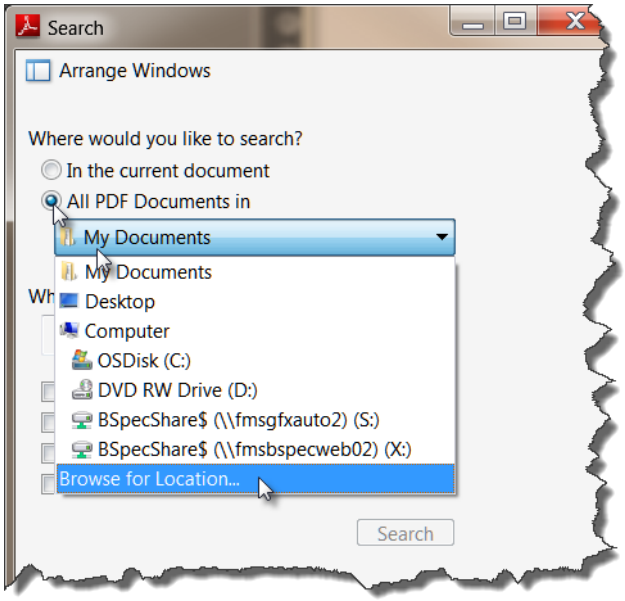
To view the detailed explanation of one of the listed registers, for this example, we will use **RING_BUFFER_TAIL ...**

1. Simply select **Search** or **Advanced Search** from the **Edit** menu, depending on the version of Acrobat you might be using. (If you are using some other PDF Reader, it probably will also have advanced search capabilities.)

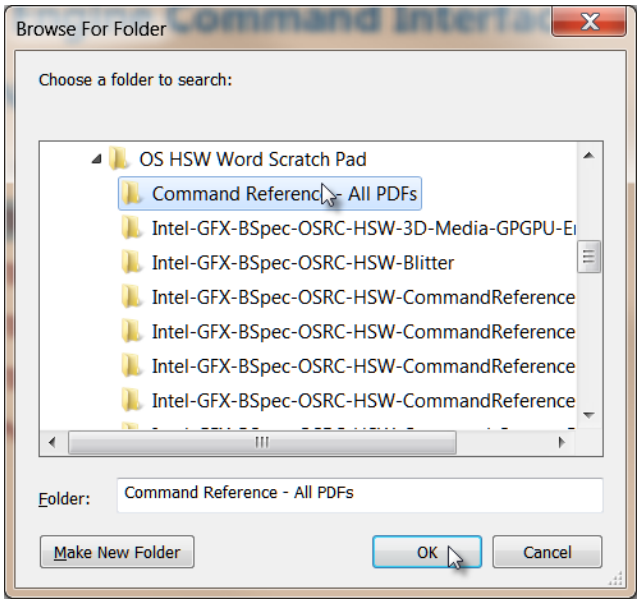
A search dialog box then displays.



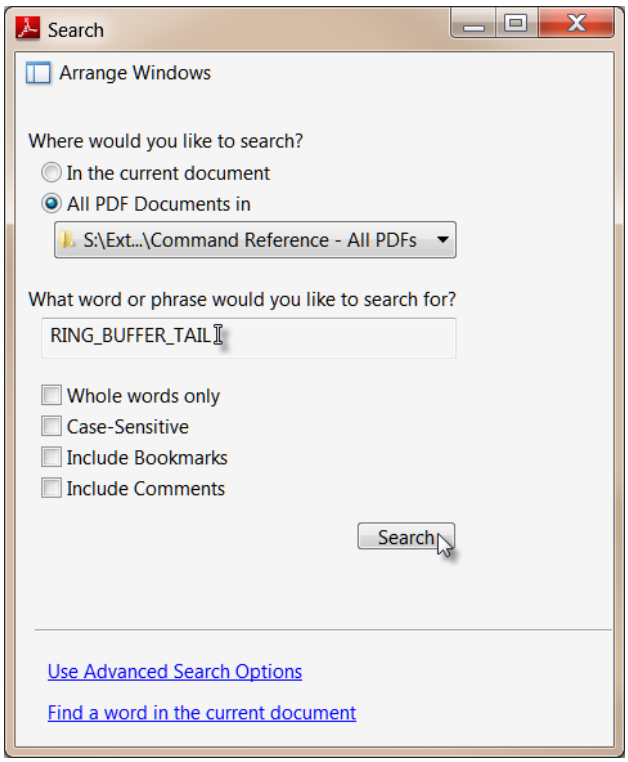
2. Select the **All PDF Documents in** radio button, click the pull-down list box, then select **Browse for Location...**



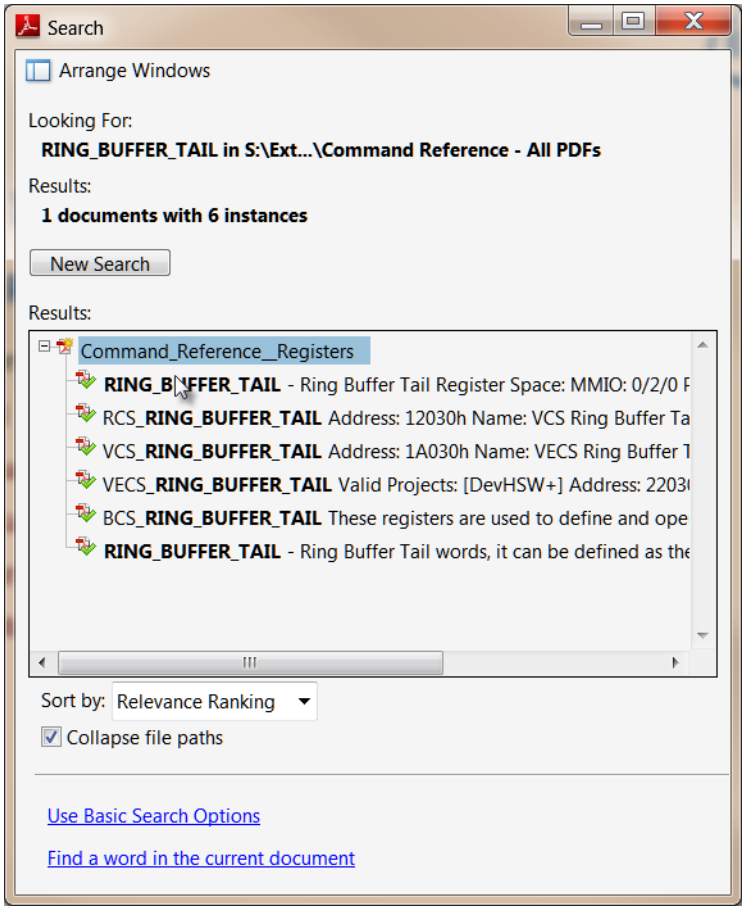
3. Browse to and select the folder containing the pertinent PDF files and click **OK**.



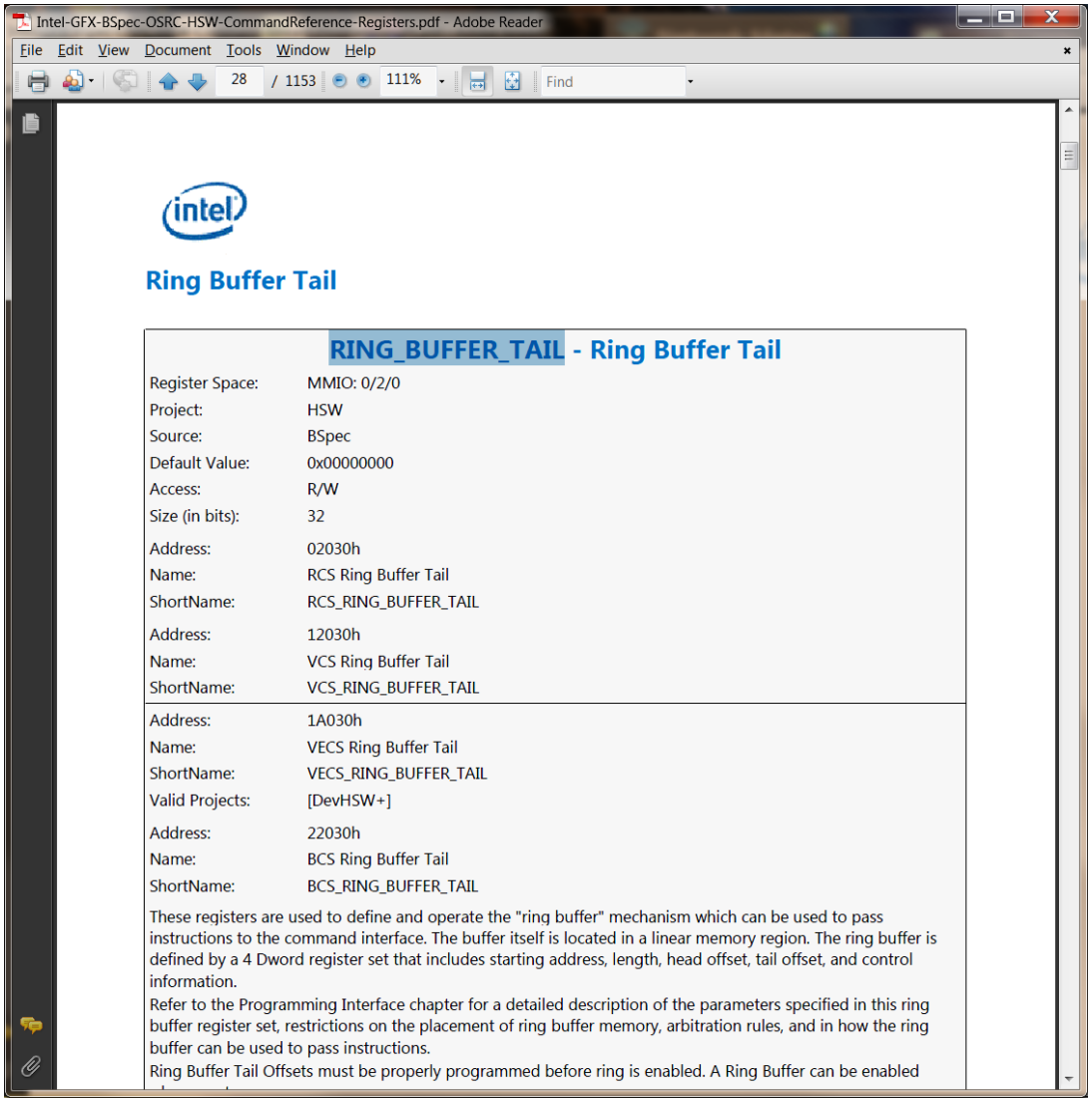
4. The Browse for Folder dialog closes and the focus shifts back to the original Search dialog. Enter the word or phrase from the red, serif text in the **What word or phrase would you like to search for?** text box and click **Search**.



As matches are found, they will be listed. When the search is complete, the dialog will appear similar to the following:



Click the most relevant result and the PDF file containing that result will open at the correct page.



Organization of the Programmer's Reference Manual

The Intel® Graphics PRM is organized into the following content areas:

Command Reference provides a tabular graphical interface tool for searching for and displaying command opcodes, registers, structures, and instructions.

GPU Overview introduces the GPU and its subsystems. This includes high level descriptions of its hardware pipelines, command formats, register maps, and supported memory/surface formats.

Configurations tracks the evolution of the graphics architecture and provides generational information for many basic architectural attributes.

Memory Views covers the Graphics Memory Interface, including memory interface functions, tiling, the physical graphics memory, page faults and error handling, memory types, common video and non-video surface formats, and other information related to the memory interface.

The **GuC** micro-controller is a scheduler for the graphics engines. This section describes the following GuC register functions:

Functions
Interrupt register functions
DMA register functions
Shim register functions

Command Stream Programming describes the programming interfaces for Command Streamer, which manages the use of the 3D and Media pipelines. It performs switching between pipelines and forwarding command streams to the currently active pipeline. It also manages the allocation of the Unified Return Buffer (URB) and helps support the Constant URB Entry (CURBE) function. It includes the following topics:

Topic
Command Formats
Blitter Engine Command Interface
Render Engine Command Interface
Video Codec Engine Command Interface
Video Enhancement Engine Command Interface
Preemption
Command Streamer (CS) ALU Programming
Resource Streamer

Media describes the VDBOX, VEBOX, VDENC engines and registers. Also covered are HEVC, HCP and SFC.

Topic
Media VDBOX covers the **Multi-Format Codec (MFX) Engine**, the hardware fixed-function pipeline that includes both multi-format decoding (MFD) and multi-format encoding (MFC).
Media VEBOX discusses the Video Enhancement pipeline, an independent pipe that manages image enhancement functions.

Topic
Topics include denoise filtering, deinterlacing, color processing, auto contrast enhancement, the capture pipe, output statistics, and video enhancement engine functions.
Media VDENC is the VDBOX Fixed Function Encode Engine. It is a dedicated encoder that does not depend on Execution Units and the Video Motion Estimation engine for motion estimation and mode decisions. This saves bandwidth and power and provides an independent encode pipeline that can be better managed for Quality of Service features. The primary usage will be low power video conferencing and video capture. This engine supports the AVC codec for encoding video with resolutions up to 3840x2160 in real-time.
HEVC covers the High Efficiency Video Coding (HEVC) Codec Pipeline (HCP), which is a fixed function hardware video codec pipeline responsible for encoding and decoding HEVC streams, with a target resolution of 4kx2k at 60 frames per second.

3D/GPGPU Engine includes the following topics:

- Command stream backend processing
- L3 cache
- Shared Local Memory (SLM)
- Messaging
- Shared functions for the 3D pipeline
- 3D fixed functions
- GPGPU Pipeline processing via both fixed functions and programmable GPU cores
- Execution units

Blitter discusses the **BL**ock **I**mage **T**ransfer**er** Engine for 2D graphics processing.

Display describes display engine registers, including display HD audio configuration and programming. It includes VGA and extended VGA registers, CPU display registers, North display engine registers, and South display engine registers. The Display section explains audio codec verbs, display audio configuration registers, and the display audio controller registers.

Topic
Wireless Display (WIDI) covers the **Wireless Display engine (WDBOX)**, a dedicated AVC video encoder. This section discusses the use of state parameters and lists the MMIO registers used for wireless display.
PCIE Config Registers lists the registers used for **Peripheral Component Interconnect Express (PCIE)** functions, including device identification and enabling, power management capabilities, per-thread and per-core status.
The PCIE Config Registers also include the UNCORE registers.
Memory Mapped I/O (**MMIO**) includes information on:
• Slice Registers and Die Recovery
SFC discusses the commands sent to the **Scalar and Format Converter (SFC)** pipeline from the Media VEBOX and Media VDBOX.

Topic
Workarounds provides software workarounds for known issues with the GPU.

Reserved Bits and Software Compatibility

In many register, instruction, and memory layout descriptions, certain bits are marked as "Reserved." When bits are marked as reserved, it is essential for compatibility with future devices that the software treat these bits as having a future, though unknown, effect. The behavior of reserved bits should be regarded as undefined *and unpredictable*. Software should follow the guidelines listed below in dealing with reserved bits:

1. Do not depend on the states of any reserved bits when testing values of registers that contain such bits.
2. Mask out the reserved bits before testing.
3. Do not depend on the states of any reserved bits when storing to an instruction or to a register.
4. When loading a register or formatting an instruction, always load the reserved bits with the values indicated in the documentation (if any), or reload them with the values previously read from the register.

Terminology

Term	Abbr.	Definition
Adjacency	--	One can consider a single line object as existing in a strip of connected lines. The neighboring line objects are called "adjacent objects", with the non-shared endpoints called the "adjacent vertices." The same concept can be applied to a single triangle object, considering it as existing in a mesh of connected triangles. Each triangle shares edges with three other adjacent triangles, each defined by a non-shared adjacent vertex. Knowledge of these adjacent objects/vertices is required by some object processing algorithms (e.g., silhouette edge detection). See *3D Pipeline*.
Application IP	AIP	Application Instruction Pointer. This is part of the control registers for exception handling for a thread. Upon an exception, hardware moves the current IP into this register and then jumps to System Instruction Pointer (SIP).
Architectural Register File	ARF	A collection of architecturally visible registers for a thread such as address registers, accumulator, flags, notification registers, IP, null, etc.
Binding Table	--	Memory-resident list of pointers to surface state blocks (also in memory).
Binding Table Pointer	BTP	Pointer to a binding table, specified as an offset from the Surface State Base Address register.
Blitter	BLT	Block Image Transferrer
Bypass Mode	--	Mode where a given fixed function (FF) unit is disabled and forwards data down the pipeline unchanged. Not supported by all FF units.
Byte	B	A numerical data type of 8 bits, B represents a signed byte integer.
Child Thread	--	A branch-node or a leaf-node thread that is created by another thread. It is a kind of thread associated with the media fixed function pipeline. A child thread is originated from a thread (the parent) executing on an EU and forwarded to the Thread Dispatcher by the TS unit. A child thread may or may not have child threads depending on whether it is a branch-node or a leaf-node thread. All pre-allocated resources such as URB and scratch memory for a child thread are managed by its parent thread. See also *Parent Thread*.
Clip Space	--	A 4-dimensional coordinate system within which a clipping frustum is defined. Object positions are projected from Clip Space to NDC space via "perspective divide" by the W coordinate, and then viewport mapped into Screen Space.
Clipper	--	3D fixed function unit that removes invisible portions of the drawing sequence by discarding (culling) primitives or by "replacing" primitives with one or more primitives that replicate only the visible portion of the original primitive.
Color Calculator	CC	Part of the Data Port shared function, the color calculator performs fixed-function pixel operations (e.g., blending) before writing a result pixel into the render cache.
Command	--	Directive fetched from a ring buffer in memory by the Command Streamer

Term	Abbr.	Definition
		and routed down a pipeline. Should not be confused with instructions which are fetched by the instruction cache subsystem and executed on an EU.
Command Streamer	CS or CSI	Functional unit of the Graphics Processing Engine that fetches commands, parses them, and routes them to the appropriate pipeline.
Constant URB Entry	CURBE	A UE that contains "constant" data for use by various stages of the pipeline.
Control Register	CR	The read-write registers are used for thread mode control and exception handling for a thread.
Data Port	DP	Shared function unit that performs a majority of the memory access types. The Data Port contains the render cache and the constant cache and performs all memory accesses requested by programs except those performed by the Sampler. See the DataPort section.
Degenerate Object	--	Object that is invisible due to coincident vertices or because it does not intersect any sample points (usually due to being tiny or a very thin sliver).
Destination	--	Describes an output or write operand.
Destination Size	--	The number of data elements in the destination of a SIMD instruction.
Destination Width	--	The size of each of (possibly) many elements of the destination of a SIMD instruction.
Domain Shader	DS	An API-supplied program that shades vertices using domain points. Also refers to the FF unit that dispatches threads to "shade" (calculate attributes for) vertices based on domain points.
Double Quad word (DQword)	DQ	A fundamental data type, DQ represents 16 bytes.
Double word (DWord)	D or DW	A fundamental data type, D or DW represents 4 bytes.
Drawing Rectangle	--	A screen-space rectangle within which 3D primitives are rendered. An object's screen-space positions are relative to the Drawing Rectangle origin. See *Strips and Fans*.
Dual Sub-slice	DSS	The collection of Execution Units (EU) that have a common set of shared function units, such as sampler, dataport, and pixel port.
End of Block	EOB	A 1-bit flag in the non-zero DCT coefficient data structure indicating the end of an 8x8 block in a DCT coefficient data buffer.
End Of Thread	EOT	A message sideband signal on the Output message bus signifying that the message requester thread is terminated. A thread must have at least one SEND instruction with the EOT bit in the message descriptor field set to properly terminate.
Exception	--	Type of (normally rare) interruption to EU execution of a thread's instructions. An exception occurrence causes the EU thread to begin executing the System Routine, which is designed to handle exceptions.
Execution Channel	--	Single lane of a SIMD operand.
Execution Size	ExecSize	Execution Size indicates the number of data elements processed by an SIMD instruction. It is one of the instruction fields and can be changed per instruction.

Term	Abbr.	Definition
Execution Unit	EU	An EU is a multi-threaded processor within the multi-processor system. Each EU is a fully-capable processor containing instruction fetch and decode, register files, source operand swizzle and SIMD ALU, etc.
Execution Unit Identifier	EUID	The 4-bit field within a thread state register (SR0) that identifies the row and column location of the EU where a thread is located. A thread can be uniquely identified by the EUID and TID.
Execution Width	ExecWidth	The width of each of several data elements that may be processed by a single SIMD instruction.
Fixed Function	FF	Function of the pipeline that is performed by dedicated (vs. programmable) hardware.
Fixed Function ID	FFID	Unique identifier for a fixed function unit.
Gateway	GW	See *Message Gateway*.
General Register File	GRF	Large read/write register file shared by all the EUs for operand sources and destinations. This is the most commonly used read-write register space organized as an array of 256-bit registers for a thread.
General State Base Address	--	The Graphics Address of a block of memory-resident "state data", which includes state blocks, scratch space, constant buffers, and kernel programs. The contents of this memory block are referenced via offsets from the contents of the General State Base Address register. See *Graphics Processing Engine*.
Geometry Shader	GS	Fixed-function unit between the vertex shader and the clipper that (if enabled) dispatches "geometry shader" threads on its input primitives. Application-supplied geometry shaders normally expand each input primitive into several output primitives to perform 3D modeling algorithms such as fur/fins.
Graphics Address	--	The GPE virtual address of some memory-resident object. This virtual address gets mapped by a GTT or PGTT to a physical memory address. Note that many memory-resident objects are referenced not with Graphics Addresses, but instead with offsets from a "base address register".
Graphics Processing Engine	GPE	Collective name for the Subsystem, the 3D and Media pipelines, and the Command Streamer.
Guardband	GB	Region that may be clipped against to make sure objects do not exceed the limitations of the renderer's coordinate space.
Horizontal Stride	HorzStride	The distance in element-sized units between adjacent elements of a region-based GRF access.
Immediate floating point vector	VF	A numerical data type of 32 bits, an immediate floating point vector of type VF contains 4 floating point elements with 8 bits each. The 8-bit floating point element contains a sign field, a 3-bit exponent field and a 4-bit mantissa field. It may be used to specify the type of an immediate operand in an instruction.
Immediate integer vector	V	A numerical data type of 32 bits, an immediate integer vector of type V contains 8 signed integer elements with 4 bits each. The 4-bit integer element is in 2's complement form. It may be used to specify the type of an

Term	Abbr.	Definition
		immediate operand in an instruction.
Index Buffer	IB	Buffer in memory containing vertex indices.
Intel Architecture	IA	
Instance	--	In the context of the VF unit, an instance is one of a sequence of sets of similar primitive data. Each set has identical vertex data but may have unique instance data that differentiates it from other sets in the sequence.
Instruction	--	Data in memory directing an EU operation. Instructions are fetched from memory, stored in a cache, and executed on one or more cores. Not to be confused with commands which are fetched and parsed by the command streamer and dispatched down the 3D or Media pipeline.
Instruction Pointer	IP	The address (really an offset) of the instruction currently being fetched by an EU. Each EU has its own IP.
Instruction Set Architecture	ISA	The ISA describes the instructions supported by an EU.
Instruction State Cache	ISC	On-chip memory that holds recently-used instructions and state variable values.
Interface Descriptor	--	Media analog of a State Descriptor.
Intermediate Z	IZ	Completion of the Z (depth) test at the front end of the Windower/Masker unit when certain conditions are met (no alpha, no pixel-shader computed Z values, etc.).
Inverse Discrete Cosine Transform	IDCT	The stage in the video decoding pipe between IQ and MC.
Inverse Quantization	IQ	A stage in the video decoding pipe between IS and IDCT.
JIT	--	Just-in-time compiler, aka "jitter".
Kernel	--	A sequence of instructions that is logically part of the driver or generated by the jitter. Differentiated from a Shader which is an application supplied program that is translated by the jitter to instructions.
Logical Ring Context Area	LRCA	Memory area used to store contents of registers and state information required for initiating and resuming communication between software application and hardware graphics pipeline via Ring Buffers.
Least Significant Bit	LSB	The bit with the lowest bit position within a group of bits, which could be a bit group, DWord, field, instruction, memory range, register, or structure. For example, bit 0 of a DWord.
MathBox	--	See *Extended Math Unit*
Media	--	Term for operations such as video decode and encode that are normally performed by the Media pipeline.
Media Pipeline	--	Fixed function stages dedicated to media and "generic" processing, sometimes referred to as the generic pipeline.
Memory-mapped Input/Output	MMIO	A methord for performing input/output between the CPU/GPU and peripheral devices.
Message	--	Messages are data packages transmitted from a thread to another thread,

Term	Abbr.	Definition
		another shared function, or another fixed function. Message passing is the primary communication mechanism of the architecture.
Message Gateway	--	Shared function that enables thread-to-thread message communication/synchronization, used solely by the Media pipeline.
Most Significant Bit	MSB	The bit with the highest bit position within a group of bits, which could be a bit group, DWord, field, instruction, memory range, register, or structure. For example, bit 31 of a DWord.
Motion Compensation	MC	Part of the video decoding pipe.
Motion Picture Expert Group	MPEG	MPEG is the international standard body JTC1/SC29/WG11 under ISO/IEC that has defined audio and video compression standards such as MPEG-1, MPEG-2, and MPEG-4.
Motion Vector Field Selection	MVFS	A four-bit field selecting reference fields for the motion vectors of the current macroblock.
Multiple Render Targets	MRT	Multiple independent surfaces that may be the target of a sequence of 3D or Media commands that use the same surface state.
Normalized Device Coordinates	NDC	Clip Space Coordinates that have been divided by the Clip Space "W" component.
Object	--	A single triangle, line, or point.
OpenGL	OGL	**O**pen **G**raphics **L**ibrary. A graphics API specification associated with Linux.
Parent Thread	--	A thread corresponding to a root-node or a branch-node in thread generation hierarchy. A parent thread may be a root thread or a child thread depending on its position in the thread generation hierarchy.
Pipeline Stage	--	An abstracted element of the 3D Pipeline, providing functions performed by a combination of the corresponding hardware FF unit and the threads spawned by that FF unit.
Pipelined State Pointers	PSP	Pointers to state blocks in memory that are passed down the pipeline.
Pixel Shader	PS	Shader that is supplied by the application, translated by the jitter and is dispatched to the EU by the Windower (conceptually) once per pixel.
Point	--	A drawing object characterized only by position coordinates and width.
Primitive	--	Synonym for object: triangle, rectangle, line, or point.
Primitive Topology	--	A composite primitive such as a triangle strip or a line list. Also includes the objects triangle, line, and point as degenerate cases.
Provoking Vertex	--	The vertex of a primitive topology from which vertex attributes that are constant across the primitive are taken.
Quad Quad word (QQword)	QQ	A fundamental data type, QQ represents 32 bytes.
Quad Word (QWord)	QW	A fundamental data type, QW represents 8 bytes.
Rasterization	--	Conversion of an object represented by vertices into the set of pixels that make up the object.
Region-based addressing	--	Collective term for the register addressing modes available in the EU instruction set that permit discontiguous register data to be fetched and

Term	Abbr.	Definition
		used as a single operand.
Render Cache	RC	Cache in which pixel color and depth information is written before being written to memory, and where prior pixel destination attributes are read in preparation for blending and Z test.
Render Target	RT	A destination surface in memory where render results are written.
Render Target Array Index	--	Selector of which of several render targets the current operation is targeting.
Resource Streamer	RS	Functional unit of the Graphics Processing Engine that examines the commands in the ring buffer in an attempt to pre-process certain long latency items for the remainder of the graphics processing.
Root Thread	--	A root-node thread. A thread corresponds to a root-node in a thread generation hierarchy. It is a kind of thread associated with the media fixed function pipeline. A root thread is originated from the VFE unit and forwarded to the Thread Dispatcher by the TS unit. A root thread may or may not have child threads. A root thread may have scratch memory managed by TS. A root thread with children has its URB resource managed by the VFE.
Sampler	--	Shared function that samples textures and reads data from buffers on behalf of EU programs.
Scratch Space	--	Memory allocated to the subsystem that is used by EU threads for data storage that exceeds their register allocation, persistent storage, storage of mask stack entries beyond the first 16, etc.
Shader	--	A program supplied by the application in a high level shader language, and translated to instructions by the jitter.
Shared Function	SF	Function unit that is shared by EUs. EUs send messages to shared functions, that consume the data and may return results. Example Sampler and Data Port.
Shared Function ID	SFID	Unique identifier used by kernels and shaders to target shared functions and to identify their returned messages.
Single Instruction Multiple Data	SIMD	A parallel processing architecture that exploits data parallelism at the instruction level. It can also be used to describe the instructions in such an architecture or to describe the amount of data parallelism in a particular instruction (SIMD8 for example).
Source	--	Describes an input or read operand.
Spawn	--	To initiate a thread for execution on an EU. Done by the thread spawner as well as most FF units in the 3D Pipeline.
Sprite Point	--	Point object using full range texture coordinates. Points that are not sprite points use the texture coordinates of the point's center across the entire point object.
State Descriptor	--	Blocks in memory that describe the state associated with a particular FF, including its associated kernel pointer, kernel resource allowances, and a pointer to its surface state.

Term	Abbr.	Definition
State Register	SR	The read-only registers containing the state information of the current thread, including the EUID/TID, Dispatcher Mask, and System IP.
State Variable	SV	An individual state element that can be varied to change the way given primitives are rendered or media objects processed. On state variables persist only in memory and are cached as needed by rendering/processing operations except for a small amount of non-pipelined state.
Stream Output	--	A term for writing the output of a FF unit directly to a memory buffer instead of, or in addition to, the output passing to the next FF unit in the pipeline. Currently only supported for the Geometry Shader (GS) FF unit.
Strips and Fans	SF	Fixed function unit whose main function is to decompose primitive topologies such as strips and fans into primitives or objects.
Sub-Register	--	Subfield of a SIMD register. A SIMD register is an aligned fixed size register for a register file or a register type. For example, a GRF register, *r2*, is a 256-bits wide, 256-bit aligned register. A sub-register, *r2.3:d*, is the fourth dword of GRF register *r2*.
Subsystem	--	The name given to the resources shared by the FF units, including shared functions and EUs.
Surface	--	A rendering operand or destination, including textures, buffers, and render targets.
Surface State Base Pointer	--	Base address used when referencing binding table and surface state data.
Synchronized Root Thread	--	A root thread that is dispatched by TS upon a 'dispatch root thread' message.
System IP	SIP	There is one global System IP register for all the threads. From a thread's point of view, this is a virtual read only register. Upon an exception, hardware performs some bookkeeping and then jumps to SIP.
System Routine	--	Sequence of instructions that handles exceptions. SIP is programmed to point to this routine, and all threads encountering an exception will call it.
Thread	--	An instance of a kernel program executed on an EU. The life cycle for a thread starts from the executing the first instruction after being dispatched from Thread Dispatcher to an EU to the execution of the last instruction - a *send* instruction with EOT that signals the thread termination. Threads in the system may be independent from each other or communicate with each other through Message Gateway share function.
Thread Dispatcher	TD, TDL	Functional unit that arbitrates thread initiation requests from Fixed Functions units and instantiates the threads on EUs.
Thread Identifier	TID	The field within a thread state register (SR0) that identifies which thread slots on an EU a thread occupies. A thread can be uniquely identified by the EUID and TID.
Thread Payload	--	Before a thread starting execution, some amount of data is pre-loaded into the thread's GRF (starting at r0). This data is typically a combination of control information provided by the spawning entity (FF Unit) and data read from the URB.
Thread Spawner	TS	The second and the last fixed function stage of the media pipeline that

Term	Abbr.	Definition
		initiates new threads on behalf of generic/media processing.
Topology	--	See *Primitive Topology*.
Unified Return Buffer	URB	The on-chip memory managed/shared by Fixed Functions in order for a thread to return data that will be consumed either by a Fixed Function or other threads.
Unsigned Byte integer	UB	A numerical data type of 8 bits.
Unsigned Double Word integer	UD	A numerical data type of 32 bits. It may be used to specify the type of an operand in an instruction.
Unsigned Word integer	UW	A numerical data type of 16 bits. It may be used to specify the type of an operand in an instruction.
Unsynchronized Root Thread	--	A root thread that is automatically dispatched by TS.
URB Dereference	--	
URB Entry	UE	URB Entry: A logical entity stored in the URB (such as a vertex), referenced via a URB Handle.
URB Entry Allocation Size	--	Number of URB entries allocated to a Fixed Function unit.
URB Fence	Fence	Virtual, movable boundaries between the URB regions owned by each FF unit.
URB Handle	--	A unique identifier for an URB entry that is passed down a pipeline.
URB Reference	--	
Variable Length Decode	VLD	The first stage of the video decoding pipe that consists mainly of bit-wide operations.
Vertex Buffer	VB	Buffer in memory containing vertex attributes.
Vertex Cache	VC	Cache of Vertex URB Entry (VUE) handles tagged with vertex indices. See the VS chapter for details on this cache.
Vertex Fetcher	VF	The first FF unit in the 3D Pipeline responsible for fetching vertex data from memory. Sometimes referred to as the Vertex Formatter.
Vertex Header	--	Vertex data required for every vertex appearing at the beginning of a Vertex URB Entry.
Vertex ID	--	Unique ID for each vertex that can optionally be included in vertex attribute data sent down the pipeline and used by kernel/shader threads.
Vertex Shader	VS	An API-supplied program that calculates vertex attributes. Also refers to the FF unit that dispatches threads to "shade" (calculate attributes for) vertices.
Vertex URB Entry	VUE	An URB entry that contains data for a specific vertex.
Vertical Stride	VertStride	The distance in element-sized units between 2 vertically-adjacent elements of a region-based GRF access.
Video Front End	VFE	The first fixed function in the generic pipeline; performs fixed-function media operations.
Viewport	VP	

Term	Abbr.	Definition
Windower IZ	WIZ	Term for Windower/Masker that encapsulates its early ("intermediate") depth test function.
Windower/Masker	WM	Fixed function triangle/line rasterizer.
Word	W	A numerical data type of 16 bits, W represents a signed word integer.

Intel® Iris® Xe and UHD Graphics Open Source

Programmer's Reference Manual

For the 2020-2021 11th Generation Intel Xeon®, Core™, Celeron®, Pentium® Gold Processors based on the "Tiger Lake" Platform

Volume 2b: Command Reference: Enumerations

December 2021, Revision 1.0

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Table of Contents

Notices and Disclaimers 2
3D_Color_Buffer_Blend_Factor 1
3D_Color_Buffer_Blend_Function 2
3D_Compare_Function 3
3D_Logic_Op_Function 4
3D_Prim_Topo_Type 5
3D_Stencil_Operation 7
3D_Vertex_Component_Control 8
AccWrCtrl 9
AddrMode 10
AtomicCtrl 11
Attribute_Component_Format 12
ChanOff 13
COMPONENT_ENABLES 14
DPASOperandPrecision 15
EU_OPCODE 16
ExecSize 19
Fixed Function ID 20
FlagModifier 21
GW_FENCE_PORTS 22
HorzStride 23
ImmDataType 24
MathFC 25
MediaCompressionFormat 29
Performance Counter Report Formats 30
PredCtrl 31
PREFERRED_SLM_SIZE 32
RegDataType 33
Registers Per Thread 34
RENDER_BARRIER_STAGE 35
RENDER_BARRIER_TYPE 36
RenderCompressionFormat 37

Saturate **40**
SFID **41**
Shader Channel Select **42**
SIMD Mode **43**
Slice Hash Control **44**
SrcMod **45**
SURFACE_FORMAT **46**
SyncFC **53**
TernaryDataType **54**
TernaryVertStride **55**
Texture Coordinate Mode **56**
VertStride **57**
Width **58**
WRAP_SHORTEST_ENABLE **59**

3D_Color_Buffer_Blend_Factor

3D_Color_Buffer_Blend_Factor	
Size (in bits):	5

Value	Name
00h	Reserved
01h	BLENDFACTOR_ONE
02h	BLENDFACTOR_SRC_COLOR
03h	BLENDFACTOR_SRC_ALPHA
04h	BLENDFACTOR_DST_ALPHA
05h	BLENDFACTOR_DST_COLOR
06h	BLENDFACTOR_SRC_ALPHA_SATURATE
07h	BLENDFACTOR_CONST_COLOR
08h	BLENDFACTOR_CONST_ALPHA
09h	BLENDFACTOR_SRC1_COLOR
0Ah	BLENDFACTOR_SRC1_ALPHA
0Bh-10h	Reserved
11h	BLENDFACTOR_ZERO
12h	BLENDFACTOR_INV_SRC_COLOR
13h	BLENDFACTOR_INV_SRC_ALPHA
14h	BLENDFACTOR_INV_DST_ALPHA
15h	BLENDFACTOR_INV_DST_COLOR
16h	Reserved
17h	BLENDFACTOR_INV_CONST_COLOR
18h	BLENDFACTOR_INV_CONST_ALPHA
19h	BLENDFACTOR_INV_SRC1_COLOR
1Ah	BLENDFACTOR_INV_SRC1_ALPHA

3D_Color_Buffer_Blend_Function

3D_Color_Buffer_Blend_Function		
Size (in bits): 3		
Value	**Name**	**Description**
0	BLENDFUNCTION_ADD	BLENDFUNCTION_ADD
1	BLENDFUNCTION_SUBTRACT	BLENDFUNCTION_SUBTRACT
2	BLENDFUNCTION_REVERSE_SUBTRACT	BLENDFUNCTION_REVERSE_SUBTRACT
3	BLENDFUNCTION_MIN	BLENDFUNCTION_MIN
4	BLENDFUNCTION_MAX	BLENDFUNCTION_MAX
5 - 7	Reserved	

3D_Compare_Function

3D_Compare_Function		
Size (in bits):	3	
Value	**Name**	**Description**
0h	COMPAREFUNCTION_ALWAYS	Always pass
1h	COMPAREFUNCTION_NEVER	Never pass
2h	COMPAREFUNCTION_LESS	Pass if the value is less than the reference
3h	COMPAREFUNCTION_EQUAL	Pass if the value is equal to the reference
4h	COMPAREFUNCTION_LEQUAL	Pass if the value is less than or equal to the reference
5h	COMPAREFUNCTION_GREATER	Pass if the value is greater than the reference
6h	COMPAREFUNCTION_NOTEQUAL	Pass if the value is not equal to the reference
7h	COMPAREFUNCTION_GEQUAL	Pass if the value is greater than or equal to the reference

3D_Logic_Op_Function

3D_Logic_Op_Function		
Size (in bits):	4	
Value	**Name**	**Description**
0h	LOGICOP_CLEAR	BLACK; all 0's
1h	LOGICOP_NOR	NOTMERGEPEN; NOT (S OR D)
2h	LOGICOP_AND_INVERTED	MASKNOTPEN; (NOT S) AND D
3h	LOGICOP_COPY_INVERTED	NOTCOPYPEN; NOT S
4h	LOGICOP_AND_REVERSE	MASKPENNOT; S AND NOT D
5h	LOGICOP_INVERT	NOT; NOT D
6h	LOGICOP_XOR	XORPEN; S XOR D
7h	LOGICOP_NAND	NOTMASKPEN; NOT (S AND D)
8h	LOGICOP_AND	MASKPEN; S AND D
9h	LOGICOP_EQUIV	NOTXORPEN; NOT (S XOR D)
Ah	LOGICOP_NOOP	NOP; D
Bh	LOGICOP_OR_INVERTED	MERGENOTPEN; (NOT S) OR D
Ch	LOGICOP_COPY	COPYPEN; S
Dh	LOGICOP_OR_REVERSE	MERGEPENNOT; S OR NOT D
Eh	LOGICOP_OR	MERGEPEN; S OR D
Fh	LOGICOP_SET	WHITE; all 1's

3D_Prim_Topo_Type

3D_Prim_Topo_Type

Source: RenderCS

Size (in bits): 6

The following table defines the encoding of the Primitive Topology Type field. See 3D Pipeline for details, programming restrictions, diagrams and a discussion of the basic primitive types.

Value	Name	Description
00h	Reserved	
01h	3DPRIM_POINTLIST	
02h	3DPRIM_LINELIST	
03h	3DPRIM_LINESTRIP	
04h	3DPRIM_TRILIST	
05h	3DPRIM_TRISTRIP	
06h	3DPRIM_TRIFAN	
07h	3DPRIM_QUADLIST	
08h	3DPRIM_QUADSTRIP	
09h	3DPRIM_LINELIST_ADJ	
0Ah	3DPRIM_LINESTRIP_ADJ	
0Bh	3DPRIM_TRILIST_ADJ	
0Ch	3DPRIM_TRISTRIP_ADJ	
0Dh	3DPRIM_TRISTRIP_REVERSE	
0Eh	3DPRIM_POLYGON	
0Fh	3DPRIM_RECTLIST	
10h	3DPRIM_LINELOOP	
11h	3DPRIM_POINTLIST _BF	
12h	3DPRIM_LINESTRIP_CONT	
13h	3DPRIM_LINESTRIP_BF	
14h	3DPRIM_LINESTRIP_CONT_BF	
15h	Reserved	
16h	3DPRIM_TRIFAN_NOSTIPPLE	
17h	Reserved	
18h	Reserved	
19h	Reserved	
1Ah	Reserved	
[1Bh-1Fh]	Reserved	
20h	3DPRIM_PATCHLIST_1	List of 1-vertex patches

3D_Prim_Topo_Type		
21h	3DPRIM_PATCHLIST_2	
22h	3DPRIM_PATCHLIST_3	
23h	3DPRIM_PATCHLIST_4	
24h	3DPRIM_PATCHLIST_5	
25h	3DPRIM_PATCHLIST_6	
26h	3DPRIM_PATCHLIST_7	
27h	3DPRIM_PATCHLIST_8	
28h	3DPRIM_PATCHLIST_9	
29h	3DPRIM_PATCHLIST_10	
2ah	3DPRIM_PATCHLIST_11	
2bh	3DPRIM_PATCHLIST_12	
2ch	3DPRIM_PATCHLIST_13	
2dh	3DPRIM_PATCHLIST_14	
2eh	3DPRIM_PATCHLIST_15	
2fh	3DPRIM_PATCHLIST_16	
30h	3DPRIM_PATCHLIST_17	
31h	3DPRIM_PATCHLIST_18	
32h	3DPRIM_PATCHLIST_19	
33h	3DPRIM_PATCHLIST_20	
34h	3DPRIM_PATCHLIST_21	
35h	3DPRIM_PATCHLIST_22	
36h	3DPRIM_PATCHLIST_23	
37h	3DPRIM_PATCHLIST_24	
38h	3DPRIM_PATCHLIST_25	
39h	3DPRIM_PATCHLIST_26	
3ah	3DPRIM_PATCHLIST_27	
3bh	3DPRIM_PATCHLIST_28	
3ch	3DPRIM_PATCHLIST_29	
3dh	3DPRIM_PATCHLIST_30	
3eh	3DPRIM_PATCHLIST_31	
3Fh	3DPRIM_PATCHLIST_32	List of 32-vertex patches

3D_Stencil_Operation

3D_Stencil_Operation

Source: RenderCS

Size (in bits): 3

Value	Name
0	STENCILOP_KEEP
1	STENCILOP_ZERO
2	STENCILOP_REPLACE
3	STENCILOP_INCRSAT
4	STENCILOP_DECRSAT
5	STENCILOP_INCR
6	STENCILOP_DECR
7	STENCILOP_INVERT

3D_Vertex_Component_Control

3D_Vertex_Component_Control	
Source:	RenderCS
Size (in bits):	3

Value	Name	Description
0	VFCOMP_NOSTORE	Don't store this component. (Not valid for Component 0, but can be used for Component 1-3). Once this setting is used for a component, all higher-numbered components (if any) MUST also use this setting. (I.e., no holes within any particular vertex element). VFCOMP_NOSTORE will not store a component if the SourceElementFormat is R64_PASSTHRU or R64G64_PASSTHRU and it is used on component 2 and 3 else 0 will be stored.
1	VFCOMP_STORE_SRC	Store corresponding component from format-converted source element. Storing a component that is not included in the Source Element Format results in an UNPREDICTABLE value being stored. VF will process Component Control fields within a VERTEX_ELEMENT_STATE structure sequentially, starting with Component 0 Control. For each Component Control field in this sequence, when VF detects (a) the Component Control field is set to STORE_SRC and (b) the component is not overwritten by an SGV, VF will store a component of the source vertex data into the destination component. The first such Component Control field satisfying this criteria will use Component 0 of the source vertex data, the second such Component Control field will use Component 1 of the source vertex data, and so on. Therefore, when a lower-numbered Component Control field (a) is set to something other than STORE_SRC (e.g., STORE_0) or (b) the component is overwritten with an SGV, the source vertex component used when a higher-numbered Component Control fields is set to STORE_SRC will be impacted.
2	VFCOMP_STORE_0	Store 0 (interpreted as 0.0f if accessed as a float value)
3	VFCOMP_STORE_1_FP	Store 1.0f
4	VFCOMP_STORE_1_INT	Store 0x1
5-6	-	Reserved
7	VFCOMP_STORE_PID	Store Primitive ID (as U32) Software can no longer use this encoding as PrimitiveID is passed down the FF pipeline - see explanation above.

AccWrCtrl

AccWrCtrl	
Size (in bits):	1
This field allows per instruction accumulator write control. When set, the result is written to both destination and accumulator.	

Value	Name
0	Don't write to ACC **[Default]**
1	Update ACC

AddrMode

AddrMode	
Source:	EuIsa
Size (in bits):	1

Addressing Mode This field determines the addressing method of the operand. Normally the destination operand and each source operand each have a distinct addressing mode field. When it is cleared, the register address of the operand is directly provided by bits in the instruction word. It is called a direct register addressing mode. When it is set, the register address of the operand is computed based on the address register value and an address immediate field in the instruction word. This is referred to as a register-indirect register addressing mode. This field applies to the destination operand and the first source operand, src0. Support for src1 is device dependent. See Table XX (Indirect source addressing support available in device hardware) in ISA Execution Environment for details.

Programming Notes
Instructions with 3 source operands use Direct Addressing.

Value	Name	Description
0	Direct	'Direct' register addressing
1	Indirect	'Register-Indirect' (or in short 'Indirect'). Register-indirect register addressing

AtomicCtrl

AtomicCtrl	
Source:	EuIsa
Size (in bits):	1
Enables the atomic instruction control option	

Value	Name	Description
0b	No Operation **[Default]**	This value leaves thread switching up to the EU's scheduler.
1b	Atomic	Prevent any thread switch immediately following this instruction. Always execute the next instruction (which may not be next sequentially if the current instruction branches). The next instruction gets highest priority in the thread arbitration for the execution pipelines.

Attribute_Component_Format

Attribute_Component_Format

Source: RenderCS

Size (in bits): 2

Value	Name	Description
00b	disabled **[Default]**	All components disabled
01b	.xy	2D attribute, z and w components disabled
10b	.xyz	3D attribute, w components disabled
11b	.xyzw	4D attribute, no disabled components

ChanOff

ChanOff	
Source:	EuIsa
Size (in bits):	3

Channel Offset
This enumeration (instruction field) provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed. Some of the ARFs affected by this offset are:

- ce access (channel enable / the execution mask): bitwise offset
- f# access: bitwise offset (e.g. in predication or conditional modifier access)
- acc# access via AccWrEn (implicit only): in subregister units of the data type

Note: ChanOff is functionally the same as a concatenation of the previous QtrCtrl and NibCtrl and supersedes these fields.

Restriction
The execution size (ExecSize) must be a factor of the chosen offset. For instance, M28 (offset 28) can only be used with SIMD1, SIMD2, and SIMD4.

Value	Name	Description
000b	M0 **[Default]**	ARF access begins at channel 0
001b	M4	ARF access begins at channel 4 (e.g. 4 bits in on ce and f#
010b	M8	
011b	M12	
100b	M16	
101b	M20	
110b	M24	
111b	M28	

COMPONENT_ENABLES

COMPONENT_ENABLES	
Source:	RenderCS
Size (in bits):	4
If enabled, the component will be stored in the URB.	

Value	Name
0000b	NONE
0001b	X
0010b	Y
0011b	XY
0100b	Z
0101b	XZ
0110b	YZ
0111b	XYZ
1000b	W
1001b	XW
1010b	YW
1011b	XYW
1100b	ZW
1101b	XZW
1110b	YZW
1111b	XYZW

DPASOperandPrecision

DPASOperandPrecision	
Source:	EuIsa
Size (in bits):	3
This operand defines the number of bits per element in the dpas instruction. E.g. s8 would indicate that a dword is chunked into 4 x 8b signed values; u2 would indicate that the operand a DWORD is broken into 16 x 2b unsigned values.	

Value	Name
000b	u1
001b	u2
010b	u4
011b	u8
100b	s1
101b	s2
110b	s4
111b	s8

EU_OPCODE

EU_OPCODE

Source: EuIsa

Size (in bits): 7

Value	Name	
40h	add	Instruction: add
4Eh	addc	Instruction: addc
65h	and	Instruction: and
6Ch	asr	Instruction: asr
42h	avg	Instruction: avg
78h	bfe	Instruction: bfe
79h	bfi1	Instruction: bfi1
7Ah	bfi2	Instruction: bfi2
77h	bfrev	Instruction: bfrev
23h	brc	Instruction: brc
21h	brd	Instruction: brd
28h	break	Instruction: break
2Ch	call	Instruction: call
2Bh	calla	Instruction: calla
4Dh	cbit	Instruction: cbit
70h	cmp	Instruction: cmp
71h	cmpn	Instruction: cmpn
29h	cont	Instruction: cont
72h	csel	Instruction: csel
24h	else	Instruction: else
25h	endif	Instruction: endif
4Bh	fbh	Instruction: fbh
4Ch	fbl	Instruction: fbl
43h	frc	Instruction: frc

EU_OPCODE		
2Eh	goto	Instruction: goto
2Ah	halt	Instruction: halt
22h	if	Instruction: if
0h	illegal	Instruction: illegal
20h	jmpi	Instruction: jmpi
2Fh	join	Instruction: join
4Ah	lzd	Instruction: lzd
48h	mac	Instruction: mac
49h	mach	Instruction: mach
5Bh	mad	Instruction: mad
38h	math	Instruction: math
61h	mov	Instruction: mov
63h	movi	Instruction: movi
41h	mul	Instruction: mul
60h	nop	Instruction: nop
64h	not	Instruction: not
66h	or	Instruction: or
2Dh	ret	Instruction: ret
45h	rndd	Instruction: rndd
46h	rnde	Instruction: rnde
44h	rndu	Instruction: rndu
47h	rndz	Instruction: rndz
6Fh	rol	Instruction: rol
6Eh	ror	Instruction: ror
62h	sel	Instruction: sel
31h	send	Instruction: send
32h	sendc	Instruction: sendc
69h	shl	Instruction: shl
68h	shr	Instruction: shr

EU_OPCODE		
4Fh	subb	Instruction: subb
1h	sync	Instruction: sync
27h	while	Instruction: while
67h	xor	Instruction: xor

ExecSize

ExecSize	
Source:	EuIsa
Size (in bits):	3

Execution Size This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

Restriction
An operand's Width must be less-than-or-equal to ExecSize

Value	Name	Programming Notes
000b	1 Channel (Scalar operation) **[Default]**	
001b	2 Channels	
010b	4 Channels	
011b	8 Channels	
100b	16 Channels	[] 4-byte or smaller data types. Excludes DF, Q, and UQ types.
101b	32 Channels	[] 2-byte or 1-byte data types. Excludes D, DF, F, Q, UD, and UQ types.
110b-111b	Reserved	

Fixed Function ID

FFID - Fixed Function ID

Size (in bits): 4

Fixed functions are hardware units that execute complex graphics or media command on behalf of application software. Some fixed functions send work down a pipeline through a series of fixed functions. Multiple fixed functions can be running at the same time. The GPU tracks activity from the fixed function with its FFID.

Programming Notes

Software does not specify the FFID, and does not normally use the FFID value. The FFID value is available to a EU thread in an ARF.

Value	Name	Description
00h	Null	
03h	POSH Vertex Shader	
04h	Hull Shader	
05h	Domain Shader	
06h	Texel Shader	Adaptive Multi-Frequency Shader
07h	General Purpose Thread Spawner	GPGPU command queue thread dispatcher
08h	General Purpose Asynchronous Thread Spawner	GPGPU command queue's thread dispatcher for secondary queue
09h	Vertex Shader	
0Ch	Geometry Shader	

FlagModifier

FlagModifier	
Source:	EuIsa
Size (in bits):	4

Flag (Conditional) Modifier - This field sets the flag register based on the internal conditional signals output from the execution pipe such as sign, zero, overflow and NaNs, etc. If this field is set to 0000, no flag registers are updated. Flag registers are not updated for instructions with embedded compares. This field may also be referred to as the flag destination control field.

Value	Name	Description
0000b	None **[Default]**	None
0001b	(ze)	Zero
0010b	(nz)	NotZero
0011b	(gt)	Greater-than
0100b	(ge)	Greater-than-or-equal
0101b	(lt)	Less-than
0110b	(le)	Less-than-or-equal
0111b	Reserved	
1000b	(ov)	Overflow
1001b	(un)	Unordered (NaN)
1110b-1111b	Reserved	

GW_FENCE_PORTS

GW_FENCE_PORTS	
Source:	BSpec
Size (in bits):	0

Bit mask specifies the list of data ports to be fenced.

Value	Name	Description
0	None **[Default]**	No fence is performed when no bits are set.

HorzStride

HorzStride	
Source:	EuIsa
Size (in bits):	2

Horizontal Stride This field provides the distance in unit of data elements between two adjacent data elements within a row (horizontal) in the register region for the operand. This field applies to both destination and source operands. This field is not present for an immediate source operand.

A horizontal stride of 0 is used for a row that is one-element wide, useful when an instruction repeats a column value or repeats a scalar value. For example, adding a single column to every column in a 2D array or adding a scalar to every element in a 2D array uses HorzStride of 0. A horizontal stride of 1 indicates that elements are adjacent within a row. References to HorzStride in this volume normally reference the value not the encoding, so there are references to HorzStride of 4, which is encoded as 11b.

Value	Name
00b	0 elements
01b	1 elements
10b	2 elements
11b	4 elements

ImmDataType

ImmDataType

Source: EuIsa

Size (in bits): 4

Numeric data type of source and destination operand. Three source instructions use a 3-bit encoding that allows fewer data types.

Value	Name	Description
0000b	:uv	Packed Unsigned Half-Byte Integer Vector, 8 x 4-Bit Unsigned Integer
0001b	:uw	Unsigned Word (16-bit) Integer
0010b	:ud	Unsigned DoubleWord (32-bit) Integer
0011b	:uq	Unsigned Quadword (64-bit) Integer
0100b	:v	Packed Signed Half-Byte Integer Vector, 8 x 4-Bit Signed Integer
0101b	:w	Signed Word (16-bit) Integer
0110b	:d	Signed DoubleWord (32-bit) Integer
0111b	:q	Signed QuadWord (64-bit) Integer
1000b	:vf	Packed Restricted Float Vector, 4 x 8-Bit Restricted Precision Floating-Point Number.
1001b	:hf	Half (16-bit) Float
1010b	:f	Single-Precision (32-bit) Float
1011b	:df	Double-Precision (64-bit) Float
1100b	Reserved	
1101b	Reserved	
[1110b-1111b]	Reserved	

MathFC

MathFC

Source: EuIsa

Size (in bits): 4

Math Function Control

Value	Name	Description	Programming Notes	
0001b	INV	Reciprocal (Multiplicative Inverse): 1/src0	Table:special value processing | Src | +inf | +0 / +Denorm | -0 / -Denorm | -inf | NaN | |Dest - IEEE mode | +0 | +inf | -inf | -0 | NaN | |Dest - ALT mode | +fmax | -fmax | -fmax | NaN | | Src | +inf | +0 | -0 | -inf | NaN | |Dest - IEEE mode | +0 | +inf | -inf | -0 | NaN |	Syntax: MATH_UNARY_REGIMM
0010b	LOG	Natural log: ln(src0)	Table:special value processing | Src | +inf | +0 / +Denorm | -0 / -Denorm | -inf | -F | NaN | |Dest - IEEE mode | +inf | -inf | -inf | NaN | NaN | NaN | NaN | |Dest - ALT mode | -fmax | -fmax | +F | NaN |	Syntax: MATH_UNARY_REGIMM

			Src | +inf | +0 | -0 | -inf | -F | NaN | |Dest - IEEE mode | +inf | -inf | -inf | NaN | NaN | NaN |	
0011b	EXP	Exponential (E^src0)	Table:special value processing | Src | +inf | +0 / +Denorm | -0 / -Denorm | -inf | -F | NaN | |Dest - IEEE mode | +inf | 1 | 1 | 0 | +F | NaN | |Dest - ALT mode | | 1 | 1 | | +F | NaN | | Src | +inf | +0 | -0 | -inf | -F | NaN | |Dest - IEEE mode | +inf | 1 | 1 | 0 | +F | NaN |	Syntax: MATH_UNARY_REGIMM
0100b	SQT	Square Root	Table:special value processing | Src | +inf | +0 / +Denorm | -0 / -Denorm | -inf | -F | NaN | |Dest - IEEE mode | +inf | 0 | -0 | NaN | NaN | NaN | |Dest - ALT mode | | 0 | 0 | | +F | NaN | | Src | +inf | +0 | -0 | -inf | -F | NaN | |Dest - IEEE mode | +inf | 0 | -0 |	Syntax: MATH_UNARY_REGIMM

			NaN | NaN | NaN |	
0101b	RSQT	Reciprocal Square Root: 1/sqt(src)	Table:special value processing | Src | +inf | +0 / +Denorm | -0 / -Denorm | -inf | -F | NaN | |Dest - IEEE mode | +0 | +inf | -inf | NaN | NaN | NaN | |Dest - ALT mode | +fmax | +fmax | +F | NaN | | Src | +inf | +0 | -0 | -inf | -F | NaN | |Dest - IEEE mode | +0 | +inf | -inf | NaN | NaN | NaN |	Syntax: MATH_UNARY_REGIMM
0110b	SIN	Sine function. sin(src0)	Table:special value processing | Src | +inf | +0 / +Denorm | -0 / -Denorm | -inf | -F | NaN | |Dest - IEEE mode | NaN | +0 | -0 | NaN | -1 to 1 | NaN | |Dest - ALT mode | +0 | -0 | -1 to 1 | NaN | | Src | +inf | +0 | -0 | -inf | -F | NaN | |Dest - IEEE mode | NaN | +0 | -0 | NaN | -1 to 1 | NaN |	Syntax: MATH_UNARY_REGIMM

MathFC				
0111b	COS	Cosine function. cos(src0)	Table:special value processing | Src | +inf | +0 / +Denorm | -0 / -Denorm | -inf | -F | NaN | |Dest - IEEE mode | NaN | +0 | -0 | NaN | -1 to 1 | NaN | |Dest - ALT mode | +1 | +1 | -1 to 1 | NaN | | Src | +inf | +0 | -0 | -inf | -F | NaN | |Dest - IEEE mode | NaN | +0 | -0 | NaN | -1 to 1 | NaN |	Syntax: MATH_UNARY_REGIMM
1000b	Reserved			
[1001b-1010b]	Reserved	Previously fdiv and pow. Fdiv x/y may be emulated via mul and inv: x*inv(y). Pow can be replaced via exp and log: X^Y = E^(Y*LOG(X))		
1011b	IDIV	Integer Divide with Quotient and Remainder. The quotient goes in the destination register; the remainder goes in the following register.		Syntax: MATH_BINARY_REG_REGIMM
1100b	IQOT	Integer Quotient only		Syntax: MATH_BINARY_REG_REGIMM
1101b	IREM	Integer Remainder only		Syntax: MATH_BINARY_REG_REGIMM
1110b	INVM	Reciprocal Macro for IEEE754-compliant fdiv		Syntax: MATH_MACRO_BINARY_REG_REG
1111b	RSQTM	Reciprocal Square Root Macro for IEEE754-compliant rsqt		Syntax: MATH_MACRO_UNARY_REG

MediaCompressionFormat

CMP_FMT_MEDIA_TGL - MediaCompressionFormat _TGL		
Size (in bits): 5		
Media Compression Format		
Value	**Name**	**Description**
00000b	CMF_0x0	Reserved
00001b	CMF_0x1	Reserved
00010b	CMF_0x2	Reserved
00011b	CMF_0x3	YUY2
00100b	CMF_0x4	Y410 (10:10:10:2)
00101b	CMF_0x5	Y216
00110b	CMF_0x6	Y416
00111b	CMF_0x7	P010
01000b	CMF_0x8	P016
01001b	CMF_0x9	AYUV
01010b	CMF_0xA	ARGB 8b
01011b	CMF_0xB	YCRCB_SwapY
01100b	CMF_0xC	YCRCB_SwapUV
01101b	CMF_0xD	YCRCB_SwapUVY
01110b	CMF_0xE	RGB 10b
01111b	CMF_0xF	NV21/NV12
10000b	CMF_0x10	RGBA_16Float

Performance Counter Report Formats

Performance Counter Report Formats	
Size (in bits):	3

Value	Name
001b	
010b	
011b	
100b	
110b	
111b	

PredCtrl

PredCtrl	
Source:	EuIsa
Size (in bits):	4

Value	Name	Exists If
0000b	No Predication (normal) **[Default]**	
0001b	Sequential Flag Channel Mapping	
0010b	Replication swizzle .x	(Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16')
0010b	.anyv (any from f0.0-f1.0 on the same channel)	(Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1')
0011b	Replication swizzle .y	(Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16')
0011b	.allv (all of f0.0-f1.0 on the same channel)	(Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1')
0100b	Replication swizzle .z	(Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16')
0100b	.any2h (any in group of 2 channels)	(Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1')
0101b	Replication swizzle .w	(Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16')
0101b	.all2h (all in group of 2 channels)	(Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1')
0110b	.any4h	
0111b	.all4h	
1000b-1111b	Reserved	(Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align16')
1000b	.any8h (any in group of 8 channels)	(Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1')
1001b	.all8h (all in group of 8 channels)	(Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1')
1010b	.any16h (any in group of 16 channels)	(Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1')
1011b	.all16h (all in group of 16 channels)	(Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1')
1100b	.any32h (any in group of 32 channels)	(Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1')
1101b	.all32h (all in group of 32 channels)	(Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1')
1110b-1111b	Reserved	(Structure[EU_INSTRUCTION_CONTROLS_A][AccessMode]=='Align1')

PREFERRED_SLM_SIZE

PREFERRED_SLM_SIZE - PREFERRED_SLM_SIZE		
Size (in bits):	0	
Specifies the preferred SLM size per subslice for this kernel.		

Restriction
Selected PREFERRED_SLM_SIZE must specify a size >= the selected SLM_SIZE

Value	Name	Description
0x0	Max **[Default]**	Preferred SLM size is the largest SLM size supported in the subslice.

RegDataType

RegDataType	
Source:	EuIsa
Size (in bits):	4

Destination Type Numeric data type of the destination operand dst. The bits of the destination operand are interpreted as the identified numeric data type, rather than coerced into a type implied by the operator. For a send or sendc instruction, this field applies to CurrDst, the current destination operand. Three source instructions use a 3-bit encoding that allows fewer data types.

Value	Name	Description
0000b	:ub	Unsigned Byte (8-bit) Integer
0001b	:uw	Unsigned Word (16-bit) Integer
0010b	:ud	Unsigned DoubleWord (32-bit) Integer
0011b	:uq	Unsigned Quadword (64-bit) Integer
0100b	:b	Signed Byte (8-bit) Integer
0101b	:w	Signed Word (16-bit) Integer
0110b	:d	Signed DoubleWord (32-bit) Integer
0111b	:q	Signed QuadWord (64-bit) Integer
1000b	Reserved	
1001b	:hf	Half (16-bit) Float
1010b	:f	Single-Precision (32-bit) Float
1011b	:df	Double-Precision (64-bit) Float
1100b	Reserved	
1101b	Reserved	
1110b	Reserved	
1111b	Reserved	

Registers Per Thread

REGISTERS_PER_THREAD_SIZE - Registers Per Thread		
Size (in bits): 0		
Specifies the minimum number of registers allocated for each thread dispatch.		
Value	**Name**	**Description**
0	Default **[Default]**	For all threads except Compute thread, use 128 registers. For Compute Threads, use either 128 or 256 registers, based on STATE_COMPUTE_MODE.

RENDER_BARRIER_STAGE

RENDER_BARRIER_STAGE	
Size (in bits):	7
Stages a render barrier can either signal or wait.	

Value	Name
0x1	TOP
0x2	Color
0x4	Gpgpu
0x6	GPGPU and Color
0x10	Geom
0x20	Z
0x40	PS

RENDER_BARRIER_TYPE

RENDER_BARRIER_TYPE	
Size (in bits):	2

Value	Name
0x1	Signal
0x2	Wait
0x3	Immediate

RenderCompressionFormat

CMP_FMT_RENDER_TGL - RenderCompressionFormat_TGL

Source: BSpec

Size (in bits): 5

Compression Format

Value	Name	Description
00001b		Format: Reserved; Casting: N/A; Encoding: 0x1; IP: N/A
00010b		Format: Reserved; Casting: N/A; Encoding: 0x2; IP: N/A
01010b		

Format	Casting	Encoding	IP
B8G8R8A8_UNORM	8b	0xA	3D
B8G8R8A8_UNORM_SRGB	8b	0xA	3D
R8G8B8A8_UNORM	8b	0xA	3D
R8G8B8A8_UNORM_SRGB	8b	0xA	3D
B5G6R5_UNORM	8b	0xA	3D
B5G6R5_UNORM_SRGB	8b	0xA	3D
B5G5R5A1_UNORM	8b	0xA	3D
B5G5R5A1_UNORM_SRGB	8b	0xA	3D
B4G4R4A4_UNORM	8b	0xA	3D
B4G4R4A4_UNORM_SRGB	8b	0xA	3D
R8G8_UNORM	8b	0xA	3D
B5G5R5X1_UNORM	8b	0xA	3D
B5G5R5X1_UNORM_SRGB	8b	0xA	3D
A1B5G5R5_UNORM	8b	0xA	3D
A4B4G4R4_UNORM	8b	0xA	3D
R8_UNORM	8b	0xA	3D
A8_UNORM	8b	0xA	3D

10000b		

Format	Casting	Encoding	IP
R16G16B16A16_FLOAT	16b	0x10	3D
R16G16B16X16_FLOAT	16b	0x10	3D
R16G16_FLOAT	16b	0x10	3D
R16_FLOAT	16b	0x10	3D

10001b		

Format	Casting	Encoding	IP

CMP_FMT_RENDER_TGL - RenderCompressionFormat_TGL

		R32G32B32A32_FLOAT	8b	0x11	3D
		R32G32B32X32_FLOAT	8b	0x11	3D
		R32G32_FLOAT	8b	0x11	3D
		R32_FLOAT	8b	0x11	3D
10010b		Format	Casting	Encoding	IP
		R32G32B32A32_SINT	8b	0x12	3D
		R32G32_SINT	8b	0x12	3D
		R32_SINT	8b	0x12	3D
10011b		Format	Casting	Encoding	IP
		R32G32B32A32_UINT	8b	0x13	3D
		R32G32_UINT	8b	0x13	3D
		R32_UINT	8b	0x13	3D
		D24_UNORM_X8	8b	0x13	3D
10100b		Format	Casting	Encoding	IP
		R16G16B16A16_UNORM	16b	0x14	3D
		R16G16_UNORM	16b	0x14	3D
		R16_UNORM	16b	0x14	3D
10101b		Format	Casting	Encoding	IP
		R16G16B16A16_SNORM	16b	0x15	3D
		R16G16_SNORM	16b	0x15	3D
		R16_SNORM	16b	0x15	3D
10110b		Format	Casting	Encoding	IP
		R16G16B16A16_SINT	16b	0x16	3D
		R16G16_SINT	16b	0x16	3D
		R16_SINT	16b	0x16	3D
10111b		Format	Casting	Encoding	IP
		R16G16B16A16_UINT	16b	0x17	3D
		R16G16_UINT	16b	0x17	3D
		R16_UINT	16b	0x17	3D
11000b		Format	Casting	Encoding	IP
		R10G10B10A2_UNORM	8b*	0x18	3D
		R10G10B10A2_UNORM_SRGB	8b*	0x18	3D
		B10G10R10A2_UNORM	8b*	0x18	3D
		B10G10R10A2_UNORM_SRGB	8b*	0x18	3D

CMP_FMT_RENDER_TGL - RenderCompressionFormat_TGL

11001b		Format: R10G10B10_FLOAT_A2_UNORM; Casting: 8b*; Encoding: 0x19; IP: 3D
11010b		Format: R10G10B10A2_UINT; Casting: 8b*; Encoding: 0x1A; IP: 3D
11011b		Format: R8G8B8A8_SNORM; Casting: 8b; Encoding: 0x1B; IP: 3D Format: R8G8_SNORM; Casting: 8b; Encoding: 0x1B; IP: 3D Format: R8_SNORM; Casting: 8b; Encoding: 0x1B; IP: 3D
11100b		Format: R8G8B8A8_SINT; Casting: 8b; Encoding: 0x1C; IP: 3D Format: R8G8_SINT; Casting: 8b; Encoding: 0x1C; IP: 3D Format: R8_SINT; Casting: 8b; Encoding: 0x1C; IP: 3D
11101b		Format: R8G8B8A8_UINT; Casting: 8b; Encoding: 0x1D; IP: 3D Format: R8G8_UINT; Casting: 8b; Encoding: 0x1D; IP: 3D Format: R8_UINT; Casting: 8b; Encoding: 0x1D; IP: 3D
11110b		Format: R11G11B10_FLOAT; Casting: 8b*; Encoding: 0x1E; IP: 3D

Saturate

Saturate	
Size (in bits):	1
This field controls the destination saturation. When it is set, output data to the destination register are saturated. The saturation operation depends on the destination data type. Saturation is the operation that converts any data that is outside the saturation target range for the data type to the closest representable value with the target range. If destination type is float, saturation target range is [0, 1]. For example, any positive number greater than 1 (including +INF) is saturated to 1 and any negative number (including -INF) is saturated to 0. A NaN is saturated to 0, For integer data types, the maximum range for the given numerical data type is the saturation target range. When it is not set, output data to the destination register are not saturated. For example, a wrapped result (modular) is output to the destination for an overflowed integer data. More details can be found in the Data Types chapter.	

Value	Name
0	No Destination modification **[Default]**
1	Saturate Destination

SFID

SFID

Source: EuIsa

Size (in bits): 4

The following table lists the assignments (encodings) of the Shared Function and Fixed Function IDs used within the GPE. A Shared Function is a valid target of a message initiated via a 'send' instruction. A Fixed Function is an identifiable unit of the 3D or Media pipeline. Note that the Thread Spawner is both a Shared Function and Fixed Function. Note: The initial intention was to combine these two ID namespaces, so that (theoretically) an agent (such as the Thread Spawner) that served both as a Shared Function and Fixed Function would have a single, unique 4-bit ID encoding. However, this combination is not a requirement of the architecture.

Programming Notes

SFID_DP_DC1 is an extension of SFID_DP_DC0 to allow for more message types. They act as a single logical entity.

SFID_DP_DC1 and SFID_DP_DC2 are extensions of SFID_DP_DC0 to allow for more message types. They act as a single logical entity.

Value	Name	Description	
0000b	SFID_NULL	Null	Syntax: SEND_BINARY
0001b	Reserved	Reserved	
0010b	SFID_SAMPLER	Sampler	Syntax: SEND_BINARY
0011b	SFID_GATEWAY	Message Gateway	Syntax: SEND_BINARY
0100b	SFID_DP_DC2	Data Cache Data Port 2	Syntax: SEND_BINARY
0101b	SFID_DP_RC	Render Cache Data Port	Syntax: SEND_BINARY
0110b	SFID_URB	URB	Syntax: SEND_BINARY
0111b	SFID_SPAWNER	Thread Spawner	Syntax: SEND_BINARY
1000b	SFID_VME	Video Motion Estimation	Syntax: SEND_BINARY
1001b	SFID_DP_DCRO	Data Cache Read Only Data Port	Syntax: SEND_BINARY
1010b	SFID_DP_DC0	Data Cache Data Port	Syntax: SEND_BINARY
1011b	SFID_PI	Pixel Interpolator	Syntax: SEND_BINARY
1100b	SFID_DP_DC1	Data Cache Data Port 1	Syntax: SEND_BINARY
1101b	SFID_CRE	Check and Refinement Engine	Syntax: SEND_BINARY
1110b-1111b	Reserved		

Shader Channel Select

Shader Channel Select		
Size (in bits):	3	
Value	**Name**	**Description**
0	ZERO	
1	ONE	
2	Reserved	
3	Reserved	
4	RED	Shader channel is set to surface red channel
5	GREEN	Shader channel is set to surface green channel
6	BLUE	Shader channel is set to surface blue channel
7	ALPHA	Shader channel is set to surface alpha channel

SIMD Mode

SIMD Mode	
Size (in bits):	3

Value	Name
0	SIMD8 + Integer Return
1	SIMD8
2	SIMD16
3	Reserved2
4	SIMD16 + Integer Return
5	SIMD8H
6	SIMD16H
7	Reserved7

Slice Hash Control

Slice Hash Control

Source: RenderCS

Size (in bits): 2

This is a sample showing the basic structure of an explicit enumeration.

Value	Name	Description
00b	Computed	Use Computed pixelhash_id
01b	Unbalanced table[0]	Use Computed pixelhash_id when balanced, Table[0] when unbalanced
10b	Table[0]	Use Table[0]
11b	Table[1]	Use Table[1]

SrcMod

SrcMod

Source: EuIsa

Size (in bits): 2

Source Modifier This field specifies the numeric modification of a source operand. The value of each data element of a source operand can optionally have its absolute value taken and/or its sign inverted prior to delivery to the execution pipe. The absolute value is prior to negate such that a guaranteed negative value can be produced. This field only applies to source operand. It does not apply to destination. This field is not present for an immediate source operand.

When used with logic instructions (and, not, or, xor), this field indicates whether the source bits are inverted (bitwise NOT) before delivery to the execution pipe, regardless of the source type.

Value	Name	Description
00b	No modification	
01b	abs	Absolute value Logic instructions: No modification (This encoding cannot be selected in the assembler syntax)
10b	negate	Negate Logic instructions: Bitwise NOT, inverting the source bits
11b	negate of abs	Negate of the absolute (forced negative value) Logic instructions: No modification (This encoding cannot be selected in the assembler syntax)

SURFACE_FORMAT

SURFACE_FORMAT

Size (in bits): 9

The following table indicates the supported surface formats and the 9-bit encoding for each. Note that some of these formats are used not only by the Sampling Engine, but also by the Data Port and the Vertex Fetch unit.

Value	Name	Description
000h	R32G32B32A32_FLOAT	
001h	R32G32B32A32_SINT	
002h	R32G32B32A32_UINT	
003h	R32G32B32A32_UNORM	
004h	R32G32B32A32_SNORM	
005h	R64G64_FLOAT	
006h	R32G32B32X32_FLOAT	
007h	R32G32B32A32_SSCALED	
008h	R32G32B32A32_USCALED	
020h	R32G32B32A32_SFIXED	
021h	R64G64_PASSTHRU	
040h	R32G32B32_FLOAT	
041h	R32G32B32_SINT	
042h	R32G32B32_UINT	
043h	R32G32B32_UNORM	
044h	R32G32B32_SNORM	
045h	R32G32B32_SSCALED	
046h	R32G32B32_USCALED	
050h	R32G32B32_SFIXED	
080h	R16G16B16A16_UNORM	
081h	R16G16B16A16_SNORM	
082h	R16G16B16A16_SINT	
083h	R16G16B16A16_UINT	
084h	R16G16B16A16_FLOAT	
085h	R32G32_FLOAT	
086h	R32G32_SINT	
087h	R32G32_UINT	
088h	R32_FLOAT_X8X24_TYPELESS	
089h	X32_TYPELESS_G8X24_UINT	
08Ah	L32A32_FLOAT	

SURFACE_FORMAT

08Bh	R32G32_UNORM	
08Ch	R32G32_SNORM	
08Dh	R64_FLOAT	
08Eh	R16G16B16X16_UNORM	
08Fh	R16G16B16X16_FLOAT	
090h	A32X32_FLOAT	
091h	L32X32_FLOAT	
092h	I32X32_FLOAT	
093h	R16G16B16A16_SSCALED	
094h	R16G16B16A16_USCALED	
095h	R32G32_SSCALED	
096h	R32G32_USCALED	
0A0h	R32G32_SFIXED	
0A1h	R64_PASSTHRU	
0C0h	B8G8R8A8_UNORM	
0C1h	B8G8R8A8_UNORM_SRGB	
0C2h	R10G10B10A2_UNORM	
0C3h	R10G10B10A2_UNORM_SRGB	
0C4h	R10G10B10A2_UINT	
0C5h	R10G10B10_SNORM_A2_UNORM	
0C7h	R8G8B8A8_UNORM	
0C8h	R8G8B8A8_UNORM_SRGB	
0C9h	R8G8B8A8_SNORM	
0CAh	R8G8B8A8_SINT	
0CBh	R8G8B8A8_UINT	
0CCh	R16G16_UNORM	
0CDh	R16G16_SNORM	
0CEh	R16G16_SINT	
0CFh	R16G16_UINT	
0D0h	R16G16_FLOAT	
0D1h	B10G10R10A2_UNORM	
0D2h	B10G10R10A2_UNORM_SRGB	
0D3h	R11G11B10_FLOAT	
0D5h	R10G10B10_FLOAT_A2_UNORM	
0D6h	R32_SINT	
0D7h	R32_UINT	

SURFACE_FORMAT		
0D8h	R32_FLOAT	
0D9h	R24_UNORM_X8_TYPELESS	
0DAh	X24_TYPELESS_G8_UINT	
0DDh	L32_UNORM	
0DEh	A32_UNORM	
0DFh	L16A16_UNORM	
0E0h	I24X8_UNORM	
0E1h	L24X8_UNORM	
0E2h	A24X8_UNORM	
0E3h	I32_FLOAT	
0E4h	L32_FLOAT	
0E5h	A32_FLOAT	
0E6h	X8B8_UNORM_G8R8_SNORM	
0E7h	A8X8_UNORM_G8R8_SNORM	
0E8h	B8X8_UNORM_G8R8_SNORM	
0E9h	B8G8R8X8_UNORM	
0EAh	B8G8R8X8_UNORM_SRGB	
0EBh	R8G8B8X8_UNORM	
0ECh	R8G8B8X8_UNORM_SRGB	
0EDh	R9G9B9E5_SHAREDEXP	
0EEh	B10G10R10X2_UNORM	
0F0h	L16A16_FLOAT	
0F1h	R32_UNORM	
0F2h	R32_SNORM	
0F3h	R10G10B10X2_USCALED	
0F4h	R8G8B8A8_SSCALED	
0F5h	R8G8B8A8_USCALED	
0F6h	R16G16_SSCALED	
0F7h	R16G16_USCALED	
0F8h	R32_SSCALED	
0F9h	R32_USCALED	
100h	B5G6R5_UNORM	
101h	B5G6R5_UNORM_SRGB	
102h	B5G5R5A1_UNORM	
103h	B5G5R5A1_UNORM_SRGB	
104h	B4G4R4A4_UNORM	

SURFACE_FORMAT		
105h	B4G4R4A4_UNORM_SRGB	
106h	R8G8_UNORM	
107h	R8G8_SNORM	
108h	R8G8_SINT	
109h	R8G8_UINT	
10Ah	R16_UNORM	
10Bh	R16_SNORM	
10Ch	R16_SINT	
10Dh	R16_UINT	
10Eh	R16_FLOAT	
10Fh	A8P8_UNORM_PALETTE0	
110h	A8P8_UNORM_PALETTE1	
111h	I16_UNORM	
112h	L16_UNORM	
113h	A16_UNORM	
114h	L8A8_UNORM	
115h	I16_FLOAT	
116h	L16_FLOAT	
117h	A16_FLOAT	
118h	L8A8_UNORM_SRGB	
119h	R5G5_SNORM_B6_UNORM	
11Ah	B5G5R5X1_UNORM	
11Bh	B5G5R5X1_UNORM_SRGB	
11Ch	R8G8_SSCALED	
11Dh	R8G8_USCALED	
11Eh	R16_SSCALED	
11Fh	R16_USCALED	
122h	P8A8_UNORM_PALETTE0	
123h	P8A8_UNORM_PALETTE1	
124h	A1B5G5R5_UNORM	
125h	A4B4G4R4_UNORM	
126h	L8A8_UINT	
127h	L8A8_SINT	
140h	R8_UNORM	
141h	R8_SNORM	
142h	R8_SINT	

SURFACE_FORMAT		
143h	R8_UINT	
144h	A8_UNORM	
145h	I8_UNORM	
146h	L8_UNORM	
147h	P4A4_UNORM_PALETTE0	
148h	A4P4_UNORM_PALETTE0	
149h	R8_SSCALED	
14Ah	R8_USCALED	
14Bh	P8_UNORM_PALETTE0	
14Ch	L8_UNORM_SRGB	
14Dh	P8_UNORM_PALETTE1	
14Eh	P4A4_UNORM_PALETTE1	
14Fh	A4P4_UNORM_PALETTE1	
150h	Y8_UNORM	
152h	L8_UINT	
153h	L8_SINT	
154h	I8_UINT	
155h	I8_SINT	
180h	DXT1_RGB_SRGB	
181h	R1_UNORM	SET0_LEGACY: Undefined behavior if used in any feature added. See Legacy sampler feature page for details
182h	YCRCB_NORMAL	
183h	YCRCB_SWAPUVY	
184h	P2_UNORM_PALETTE0	
185h	P2_UNORM_PALETTE1	
186h	BC1_UNORM	(DXT1)
187h	BC2_UNORM	(DXT2/3)
188h	BC3_UNORM	(DXT4/5)
189h	BC4_UNORM	
18Ah	BC5_UNORM	
18Bh	BC1_UNORM_SRGB	(DXT1_SRGB)
18Ch	BC2_UNORM_SRGB	(DXT2/3_SRGB)
18Dh	BC3_UNORM_SRGB	(DXT4/5_SRGB)
18Eh	MONO8	SET0_LEGACY: Undefined behavior if used in any feature added. See Legacy sampler feature page for details
18Fh	YCRCB_SWAPUV	
190h	YCRCB_SWAPY	

SURFACE_FORMAT		
191h	DXT1_RGB	
192h	RESERVED_192	This value is reserved for internal use.
193h	R8G8B8_UNORM	
194h	R8G8B8_SNORM	
195h	R8G8B8_SSCALED	
196h	R8G8B8_USCALED	
197h	R64G64B64A64_FLOAT	
198h	R64G64B64_FLOAT	
199h	BC4_SNORM	
19Ah	BC5_SNORM	
19Bh	R16G16B16_FLOAT	
19Ch	R16G16B16_UNORM	
19Dh	R16G16B16_SNORM	
19Eh	R16G16B16_SSCALED	
19Fh	R16G16B16_USCALED	
1A1h	BC6H_SF16	
1A2h	BC7_UNORM	
1A3h	BC7_UNORM_SRGB	
1A4h	BC6H_UF16	
1A5h	PLANAR_420_8	
1A6h	PLANAR_420_16	
1A8h	R8G8B8_UNORM_SRGB	
1A9h	ETC1_RGB8	
1AAh	ETC2_RGB8	
1ABh	EAC_R11	
1ACh	EAC_RG11	
1ADh	EAC_SIGNED_R11	
1AEh	EAC_SIGNED_RG11	
1AFh	ETC2_SRGB8	
1B0h	R16G16B16_UINT	
1B1h	R16G16B16_SINT	
1B2h	R32_SFIXED	
1B3h	R10G10B10A2_SNORM	
1B4h	R10G10B10A2_USCALED	
1B5h	R10G10B10A2_SSCALED	
1B6h	R10G10B10A2_SINT	

SURFACE_FORMAT		
1B7h	B10G10R10A2_SNORM	
1B8h	B10G10R10A2_USCALED	
1B9h	B10G10R10A2_SSCALED	
1BAh	B10G10R10A2_UINT	
1BBh	B10G10R10A2_SINT	
1BCh	R64G64B64A64_PASSTHRU	
1BDh	R64G64B64_PASSTHRU	
1C0h	ETC2_RGB8_PTA	
1C1h	ETC2_SRGB8_PTA	
1C2h	ETC2_EAC_RGBA8	
1C3h	ETC2_EAC_SRGB8_A8	
1C8h	R8G8B8_UINT	
1C9h	R8G8B8_SINT	
1FFh	RAW	

SyncFC

SyncFC		
Source:	EuIsa	
Size (in bits):	4	
Subfunctions that the sync instruction supports.		

Value	Name	Description	
0000b	No Operation	Performs no operation. Regular SWSB constraints are checked.	Syntax: SYNC_UNARY
0010b	SBID Read Wait	Blocks until pending out-of-order source accesses are complete. If a mask is provided as immediate value in src0 then specific SBID resources can be checked, else all SBID resources are checked for source access complete status.	Syntax: SYNC_UNARY
0011b	SBID Write Wait	Blocks until pending out-of-order writebacks are complete. If a mask is provided as immediate value in src0 then specific SBID resources can be checked, else all SBID resources are checked for writeback complete status.	Syntax: SYNC_UNARY
1110b	Wait on Barrier	[] Blocks until the notification count reaches 0. The wait instruction evaluates the value of the notification count register nreg. If nreg is zero, thread execution is suspended and the thread is put in 'wait_for_notification' state. If nreg is not zero (i.e., one or more notifications have been received), nreg is decremented by one and the thread continues executing on the next instruction. If a thread is in the 'wait_for_notification' state, when a notification arrives, the notification count register is incremented by one. As the notification count register becomes nonzero, the thread wakes up to continue execution and at the same time the notification register is decremented by one. If only one notification arrived, the notification register value becomes zero. However, during the above mentioned time period, it is possible that more notifications may arrive, making the notification register nonzero again. This operation implicitly accesses n0 (typically n0.0 for barriers).	Syntax: SYNC_UNARY
1111b	Wait on Host Notification	Similar to .bar, but waits the host's notification register (typically n0.1). See that element for more information.	Syntax: SYNC_UNARY

TernaryDataType

TernaryDataType	
Source:	EuIsa
Size (in bits):	3
This field provide the datatype of the source and destination operands for ternary instruction.	

Value	Name	Description	Exists If
000b	:ub	Unsigned Byte (8-bit) Integer	(Structure[EU_INSTRUCTION_BASIC_THREE_SRC][ExecDataType]==Integer)
001b	:uw	Unsigned Word (16-bit) Integer	(Structure[EU_INSTRUCTION_BASIC_THREE_SRC][ExecDataType]==Integer)
010b	:ud	Unsigned DoubleWord (32-bit) Integer	(Structure[EU_INSTRUCTION_BASIC_THREE_SRC][ExecDataType]==Integer)
011b	:uq	Unsigned Quadword (64-bit) Integer	(Structure[EU_INSTRUCTION_BASIC_THREE_SRC][ExecDataType]==Integer)
100b	:b	Signed Byte (8-bit) Integer	(Structure[EU_INSTRUCTION_BASIC_THREE_SRC][ExecDataType]==Integer)
101b	:w	Signed Word (16-bit) Integer	(Structure[EU_INSTRUCTION_BASIC_THREE_SRC][ExecDataType]==Integer)
110b	:d	Signed DoubleWord (32-bit) Integer	(Structure[EU_INSTRUCTION_BASIC_THREE_SRC][ExecDataType]==Integer)
111b	:q	Signed QuadWord (64-bit) Integer	(Structure[EU_INSTRUCTION_BASIC_THREE_SRC][ExecDataType]==Integer)
000b	Reserved		(Structure[EU_INSTRUCTION_BASIC_THREE_SRC][ExecDataType]==Float)
001b	:hf	Half (16-bit) Float	(Structure[EU_INSTRUCTION_BASIC_THREE_SRC][ExecDataType]==Float)
010b	:f	Single-Precision (32-bit) Float	(Structure[EU_INSTRUCTION_BASIC_THREE_SRC][ExecDataType]==Float)
011b	:df	Double-Precision (64-bit) Float	(Structure[EU_INSTRUCTION_BASIC_THREE_SRC][ExecDataType]==Float)
100b	Reserved		(Structure[EU_INSTRUCTION_BASIC_THREE_SRC][ExecDataType]==Float)
101b	Reserved		(Structure[EU_INSTRUCTION_BASIC_THREE_SRC][ExecDataType]==Float)
110b	Reserved		(Structure[EU_INSTRUCTION_BASIC_THREE_SRC][ExecDataType]==Float)
111b	Reserved		(Structure[EU_INSTRUCTION_BASIC_THREE_SRC][ExecDataType]==Float)

TernaryVertStride

TernaryVertStride	
Source:	EuIsa
Size (in bits):	2
Source Vertical Stride is required for regioning/accessing datatypes of varied size. It is one way to obtain a vector of scalars.	

Value	Name
00b	0 - elements
01b	1 - elements
10b	4 - elements
11b	8 - elements

Texture Coordinate Mode

Texture Coordinate Mode		
Size (in bits): 3		
Value	**Name**	**Description**
0h	WRAP	Map is repeated in the U direction
1h	MIRROR	Map is mirrored in the U direction
2h	CLAMP	Map is clamped to the edges of the accessed map
3h	CUBE	For cube-mapping, filtering in edges access adjacent map faces
4h	CLAMP_BORDER	Map is infinitely extended with the border color
5h	MIRROR_ONCE	Map is mirrored once about origin, then clamped
6h	HALF_BORDER	Map is infinitely extended with the average of the nearest edge texel and the border color
7h	MIRROR_101	Map is mirrored one time in each direction, but the first pixel of the reflected image is skipped, and the reflected image is effectively 1 pixel less in that direction. May only be used on 2D surfaces.

VertStride

VertStride	
Source:	EuIsa
Size (in bits):	4

Description
Vertical Stride. The field provides the vertical stride of the register region in unit of data elements for an operand. Encoding of this field provides values of 0 or powers of 2, ranging from 1 to 32 elements. Larger values are not supported due to the restriction that a source operand must reside within two adjacent 256-bit registers (64 bytes total). Special encoding 1111b (0xF) is only valid when the operand is in register-indirect addressing mode (AddrMode = 1). If this field is set to 0xF, one or more sub-registers of the address registers may be used to compute the addresses. Each address sub-register provides the origin for a row of data element. The number of address sub-registers used is determined by the division of ExecSize of the instruction by the Width fields of the operand. This field only applies to source operand. It does not apply to destination. This field is not present for an immediate source operand.

Programming Notes
Note 1: If indirect address is supported for src1, encoding 0xF is reserved for src1 - only single-index indirect addressing is supported.
Note 2: Encoding 0010 applies for QWord-size operands.

Value	Name	Programming Notes
0000b	0 elements	
0001b	1 element	Align1 mode only.
0010b	2 elements	
0011b	4 elements	
0100b	8 elements	Align1 mode only.
0101b	16 elements	Applies to byte or word operand only. Align1 mode only.
0110b	32 elements	Applies to byte operand only. Align1 mode only.
0111b-1110b	Reserved	
1111b	VxH or Vx1 mode	Only valid for register-indirect addressing in Align1 mode.

Width

Width	
Source:	EuIsa
Size (in bits):	3

This field specifies the number of elements in the horizontal dimension of the region for a source operand. This field cannot exceed the ExecSize field of the instruction. This field only applies to source operand. It does not apply to destination. This field is not present for an immediate source operand.

Programming Notes
Note that with ExecSize of 32, because the maximum Width is 16, there are at least two rows in a source region.

Value	Name
000b	1 elements
001b	2 elements
010b	4 elements
011b	8 elements
100b	16 elements
101b-111b	Reserved

WRAP_SHORTEST_ENABLE

WRAP_SHORTEST_ENABLE	
Source:	RenderCS
Size (in bits):	4

This state selects which components (if any) of Attribute [n] are to be interpolated in a "wrap shortest" fashion. Operation is UNDEFINED if any of these bits are set and the Constant Interpolation Enable bit associated with this attribute is set. Note that wrap-shortest interpolation is only supported for Attributes 0-15.

Value	Name	Description
0001b	X	Wrap Shortest X Component
0010b	Y	Wrap Shortest Y Component
0011b	XY	Wrap Shortest XY Components
0100b	Z	Wrap Shortest Z Component
0101b	XZ	Wrap Shortest XZ Components
0110b	YZ	Wrap Shortest YZ Components
0111b	XYZ	Wrap Shortest XYZ Components
1000b	W	Wrap Shortest W Component
1001b	XW	Wrap Shortest XW Components
1010b	YW	Wrap Shortest YW Components
1011b	XYW	Wrap Shortest XYW Components
1100b	ZW	Wrap Shortest ZW Components
1101b	XZW	Wrap Shortest XZW Components
1110b	YZW	Wrap Shortest YZW Components
1111b	XYZW	Wrap Shortest XYZW Components

Intel® Iris® Xe and UHD Graphics Open Source

Programmer's Reference Manual

For the 2020-2021 11th Generation Intel Xeon®, Core™, Celeron®, Pentium® Gold Processors based on the "Tiger Lake" Platform

Volume 2c: Command Reference: Registers
Part 1 – Registers A through L

December 2021, Revision 1.0

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Table of Contents

Active Doorbell Register 0 1
Active Doorbell Register 1 5
Active Doorbell Register 2 10
Active Doorbell Register 3 15
Active Doorbell Register 4 20
Active Doorbell Register 5 25
Active Doorbell Register 6 30
Active Doorbell Register 7 35
Active Head Pointer Register 40
Advanced Scheduler Reset Request Messages 42
Aggregate Perf Counter SPM0 Lower DWord Free 45
Aggregate Perf Counter SPM0 Upper DWord Free 46
Aggregate Perf Counter SPM1 Lower DWord Free 47
Aggregate Perf Counter SPM1 Upper DWord Free 48
ARAT C6 Disallow Threshold 49
ARAT Delta (LSB) 50
ARB_CTL 51
ARB_CTL2 54
ASL Storage 57
ATS Capability 58
ATS Control 59
ATS Extended Capability Header 60
AUD_CONFIG 61
AUD_CONFIG_2 63
AUD_CONFIG_BE 65
AUD_DIP_ELD_CTRL_ST 70
AUD_EDID_DATA 73
AUD_FREQ_CNTRL 75
AUD_INFOFR 77
AUD_M_CTS_ENABLE 78
AUD_MISC_CTRL 80
AUD_PIN_ELD_CP_VLD 82

AUD_PIN_PIPE_CONN_ENTRY_LNGTH....86
AUD_PIPE_CONN_SEL_CTRL....87
AUD_PWRST....88
AUD_RID....91
AUD_VID_DID....92
AUD_WD_CNTRL....93
AUD_WD_DMA_HDPTR....96
AUD_WD_DMA_TAILPTR....97
AUD_WD_EDID_DATA....98
AUD_WNIC_2_AUD_RESET....99
AUD_WNIC_PCR_LOWADDROFFSET....100
AUD_WNIC_PCR_UPADDROFFSET....101
AUD_WNIC_TAILPTR_ADDROFFSET....102
AUD_WNIC_TAILPTR_UPADDROFFSET....103
AUDIO_PIN_BUF_CTL....104
Audio Codec Interrupt Definition....105
Auto Draw End Offset....107
AVP av1 status....108
AVP av1 unit done....110
Base of DMA Protected Range....112
Batch Address Difference Register....113
Batch Buffer Head Pointer Preemption Register....115
Batch Buffer Head Pointer Register....118
Batch Buffer Per Context Pointer....121
Batch Buffer Stack Write Port....125
Batch Buffer Start Head Pointer Register....127
Batch Buffer Start Upper Head Pointer Register....129
Batch Buffer State Register....131
Batch Buffer Upper Head Pointer Preemption Register....134
Batch Buffer Upper Head Pointer Register....136
Batch Offset Register....139
BCS Context Sizes....142
BCS CSB....143
BCS CSB Fifo Status Register....144

BCS Ring Buffer Next Context ID Register 145
BCS SW Control 146
BIOS2DRIVER Scratch0 148
BIOS2DRIVER Scratch1 152
BIOS2DRIVER Scratch2 155
BIOS2DRIVER Scratch3 156
BIOS2DRIVER Scratch4 157
BIOS2DRIVER Scratch5 158
BIOS2DRIVER Scratch6 159
BIOS2DRIVER Scratch7 160
Bitstream Output Bit Count for the last Syntax Element Report Register 161
Bitstream Output Byte Count Per Slice Report Register 162
Bitstream Output Minimal Size Padding Count Report Register 163
BLC_PWM_CTL 164
BLC_PWM_DATA 166
Blitter Cache Control Register 167
Boot Hash Check Status 168
BOOT VECTOR 169
Built In Self Test 170
BW_BUDDY_CTL 171
BW_BUDDY_PAGE_MASK 173
C6 Entry latency 174
Cache Line Size 175
Cache Mode Register 0 176
Cache Mode Register 1 179
Cache Mode Subslice Register 183
Capabilities A 185
Capabilities B 187
Capabilities Control 188
Capabilities Pointer 189
Capability Identifier 190
CAPTURE_0_DSSM0 191
CAPTURE_0_DSSM1 192
CAPTURE_0_L3 193

CAPTURE_1_DSSM0 194
CAPTURE_1_DSSM1 195
CAPTURE_1_L3 196
CAPTURE_2_DSSM0 197
CAPTURE_2_DSSM1 198
CAPTURE_2_L3 199
CAPTURE_3_DSSM0 200
CAPTURE_3_DSSM1 201
CAPTURE_3_L3 202
CAPTURE_4_DSSM0 203
CAPTURE_4_DSSM1 204
CAPTURE_4_L3 205
CAPTURE_5_DSSM0 206
CAPTURE_5_DSSM1 207
CAPTURE_5_L3 208
CAPTURE_6_DSSM0 209
CAPTURE_6_DSSM1 210
CAPTURE_6_L3 211
CAPTURE_7_DSSM0 212
CAPTURE_7_DSSM1 213
CAPTURE_7_L3 214
Catastrophic Event FIFO Status 215
Catastrophic Interrupt Context ID 217
CCS CSB 218
CCS CSB Fifo Status Register 219
CDCLK_CTL 220
CDCLK_PLL_ENABLE 223
CGE_CTRL 225
CGE_WEIGHT 227
Clipper Invocation Counter 231
Clipper Primitives Counter 232
Command Buffer Caching Control Register 233
Comp Ctx TLB Invalidation Register 236
Compute Engine DSS Reservation 239

Config to MCI HI 240
Config to MCI LO 241
Config to MCI STATUS1 242
Configuration Register0 for RPMunit 243
Configuration Register1 for RPMunit 245
CONTEXT_SCHEDULING_ATTRIBUTES....246
Context Info of Per-process GTT Space HDW 248
Context Info of Per-process GTT Space LDW 249
Context Restore Request To TDL 251
Context Sizes 252
Context Status Buffer1 Contents 253
Context Status Buffer Contents 256
Context Status Buffer Interrupt Mask Register....259
Context Status Buffer Read Register 263
Context Timestamp Count....265
Control Register for Power Management....267
Count Active Channels Dispatched....272
Count of C6 Latency maximum....273
Count of Media License latency 274
Count of Media License latency Maximum....275
Count of Media License Request....276
Count of Media Sampler License latency....277
Count of Media Sampler License latency Maximum 278
Count of Media Sampler License Request 279
Count of Render License latency....280
Count of Render License latency Maximum 281
Count of Render License Request 282
Count of spec License Request....283
Count of Speculative License latency 284
Count of Speculative License latency Maximum....285
CPS Invocation Counter 286
CSC_COEFF 287
CSC_MODE 289
CSC_POSTOFF 291

CSC_PREOFF 293
CS CSB 295
CS CSB _FSR 296
CSFE FSM2 297
CS Indirect Base 300
CS-LTISEQ Flush Message Register 301
CSPREEMPT 302
CUR_BASE 303
CUR_COLOR_CTL 305
CUR_CSC_COEFF 307
CUR_CTL 309
CUR_FBC_CTL 314
CUR_PAL 316
CUR_POS 319
CUR_PRE_CSC_GAMC_DATA 321
CUR_PRE_CSC_GAMC_INDEX 323
CUR_SURFLIVE 325
Customizable Event Creation 5-0 326
DATAM 328
DATAN 330
DBUF_CTL 331
DBUF_CTL2 333
DBUF_ECC_STAT 334
DBUF_STATUS 336
DC_STATE_EN 338
DCPR_STATUS_1 341
DDI_AUX_CTL 342
DDI_AUX_DATA 346
DDI_BUF_CTL 352
DDI_CLK_SEL 356
DE_FUSA_IOSF_PARITY_CNTRL 358
DE_PIPE_INTERRUPT 361
DE_POWER1 365
DE_POWER2 367

DE_RR_DEST 368
DE_RRMR 370
DE_RRMR_DW1 375
DE_RRMR_DW2 379
Decouple Register 0 DW0 383
Decouple Register 0 DW1 384
Decouple Register 1 DW0 385
Decouple Register 1 DW1 386
Decouple Register 2 DW0 387
Decouple Register 2 DW1 388
Decouple Register 3 DW0 389
Decouple Register 3 DW1 390
Decouple Register 4 DW0 391
Decouple Register 4 DW1 392
Decouple Register 5 DW0 393
Decouple Register 5 DW1 394
Decouple Register 6 DW0 395
Decouple Register 6 DW1 396
Decouple Register 7 DW0 397
Decouple Register 7 DW1 398
Decouple Register 8 DW0 399
Decouple Register 8 DW1 400
Decouple Register 9 DW0 401
Decouple Register 9 DW1 402
Decouple Register 10 DW0 403
Decouple Register 10 DW1 404
Decouple Register 11 DW0 405
Decouple Register 11 DW1 406
Decouple Register 12 DW0 407
Decouple Register 12 DW1 408
Decouple Register 13 DW0 409
Decouple Register 13 DW1 410
Decouple Register 14 DW0 411
Decouple Register 14 DW1 412

Decouple Register 15 DW0 413
Decouple Register 15 DW1 414
Dedicated Path Arbiter Credits 415
DE HPD Interrupt Definition 416
DE Misc Interrupt Definition 419
DE Port Interrupt Definition 421
Device 2 Control 423
Device Capabilities 424
Device Enable 427
Device Identification 430
DFSDONE 431
DFSM 432
DG_CLKREQ_POLICY 435
Discard counter status register 436
Discard Enables for Z streams 437
DISMBASE_LSB 438
DISMBASE_MSB 439
DISMLIMIT_LSB 440
DISMLIMIT_MSB 441
DISPLAY_INT_CTL 442
DKL_BIAS 444
DKL_CLKTOP2_CORECLKCTL1 445
DKL_CLKTOP2_HSCLKCTL 447
DKL_CMN_ANA_DWORD28 451
DKL_CMN_DIG_PLL_MISC 453
DKL_CMN_UC_DW27 454
DKL_DP_MODE 458
DKL_PLL_DIV0 460
DKL_PLL_DIV1 462
DKL_PLL_FRAC_LOCK 464
DKL_PLL_LF 466
DKL_PLL1_CNTR_XXXX_SETTINGS 468
DKL_PMD_FORCE_PCS_IF_TX1 470
DKL_REFCLKIN_CTL 472

DKL_SSC 473
DKL_TDC_COLDST_BIAS 475
DKL_TX_DPCNTL0 476
DKL_TX_DPCNTL1 477
DKL_TX_DPCNTL2 478
DKL_TX_DW17 479
DKL_TX_DW18 480
DKL_TX_FW_CALIB 481
DKL_TX_PMD_LANE_SUS_LN0 482
DKL_TX_PMD_LANE_SUS_LN1 483
DKL_XXX_TDC_CRO 484
DKLP_ACU_ACU_DWORD4 486
DKLP_ACU_ACU_DWORD8 488
DKLP_ACU_ICL_INDEXED_ACU_INDEXED_DWORD46 492
DKLP_CMN_ANA_CMN_ANA_DWORD0 493
DKLP_CMN_ANA_CMN_ANA_DWORD1 496
DKLP_CMN_ANA_CMN_ANA_DWORD5 498
DKLP_CMN_ANA_CMN_ANA_DWORD6 501
DKLP_CMN_ANA_CMN_ANA_DWORD7 504
DKLP_CMN_ANA_CMN_ANA_DWORD27 506
DKLP_CMN_DIG_CMN_DIG_DWORD6 508
DKLP_CMN_DIG_CMN_DIG_DWORD29 512
DKLP_CMN_DIG_CMN_DIG_DWORD33 514
DKLP_CMN_UC_CMN_UC_DWORD27 516
DKLP_CMN_UC_CMN_UC_DWORD30 521
DKLP_CMN2_DIG_CMN_DIG_DWORD0 522
DKLP_PCS_GLUE_PMD_IRQ_VEC_EN 525
DKLP_PCS_GLUE_RTT_CR_SPARE 526
DKLP_PCS_GLUE_TX_DPCNTL0 527
DKLP_PCS_GLUE_TX_DPCNTL1 529
DKLP_PCS_GLUE_TX_DPCNTL2 531
DKLP_PCS_GLUE_TX1_FW_CALIB 533
DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD0 535
DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD1 538

DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD7 542
DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD8 544
DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD9 545
DKLP_PCS_PCS_DWORD3 546
DKLP_PCS_PCS_DWORD23 547
DKLP_PCS_PCS_DWORD25 549
DKLP_PLL0_BIAS 550
DKLP_PLL0_DIV0 551
DKLP_PLL0_DIV1 553
DKLP_PLL0_FRAC_LOCK 555
DKLP_PLL0_LF 557
DKLP_PLL0_TDC_COLDST_BIAS 559
DKLP_PLL1_BIAS 560
DKLP_PLL1_DIV0 561
DKLP_PLL1_DIV1 563
DKLP_PLL1_FRAC_LOCK 565
DKLP_PLL1_LF 567
DKLP_PLL1_TDC_COLDST_BIAS 569
DKLP_PMD_LANE_ANA_CSR_DIG_LFPS_CAL 570
DKLP_PMD_LANE_CDR_CDR_SAVE_RESTORE1 572
DKLP_PMD_LANE_SUSWELL_LFPS_SPARE 573
DKLP_PMD_LANE_SUSWELL_TX1_RCV_DETECT_CTRL 575
DKLP_TX_GLUE_TX_DWORD14 576
DKLP_TX2_PMD_LANE_MISC_TX1_SPARE 577
DMA Address Register 0 High 578
DMA Address Register 0 Low 579
DMA Address Register 1 High 580
DMA Address Register 1 Low 581
DMA Configuration 582
DMA Control 583
DMA Copy Size 585
DMA Global MicroController WOPCM Offset 586
DMA Microkernel Base 587
DMA Microkernel Top 588

DMA Protected Range....589
Doorbell Control....591
Doorbell Cookie Register 0..255....594
Doorbell IDI Register....595
Doorbell Lower Address Register 0..255....596
Doorbell Upper Address Register 0..255....598
DOUBLE_BUFFER_CTL....599
DP_TP_CTL....600
DP_TP_STATUS....604
DPCLKA_CFGCR0....608
DPFC_CONTROL_SA....611
DPFC_CPU_FENCE_OFFSET....612
DPHY_CLK_TIMING_PARAM....613
DPHY_DATA_TIMING_PARAM....616
DPHY_ESC_CLK_DIV....619
DPHY_TA_TIMING_PARAM....621
DPHY_TRIG_EXT....624
DPLC_CTL....625
DPLC_FA_IIR....629
DPLC_FA_IMR....632
DPLC_FA_STATUS....635
DPLC_FA_SURF....637
DPLC_FA_SURF_LIVE....638
DPLC_HIST_DATA....639
DPLC_HIST_INDEX....640
DPLC_IE_DATA....642
DPLC_IE_INDEX....643
DPLC_PART_CTL....645
DPLC_PTRCFG....646
DPLC_PTRCFG_LIVE....647
DPLC_RDLENGTH....648
DPLC_RDLENGTH_LIVE....649
DPLL_CFGCR0....650
DPLL_CFGCR1....651

DPLL_DIV0 653
DPLL_ENABLE 655
DPLL_SSC 658
DPST_BIN 660
DPST_CTL 662
DPST_GUARD 665
Driver Media Force Wake Ack 667
DRIVER MEDIA FORCE WAKE REQ 668
Driver Render Force Wake Ack 671
Driver VDBox0 Force Wake Ack 672
Driver VDBox1 Force Wake Ack 673
Driver VDBox2 Force Wake Ack 674
Driver VDBox3 Force Wake Ack 675
Driver VDBox4 Force Wake Ack 676
Driver VDBox5 Force Wake Ack 677
Driver VDBox6 Force Wake Ack 678
Driver VDBox7 Force Wake Ack 679
Driver VEBox0 Force Wake Ack 680
Driver VEBox1 Force Wake Ack 681
Driver Vebox2 Force Wake Ack 682
Driver Vebox3 Force Wake Ack 683
DSB_BUFRPT_CNT 684
DSB_CTRL 686
DSB_CURRENT_HEAD_PTR 690
DSB_HEAD_PTR 692
DSB_INTERRUPT 694
DSB_MMIOCTRL 698
DSB_PF_LN_LOWER 701
DSB_PF_LN_UPPER 703
DSB_PMCTRL_2 705
DSB_PMCTRL 708
DSB_POLLFUNC 710
DSB_POLLMASK 713
DSB_RM_TIMEOUT 715

DSB_RMTIMEOUTREG_CAPTURE 718
DSB_STATUS 720
DSB_TAIL_PTR 725
DSC_CRC_CTL 727
DSC_CRC_RES 729
DSC_PICTURE_PARAMETER_SET_0 731
DSC_PICTURE_PARAMETER_SET_1 734
DSC_PICTURE_PARAMETER_SET_2 736
DSC_PICTURE_PARAMETER_SET_3 738
DSC_PICTURE_PARAMETER_SET_4 740
DSC_PICTURE_PARAMETER_SET_5 742
DSC_PICTURE_PARAMETER_SET_6 744
DSC_PICTURE_PARAMETER_SET_7 746
DSC_PICTURE_PARAMETER_SET_8 748
DSC_PICTURE_PARAMETER_SET_9 750
DSC_PICTURE_PARAMETER_SET_10 752
DSC_PICTURE_PARAMETER_SET_11 754
DSC_PICTURE_PARAMETER_SET_12 756
DSC_PICTURE_PARAMETER_SET_13 758
DSC_PICTURE_PARAMETER_SET_14 760
DSC_PICTURE_PARAMETER_SET_15 762
DSC_PICTURE_PARAMETER_SET_16 764
DSC_RC_BUF_THRESH_0 766
DSC_RC_BUF_THRESH_1 768
DSC_RC_RANGE_PARAMETERS_0 770
DSC_RC_RANGE_PARAMETERS_1 772
DSC_RC_RANGE_PARAMETERS_2 774
DSC_RC_RANGE_PARAMETERS_3 776
DSI_CALIB_TO 778
DSI_CLK_TIMING_PARAM 780
DSI_CMD_FRMCTL 783
DSI_CMD_RXCTL 787
DSI_CMD_RXHDR 790
DSI_CMD_RXPYLD 791

DSI_CMD_TXCTL....792
DSI_CMD_TXHDR....794
DSI_CMD_TXPYLD....796
DSI_DATA_TIMING_PARAM....797
DSI_ESC_CLK_DIV....800
DSI_HTX_TO....802
DSI_INTER_IDENT_REG....803
DSI_INTER_MSK_REG....806
DSI_IO_MODECTL....812
DSI_LP_MSG....814
DSI_LRX_H_TO....817
DSI_PWAIT_TO....818
DSI_T_INIT_PRIMARY....819
DSI_T_WAKEUP....820
DSI_TA_TO....821
DSI_TRIG_TX_TIME....822
DS Invocation Counter....823
DSMBASE....824
DSSM....825
Dummy Context Save Register....828
EDRAMCAP....829
EMRR Mask LSB....830
EMRR Mask MSB....831
Encoded row max QP vector....832
Error Identity Register....837
Error Mask Register....840
Error Status Register....843
EU_GRF_CLEAR....845
EUP1 BONUS2 Reg....846
EUP1 BONUS11 Reg....848
EUP2 BONUS1 Reg....850
EUP2 BONUS2 Reg....852
EUP 2 Power Down FSM control register with lock....854
EUP 2 Power on FSM control register with lock....857

EUP3 BONUS1 Reg....859
EUP3 BONUS2 Reg....861
EU PAIR 1 PFET control register with lock....863
EU PAIR 1 Power Context Save request....865
EU PAIR 1 Power Down FSM control register with lock....866
EU PAIR 1 Power Gate Control Request....869
EU PAIR 1 Power on FSM control register with lock....870
EU PAIR 2 PGFET control register with lock....872
EU PAIR 2 Power Context Save request....874
EU PAIR 2 Power Gate Control Request....875
EU PAIR 3 PGFET control register with lock....876
EU PAIR 3 Power Context Save request....878
EU PAIR 3 Power Down FSM control register with lock....879
EU PAIR 3 Power Gate Control Request....882
EU PAIR 3 Power on FSM control register with lock....883
Eviction counter status register....885
Exec-List Context Offset....886
Execlist Control Register....889
Execlist Status....892
Execlist Submission Queue Contents....896
Execlist Submit Port Register....899
Execute Condition Code Register....901
Execution Queue Element Mask....904
FAULT_TLB_RD_DATA1 Register....906
FBC_CFB_BASE....907
FBC_CTL....908
FBC_RT_BASE_ADDR_REGISTER....911
FBC_RT_BASE_ADDR_REGISTER_UPPER....913
FBC LLC Config Read Control Register....914
FENCE_LSB....915
FENCE_MSB....919
Fence Control Register....923
FF Performance....925
First MGSR read done....927

First VF Offset 929
FIX BONUS1 Reg 930
FIX BONUS2 Reg 932
FIX PGFET control register with lock 934
Fix Power Context Save request 936
FIX Power Down FSM control register with lock 937
Fix Power Gate Control Request 940
FIX Power on FSM control register with lock 941
Flexible EU Event Control 0 943
Flexible EU Event Control 1 945
Flexible EU Event Control 2 947
Flexible EU Event Control 3 949
Flexible EU Event Control 4 951
Flexible EU Event Control 5 953
Flexible EU Event Control 6 955
FLT_RPT0 957
FLT_RPT1 958
FLT_RPT2 959
FLT_RPT3 960
FORCE_TO_NONPRIV 961
FUSE_STATUS 990
GAC_GAM Arbitration Counters Register 0 992
GAC_GAM Arbitration Counters Register 1 993
GAC_GAM RO Arbitration Register 0 994
GAC_GAM RO Arbitration Register 1 995
GAC_GAM RO Arbitration Register 2 996
GAC_GAM RO Arbitration Register 3 997
GAF Config Register 0 998
GAF Config Register 1 1002
GAMMA_MODE 1004
Gang Timer Register 1006
Gated Clock Counter for DFR Testability 1008
General Purpose Register 1009
GFX_FLSH_CNT 1013

GFX_VTDBAR_LSB 1014
GFXBDF 1015
GGTT Pinned Range Base Register 1016
GGTT Pinned Range Limit Register 1017
Global MicroController Status 1018
Global System Interrupt Routine 1019
GMCH Graphics Control 1020
GPGPU Dispatch Dimension X 1023
GPGPU Dispatch Dimension Y 1024
GPGPU Dispatch Dimension Z 1025
GP Thread Time 1026
Graphics Device Reset Control 1027
Graphics Memory Range Address 1032
Graphics Mode Register 1034
Graphics Primary Interrupt 1039
Graphics System Event 1041
Graphics Translation Table Memory Mapped Range Address 1042
Graphics Virtual Primary Interrupt 1044
GSCPowerGoodDelay 1046
GS Invocation Counter 1047
GSI Power Good Delay 1048
GSMBASE 1049
GS Primitives Counter 1050
GT_FLUSH_BCLD_ACK 1051
GT_RELOAD_FLUSH 1052
GTACK 1053
GTC_CTL 1055
GTC_DDA_M 1056
GTC_DDA_N 1057
GTC_IIR 1058
GTC_IMR 1059
GTC_LIVE 1060
GTC_PORT_CTL 1061
GTC_PORT_MISC 1063

GTC_PORT_TX_CURR 1065
GTC_PORT_TX_PREV 1067
GT C6 Entry TSC LSB 1069
GT C6 Entry TSC MSB 1070
GT C6 Residency LSB 1071
GT C6 Residency MSB 1072
GT Correctable Err Status Register 1073
GTDRIVER_MAILBOX_DATA1 1074
GTDRIVER_MAILBOX_INTERFACE 1075
GTDRIVER_P2G_EVENTS 1076
GT Engine Interrupt Enable 1077
GT Engine Interrupt Mask 1079
GT Fatal Err Status Register 1081
GT Function Level Reset Control Message 1083
GTICP BONUS1 Reg 1084
GTICP BONUS2 Reg 1086
GT Interrupt DW0 1088
GT Interrupt DW1 1090
GT Interrupt Identity 1091
GT Interrupt IIR Selector 1092
GTI PGFET control register with lock 1093
GTI Power Gate Control Request 1096
GT Mode Register 1097
GT Non Fatal Err Status Register 1100
GTPO Triggering Block 1 Mask A 1101
GTPO Triggering Block 1 Mask B 1102
GTPO Triggering Block 1 Match A 1103
GTPO Triggering Block 1 Match B 1104
GTPO Triggering Block Mode Enable 1105
GTSCRATCH 1106
GTSP 1107
GT Virtual Function Engine Interrupt Enable 1108
GT Virtual Function Engine Interrupt Mask 1110
GT Virtual Function IIR Selector 1115

GT Virtual Function Interrupt DW0 1117
GT Virtual Function Interrupt DW1 1119
GT Virtual Function Interrupt Identity 1121
GUC_HOST_INTR_IIR 1123
GuC DMA Interrupt Input 1124
GuC Doorbell Group 0 Interrupt Status 1125
GuC Doorbell Group 1 Interrupt Status 1126
GuC Doorbell Group 2 Interrupt Status 1127
GuC Doorbell Group 3 Interrupt Status 1128
GuC Doorbell Group 4 Interrupt Status 1129
GuC Doorbell Group 5 Interrupt Status 1130
GuC Doorbell Group 6 Interrupt Status 1131
GuC Doorbell Group 7 Interrupt Status 1132
GuC Engine Interrupt Mask 1133
Guc GT Interrupt DW1 1134
GuC Host Interrupt Interrupt Input 0 1135
GuC Host Interrupt Mask 0 1136
GuC Host Interupt Enable 0 1137
GuC Interupt IIR Selector 1138
Guc Peek Register 0 1139
Guc Peek Register 1 1145
Guc Peek Register 2 1151
Guc Peek Register 3 1157
Guc Peek Register 4 1163
Guc Peek Register 5 1169
Guc Peek Register 6 1175
Guc Peek Register 7 1181
GuC PM Time Stamp Counter 0 1187
GuC PM Time Stamp Counter 1 1188
GU Misc Interrupt Definition 1189
Hardware Status Mask Register 1193
Hardware Status Page Address Register 1195
HCP Bitstream Output Minimal Size Padding Count Report Register 1197
HCP CABAC Status 1198

HCP Decode Status 1201
HCP Frame BitStream BIN Count 1203
HCP Image Status Control 1204
HCP Image Status Mask 1206
HCP Last Position 1207
HCP Picture Checksum cIdx0 1209
HCP Picture Checksum cIdx1 1211
HCP Picture Checksum cIdx2 1213
HCP PMU Status 1215
HCP Power Context Save request 1217
HCP Power Down FSM control register with lock 1218
HCP Power Gate Control Request 1220
HCP Qp Status Count 1221
HCP Reported Bitstream Output Byte Count with header per Frame Register 1222
HCP Reported Bitstream Output Byte Count without header per Frame Register 1223
HCP Reported Bitstream Output CABAC Bin Count Register 1224
HCP SLICE COUNT 1225
HCP Unit Done 1226
Header Type 1229
Head pointer update 1230
HEVC Local APIC Retry Vector 1232
HIP_INDEX_REG0 1233
HIP_INDEX_REG1 1234
HOTPLUG_CTL 1235
HS Invocation Counter 1239
IA Vertices Count 1240
Idle Switch Delay 1241
Indirect Context Offset Pointer 1243
Indirect Context Pointer 1246
infcp Vdbox unit Level Clock Gating override during rstflow 1249
INF Power Context Save request 1250
INF unit Level Clock Gating Control 9560 1251
Instruction Parser Mode Register 1252
Interrupt Group 0 1256

Interrupt Group 1 ..... **1258**
Interrupt Line **1260**
Interrupt Pin **1261**
I/O Base Address **1262**
Jump Location **1263**
KVMR_SPR_COLOR_CTL **1264**
L3 Allocation Control Register **1265**
L3 Cache Runtime ECC capture Register **1267**
L3 Cache Runtime ECC Test Control Register **1268**
L3 Control Register **1269**
L3 Multi Context Reserved1 ..... **1270**
L3 Multi Context Reserved2 ..... **1271**
L3 Node Units Idle Status **1272**
L3 Parameter Information Register **1273**
L3 SQC register 4 ..... **1274**
L3 SQC register 5 ..... **1276**
L3 SQC register 6 ..... **1278**
L3 SQC registers 1 ..... **1279**
L3 Unslice IDLE Status Register **1283**
LBCF Render config save msg **1284**
LINKM **1285**
LINKN **1286**
LNCF MOCS Register 0 ..... **1287**
LNCF MOCS Register 1 ..... **1290**
LNCF MOCS Register 2 ..... **1293**
LNCF MOCS Register 3 ..... **1296**
LNCF MOCS Register 4 ..... **1299**
LNCF MOCS Register 5 ..... **1302**
LNCF MOCS Register 6 ..... **1305**
LNCF MOCS Register 7 ..... **1308**
LNCF MOCS Register 8 ..... **1311**
LNCF MOCS Register 9 ..... **1314**
LNCF MOCS Register 10 ..... **1317**
LNCF MOCS Register 11 ..... **1320**

LNCF MOCS Register 12 1323
LNCF MOCS Register 13 1326
LNCF MOCS Register 14 1329
LNCF MOCS Register 15 1332
LNCF MOCS Register 16 1335
LNCF MOCS Register 17 1338
LNCF MOCS Register 18 1341
LNCF MOCS Register 19 1344
LNCF MOCS Register 20 1347
LNCF MOCS Register 21 1350
LNCF MOCS Register 22 1353
LNCF MOCS Register 23 1356
LNCF MOCS Register 24 1359
LNCF MOCS Register 25 1362
LNCF MOCS Register 26 1365
LNCF MOCS Register 27 1368
LNCF MOCS Register 28 1371
LNCF MOCS Register 29 1374
LNCF MOCS Register 30 1377
LNCF MOCS Register 31 1380
Load Indirect Base Vertex 1383
Load Indirect Extended Parameter 0 1384
Load Indirect Extended Parameter 1 1385
Load Indirect Extended Parameter 2 1386
Load Indirect Instance Count 1387
Load Indirect Start Instance 1388
Load Indirect Start Vertex 1389
Load Indirect Vertex Count 1390
Lock register for Bank 1391
Lock register for LPFC 1395
Lock register for Node 1396
LPFC FUSA Register 1398
LSN Arbitration Priority Register 0 1400
LSN Arbitration Priority Register 1 1402

LSN Miscellaneous Configuration **1404**
LSN Slice Client Virtual Channel Assignment **1406**
LSQC FIFO Arbitration **1408**
LUT_3D_CTL **1413**
LUT_3D_DATA **1415**
LUT_3D_INDEX **1416**

Active Doorbell Register 0

DRB0ACT - Active Doorbell Register 0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	01900h	
DWord	**Bit**	**Description**
0	31	**Doorbell #(0*32+31) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+31) has been rung.
	30	**Doorbell #(0*32+30) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+30) has been rung.
	29	**Doorbell #(0*32+29) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+29) has been rung.
	28	**Doorbell #(0*32+28) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+28) has been rung.
	27	**Doorbell #(0*32+27) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+27) has been rung.
	26	**Doorbell #(0*32+26) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+26) has been rung.
	25	**Doorbell #(0*32+25) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+25) has been rung.
	24	**Doorbell #(0*32+24) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+24) has been rung.

DRB0ACT - Active Doorbell Register 0

23	**Doorbell #(0*32+23) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+23) has been rung.
22	**Doorbell #(0*32+22) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+22) has been rung.
21	**Doorbell #(0*32+21) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+21) has been rung.
20	**Doorbell #(0*32+20) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+20) has been rung.
19	**Doorbell #(0*32+19) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+19) has been rung.
18	**Doorbell #(0*32+18) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+18) has been rung.
17	**Doorbell #(0*32+17) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+17) has been rung.
16	**Doorbell #(0*32+16) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+16) has been rung.
15	**Doorbell #(0*32+15) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+15) has been rung.
14	**Doorbell #(0*32+14) Active** Access: RO

DRB0ACT - Active Doorbell Register 0

		_Custom_GTIReset: BUS Doorbell #(0*32+14) has been rung.
	13	**Doorbell #(0*32+13) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+13) has been rung.
	12	**Doorbell #(0*32+12) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+12) has been rung.
	11	**Doorbell #(0*32+11) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+11) has been rung.
	10	**Doorbell #(0*32+10) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+10) has been rung.
	9	**Doorbell #(0*32+9) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+9) has been rung.
	8	**Doorbell #(0*32+8) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+8) has been rung.
	7	**Doorbell #(0*32+7) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+7) has been rung.
	6	**Doorbell #(0*32+6) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+6) has been rung.
	5	**Doorbell #(0*32+5) Active** Access: RO _Custom_GTIReset: BUS

DRB0ACT - Active Doorbell Register 0

		Doorbell #(0*32+5) has been rung.
	4	**Doorbell #(0*32+4) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+4) has been rung.
	3	**Doorbell #(0*32+3) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+3) has been rung.
	2	**Doorbell #(0*32+2) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+2) has been rung.
	1	**Doorbell #(0*32+1) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+1) has been rung.
	0	**Doorbell #(0*32+0) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+0) has been rung.

Active Doorbell Register 1

DRB1ACT - Active Doorbell Register 1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	01904h	
DWord	**Bit**	**Description**
0	31	**Doorbell #(1*32+31) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+31) has been rung.
	30	**Doorbell #(1*32+30) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+30) has been rung.
	29	**Doorbell #(1*32+29) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+29) has been rung.
	28	**Doorbell #(1*32+28) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+28) has been rung.
	27	**Doorbell #(1*32+27) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+27) has been rung.
	26	**Doorbell #(1*32+26) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+26) has been rung.

DRB1ACT - Active Doorbell Register 1

	25	**Doorbell #(1*32+25) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+25) has been rung.
	24	**Doorbell #(1*32+24) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+24) has been rung.
	23	**Doorbell #(1*32+23) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+23) has been rung.
	22	**Doorbell #(1*32+22) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+22) has been rung.
	21	**Doorbell #(1*32+21) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+21) has been rung.
	20	**Doorbell #(1*32+20) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+20) has been rung.
	19	**Doorbell #(1*32+19) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+19) has been rung.
	18	**Doorbell #(1*32+18) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+18) has been rung.

DRB1ACT - Active Doorbell Register 1

17	**Doorbell #(1*32+17) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+17) has been rung.
16	**Doorbell #(1*32+16) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+16) has been rung.
15	**Doorbell #(1*32+15) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+15) has been rung.
14	**Doorbell #(1*32+14) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+14) has been rung.
13	**Doorbell #(1*32+13) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+13) has been rung.
12	**Doorbell #(1*32+12) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+12) has been rung.
11	**Doorbell #(1*32+11) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+11) has been rung.
10	**Doorbell #(1*32+10) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+10) has been rung.

DRB1ACT - Active Doorbell Register 1

9	**Doorbell #(1*32+9) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+9) has been rung.
8	**Doorbell #(1*32+8) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+8) has been rung.
7	**Doorbell #(1*32+7) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+7) has been rung.
6	**Doorbell #(1*32+6) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+6) has been rung.
5	**Doorbell #(1*32+5) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+5) has been rung.
4	**Doorbell #(1*32+4) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+4) has been rung.
3	**Doorbell #(1*32+3) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+3) has been rung.
2	**Doorbell #(1*32+2) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+2) has been rung.

DRB1ACT - Active Doorbell Register 1

	1	**Doorbell #(1*32+1) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+1) has been rung.
	0	**Doorbell #(1*32+0) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+0) has been rung.

Active Doorbell Register 2

DRB2ACT - Active Doorbell Register 2		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	01908h	
DWord	**Bit**	**Description**
0	31	**Doorbell #(2*32+31) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+31) has been rung.
	30	**Doorbell #(2*32+30) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+30) has been rung.
	29	**Doorbell #(2*32+29) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+29) has been rung.
	28	**Doorbell #(2*32+28) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+28) has been rung.
	27	**Doorbell #(2*32+27) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+27) has been rung.
	26	**Doorbell #(2*32+26) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+26) has been rung.

DRB2ACT - Active Doorbell Register 2

25	**Doorbell #(2*32+25) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(2*32+25) has been rung.	
24	**Doorbell #(2*32+24) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(2*32+24) has been rung.	
23	**Doorbell #(2*32+23) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(2*32+23) has been rung.	
22	**Doorbell #(2*32+22) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(2*32+22) has been rung.	
21	**Doorbell #(2*32+21) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(2*32+21) has been rung.	
20	**Doorbell #(2*32+20) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(2*32+20) has been rung.	
19	**Doorbell #(2*32+19) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(2*32+19) has been rung.	
18	**Doorbell #(2*32+18) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(2*32+18) has been rung.	

DRB2ACT - Active Doorbell Register 2

	17	**Doorbell #(2*32+17) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+17) has been rung.
	16	**Doorbell #(2*32+16) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+16) has been rung.
	15	**Doorbell #(2*32+15) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+15) has been rung.
	14	**Doorbell #(2*32+14) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+14) has been rung.
	13	**Doorbell #(2*32+13) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+13) has been rung.
	12	**Doorbell #(2*32+12) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+12) has been rung.
	11	**Doorbell #(2*32+11) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+11) has been rung.
	10	**Doorbell #(2*32+10) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+10) has been rung.

DRB2ACT - Active Doorbell Register 2

9	**Doorbell #(2*32+9) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(2*32+9) has been rung.	
8	**Doorbell #(2*32+8) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(2*32+8) has been rung.	
7	**Doorbell #(2*32+7) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(2*32+7) has been rung.	
6	**Doorbell #(2*32+6) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(2*32+6) has been rung.	
5	**Doorbell #(2*32+5) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(2*32+5) has been rung.	
4	**Doorbell #(2*32+4) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(2*32+4) has been rung.	
3	**Doorbell #(2*32+3) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(2*32+3) has been rung.	
2	**Doorbell #(2*32+2) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(2*32+2) has been rung.	

DRB2ACT - Active Doorbell Register 2

	1	**Doorbell #(2*32+1) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+1) has been rung.
	0	**Doorbell #(2*32+0) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+0) has been rung.

Active Doorbell Register 3

DRB3ACT - Active Doorbell Register 3		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	0190Ch	
DWord	**Bit**	**Description**
0	31	**Doorbell #(3*32+31) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+31) has been rung.
	30	**Doorbell #(3*32+30) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+30) has been rung.
	29	**Doorbell #(3*32+29) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+29) has been rung.
	28	**Doorbell #(3*32+28) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+28) has been rung.
	27	**Doorbell #(3*32+27) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+27) has been rung.
	26	**Doorbell #(3*32+26) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+26) has been rung.

DRB3ACT - Active Doorbell Register 3

25	**Doorbell #(3*32+25) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+25) has been rung.
24	**Doorbell #(3*32+24) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+24) has been rung.
23	**Doorbell #(3*32+23) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+23) has been rung.
22	**Doorbell #(3*32+22) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+22) has been rung.
21	**Doorbell #(3*32+21) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+21) has been rung.
20	**Doorbell #(3*32+20) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+20) has been rung.
19	**Doorbell #(3*32+19) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+19) has been rung.
18	**Doorbell #(3*32+18) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+18) has been rung.

DRB3ACT - Active Doorbell Register 3

17	**Doorbell #(3*32+17) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+17) has been rung.
16	**Doorbell #(3*32+16) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+16) has been rung.
15	**Doorbell #(3*32+15) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+15) has been rung.
14	**Doorbell #(3*32+14) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+14) has been rung.
13	**Doorbell #(3*32+13) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+13) has been rung.
12	**Doorbell #(3*32+12) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+12) has been rung.
11	**Doorbell #(3*32+11) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+11) has been rung.
10	**Doorbell #(3*32+10) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+10) has been rung.

DRB3ACT - Active Doorbell Register 3

9	**Doorbell #(3*32+9) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(3*32+9) has been rung.	
8	**Doorbell #(3*32+8) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(3*32+8) has been rung.	
7	**Doorbell #(3*32+7) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(3*32+7) has been rung.	
6	**Doorbell #(3*32+6) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(3*32+6) has been rung.	
5	**Doorbell #(3*32+5) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(3*32+5) has been rung.	
4	**Doorbell #(3*32+4) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(3*32+4) has been rung.	
3	**Doorbell #(3*32+3) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(3*32+3) has been rung.	
2	**Doorbell #(3*32+2) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(3*32+2) has been rung.	

DRB3ACT - Active Doorbell Register 3

	1	**Doorbell #(3*32+1) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+1) has been rung.
	0	**Doorbell #(3*32+0) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+0) has been rung.

Active Doorbell Register 4

DRB4ACT - Active Doorbell Register 4		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	01910h	
DWord	**Bit**	**Description**
0	31	**Doorbell #(4*32+31) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+31) has been rung.
	30	**Doorbell #(4*32+30) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+30) has been rung.
	29	**Doorbell #(4*32+29) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+29) has been rung.
	28	**Doorbell #(4*32+28) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+28) has been rung.
	27	**Doorbell #(4*32+27) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+27) has been rung.
	26	**Doorbell #(4*32+26) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+26) has been rung.

DRB4ACT - Active Doorbell Register 4

25	**Doorbell #(4*32+25) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(4*32+25) has been rung.	
24	**Doorbell #(4*32+24) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(4*32+24) has been rung.	
23	**Doorbell #(4*32+23) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(4*32+23) has been rung.	
22	**Doorbell #(4*32+22) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(4*32+22) has been rung.	
21	**Doorbell #(4*32+21) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(4*32+21) has been rung.	
20	**Doorbell #(4*32+20) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(4*32+20) has been rung.	
19	**Doorbell #(4*32+19) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(4*32+19) has been rung.	
18	**Doorbell #(4*32+18) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(4*32+18) has been rung.	

DRB4ACT - Active Doorbell Register 4

17	**Doorbell #(4*32+17) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(4*32+17) has been rung.	
16	**Doorbell #(4*32+16) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(4*32+16) has been rung.	
15	**Doorbell #(4*32+15) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(4*32+15) has been rung.	
14	**Doorbell #(4*32+14) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(4*32+14) has been rung.	
13	**Doorbell #(4*32+13) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(4*32+13) has been rung.	
12	**Doorbell #(4*32+12) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(4*32+12) has been rung.	
11	**Doorbell #(4*32+11) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(4*32+11) has been rung.	
10	**Doorbell #(4*32+10) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(4*32+10) has been rung.	

DRB4ACT - Active Doorbell Register 4

	9	**Doorbell #(4*32+9) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+9) has been rung.
	8	**Doorbell #(4*32+8) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+8) has been rung.
	7	**Doorbell #(4*32+7) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+7) has been rung.
	6	**Doorbell #(4*32+6) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+6) has been rung.
	5	**Doorbell #(4*32+5) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+5) has been rung.
	4	**Doorbell #(4*32+4) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+4) has been rung.
	3	**Doorbell #(4*32+3) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+3) has been rung.
	2	**Doorbell #(4*32+2) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+2) has been rung.

DRB4ACT - Active Doorbell Register 4

	1	**Doorbell #(4*32+1) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+1) has been rung.
	0	**Doorbell #(4*32+0) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+0) has been rung.

Active Doorbell Register 5

DRB5ACT - Active Doorbell Register 5		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	01914h	
DWord	**Bit**	**Description**
0	31	**Doorbell #(5*32+31) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+31) has been rung.
	30	**Doorbell #(5*32+30) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+30) has been rung.
	29	**Doorbell #(5*32+29) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+29) has been rung.
	28	**Doorbell #(5*32+28) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+28) has been rung.
	27	**Doorbell #(5*32+27) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+27) has been rung.
	26	**Doorbell #(5*32+26) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+26) has been rung.

DRB5ACT - Active Doorbell Register 5

25	**Doorbell #(5*32+25) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(5*32+25) has been rung.	
24	**Doorbell #(5*32+24) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(5*32+24) has been rung.	
23	**Doorbell #(5*32+23) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(5*32+23) has been rung.	
22	**Doorbell #(5*32+22) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(5*32+22) has been rung.	
21	**Doorbell #(5*32+21) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(5*32+21) has been rung.	
20	**Doorbell #(5*32+20) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(5*32+20) has been rung.	
19	**Doorbell #(5*32+19) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(5*32+19) has been rung.	
18	**Doorbell #(5*32+18) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(5*32+18) has been rung.	

DRB5ACT - Active Doorbell Register 5

	17	**Doorbell #(5*32+17) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+17) has been rung.
	16	**Doorbell #(5*32+16) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+16) has been rung.
	15	**Doorbell #(5*32+15) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+15) has been rung.
	14	**Doorbell #(5*32+14) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+14) has been rung.
	13	**Doorbell #(5*32+13) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+13) has been rung.
	12	**Doorbell #(5*32+12) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+12) has been rung.
	11	**Doorbell #(5*32+11) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+11) has been rung.
	10	**Doorbell #(5*32+10) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+10) has been rung.

DRB5ACT - Active Doorbell Register 5

9	**Doorbell #(5*32+9) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+9) has been rung.
8	**Doorbell #(5*32+8) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+8) has been rung.
7	**Doorbell #(5*32+7) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+7) has been rung.
6	**Doorbell #(5*32+6) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+6) has been rung.
5	**Doorbell #(5*32+5) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+5) has been rung.
4	**Doorbell #(5*32+4) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+4) has been rung.
3	**Doorbell #(5*32+3) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+3) has been rung.
2	**Doorbell #(5*32+2) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+2) has been rung.

DRB5ACT - Active Doorbell Register 5

	1	**Doorbell #(5*32+1) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+1) has been rung.
	0	**Doorbell #(5*32+0) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+0) has been rung.

Active Doorbell Register 6

DRB6ACT - Active Doorbell Register 6		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	01918h	
DWord	**Bit**	**Description**
0	31	**Doorbell #(6*32+31) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+31) has been rung.
	30	**Doorbell #(6*32+30) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+30) has been rung.
	29	**Doorbell #(6*32+29) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+29) has been rung.
	28	**Doorbell #(6*32+28) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+28) has been rung.
	27	**Doorbell #(6*32+27) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+27) has been rung.
	26	**Doorbell #(6*32+26) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+26) has been rung.

DRB6ACT - Active Doorbell Register 6

25	**Doorbell #(6*32+25) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+25) has been rung.
24	**Doorbell #(6*32+24) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+24) has been rung.
23	**Doorbell #(6*32+23) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+23) has been rung.
22	**Doorbell #(6*32+22) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+22) has been rung.
21	**Doorbell #(6*32+21) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+21) has been rung.
20	**Doorbell #(6*32+20) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+20) has been rung.
19	**Doorbell #(6*32+19) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+19) has been rung.
18	**Doorbell #(6*32+18) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+18) has been rung.

DRB6ACT - Active Doorbell Register 6

	17	**Doorbell #(6*32+17) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+17) has been rung.
	16	**Doorbell #(6*32+16) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+16) has been rung.
	15	**Doorbell #(6*32+15) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+15) has been rung.
	14	**Doorbell #(6*32+14) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+14) has been rung.
	13	**Doorbell #(6*32+13) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+13) has been rung.
	12	**Doorbell #(6*32+12) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+12) has been rung.
	11	**Doorbell #(6*32+11) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+11) has been rung.
	10	**Doorbell #(6*32+10) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+10) has been rung.

DRB6ACT - Active Doorbell Register 6

9	**Doorbell #(6*32+9) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(6*32+9) has been rung.	
8	**Doorbell #(6*32+8) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(6*32+8) has been rung.	
7	**Doorbell #(6*32+7) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(6*32+7) has been rung.	
6	**Doorbell #(6*32+6) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(6*32+6) has been rung.	
5	**Doorbell #(6*32+5) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(6*32+5) has been rung.	
4	**Doorbell #(6*32+4) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(6*32+4) has been rung.	
3	**Doorbell #(6*32+3) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(6*32+3) has been rung.	
2	**Doorbell #(6*32+2) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(6*32+2) has been rung.	

DRB6ACT - Active Doorbell Register 6

	1	**Doorbell #(6*32+1) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+1) has been rung.
	0	**Doorbell #(6*32+0) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+0) has been rung.

Active Doorbell Register 7

DRB7ACT - Active Doorbell Register 7

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	0191Ch

DWord	Bit	Description
0	31	**Doorbell #(7*32+31) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+31) has been rung.
	30	**Doorbell #(7*32+30) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+30) has been rung.
	29	**Doorbell #(7*32+29) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+29) has been rung.
	28	**Doorbell #(7*32+28) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+28) has been rung.
	27	**Doorbell #(7*32+27) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+27) has been rung.
	26	**Doorbell #(7*32+26) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+26) has been rung.

DRB7ACT - Active Doorbell Register 7

25	**Doorbell #(7*32+25) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(7*32+25) has been rung.	
24	**Doorbell #(7*32+24) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(7*32+24) has been rung.	
23	**Doorbell #(7*32+23) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(7*32+23) has been rung.	
22	**Doorbell #(7*32+22) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(7*32+22) has been rung.	
21	**Doorbell #(7*32+21) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(7*32+21) has been rung.	
20	**Doorbell #(7*32+20) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(7*32+20) has been rung.	
19	**Doorbell #(7*32+19) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(7*32+19) has been rung.	
18	**Doorbell #(7*32+18) Active**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Doorbell #(7*32+18) has been rung.	

DRB7ACT - Active Doorbell Register 7

	17	**Doorbell #(7*32+17) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+17) has been rung.
	16	**Doorbell #(7*32+16) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+16) has been rung.
	15	**Doorbell #(7*32+15) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+15) has been rung.
	14	**Doorbell #(7*32+14) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+14) has been rung.
	13	**Doorbell #(7*32+13) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+13) has been rung.
	12	**Doorbell #(7*32+12) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+12) has been rung.
	11	**Doorbell #(7*32+11) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+11) has been rung.
	10	**Doorbell #(7*32+10) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+10) has been rung.

DRB7ACT - Active Doorbell Register 7

9	**Doorbell #(7*32+9) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+9) has been rung.
8	**Doorbell #(7*32+8) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+8) has been rung.
7	**Doorbell #(7*32+7) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+7) has been rung.
6	**Doorbell #(7*32+6) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+6) has been rung.
5	**Doorbell #(7*32+5) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+5) has been rung.
4	**Doorbell #(7*32+4) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+4) has been rung.
3	**Doorbell #(7*32+3) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+3) has been rung.
2	**Doorbell #(7*32+2) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+2) has been rung.

DRB7ACT - Active Doorbell Register 7

	1	**Doorbell #(7*32+1) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+1) has been rung.
	0	**Doorbell #(7*32+0) Active** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+0) has been rung.

Active Head Pointer Register

ACTHD - Active Head Pointer Register	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
_Custom_GTIReset:	DEV
_Custom_GTIStorage:	FLOP
Address:	02074h-02077h
Name:	Active Head Pointer Register
ShortName:	ACTHD_RCSUNIT
Address:	18074h-18077h
Name:	Active Head Pointer Register
ShortName:	ACTHD_POCSUNIT
Address:	22074h-22077h
Name:	Active Head Pointer Register
ShortName:	ACTHD_BCSUNIT
Address:	1C0074h-1C0077h
Name:	Active Head Pointer Register
ShortName:	ACTHD_VCSUNIT0
Address:	1C4074h-1C4077h
Name:	Active Head Pointer Register
ShortName:	ACTHD_VCSUNIT1
Address:	1C8074h-1C8077h
Name:	Active Head Pointer Register
ShortName:	ACTHD_VECSUNIT0
Address:	1D0074h-1D0077h
Name:	Active Head Pointer Register
ShortName:	ACTHD_VCSUNIT2
Address:	1D4074h-1D4077h
Name:	Active Head Pointer Register
ShortName:	ACTHD_VCSUNIT3
Address:	1D8074h-1D8077h
Name:	Active Head Pointer Register
ShortName:	ACTHD_VECSUNIT1
Address:	1E0074h-1E0077h
Name:	Active Head Pointer Register

ACTHD - Active Head Pointer Register

ShortName:	ACTHD_VCSUNIT4
Address:	1E4074h-1E4077h
Name:	Active Head Pointer Register
ShortName:	ACTHD_VCSUNIT5
Address:	1E8074h-1E8077h
Name:	Active Head Pointer Register
ShortName:	ACTHD_VECSUNIT2
Address:	1F0074h-1F0077h
Name:	Active Head Pointer Register
ShortName:	ACTHD_VCSUNIT6
Address:	1F4074h-1F4077h
Name:	Active Head Pointer Register
ShortName:	ACTHD_VCSUNIT7
Address:	1F8074h-1F8077h
Name:	Active Head Pointer Register
ShortName:	ACTHD_VECSUNIT3
Address:	1A074h-1A077h
Name:	Active Head Pointer Register
ShortName:	ACTHD_CCSUNIT0

This register contains the address details of the data dword being parsed by command streamer.

- When the commands are being executed from a batch buffer this register contains the Dword aligned Graphics Memory Address.
- When the commands are being executed from a ring buffer this register contains the Dword aligned offset into the ring buffer (offset from Ring Buffer start address).

DWord	Bit	Description
0	31:2	**Head Pointer** Format: GraphicsAddress[31:2] • When the commands are being executed from a batch buffer this register contains the Dword aligned Graphics Memory Address. • When the commands are being executed from a ring buffer this register contains the Dword aligned offset into the ring buffer (offset from Ring Buffer start address).
	1:0	**Reserved** Access: RO Format: MBZ

Advanced Scheduler Reset Request Messages

SSRREQ - Advanced Scheduler Reset Request Messages

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 0810Ch

Hardware (CS, VCS) initiated Advanced Scheduler reset request messages.

DWord	Bit	Description
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask In order to write to bits 15:0, the corresponding message mask bits must be written. For example, for bit 14 to be set, bit 30 needs to be 1 : 40004000
	15:14	**Reserved** Access: RO Format: MBZ
	13	**SFC3 gracefull reset request message** Access: R/W Set _Custom_GTIReset: BUS SFC3 gracefull Reset Request Message for 2nd Vbox: '1' : cmsfc Reset Requested - This bit is cleared by the CP upon completion of the reset request '0' : cmsfc Reset Not Requested
	12	**SFC2 gracefull reset request message** Access: R/W Set _Custom_GTIReset: BUS SFC2 gracefull Reset Request Message for 2nd Vbox: '1' : cmsfc Reset Requested - This bit is cleared by the CP upon completion of the reset request '0' : cmsfc Reset Not Requested
	11	**SFC1 gracefull reset request message (2nd Vbox)** Access: R/W Set _Custom_GTIReset: BUS SFC1 gracefull Reset Request Message for 2nd Vbox: '1' : cmsfc Reset Requested - This bit is cleared by the CP upon completion of the reset request '0' : cmsfc Reset Not Requested

SSRREQ - Advanced Scheduler Reset Request Messages

	10	**SFC0 gracefull reset request message (1st Vbox)** Access: R/W Set _Custom_GTIReset: BUS SFC0 gracefull Reset Request Message for 1st Vbox: '1' : cmsfc Reset Requested - This bit is cleared by the CP upon completion of the reset request '0' : cmsfc Reset Not Requested
	9	**VINunit cmfxrst reset request message (8nd Vbox)** Access: R/W Set _Custom_GTIReset: BUS CMFX Reset Request Message from the VINunit in 8nd Vbox: '1' : CMFX Reset Requested - This bit is cleared by the CP upon completion of the reset request '0' : CMFX Reset Not Requested
	8	**VINunit cmfxrst reset request message (7nd Vbox)** Access: R/W Set _Custom_GTIReset: BUS CMFX Reset Request Message from the VINunit in 7nd Vbox: '1' : CMFX Reset Requested - This bit is cleared by the CP upon completion of the reset request '0' : CMFX Reset Not Requested
	7	**VINunit cmfxrst reset request message (6nd Vbox)** Access: R/W Set _Custom_GTIReset: BUS CMFX Reset Request Message from the VINunit in 6nd Vbox: '1' : CMFX Reset Requested - This bit is cleared by the CP upon completion of the reset request '0' : CMFX Reset Not Requested
	6	**VINunit cmfxrst reset request message (5nd Vbox)** Access: R/W Set _Custom_GTIReset: BUS CMFX Reset Request Message from the VINunit in 5nd Vbox: '1' : CMFX Reset Requested - This bit is cleared by the CP upon completion of the reset request '0' : CMFX Reset Not Requested
	5	**VINunit cmfxrst reset request message (4nd Vbox)** Access: R/W Set _Custom_GTIReset: BUS CMFX Reset Request Message from the VINunit in 4nd Vbox: '1' : CMFX Reset Requested - This bit is cleared by the CP upon completion of the reset request

SSRREQ - Advanced Scheduler Reset Request Messages

		'0' : CMFX Reset Not Requested
	4	**VINunit cmfxrst reset request message (3nd Vbox)** Access: R/W Set _Custom_GTIReset: BUS CMFX Reset Request Message from the VINunit in 3nd Vbox: '1' : CMFX Reset Requested - This bit is cleared by the CP upon completion of the reset request '0' : CMFX Reset Not Requested
	3	**VINunit cmfxrst reset request message (2nd Vbox)** Access: R/W Set _Custom_GTIReset: BUS CMFX Reset Request Message from the VINunit in 2nd Vbox: '1' : CMFX Reset Requested - This bit is cleared by the CP upon completion of the reset request '0' : CMFX Reset Not Requested
	2	**VINunit cmfxrst Reset Request message** Access: R/W Set _Custom_GTIReset: BUS CMFX Reset Request Message from the VINunit: '1' : CMFX Reset Requested - This bit is cleared by the CP upon completion of the reset request '0' : CMFX Reset Not Requested
	1	**Render AS Reset Request Message** Access: R/W Set _Custom_GTIReset: BUS Render AS Reset Request Message from the CSunit: '1' : Render AS Reset Requested - This bit is cleared by the CP upon completion of the reset request '0' : Render AS Reset Not Requested
	0	**Media AS Reset Request Message** Access: R/W Set _Custom_GTIReset: BUS Media AS Reset Request Message from the VCSunit: '1' : Media AS Reset Requested - This bit is cleared by the CP upon completion of the reset request '0' : Media AS Reset Not Requested

Aggregate Perf Counter SPM0 Lower DWord Free

OAPERF_SPM0_LOWER_FREE - Aggregate Perf Counter SPM0 Lower DWord Free	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02980h

This counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted used this register.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO,so the value returned from this register may be different on back-to-back reads.

Aggregate Perf Counter SPM0 Upper DWord Free

OAPERF_SPM0_UPPER_FREE - Aggregate Perf Counter SPM0 Upper DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02984h

This counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted using this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

Aggregate Perf Counter SPM1 Lower DWord Free

OAPERF_SPM1_LOWER_FREE - Aggregate Perf Counter SPM1 Lower DWord Free		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02988h	
This counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted used this register.		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

Aggregate Perf Counter SPM1 Upper DWord Free

OAPERF_SPM1_UPPER_FREE - Aggregate Perf Counter SPM1 Upper DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0298Ch

This counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted using this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

ARAT C6 Disallow Threshold

ARAT_C6DIS - ARAT C6 Disallow Threshold		
Register Space: MMIO: 0/2/0 Size (in bits): 32		
Address: 0A178h		
DWord	**Bit**	**Description**
0	31:0	**C6 Disallow Threshold for ARAT** Access: R/W _Custom_GTIReset: DEV Threshold, in 10ns increments to prevent short C6.

ARAT Delta (LSB)

ARAT_TDELTA_LOW - ARAT Delta (LSB)		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	0A174h	
DWord	**Bit**	**Description**
0	31:2	**Lower Bits of Delta Time for ARAT** Access: R/W _Custom_GTIReset: DEV Bits [31:2] of Delta Time, in 10ns increments. Bits 1:0 dropped. This means the granularity is 40ns increments. For example, [31:2]=b1 means the delta time is 40ns.
	1	**ARAT Mode** Access: R/W _Custom_GTIReset: DEV 0b: One-Shot Mode (default). 1b: Periodic Mode.
	0	**ARAT Enable** Access: R/W _Custom_GTIReset: DEV 0b: ARAT Disabled (default). 1b: ARAT Enabled.

ARB_CTL

ARB_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	45000h-45003h
Name:	Display Arbitration Control 1
ShortName:	ARB_CTL
Reset:	soft
Address:	45800h-45803h
Name:	Display Arbitration Control 1 Abox 1
ShortName:	ARB_CTL_ABOX1
Reset:	soft
Address:	45808h-4580Bh
Name:	Display Arbitration Control 1 Abox 2
ShortName:	ARB_CTL_ABOX2
Reset:	soft

DWord	Bit	Description
0	31	**FBC Memory Wake** Setting this bit allows FBC compressed write requests to wake memory. Value: 1b — Name: Wake On **[Default]** Value: 0b — Name: Wake Off
	30	**Reserved**
	29	**Reserved** Access: RO Format: MBZ
	28:26	**HP Queue Watermark** The value in this register indicates the number of entries the high priority queue should have before it can be read. The value is zero based. Program the values as N-1, where 3'b011 indicates 4 entries. Value: 011b — Name: 4 entries **[Default]** Value: [0,7] — Name:
	25:24	**LP Write Request Limit** The value in this register indicates the maximum number of back-to-back LP write requests that will be accepted from a single client before re-arbitrating.

ARB_CTL

Value	Name
00b	1
01b	2
10b	4 **[Default]**
11b	8

23:20 **TLB Request Limit**

The value in this register indicates the maximum number of TLB requests that can be made in an arbitration loop. Zero is not a valid programming.

Value	Name
0110b	6 **[Default]**
[1,15]	

19:16 **TLB Request InFlight Limit**

The value in this register indicates the maximum number of TLB (or VTd) requests that can be in flight at any given time. Zero is not a valid programming.

Value	Name
0110b	6 **[Default]**
[1,15]	

15 **FBC Watermark Disable**

Setting this bit disables the FBC watermarks.

Value	Name
0b	Enable
1b	Disable

14:13 **Tiled Address Swizzling**

DRAM configuration registers show if memory address swizzling is needed.

Value	Name	Description
00b	No Display	No display request address swizzling
01b	Reserved	Address bit[6] swizzling for tiled surfaces is not used
10b	Reserved	
11b	Reserved	

12:8 **HP Page Break Limit**

The value in this register represents the maximum number of page breaks allowed in a HP request chain. Zero is not a valid programming.

Value	Name
10000b	16 **[Default]**
[1,31]	

7 **Reserved**

Access:	RO
Format:	MBZ

ARB_CTL

	6:0	**HP Data Request Limit** The value in this register represents the maximum number of cachelines allowed in a HP request chain. **Value** / **Name** 1010110b / 86 **[Default]** [1,127] / **Restriction** This value must always be programmed greater than 8.

ARB_CTL2

ARB_CTL2

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	45004h-45007h
Name:	Display Arbitration Control 2
ShortName:	ARB_CTL2
Reset:	soft
Address:	45804h-45807h
Name:	Display Arbitration Control 2 Abox 1
ShortName:	ARB_CTL2_ABOX1
Reset:	soft
Address:	4580Ch-4580Fh
Name:	Display Arbitration Control 2 Abox 2
ShortName:	ARB_CTL2_ABOX2
Reset:	soft

DWord	Bit	Description
0	31	**Reserved**
	30	**Reserved** Access: RO Format: MBZ
	29:28	**LP WD Write Request Limit** The value in this register indicates the maximum number of back to back LP write requests that will be accepted from WD before re-arbitrating. Value / Name: 00b: 1 01b: 2 10b: 4 **[Default]** 11b: 8
	27:26	**DSB LP Write Request Limit** The value in this register indicates the maximum number of back to back DSB LP write requests that will be accepted by a single client before re-arbitrating. Value / Name: 00b: 1 01b: 2 10b: 4 **[Default]**

ARB_CTL2

	11b	8

25:23 Reserved

Access:	RO
Format:	MBZ

22:20 Reserved

Access:	RO
Format:	MBZ

19:18 Par5 Request Limit

This field sets the maximum number of par5 requests before arbitration switches to another client.

Value	Name
00b	1
01b	2
10b	4 **[Default]**
11b	16

17:16 FBC Request Limit

This field sets the maximum number of FBC requests before arbitration switches to another client.

Value	Name
00b	1
01b	2 **[Default]**
10b	4
11b	8

15:14 Reserved

Access:	RO
Format:	MBZ

13 Reserved

12 Arbiter Trickle Feed Allow On HP Request

If enabled, Arbiter will allow trickle feed request from all clients if any of the client sends a high priority request

Value	Name
0b	Disable **[Default]**
1b	Enable

11 Reserved

10:9 Inflight LP Read Request Limit

The value in this register represents the maximum number of LP read request transactions that can be inflight at any given time.

ARB_CTL2

Value	Name
00b	1 LP
01b	2 LP
10b	3 LP
11b	4 LP **[Default]**

8:6 **Reserved**

Access:	RO
Format:	MBZ

5:4 **Inflight HP Read Request Limit**

The value in this register represents the maximum number of HP read request transactions that can be inflight at any given time.

Value	Name
00b	128 HP
01b	64 HP
10b	32 HP
11b	16 HP

3 **Enable IPC**

Enables the Isochronous Priority Control. If enabled, Display sends demoted requests once the transition watermark is reached. If transition watermark is not enabled, Display sends demoted requests when the display buffer is full.

Value	Name
0b	Disable
1b	Enable

Programming Notes
This field is not connected in ARB_CTL2_ABOX1 andARB_CTL2_ABOX2. The "Enable IPC" field in ARB_CTL2 enables/disabled IPC in all ABOXs.

2 **Reserved**

Access:	RO
Format:	MBZ

1:0 **RTID FIFO Watermark**

The value in this register represents the watermark value for the RTID FIFO. HP transactions will start only when the FIFO level is above or equal the watermark.

Value	Name
00b	8 RTIDs
01b	16 RTIDs
10b	32 RTIDs
11b	Reserved

ASL Storage

ASLS_0_2_0_PCI - ASL Storage

Register Space:	PCI: 0/2/0
Size (in bits):	32
Address:	000FCh

This is a software scratch register. The exact bit register usage must be worked out in common between System BIOS and driver software. For each device, the ASL control method requires two bits for DOD (BIOS detectable yes or no, VGA/NonVGA), one bit for DGS (enable/disable requested), and two bits for DCS (enabled now/disabled now, connected or not).

DWord	Bit	Description
0	31:0	**Device Switching Storage** Default Value: 00000000000000000000000000000000b Access: R/W _Custom_GTIReset: BUS Software controlled usage to support device switching.

ATS Capability

ATS_CAP_0_2_0_PCI - ATS Capability

Register Space:	PCI: 0/2/0
Size (in bits):	16
Address:	00204h

ATS Capability reports support for Device-TLBs on Device-2, compliant to PCI Express ATS specification.

DWord	Bit	Description
0	15:7	**Reserved** Access: RO Format: MBZ
	6	**Global Invalidate Supported** Default Value: 1b Access: RO _Custom_GTIReset: BUS If Set, the Function supports Invalidation Requests that have the Global Invalidate bit Set. If Clear, the Function ignores the Global Invalidate bit in all Invalidate requests. Reserved
	5	**Page Aligned Request** Default Value: 1b Access: RO _Custom_GTIReset: BUS Hardwired to 1, the Untranslated Address is always aligned to a 4096 byte boundary. Processor Graphics reports value of 1b indicating all VT-d and SVM translations are page-aligned.
	4:0	**Invalidate Queue Depth** Default Value: 00000b Access: RO _Custom_GTIReset: BUS The number of Invalidate Requests that the endpoint can accept before putting back pressure on the upstream connection. Hardwired to 0h, the function can accept 32 Invalidate Requests.

ATS Control

ATS_CTRL_0_2_0_PCI - ATS Control

Register Space: PCI: 0/2/0
Size (in bits): 16

Address: 00206h

DWord	Bit	Description
0	15	**ATS Enable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS When Set, the function is enabled to cache translations. Processor graphics ignores this field, as GT uses GTLB as IOTLB and only pretends to software that it has a Device-TLB. Software is expected to Set this field before configuring extended context-entry for Device2 with Page Request Enable field Set. For compatibility, this field is implemented as RW as software can read it to determine ATS enable status.
	14:5	**Reserved** Access: RO Format: MBZ
	4:0	**Smallest Translation Unit** Default Value: 00000b Access: R/W _Custom_GTIReset: BUS This value indicates to the Endpoint the minimum number of 4096-byte blocks that is indicated in a Translation Completion or Invalidate Request. This is a power of 2 multiple and the number of blocks is 2^STU. A value of 0 indicates one block and value 1F indicates 2^31 blocks. For IGD this must be programmed to 0h for 4KB as smallest translation unit.

ATS Extended Capability Header

ATS_EXTCAP_0_2_0_PCI - ATS Extended Capability Header

Register Space:	PCI: 0/2/0
Size (in bits):	32
Address:	00200h

ATS Capability reports support for Device-TLBs on Device-2, compliant to PCI Express ats specification.

DWord	Bit	Description
0	31:20	**Next Capability Offset** Default Value: 001100000000b Access: RO _Custom_GTIReset: BUS This is a hardwired pointer to the next item in the capabilities list. Value 300h in this field provides the offset for Page-Request Capability.
	19:16	**Version** Default Value: 0001b Access: RO _Custom_GTIReset: BUS Hardwired to capability version 1.
	15:0	**Capability ID** Default Value: 0000000000001111b Access: RO _Custom_GTIReset: BUS Hardwired to the ATS Extended Capability ID

AUD_CONFIG

AUD_CONFIG	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	65000h-65003h
Name:	Audio Configuration Transcoder A
ShortName:	AUD_TCA_CONFIG
Reset:	soft
Address:	65100h-65103h
Name:	Audio Configuration Transcoder B
ShortName:	AUD_TCB_CONFIG
Reset:	soft
Address:	65200h-65203h
Name:	Audio Configuration Transcoder C
ShortName:	AUD_TCC_CONFIG
Reset:	soft
Address:	65300h-65303h
Name:	Audio Configuration Transcoder D
ShortName:	AUD_TCD_CONFIG
Reset:	soft

This register configures the audio output. There is one instance of this register per transcoder A/B/C. Each Transcoder is independent of the other.

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29	**N value Index** 0b – HDMI **[Default]** – N value read on bits 27:20 and 15:4 reflects HDMI N value. Bits 27:20 and 15:4 are programmable to any N value. Default h7FA6 when bit 28 is not set. 1b – DisplayPort – N value read on bits 27:20 and 15:4 reflects DisplayPort N value. Set this bit to 1 before programming N value register. When this bit is set to 1, 27:20 and 15:4 will reflect the current N value. Default is h8000 when bit 28 is not set.

AUD_CONFIG

28	**N programming enable** This bit enables programming of N values for non-CEA modes. Please note that the transcoder to which audio is attached must be disabled when changing this field.
27:20	**Upper N value** Default Value: 00000111b These are bits [19:12] of programmable N values for non-CEA modes. Bit 29 of this register must also be written in order to enable programming. Please note that the transcoder to which audio is attached must be disabled when changing this field. See bit 29 description for default values.
19:16	**Pixel Clock HDMI** This is the target frequency of the CEA/HDMI video mode to which the audio stream is added. This value is used for generating N_CTS packets. This refers to only HDMI Pixel clock and does not refer to DisplayPort Link clock. DisplayPort Link clock does not require this programming. Note: The transcoder on which audio is attached must be disabled when changing this field.

Value	Name	Description
0b	**[Default]**	
0000b	25.2 / 1.001 MHz	25.2 / 1.001 MHz
0001b	25.2 MHz	25.2 MHz (Program this value for pixel clocks not listed in this field)
0010b	27 MHz	27 MHz
0011b	27 * 1.001 MHz	27 * 1.001 MHz
0100b	54 MHz	54 MHz
0101b	54 * 1.001 MHz	54 * 1.001 MHz
0110b	74.25 / 1.001 MHz	74.25 / 1.001 MHz
0111b	74.25 MHz	74.25 MHz
1000b	148.5 / 1.001 MHz	148.5 / 1.001 MHz
1001b	148.5 MHz	148.5 MHz
1010b	297 / 1.001 MHz	297 / 1.001 MHz
1011b	297 MHz	297 MHz
1100b	594 / 1.001 MHz	594 / 1.001 MHz
1101b	594 MHz	594 MHz
Others	Reserved	Reserved

15:4	**Lower N value** Default Value: 111110100110b These are bits [11:0] of programmable N values for non-CEA modes. Bit 29 of this register must also be written in order to enable programming. Please note that the transcoder to which audio is attached must be disabled when changing this field. See bit 29 description for default values
3	**Reserved**
2:0	**Reserved** Access: RO Format: MBZ

AUD_CONFIG_2

AUD_CONFIG_2	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	65004h-65007h
Name:	Audio Configuration Register 2 Transcoder A
ShortName:	AUD_TCA_CONFIG_2
Reset:	soft
Address:	65104h-65107h
Name:	Audio Configuration Register 2 Transcoder B
ShortName:	AUD_TCB_CONFIG_2
Reset:	soft
Address:	65204h-65207h
Name:	Audio Configuration Register 2 Transcoder C
ShortName:	AUD_TCC_CONFIG_2
Reset:	soft
Address:	65304h-65307h
Name:	Audio Configuration Register 2 Transcoder D
ShortName:	AUD_TCD_CONFIG_2
Reset:	soft

This is a new register to add 297 and 584MHz frequencies support for HDMI TMDS clocks. These are programmed along with the other lower bits of the N and CTS values in the Audio Config register. There is one instance of this register per transcoder A/B/C. Each Transcoder is independent of the other.

DWord	Bit	Description
0	31	**Reserved**
	30:21	**Reserved** Access: RO Format: MBZ
	20:16	**DPSpecVersion** Default Value: 00010010b DP spec version number that goes in the header of the samples. Default 12h for DP MST. Currently DP Compliance is expecting this field to be 00010001 (DP1.1) as the Compliance spec has not been updated. SW must program this register to "00010001" to overcome this compliance failure. This programming can be updated after the Compliance Specification is updated.

AUD_CONFIG_2		
	15:12	**Reserved** Access: RO Format: MBZ
	11:8	**Upper bits for N value** These are bits are concatenated as the upper 4 bits to the N value in the AUD_CONFIG register. Please note that the transcoder to which audio is attached must be disabled when changing this field. See bit 29 description for default values
	7:4	**Reserved** Access: RO Format: MBZ
	3:0	**Upper bits for MCTS value** These are the upper 4bits concatenated to CTS or M generation for CTM modes.

AUD_CONFIG_BE

AUD_CONFIG_BE

Register Space: MMIO: 0/2/0
Access: R/W
Size (in bits): 32

Address: 65EF0h-65EF3h
Name: Audio Config Register for Dacbeunit
ShortName: AUD_CONFIG_BE
Reset: soft

DWord	Bit	Description
0	31	**Delay sample count latch Pipe D** Access: R/W Pipe D Hblank SM arc delay for samplecount. Value / Name / Description: 0b / Delay by 16 clocks **[Default]** / When set to 0, sample count latch is delayed by 16 clocks after Hblank starts. 1b / Delay by 32 clocks / When set to 1, sample count latch is delayed by 32 clocks after Hblank starts.
	30	**Delay sample count latch Pipe C** Access: R/W Pipe C Hblank SM arc delay for samplecount. Value / Name / Description: 0b / Delay by 16 clocks **[Default]** / When set to 0, sample count latch is delayed by 16 clocks after Hblank starts. 1b / Delay by 32 clocks / When set to 1, sample count latch is delayed by 32 clocks after Hblank starts.
	29	**Delay sample count latch Pipe B** Access: R/W Pipe B Hblank SM arc delay for samplecount. Value / Name / Description: 0b / Delay by 16 clocks **[Default]** / When set to 0, sample count latch is delayed by 16 clocks after Hblank starts. 1b / Delay by 32 clocks / When set to 1, sample count latch is delayed by 32 clocks after Hblank starts.
	28	**Delay sample count latch Pipe A** Access: R/W Pipe A Hblank SM arc delay for samplecount.

Value	Name	Description
0b	Delay by 16 clocks **[Default]**	When set to 0, sample count latch is delayed by 16 clocks after Hblank starts.
1b	Delay by 32 clocks	When set to 1, sample count latch is delayed by 32 clocks after Hblank starts.

27 **HBlank Early Enable for PipeD**

Access:	R/W

Enables using the HBlank_start count bit-field value for triggering the samplecount calculation.

Value	Name	Description
0b	Hblank Early Disable **[Default]**	The default Hblank is used and a delay of 16 tcclks is added before the SM arcs during each hblank for Pipe D
1b	Hblank Early Enable	The early hblank programmed by fields hblank_start count for Pipe D will be used to trigger samplecount calculation for Pipe D.

26 **HBlank Early Enable for PipeC**

Access:	R/W

Enables using the HBlank_start count bit-field value for triggering the samplecount calculation.

Value	Name	Description
0b	Hblank Early Disable **[Default]**	The default Hblank is used and a delay of 16 tcclks is added before the SM arcs during each hblank for Pipe C
1b	Hblank Early Enable	The early hblank programmed by fields hblank_start count for Pipe C will be used to trigger samplecount calculation for Pipe C.

25 **HBlank Early Enable for PipeB**

Access:	R/W

Enables using the HBlank_start count bit-field value for triggering the samplecount calculation.

Value	Name	Description
0b	Hblank Early Disable **[Default]**	The default Hblank is used and a delay of 16 tcclks is added before the SM arcs during each hblank for Pipe B
1b	Hblank Early Enable	The early hblank programmed by fields hblank_start count for Pipe B will be used to trigger samplecount calculation for Pipe B.

24 **HBlank Early Enable for PipeA**

Access:	R/W

Enables using the HBlank_start count bit-field value for triggering the samplecount calculation.

Value	Name	Description
0b	Hblank Early Disable **[Default]**	The default Hblank is used and a delay of 16 tcclks is added before the SM arcs during each hblank for Pipe A.
1b	Hblank Early Enable	The early hblank programmed by fields hblank_start count for Pipe A will be used to trigger samplecount calculation for Pipe A.

AUD_CONFIG_BE

23:21 HBlank_start count for Pipe D

Access:	R/W

The number of tcclk cycles that Hblank early is generated.

Value	Name	Description
000b	Delay of 8 tcaclks	Hblank is generated 8 tcclks early.
001b	Delay of 16 tcaclks	Hblank is generated 16 tcclks early.
010b	Delay of 32 tcaclks **[Default]**	Hblank is generated 32 tcclks early.
011b	Delay of 64 tcaclks	Hblank is generated 64 tcclks early.
100b	Delay of 96 tcaclks	Hblank is generated 96 tcclks early.
101b	Delay of 128 tcaclks	Hblank is generated 128 tcclks early.

20 DP Mixer Mainstream priority enable for Pipe D

Access:	R/W

When set, this will prioritize sending of mainstream data. No Timestamps/DIPs can be sent when there is mainstream data to send except in lines 2-7 of the vblank..

19:18 Number of samples per line for Pipe D

Access:	R/W

When prorgammed to non zero value, this field determines how many samples are sent per line. This is to avoid the audio overflow for high resolutions with small hblanks regions.

Value	Name	Description
00b	All Samples available in buffer **[Default]**	When set to this value, all the collected samples in the buffer are unloaded on the line in the hblank region.
01b	1 sample per line	When set to this value, maximum of one sample(each sample has 2 channels data for layout0 and 8 channels data in layout 1 mode) in the buffer is unloaded on the line in the hblank region.
10b	2 sample per line	When set to this value, maximum of two samples(each sample has 2 channels data for layout0 and 8 channels data in layout 1 mode) in the buffer are unloaded on the line in the hblank region.

17:15 HBlank_start count for Pipe C

Access:	R/W

The number of tcclk cycles that Hblank early is generated.

Value	Name	Description
000b	Delay of 8 tccclks	Hblank is generated 8 tcclks early.
001b	Delay of 16 tccclks	Hblank is generated 16 tcclks early.
010b	Delay of 32 tccclks **[Default]**	Hblank is generated 32 tcclks early.
011b	Delay of 64 tccclks	Hblank is generated 64 tcclks early.
100b	Delay of 96 tccclks	Hblank is generated 96 tcclks early.
101b	Delay of 128 tccclks	Hblank is generated 128 tcclks early.

AUD_CONFIG_BE

	14	**DP Mixer Mainstream priority enable for Pipe C** Access: R/W When set, this will prioritize sending of mainstream data. No Timestamps/DIPs can be sent when there is mainstream data to send except in lines 2-7 of the vblank..
	13:12	**Number of samples per line for Pipe C** Access: R/W When prorgammed to non zero value, this field determines how many samples are sent per line. This is to avoid the audio overflow for high resolutions with small hblanks regions.

Value	Name	Description
00b	All Samples available in buffer **[Default]**	When set to this value, all the collected samples in the buffer are unloaded on the line in the hblank region.
01b	1 sample per line	When set to this value, maximum of one sample(each sample has 2 channels data for layout0 and 8 channels data in layout 1 mode) in the buffer is unloaded on the line in the hblank region.
10b	2 sample per line	When set to this value, maximum of two samples(each sample has 2 channels data for layout0 and 8 channels data in layout 1 mode) in the buffer are unloaded on the line in the hblank region.

	11:9	**HBlank_start count for Pipe B** Access: R/W The number of tcclk cycles that Hblank early is generated.

Value	Name	Description
000b	Delay of 8 tcbclks	Hblank is generated 8 tcclks early.
001b	Delay of 16 tcbclks	Hblank is generated 16 tcclks early.
010b	Delay of 32 tcbclks **[Default]**	Hblank is generated 32 tcclks early.
011b	Delay of 64 tcbclks	Hblank is generated 64 tcclks early.
100b	Delay of 96 tcbclks	Hblank is generated 96 tcclks early.
101b	Delay of 128 tcbclks	Hblank is generated 128 tcclks early.

	8	**DP Mixer Mainstream priority enable for Pipe B** Access: R/W When set, this will prioritize sending of mainstream data. No Timestamps/DIPs can be sent when there is mainstream data to send except in lines 2-7 of the vblank..
	7:6	**Number of samples per line for Pipe B** Access: R/W When programmed to non zero value, this field determines how many samples are sent per line. This is to avoid the audio overflow for high resolutions with small hblanks regions.

Value	Name	Description
00b	All Samples available in buffer **[Default]**	When set to this value, all the collected samples in the buffer are unloaded on the line in the hblank region.

		01b	1 sample per line	When set to this value, maximum of one sample(each sample has 2 channels data for layout0 and 8 channels data in layout 1 mode) in the buffer is unloaded on the line in the hblank region.
		10b	2 sample per line	When set to this value, maximum of two samples(each sample has 2 channels data for layout0 and 8 channels data in layout 1 mode) in the buffer are unloaded on the line in the hblank region.

5:3 HBlank_start count for Pipe A

Access:	R/W

The number of tcclk cycles that Hblank early is generated.

Value	Name	Description
000b	Delay of 8 tcaclks	Hblank is generated 8 tcclks early.
001b	Delay of 16 tcaclks	Hblank is generated 16 tcclks early.
010b	Delay of 32 tcaclks **[Default]**	Hblank is generated 32 tcclks early.
011b	Delay of 64 tcaclks	Hblank is generated 64 tcclks early.
100b	Delay of 96 tcaclks	Hblank is generated 96 tcclks early.
101b	Delay of 128 tcaclks	Hblank is generated 128 tcclks early.

2 DP Mixer Mainstream priority enable for Pipe A

Access:	R/W

When set, this will prioritize sending of mainstream data. No Timestamps/DIPs can be sent when there is mainstream data to send except in lines 2-7 of the vblank.

1:0 Number of samples per line for Pipe A

Access:	R/W

When programmed to non zero value, this field determines how many samples are sent per line. This is to avoid the audio overflow for high resolutions with small hblanks regions.

Value	Name	Description
00b	All Samples available in buffer **[Default]**	When set to this value, all the collected samples in the buffer are unloaded on the line in the hblank region.
01b	1 sample per line	When set to this value, maximum of one sample(each sample has 2 channels data for layout0 and 8 channels data in layout 1 mode) in the buffer is unloaded on the line in the hblank region.
10b	2 sample per line	When set to this value, maximum of two samples(each sample has 2 channels data for layout0 and 8 channels data in layout 1 mode) in the buffer are unloaded on the line in the hblank region.

AUD_DIP_ELD_CTRL_ST

AUD_DIP_ELD_CTRL_ST

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	650B4h-650B7h
Name:	Audio Control State for DIP and ELD Transcoder A
ShortName:	AUD_TCA_DIP_ELD_CTRL_ST
Reset:	soft
Address:	651B4h-651B7h
Name:	Audio Control State for DIP and ELD Transcoder B
ShortName:	AUD_TCB_DIP_ELD_CTRL_ST
Reset:	soft
Address:	652B4h-652B7h
Name:	Audio Control State for DIP and ELD Transcoder C
ShortName:	AUD_TCC_DIP_ELD_CTRL_ST
Reset:	soft
Address:	653B4h-653B7h
Name:	Audio Control State for DIP and ELD Transcoder D
ShortName:	AUD_TCD_DIP_ELD_CTRL_ST
Reset:	soft

There is one instance of this register per transcoder A/B/C.

DWord	Bit	Description
0	31:28	**DIP Port Select** Access: RO This read-only bit reflects which port is used to transmit the DIP data. This can only change when DIP is disabled. If one or more audio-related DIP packets is enabled and audio is enabled on a digital port, these bits will reflect the digital port to which audio is directed. For DP MST, this is the device select/pipe select.

Value	Name	Description
0000b	Reserved **[Default]**	Reserved
0001b	Digital Port B	Digital Port B
0010b	Digital Port C	Digital Port C
0011b	USBC1	USBC1
0100b	USBC2	USBC2
0101b	USBC3	USBC3
0110b	USBC4	USBC4

0111b	USBC5	USBC5
1000b	USBC6	USBC6

27:25 **Reserved**

Access:	RO
Format:	MBZ

24:21 **DIP type enable status**

Access:	RO

These bits reflect the DIP types enabled. It can be updated while the port is enabled. Within 2 vblank periods, the DIP is guaranteed to have been transmitted. Disabling a DIP type results in setting the contents of that DIP buffer to zero. A reserved setting reflects a disabled DIP.

Value	Name	Description
0000b	**[Default]**	
XXX0b	DIP Disable	Audio DIP disabled
XXX1b	DIP Enable	Audio DIP enabled
XX0Xb	ACP Disable	Generic 1 (ACP) DIP disabled
XX1Xb	ACP Enable	Generic 1 (ACP) DIP enabled
X0XXb	Generic 2 Disable	Generic 2 DIP disabled
X1XXb	Generic 2 Enable	Generic 2 DIP enabled, can be used by ISRC1 or ISRC2
1XXXb	Reserved	Reserved

20:18 **DIP buffer index**

This field is used during read of different DIPs, and during read or write of ELD data. These bits are used as an index to their respective DIP or ELD buffers. When the index is not valid, the contents of the DIP will return all 0s.

Value	Name	Description
000b	Audio **[Default]**	Audio DIP (31 bytes of address space, 31 bytes of data)
001b	Gen 1	Generic 1 (ACP) Data Island Packet (31 bytes of address space, 31 bytes of data)
010b	Gen 2	Generic 2 (ISRC1) Data Island Packet (31 bytes of address space, 31 bytes of data)
011b	Gen 3	Generic 3 (ISRC2) Data Island Packet (31 bytes of address space, 31 bytes of data)
Others	Reserved	Reserved

17:16 **DIP transmission frequency**

Access:	RO

These bits reflect the frequency of DIP transmission for the DIP buffer type designated in bits 20:18. When writing DIP data, this value is also latched when the first DW of the DIP is written. When read, this value reflects the DIP transmission frequency for the DIP buffer designated in bits 20:18.

AUD_DIP_ELD_CTRL_ST

		Value / **Name** / **Description** 00b / Disable **[Default]** / Disabled 01b / Reserved / Reserved 10b / Send Once / Send Once 11b / Best Effort / Best effort (Send at least every other vsync)
	15	**Reserved** Access: RO Format: MBZ
	14:10	**ELD buffer size** Default Value: 10101b Access: RO This field reflects the size of the ELD buffer in DWORDs (84 Bytes of ELD)
	9:5	**ELD access address** Selects the DWORD address for access to the ELD buffer (84 bytes). The value wraps back to zero when incremented past the max addressing value 0x1F. This field change takes effect immediately after being written. The read value indicates the current access address.
	4	**ELD ACK** Acknowledgement from the audio driver that ELD read has been completed
	3:0	**DIP access address** Selects the DWORD address for access to the DIP buffers. The value wraps back to zero when it incremented past the max addressing value of 0xF. This field change takes effect immediately after being written. The read value indicates the current access address.

AUD_EDID_DATA

AUD_EDID_DATA	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	65050h-65053h
Name:	Audio EDID Data Block Transcoder A
ShortName:	AUD_TCA_EDID_DATA
Reset:	soft
Address:	65150h-65153h
Name:	Audio EDID Data Block Transcoder B
ShortName:	AUD_TCB_EDID_DATA
Reset:	soft
Address:	65250h-65253h
Name:	Audio EDID Data Block Transcoder C
ShortName:	AUD_TCC_EDID_DATA
Reset:	soft
Address:	65350h-65353h
Name:	Audio EDID Data Block Transcoder D
ShortName:	AUD_TCD_EDID_DATA
Reset:	soft

These registers contain the HDMI/DP data block from the EDID. The graphics driver reads the EDID and writes the structure to these registers. The vendor specific data block may be longer than 8 bytes, but the driver must not write more than 48 bytes to the buffer. The EDID format is Version 3 within the CEA-861B specification. The HDMI/DP Vendor Specific Data Block is described in version 1.1 of the HDMI specification. These values are returned from the device as the HDMI/DP Vendor Specific Data Block response to a Get HDMI/DP Widget command. Writing sequence:

- Video software sets ELD invalid, and sets the ELD access address to 0, or to the desired DWORD to be written.
- Video software writes ELD data 1 DWORD at a time. The ELD access address autoincrements with each DWORD write, wrapping around to address 0 when the max buffer address size of 0xF has been reached.
- Please note that software must write an entire DWORD at a time.
- Please note that the audio driver checks the valid bit with each byte read of the ELD. This means that the video driver can unilaterally write ELD irrespective of audio driver ELD read status.

Reading sequence:

- Video software sets the ELD access address to 0, or to the desired DWORD to be read.
- Video software reads ELD data 1 DWORD at a time. The ELD access address autoincrements with each DWORD read, wrapping around to address 0 when the max buffer address size of 0xF has been reached.

AUD_EDID_DATA

There is one instance of this register per transcoder A/B/C.

DWord	Bit	Description
0	31:0	**EDID Data Block** Please note that the contents of this buffer are not cleared when ELD is disabled. The contents of this buffer are cleared during FLR.

AUD_FREQ_CNTRL

AUD_FREQ_CNTRL

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	65900h-65903h
Name:	Audio BCLK Frequency Control
ShortName:	AUD_FREQ_CNTRL
Reset:	soft

Please refer "Audio Link Settings" of the **"Audio Bios Programming Sequence"** section for details about per-project programming requirements for this register.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:14	**T-Mode** Indicates the T mode SDI is operating in. BIOS or System Software must pre-program the T-mode register.a. before the iDISPLAY Audio Link is brought out from Link Reset,b. to a value which is consistent with the value of the its counterpart T-mode bit in the Audio Controller.c. to a value which is within the electrical capabilities of the platform. Note that even T modes are prohibited from being used with any BCLK frequency which has an odd number of bit cells. Example, 2T mode is incompatible with BCLK=6MHz (125 bit cells). **Value** / **Name** / **Description** 00b / 4T / 4T mode with sdi data held for 4 bit clks. 01b / 2T / 2T Mode with sdi data held for 2 bit clocks. To use 2T mode, the bclk has to be 48MHz and flop in the IO needs to bypass by setting bit 13 of this register. BIOS has to program 48MHz in the controller also to use this mode. 10b / 8T **[Default]** / 8T Mode with sdi data held for 8 bit clocks. 11b / 16T / 16T Mode with sdi data held for 16 bit clocks.
	13	**Bypass Flop** Setting this bit will bypass the flop in the IO in the Audout path. **Value** / **Name** / **Description** 0b / No bypass **[Default]** / Flop in the AUDIO OUT IO is not bypassed. 1b / Bypass / Flop in the AUDIO OUT IO is bypassed.
	12:11	**Detect Frame sync early** These bits are used to pullin the frame sync detection logic earlier to compensate for PV issues if any. Audio codec starts driving the SDI pin earlier by the number of clocks programmed by this register.

AUD_FREQ_CNTRL

Value	Name	Description
00b	Pull in by 0 bclks	Frame sync is detected at bclk = 1998.
01b	Pull in by 1 bclks	Frame sync is detected at bclk = 1997.
10b	Pull in by 2 bclks **[Default]**	Frame sync is detected at bclk = 1996.
11b	Pull in by 3 bclks	Frame sync is detected at bclk = 1995.

10:5 Reserved

Access:	RO
Format:	MBZ

4 96MHz BCLK

Default Value:	1b

Indicates that iDISPLAY Audio Link will run at 96MHz. This bit is defaulted to 1. BIOS or System Software must pre-program B96 before the iDISPLAY Audio Link is brought out from reset.

3 48MHz BCLK

Default Value:	0b

Indicates that iDISPLAY Audio Link will run at 48MHz. This bit is defaulted to 0. BIOS or System Software must pre-program B96 before the iDISPLAY Audio Link is brought out from reset.

2:0 Reserved

Access:	RO
Format:	MBZ

AUD_INFOFR

AUD_INFOFR	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	65054h-65057h
Name:	Audio Widget Data Island Packet Transcoder A
ShortName:	AUD_TCA_INFOFR
Reset:	soft
Address:	65154h-65157h
Name:	Audio Widget Data Island Packet Transcoder B
ShortName:	AUD_TCB_INFOFR
Reset:	soft
Address:	65254h-65257h
Name:	Audio Widget Data Island Packet Transcoder C
ShortName:	AUD_TCC_INFOFR
Reset:	soft
Address:	65354h-65357h
Name:	Audio Widget Data Island Packet Transcoder D
ShortName:	AUD_TCD_INFOFR
Reset:	soft

When the IF type or dword index is not valid, the contents of the DIP will return all 0s.
These values are programmed by the audio driver in an HDMI/DP Widget Set command. They are returned one byte at a time from the device on the HD audio bus as the HDMI/DP DIP response to a Get HDMI/DP Widget command. To fetch a specific byte, the audio driver should send an HDMI/DP Widget HDMI/DP DIP Index Pointer Set command to set the index, then fetch the indexed byte using the HDMI/DP DIP get.

DWord	Bit	Description
0	31:0	**Data Island Packet Data** This reflects the contents of the DIP indexed by the DIP access address. The contents of this buffer are cleared during function reset or HD audio link reset.

AUD_M_CTS_ENABLE

AUD_M_CTS_ENABLE

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	65028h-6502Bh
Name:	Audio M and CTS Programming Enable Transcoder A
ShortName:	AUD_TCA_M_CTS_ENABLE
Reset:	soft
Address:	65128h-6512Bh
Name:	Audio M and CTS Programming Enable Transcoder B
ShortName:	AUD_TCB_M_CTS_ENABLE
Reset:	soft
Address:	65228h-6522Bh
Name:	Audio M and CTS Programming Enable Transcoder C
ShortName:	AUD_TCC_M_CTS_ENABLE
Reset:	soft
Address:	65328h-6532Bh
Name:	Audio M and CTS Programming Enable Transcoder D
ShortName:	AUD_TCD_M_CTS_ENABLE
Reset:	soft

There is one instance of this register per transcoder A/B/C.

DWord	Bit	Description
0	31:22	**Reserved** Access: RO Format: MBZ
	21	**CTS M value Index** 0b – CTS **[Default]** – CTS value read on bits 19:0 reflects CTS value. Bit 19:0 is programmable to any CTS value. default is 0 1b – M – M value read on bits 19:0 reflects DisplayPort M value. Set this bit to 1 before programming M value register. When this is set to 1 19:0 will reflect the current M value
	20	**Enable CTS or M prog** When set will enable CTS or M programming.

AUD_M_CTS_ENABLE		
	19:0	**CTS programming** These are bits [19:0] of programmable CTS values for non-CEA modes. Bit 21 of this register must also be written in order to enable programming. Please note that the transcoder to which audio is attached must be disabled when changing this field.

AUD_MISC_CTRL

AUD_MISC_CTRL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	65010h-65013h
Name:	Audio Converter 1 Misc Control
ShortName:	AUD_C1_MISC_CTRL
Reset:	soft
Address:	65110h-65113h
Name:	Audio Converter 2 Misc Control
ShortName:	AUD_C2_MISC_CTRL
Reset:	soft
Address:	65210h-65213h
Name:	Audio Converter 3 Misc Control
ShortName:	AUD_C3_MISC_CTRL
Reset:	soft
Address:	65310h-65313h
Name:	Audio Converter 4 Misc Control
ShortName:	AUD_C4_MISC_CTRL
Reset:	soft

There is one instance of this register per audio converter 1/2/3.

DWord	Bit	Description
0	31:9	**Reserved** Access: RO Format: MBZ
	8	**Reserved**
	7:4	**Output Delay** Default Value: 0100b The number of samples between when the sample is received from the HD Audio link and when it appears as an analog signal at the pin.
	3	**Reserved** Access: RO Format: MBZ
	2	**Sample Fabrication EN bit** Access: R/W This bit indicates whether internal fabrication of audio samples is enabled during a link underrun.

AUD_MISC_CTRL

Value	Name	Description
0b	Disable	Audio fabrication disabled
1b	Enable **[Default]**	Audio fabrication enabled

1 **Pro Allowed**

Access:	R/W

By default, the audio device is configured to consumer mode and does not allow the mode to be changed to professional mode by an HD Audio verb. When Pro is allowed by setting this configuration bit, the HD Audio codec allows a verb to set the device into professional mode. Note: Setting this configuration bit does not change the default Pro bit value to be 1. Pro must be set to 1 through the normal process, using a verb.

Value	Name	Description
0b	Consumer **[Default]**	Consumer use only
1b	Professional	Professional use allowed

0 **Reserved**

Access:	RO
Format:	MBZ

AUD_PIN_ELD_CP_VLD

AUD_PIN_ELD_CP_VLD

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	650C0h-650C3h
Name:	Audio Pin ELD and CP Ready Status
ShortName:	AUD_PIN_ELD_CP_VLD
Reset:	soft

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15	**Audio InactiveD** Inactive: When this bit is set, a digital display sink device has been attached but not active for streaming audio. 0b – Disable – Device is active for streaming audio data 1b – Enable – Device is connected but not active
	14	**Audio Output EnableD** This bit directs audio to the device connected to this transcoder. When enabled along with Inactive set to 0 and audio data is available, the audio data will be combined with the video data and sent over this transcoder. The audio unit uses the status of this bit to indicate presence of the HDMI/DP output to the audio driver. 0b – Disable – No Audio output 1b – Valid – Audio is enabled
	13	**CP ReadyD** This R/W bit reflects the state of CP request from the audio unit. When an audio CP request has been serviced, it must be reset to 1 by the video software to indicate that the CP request has been serviced. This is transcoder based. Software should add a delay of 1ms before updating the CP ready bit. This is needed to make sure that all the pending unsolicited responses are cleared (transmitted to HD audio) before CP ready unsolicited responses is generated. This is needed in case of DP MST is enabled and when many changes to PD, ELDV and CP ready bits are done during mode set. 0b – Pending or Not Ready – CP request pending or not ready to receive requests. 1b – Ready – CP request ready

AUD_PIN_ELD_CP_VLD

12 **ELD validD**

This R/W bit reflects the state of the ELD data written to the ELD RAM. After writing the ELD data, the video software must set this bit to 1 to indicate that the ELD data is valid. At audio codec initialization, or on a hotplug event, this bit is set to 0 by the video software. This bit is reflected in the audio pin complex widget as the ELD valid status bit. This is transcoder based.

Value	Name	Description
0b	Invalid	ELD data invalid (default, when writing ELD data, set 0 by software)
1b	Valid	ELD data valid (Set by video software only)

11 **Audio InactiveC**

Inactive: When this bit is set, a digital display sink device has been attached but not active for streaming audio.

Value	Name	Description
0b	Disable	Device is active for streaming audio data
1b	Enable	Device is connected but not active

10 **Audio Output EnableC**

This bit directs audio to the device connected to this transcoder. When enabled along with Inactive set to 0 and audio data is available, the audio data will be combined with the video data and sent over this transcoder. The audio unit uses the status of this bit to indicate presence of the HDMI/DP output to the audio driver.

Value	Name	Description
0b	Disable	No Audio output
1b	Valid	Audio is enabled

9 **CP ReadyC**

This R/W bit reflects the state of CP request from the audio unit. When an audio CP request has been serviced, it must be reset to 1 by the video software to indicate that the CP request has been serviced. This is transcoder based. Software should add a delay of 1ms before updating the CP ready bit. This is needed to make sure that all the pending unsolicited responses are cleared (transmitted to HD audio) before CP ready unsolicited responses is generated. This is needed in case of DP MST is enabled and when many changes to PD, ELDV and CP ready bits are done during mode set.

Value	Name	Description
0b	Pending or Not Ready	CP request pending or not ready to receive requests
1b	Ready	CP request ready

8 **ELD validC**

This R/W bit reflects the state of the ELD data written to the ELD RAM. After writing the ELD data, the video software must set this bit to 1 to indicate that the ELD data is valid. At audio codec initialization, or on a hotplug event, this bit is set to 0 by the video software. This bit is reflected in the audio pin complex widget as the ELD valid status bit. This is transcoder based.

Value	Name	Description
0b	Invalid	ELD data invalid (default, when writing ELD data, set 0 by software)
1b	Valid	ELD data valid (Set by video software only)

AUD_PIN_ELD_CP_VLD

7	**Audio InactiveB** Inactive: When this bit is set, a digital display sink device has been attached but not active for streaming audio.

Value	Name	Description
0b	Disable	Device is active for streaming audio data
1b	Enable	Device is connected but not active

6	**Audio Output EnableB** This bit directs audio to the device connected to this transcoder. When enabled along with Inactive set to 0 and audio data is available, the audio data will be combined with the video data and sent over this transcoder. The audio unit uses the status of this bit to indicate presence of the HDMI/DP output to the audio driver. This is transcoder based.

Value	Name	Description
0b	Disable	No audio output
1b	Enable	Audio is enabled

5	**CP ReadyB** See CP_ReadyC description.

Value	Name	Description
0b	Not Ready	CP request pending or not ready to receive requests
1b	Ready	CP request ready

4	**ELD validB** See ELD_validC descripion.

Value	Name	Description
0b	Invalid	ELD data invalid (default, when writing ELD data, set 0 by software)
1b	Valid	ELD data valid (Set by video software only)

3	**Audio InactiveA** Inactive: When this bit is set, a digital display sink device has been attached but not active for streaming audio.

Value	Name	Description
0b	Disable	Device is active for streaming audio data
1b	Enable	Device is connected but not active

2	**Audio Output EnableA** This bit directs audio to the device connected to this transcoder. When enabled along with Inactive set to 0 and audio data is available, the audio data will be combined with the video data and sent over this transcoder. The audio unit uses the status of this bit to indicate presence of the HDMI/DP output to the audio driver. This is transcoder based.

Value	Name	Description
0b	Disable	No audio output
1b	Enable	Audio is enabled

AUD_PIN_ELD_CP_VLD

	1	**CP ReadyA** See CP_ReadyC description. **Value** / **Name** / **Description** 0b / Not Ready / CP request pending or not ready to receive requests 1b / Ready / CP request ready
	0	**ELD validA** See ELD_validC descripion. **Value** / **Name** / **Description** 0b / Invalid / ELD data invalid (default, when writing ELD data, set 0 by software) 1b / Valid / ELD data valid (Set by video software only)

AUD_PIN_PIPE_CONN_ENTRY_LNGTH

AUD_PIN_PIPE_CONN_ENTRY_LNGTH

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	650A8h-650ABh
Name:	Audio Connection List Entry and Length Transcoder A
ShortName:	AUD_TCA_PIN_PIPE_CONN_ENTRY_LNGTH_RO
Reset:	soft
Address:	651A8h-651ABh
Name:	Audio Connection List Entry and Length Transcoder B
ShortName:	AUD_TCB_PIN_PIPE_CONN_ENTRY_LNGTH_RO
Reset:	soft
Address:	652A8h-652ABh
Name:	Audio Connection List Entry and Length Transcoder C
ShortName:	AUD_TCC_PIN_PIPE_CONN_ENTRY_LNGTH_RO
Reset:	soft
Address:	653A8h-653ABh
Name:	Audio Connection List Entry and Length Transcoder D
ShortName:	AUD_TCD_PIN_PIPE_CONN_ENTRY_LNGTH_RO
Reset:	soft

These values are returned from the device as the Connection List Length response to a Get Pin Widget command or Get Device Widget command if DP MST. There is one instance of this register per transcoder A/B/C.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:8	**Connection List Entry** Connection to Convertor Widget Node 0x03
	7	**Long Form** This bit indicates whether the items in the connection list are long form or short form. This bit is hardwired to 0 (items in connection list are short form)
	6:0	**Connection List Length** Default Value: 0000001b This field indicates the number of items in the connection list. If this field is 2, there is only one hardwired input possible, which is read from the Connection List, and there is no Connection Select Control.

AUD_PIPE_CONN_SEL_CTRL

AUD_PIPE_CONN_SEL_CTRL	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	650ACh-650AFh
Name:	Audio Pipe Connection Select Control
ShortName:	AUD_PIN_PIPE_CONN_SEL_CTRL_RO
Reset:	soft

These values are returned from the device as the Connection List Length response to a Get Pin Widget command or Get Device Widget command for DP MST.

DWord	Bit	Description
0	31:24	**Connection select Control PipeD** Default Value: 0Fh Connection Index Currently Set [Default 0x00], PipeD Widget is set to 0x03
	23:16	**Connection select Control PipeC** Default Value: 0Fh Connection Index Currently Set [Default 0x00], PipeC Widget is set to 0x02
	15:8	**Connection select Control PipeB** Default Value: 0Fh Connection Index Currently Set [Default 0x00], PipeB Widget is set to 0x01
	7:0	**Connection select Control PipeA** Default Value: 0Fh Connection Index Currently Set [Default 0x00], PipeA Widget is set to 0x00

AUD_PWRST

AUD_PWRST

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	6504Ch-6504Fh
Name:	Audio Power State Read Only
ShortName:	AUD_PWRST_RO
Reset:	soft

These values are returned from the device as the Power State response to a Get Audio Function Group command.

DWord	Bit	Description
0	31:30	**Converter4 Widget PwrSt Curr** Format: **Audio Power State Format** Converter4 Widget current power state Value: 11b — Name:
	29:28	**Converter4 Widget PwrSt Req** Format: **Audio Power State Format** Converter4 Widget power state that was requested by audio software Value: 11b — Name:
	27:26	**Func Grp Dev PwrSt Curr** Format: **Audio Power State Format** Function Group Device current power state Value: 11b — Name:
	25:24	**Func Grp Dev PwrSt Set** Format: **Audio Power State Format** Function Group Device power state that was set Value: 11b — Name:
	23:22	**Converter3 Widget PwrSt Curr** Format: **Audio Power State Format** Converter3 Widget current power state Value: 11b — Name:

AUD_PWRST

	21:20	**Converter3 Widget PwrSt Req** Format: **Audio Power State Format** Converter3 Widget power state that was requested by audio software **Value** / **Name** 11b
	19:18	**Convertor2 Widget PwrSt Curr** Format: **Audio Power State Format** Converor2 Widget current power state **Value** / **Name** 11b
	17:16	**Convertor2 Widget PwrSt Req** Format: **Audio Power State Format** Converter2 Widget power state that was requested by audio software **Value** / **Name** 11b
	15:14	**Convertor1 Widget PwrSt Curr** Format: **Audio Power State Format** Converter1 Widget current power state **Value** / **Name** 11b
	13:12	**Convertor1 Widget PwrSt Req** Format: **Audio Power State Format** Converter1 Widget power state that was requested by audio software **Value** / **Name** 11b
	11:10	**PinD Widget PwrSt Curr** Format: **Audio Power State Format** PinD Widget current power stateFor DP MST this represents Device3 power state **Value** / **Name** 11b
	9:8	**PinD Widget PwrSt Set** Format: **Audio Power State Format** PinD Widget power state that was setFor DP MST this represents Device3 power state **Value** / **Name** 11b
	7:6	**PinC Widget PwrSt Curr** Format: **Audio Power State Format** PinC Widget current power stateFor DP MST this represents Device2 power state

AUD_PWRST

		Value: 11b — Name:
	5:4	**PinC Widget PwrSt Set** Format: Audio Power State Format PinC Widget power state that was setFor DP MST this represents Device2 power state Value: 11b — Name:
	3:2	**PinB Widget PwrSt Curr** Format: Audio Power State Format PinB Widget current power stateFor DP MST this represents Device1 power state Value: 11b — Name:
	1:0	**PinB Widget PwrSt Set** Format: Audio Power State Format PinB Widget power state that was setFor DP MST this represents Device1 power state Value: 11b — Name:

AUD_RID

AUD_RID	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	65024h-65027h
Name:	Audio Revision ID Read Only
ShortName:	AUD_RID_RO
Reset:	soft

These values are returned from the device as the Revision ID response to a Get Root Node command.

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23:20	**Major Revision** Default Value: 1h The major revision number (left of the decimal) of the HD Audio Spec to which the codec is fully compliant. This field is hardwired within the device.
	19:16	**Minor Revision** Default Value: 0h The minor revision number (rights of the decimal) or dot number of the HD Audio Spec to which the codec is fully compliant. This field is hardwired within the device.
	15:8	**Revision ID** Default Value: 00h The vendor revision number for this given Device ID. This field is hardwired within the device.
	7:0	**Stepping ID** Default Value: 00h An optional vendor stepping number within the given Revision ID. This field is hardwired within the device.

AUD_VID_DID

AUD_VID_DID	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	65020h-65023h
Name:	Audio Vendor ID / Device ID Read Only
ShortName:	AUD_VID_DID_RO
Reset:	soft

These values are returned from the device as the Vendor ID/ Device ID response to a Get Root Node command.

DWord	Bit	Description
0	31:16	**Vendor ID** Default Value: 8086h Used to identify the codec within the PnP system. This field is hardwired within the device.
	15:8	**Device ID Upper byte** Default Value: 28h Constant used to identify the codec within the PnP system. This field is set by the device hardware.
	7:0	**Device ID Lower byte** Constant used to identify the codec within the PnP system. This field is set by fuse download. For correct values refer to the Codec root node parameter 00h.

AUD_WD_CNTRL

AUD_WD_CNTRL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6580Ch-6580Fh
Name:	Audio WD Control for slice 1
ShortName:	AUD_WD_CNTRL_1
Reset:	soft
Address:	6590Ch-6590Fh
Name:	Audio WD Control for slice 2
ShortName:	AUD_WD_CNTRL_2
Reset:	soft

There are two sets of these registers. One for each Wireless Audio Slice

DWord	Bit	Description
0	31	**Audio Inactive WD** Inactive: When this bit is set, wireless display device has been attached but not active for streaming audio. Value / Name / Description: 0b / Disable / Device is active for streaming audio data 1b / Enable / Device is connected but not active
	30	**Audio Output Enable WD** This bit directs audio to the Wireless Device. When enabled along with Inactive set to 0 and audio data is available, the audio data will be sent to the Wireless memory where WDBOX can read and send it to the Wireless NIC. Value / Name / Description: 0b / Disable / No Audio output 1b / Valid / Audio is enabled
	29	**CP Ready WD** This R/W bit reflects the state of CP request from the audio unit. When an audio CP request has been serviced, it must be reset to 1 by the display software to indicate that the CP request has been serviced. Value / Name / Description: 0b / Pending or Not Ready / CP request pending or not ready to receive requests 1b / Ready / CP request ready
	28	**ELD valid WD** This R/W bit reflects the state of the ELD data written to the ELD RAM. After writing the ELD data, the video software must set this bit to 1 to indicate that the ELD data is valid. At audio codec initialization, or on a hotplug event, this bit is set to 0 by the video software. This bit is

AUD_WD_CNTRL

reflected in the audio pin complex widget as the ELD valid status bit.

Value	Name	Description
0b	Invalid	ELD data invalid (default, when writing ELD data, set 0 by software)
1b	Valid	ELD data valid (Set by video software only)

27:24 **Reserved**

Access:	RO
Format:	MBZ

23 **WiGig Widi Mode**

Access:	R/W

This field indicates Wigig or Widi Mode.

Value	Name	Description
0b	WidiMode **[Default]**	default mode is Widi mode. When bit 30 is set to 1 and this bit is set to 0 Widi is enabled
1b	WiGigMode	When bit 30 is set to 1 and this bit is set to 1 WiGig is enabled

22:20 **WiGig TFD Size**

Access:	R/W

This is TFD size. The total circular buffer should be a multiple of this number.

Value	Name	Description
000b	512B **[Default]**	default size of TFD is 512B
001b	1024B	TFD Size is 1024B
010b	2048B	TFD Size is 2048B
011b	4096B	TFD Size is 4096B
100b	8192B	TFD Size is 8192B
101b	Reserved	Reserved
110b	Reserved	Reserved
111b	Reserved	Reserved

19:11 **Reserved**

Access:	RO
Format:	MBZ

10:6 **ELD buffer size**

Default Value:	10101b
Access:	RO

This field reflects the size of the ELD buffer in DWORDs (84 Bytes of ELD)

5:1 **ELD access address**

Selects the DWORD address for access to the ELD buffer (84 bytes). The value wraps back to zero when incremented past the max addressing value 0x1F. This field change takes effect immediately after being written. The read value indicates the current access address.

AUD_WD_CNTRL		
	0	**ELD ACK** Acknowledgement from the audio driver that ELD read has been completed

AUD_WD_DMA_HDPTR

AUD_WD_DMA_HDPTR		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	65880h-65883h	
Name:	Audio WD Input DMA Head Pointer for slice 1	
ShortName:	AUD_WD_DMA_HDPTR_1	
Reset:	soft	
Address:	65980h-65983h	
Name:	Audio WD Input DMA Head Pointer for slice 2	
ShortName:	AUD_WD_DMA_HDPTR_2	
Reset:	soft	

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Aud WD HEAD PTR** Access: R/W This field provides write of the head pointer from WDBOX/WNIC. Comes in as message to Display Engine. Head Pointer always rolls over after it reaches the Audio WD Buffer size.

AUD_WD_DMA_TAILPTR

AUD_WD_DMA_TAILPTR	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	65890h-65893h
Name:	Audio WD Input DMA Tail Pointer for slice 1
ShortName:	AUD_WD_DMA_TAILPTR_1
Reset:	soft
Address:	65990h-65993h
Name:	Audio WD Input DMA Tail Pointer for slice 2
ShortName:	AUD_WD_DMA_TAILPTR_2
Reset:	soft

There are two sets of these registers. One for each wireless Audio slice.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Aud WD TAILPTR** Access: RO This field provides readback of the tail pointer sent to the WDBOX. Tail Pointer always rolls over after it reaches the Audio WD Buffer size.

AUD_WD_EDID_DATA

AUD_WD_EDID_DATA	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	65810h-65813h
Name:	Audio WD EDID Data for slice 1
ShortName:	AUD_WD_EDID_DATA_1
Reset:	soft
Address:	65910h-65913h
Name:	Audio WD EDID Data for slice 2
ShortName:	AUD_WD_EDID_DATA_2
Reset:	soft

THere are two sets of these registers. One for each Wireless Audio Slice.

This register contains the Wireless data block from the EDID. The graphics driver reads the EDID and writes the structure to these registers. The vendor specific data block may be longer than 8 bytes, but the driver must not write more than 48 bytes to the buffer. The EDID format is Version 3 within the CEA-861B specification. The HDMI/DP Vendor Specific Data Block is described in version 1.1 of the HDMI specification. These values are returned from the device as the HDMI/DP Vendor Specific Data Block response to a Get Wireless Widget command. Writing sequence:

- Video software sets ELD invalid, and sets the ELD access address to 0, or to the desired DWORD to be written.
- Video software writes ELD data 1 DWORD at a time. The ELD access address autoincrements with each DWORD write, wrapping around to address 0 when the max buffer address size has been reached.
- Please note that software must write an entire DWORD at a time.
- Please note that the audio driver checks the valid bit with each byte read of the ELD. This means that the video driver can unilaterally write ELD irrespective of audio driver ELD read status.

Reading sequence:

- Video software sets the ELD access address to 0, or to the desired DWORD to be read.
- Video software reads ELD data 1 DWORD at a time. The ELD access address autoincrements with each DWORD read, wrapping around to address 0 when the max buffer address size has been reached.

DWord	Bit	Description
0	31:0	**EDID Data Block** This field specifies the EDID data block. Please note that the contents of this buffer are not cleared when ELD is disabled. The contents of this buffer are cleared during a device 2 FLR.

AUD_WNIC_2_AUD_RESET

AUD_WNIC_2_AUD_RESET	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6589Ch-6589Fh
Name:	WNIC to Audio RESET
ShortName:	AUD_WNIC_2_AUD_RESET_1
Reset:	soft
Address:	6599Ch-6599Fh
Name:	WNIC to Audio RESET
ShortName:	AUD_WNIC_2_AUD_RESET_2
Reset:	soft

There are two sets of these registers. One for each Audio wireless slice.

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**WNIC to AUDIO Reset** When set, • HW shall reset the audio TFD head and tail pointers. • Audio immediately stops sending audio TFD tail pointer to WNIC. • HW shall drain and throw away any remaining audio WDE packets left in the pipeline. Audio packets continue to stream internally; however, none of these shall be put in the TFD. • When reset request is clear, WDBOX resumes transmission as soon as enough audio packets exist.

AUD_WNIC_PCR_LOWADDROFFSET

AUD_WNIC_PCR_LOWADDROFFSET	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	658A0h-658A3h
Name:	Audio WNIC PCR Low offset address for slice 1
ShortName:	AUD_WNIC_PCR_LOWADDROFFSET_1
Reset:	soft
Address:	659A0h-659A3h
Name:	Audio WNIC PCR Low offset address for slice 2
ShortName:	AUD_WNIC_PCR_LOWADDROFFSET_2
Reset:	soft
There are two sets of these registers. One for each Audio wireless slice.	

DWord	Bit	Description
0	31:0	**Aud WNIC PCR LOW ADDRESS OFFSET** This field provides low base address offset which will be used to sent the PCR updates to WNIC.

AUD_WNIC_PCR_UPADDROFFSET

AUD_WNIC_PCR_UPADDROFFSET	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	658A4h-658A7h
Name:	Audio WNIC PCR Upper offset address for slice 1
ShortName:	AUD_WNIC_PCR_UPADDROFFSET_1
Reset:	soft
Address:	659A4h-659A7h
Name:	Audio WNIC PCR Upper offset address for slice 2
ShortName:	AUD_WNIC_PCR_UPADDROFFSET_2
Reset:	soft
There are two sets of these registers. One for each Audio Wireless slice.	

DWord	Bit	Description
0	31:0	**Aud WNIC PCR UPPER ADDRESS OFFSET** This field provides upper base address offset which will be used to sent the PCR updates to WNIC. Only bits 7:0 should be programmed as total address space of 39:0 is available.

AUD_WNIC_TAILPTR_ADDROFFSET

AUD_WNIC_TAILPTR_ADDROFFSET	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	65894h-65897h
Name:	Audio WNIC Tail Pointer offset address for slice 1
ShortName:	AUD_WNIC_TAILPTR_ADDROFFSET_1
Reset:	soft
Address:	65994h-65997h
Name:	Audio WNIC Tail Pointer offset address for slice 2
ShortName:	AUD_WNIC_TAILPTR_ADDROFFSET_2
Reset:	soft
There are two sets of these registers. One for each Audio Wireless slice.	

DWord	Bit	Description
0	31:0	**Aud WNIC TAIL PTR OFFSET** This field provides address offset which will be used to sent the tail point updates to WNIC.

AUD_WNIC_TAILPTR_UPADDROFFSET

AUD_WNIC_TAILPTR_UPADDROFFSET	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	65898h-6589Bh
Name:	Audio WNIC Tail Pointer offset upper address for slice 1
ShortName:	AUD_WNIC_TAILPTR_UPADDROFFSET_1
Reset:	soft
Address:	65998h-6599Bh
Name:	Audio WNIC Tail Pointer offset upper address for slice 2
ShortName:	AUD_WNIC_TAILPTR_UPADDROFFSET_2
Reset:	soft
There are two sets of these registers. One for each Audio Wireless Slice.	

DWord	Bit	Description
0	31:0	**Aud WNIC TAIL PTR Upper address offset** This field provides upper address offset which will be used to sent the tail point updates to WNIC. Only bits 7:0 should be programmed as total address space of 39:0 is available.

AUDIO_PIN_BUF_CTL

AUDIO_PIN_BUF_CTL

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	48414h-48417h
Name:	Audio Pins Buffer Control
ShortName:	AUDIO_PIN_BUF_CTL
Reset:	soft

This register controls the display audio pins I/O buffers.

DWord	Bit	Description
0	31	**Enable** This field enables the audio buffer. Value: 0b = Disable; 1b = Enable
	30	**Reserved** Access: RO Format: MBZ
	29:28	**Hysteresis**
	27	**Reserved** Access: RO Format: MBZ
	26:24	**Spare**
	23:21	**Reserved** Access: RO Format: MBZ
	20:16	**Pulldown Strength**
	15:12	**Pulldown Slew**
	11:9	**Reserved** Access: RO Format: MBZ
	8:4	**Pullup Strength**
	3:0	**Pullup Slew**

Audio Codec Interrupt Definition

Audio Codec Interrupt Definition	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	44480h-4448Fh
Name:	Audio Codec Interrupts
ShortName:	AUD_INTERRUPT
Reset:	soft

This table indicates which events are mapped to each bit of the Audio Codec Interrupt registers.
0x44480 = ISR
0x44484 = IMR
0x44488 = IIR
0x4448C = IER

DWord	Bit	Description
0	31	**Audio_Power_State_change_Transcoder_D** The ISR is an active high pulse when there is a power state change for audio for DDI D.
	30	**Audio_Power_State_change_Transcoder_C** The ISR is an active high pulse when there is a power state change for audio for DDI C.
	29	**Audio_Power_State_change_Transcoder_B** The ISR is an active high pulse when there is a power state change for audio for DDI B.
	28	**Reserved**
	27	**Reserved**
	26	**Audio_Function_Group_Power_State_change** The ISR is an active high pulse when there is a power state change for audio of function group widget.
	25	**Audio_Conv1_Power_State_change** The ISR is an active high pulse when there is a power state change for audio of Converter 1 widget.
	24	**Audio_Conv2_Power_State_change** The ISR is an active high pulse when there is a power state change for audio of Converter 2 widget.
	23	**Audio_Conv3_Power_State_change** The ISR is an active high pulse when there is a power state change for audio of Converter 3 widget.
	22	**Audio_Conv4_Power_State_change** The ISR is an active high pulse when there is a power state change for audio of Converter 4 widget.
	21	**Spare 21**
	20	**Spare 20**
	19	**Spare 19**
	18	**Reserved**
	17	**Reserved**
	16	**Reserved**

Audio Codec Interrupt Definition		
	15	**Reserved**
	14	**Reserved**
	13	**Reserved**
	12	**Spare 12**
	11	**Audio_Power_State_change_Transcoder_A** The ISR is an active high pulse when there is a power state change for audio for DDI F.
	10	**Reserved**
	9	**Reserved**
	8	**Reserved**
	7	**Reserved**
	6	**Reserved**
	5	**Reserved**
	4:3	**Unused_Int_4_3** These interrupts are currently unused.
	2	**Reserved**
	1	**Reserved**
	0	**Audio_Mailbox_Write** The ISR is an active high pulse when there is a write to any of the four Audio Mail box verbs in vendor defined node ID 8

Auto Draw End Offset

3DPRIM_END_OFFSET - Auto Draw End Offset	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02420h-02423h
Name:	Auto Draw End Offset
ShortName:	3DPRIM_END_OFFSET_RCSUNIT_BE_GEOMETRY
Address:	18420h-18423h
Name:	Auto Draw End Offset
ShortName:	3DPRIM_END_OFFSET_POCSUNIT_BE_GEOMETRY

DWord	Bit	Description
0	31:0	**End Offset** Format: U32 This register is used to store the end offset value used by the Vertex Fetch to determine when to stop processing the 3D_PRIMITIVE command. This register is valid when the End Offset Enable is set in the 3D_PRIMITIVE command.

AVP av1 status

AVP_STATUS_ADDR - AVP av1 status	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	BUS
Address:	1C2B04h-1C2B07h
Name:	AVP av1 status
ShortName:	AVP_STATUS_ADDR_AWM_REG0
Address:	1C6B04h-1C6B07h
Name:	AVP av1 status
ShortName:	AVP_STATUS_ADDR_AWM_REG1
Address:	1D2B04h-1D2B07h
Name:	AVP av1 status
ShortName:	AVP_STATUS_ADDR_AWM_REG2
Address:	1D6B04h-1D6B07h
Name:	AVP av1 status
ShortName:	AVP_STATUS_ADDR_AWM_REG3
Address:	1E2B04h-1E2B07h
Name:	AVP av1 status
ShortName:	AVP_STATUS_ADDR_AWM_REG4
Address:	1E6B04h-1E6B07h
Name:	AVP av1 status
ShortName:	AVP_STATUS_ADDR_AWM_REG5
Address:	1F2B04h-1F2B07h
Name:	AVP av1 status
ShortName:	AVP_STATUS_ADDR_AWM_REG6
Address:	1F6B04h-1F6B07h
Name:	AVP av1 status
ShortName:	AVP_STATUS_ADDR_AWM_REG7

This register stores the number of clock cycles spent decoding/encoding the current frame.

DWord	Bit	Description
0	31	**AVP Pipe Active** Access: RO Indicate AVP Pipe is currently active in processing the frame/tile

AVP_STATUS_ADDR - AVP av1 status		
	30:2	**Reserved** Access: RO Format: MBZ
	1	**Final MV Overflow Error** Access: RO Format: U1 Indicate the final MV is overflowing
	0	**Bitstream Upper Bound Error** Access: RO Format: U1 Indicate the bitstream access has reached the upper bound of the buffer and overflow

AVP av1 unit done

AVP_UNIT_DONE_ADDR - AVP av1 unit done	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	BUS
Address:	1C2B00h-1C2B03h
Name:	AVP av1 unit done
ShortName:	AVP_UNIT_DONE_ADDR_AWM_REG0
Address:	1C6B00h-1C6B03h
Name:	AVP av1 unit done
ShortName:	AVP_UNIT_DONE_ADDR_AWM_REG1
Address:	1D2B00h-1D2B03h
Name:	AVP av1 unit done
ShortName:	AVP_UNIT_DONE_ADDR_AWM_REG2
Address:	1D6B00h-1D6B03h
Name:	AVP av1 unit done
ShortName:	AVP_UNIT_DONE_ADDR_AWM_REG3
Address:	1E2B00h-1E2B03h
Name:	AVP av1 unit done
ShortName:	AVP_UNIT_DONE_ADDR_AWM_REG4
Address:	1E6B00h-1E6B03h
Name:	AVP av1 unit done
ShortName:	AVP_UNIT_DONE_ADDR_AWM_REG5
Address:	1F2B00h-1F2B03h
Name:	AVP av1 unit done
ShortName:	AVP_UNIT_DONE_ADDR_AWM_REG6
Address:	1F6B00h-1F6B03h
Name:	AVP av1 unit done
ShortName:	AVP_UNIT_DONE_ADDR_AWM_REG7

This register stores all the done signals for AVP pipe.
"0" means unit is done and idle; "1" means unit is active.

DWord	Bit	Description
0	31:18	**Reserved** Access: RO Format: MBZ

AVP_UNIT_DONE_ADDR - AVP av1 unit done

	17	**APP Done Signal (Inverted)** Access: RO Format: U1
	16	**APR Done Signal (Inverted)** Access: RO Format: U1
	15	**AIT Done Signal (Inverted)** Access: RO Format: U1
	14	**AIQ Done Signal (Inverted)** Access: RO Format: U1
	13	**AMC Done Signal (Inverted)** Access: RO Format: U1
	12	**AED Done Signal (Inverted)** Access: RO Format: U1
	11	**AMX Done Signal (Inverted)** Access: RO Format: U1
	10	**ALF Done Signal (Inverted)** Access: RO Format: U1
	9	**ALN Done Signal (Inverted)** Access: RO Format: U1
	8:0	**Reserved** Access: RO Format: MBZ

Base of DMA Protected Range

BDPR - Base of DMA Protected Range

Register Space:	MMIO: 0/2/0
Size (in bits):	64
SOC_Consumer:	BIOS
Address:	108140h

This register indicates the Base of DMA Protected Range. It contains TSEGMB, lowered by the DPR size (if enabled).

DWord	Bit	Description
0..1	63:32	**BDPR_MSB** Default Value: 000h Access: RO _Custom_GTIReset: BUS 1MB aligned base of DMA Range.
	31:20	**BDPR_LSB** Default Value: 000h Access: RO _Custom_GTIReset: BUS 1MB aligned base of DMA Protected Memory Range.
	19:0	**Reserved** Access: RO Format: MBZ

Batch Address Difference Register

BB_ADDR_DIFF - Batch Address Difference Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02154h-02157h
Name:	Batch Address Difference Register
ShortName:	BB_ADDR_DIFF_RCSUNIT
Address:	18154h-18157h
Name:	Batch Address Difference Register
ShortName:	BB_ADDR_DIFF_POCSUNIT
Address:	22154h-22157h
Name:	Batch Address Difference Register
ShortName:	BB_ADDR_DIFF_BCSUNIT
Address:	1C0154h-1C0157h
Name:	Batch Address Difference Register
ShortName:	BB_ADDR_DIFF_VCSUNIT0
Address:	1C4154h-1C4157h
Name:	Batch Address Difference Register
ShortName:	BB_ADDR_DIFF_VCSUNIT1
Address:	1C8154h-1C8157h
Name:	Batch Address Difference Register
ShortName:	BB_ADDR_DIFF_VECSUNIT0
Address:	1D0154h-1D0157h
Name:	Batch Address Difference Register
ShortName:	BB_ADDR_DIFF_VCSUNIT2
Address:	1D4154h-1D4157h
Name:	Batch Address Difference Register
ShortName:	BB_ADDR_DIFF_VCSUNIT3
Address:	1D8154h-1D8157h
Name:	Batch Address Difference Register
ShortName:	BB_ADDR_DIFF_VECSUNIT1
Address:	1E0154h-1E0157h
Name:	Batch Address Difference Register
ShortName:	BB_ADDR_DIFF_VCSUNIT4

BB_ADDR_DIFF - Batch Address Difference Register

Address:	1E4154h-1E4157h
Name:	Batch Address Difference Register
ShortName:	BB_ADDR_DIFF_VCSUNIT5

Address:	1E8154h-1E8157h
Name:	Batch Address Difference Register
ShortName:	BB_ADDR_DIFF_VECSUNIT2

Address:	1F0154h-1F0157h
Name:	Batch Address Difference Register
ShortName:	BB_ADDR_DIFF_VCSUNIT6

Address:	1F4154h-1F4157h
Name:	Batch Address Difference Register
ShortName:	BB_ADDR_DIFF_VCSUNIT7

Address:	1F8154h-1F8157h
Name:	Batch Address Difference Register
ShortName:	BB_ADDR_DIFF_VECSUNIT3

Address:	1A154h-1A157h
Name:	Batch Address Difference Register
ShortName:	BB_ADDR_DIFF_CCSUNIT0

This register contains the difference between the start of the last batch and where the last initiated Batch Buffer is currently fetching commands.

Programming Notes

Programming Restriction:
This register should NEVER be programmed by driver, this is for HW internal use only.

DWord	Bit	Description
0	31:2	**Batch Buffer Address Difference** Format: GraphicsAddress[31:2] This field specifies the DWord-aligned difference between the starting address of the batch buffer and where the last initiated Batch Buffer is currently fetching commands.
	1:0	**Reserved** Access: RO Format: MBZ

Batch Buffer Head Pointer Preemption Register

BB_PREEMPT_ADDR - Batch Buffer Head Pointer Preemption Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02148h-0214Bh
Name:	Batch Buffer Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_RCSUNIT
Address:	18148h-1814Bh
Name:	Batch Buffer Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_POCSUNIT
Address:	22148h-2214Bh
Name:	Batch Buffer Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_BCSUNIT
Address:	1C0148h-1C014Bh
Name:	Batch Buffer Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_VCSUNIT0
Address:	1C4148h-1C414Bh
Name:	Batch Buffer Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_VCSUNIT1
Address:	1C8148h-1C814Bh
Name:	Batch Buffer Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_VECSUNIT0
Address:	1D0148h-1D014Bh
Name:	Batch Buffer Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_VCSUNIT2
Address:	1D4148h-1D414Bh
Name:	Batch Buffer Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_VCSUNIT3
Address:	1D8148h-1D814Bh
Name:	Batch Buffer Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_VECSUNIT1
Address:	1E0148h-1E014Bh

BB_PREEMPT_ADDR - Batch Buffer Head Pointer Preemption Register

Name:	Batch Buffer Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_VCSUNIT4

Address:	1E4148h-1E414Bh
Name:	Batch Buffer Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_VCSUNIT5

Address:	1E8148h-1E814Bh
Name:	Batch Buffer Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_VECSUNIT2

Address:	1F0148h-1F014Bh
Name:	Batch Buffer Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_VCSUNIT6

Address:	1F4148h-1F414Bh
Name:	Batch Buffer Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_VCSUNIT7

Address:	1F8148h-1F814Bh
Name:	Batch Buffer Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_VECSUNIT3

Address:	1A148h-1A14Bh
Name:	Batch Buffer Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_CCSUNIT0

This register gets updated with the DWord-aligned graphics memory address of the PREEMPTABLE command in the batch buffer on which preemption has occurred.
This register gets updated with the DWord-aligned graphics memory address of the command following the MI_BATCH_START corresponding to the second level batch buffer, when the preemption has occurred in the second level batch buffer.
This register value should be looked at only when the preemption has occurred in the batch buffer. This is indicated by "Ring/Batch Indicator" in "RING_BUFFER_HEAD_PREEMPT_REG". This register value retains its previous value and doesn't change when the preemption occurs on a preemptable command in ring buffer.
Preemption is triggered by valid UHPTR in ring buffer mode of scheduling and by a pending execlist in Exec-List mode of scheduling.
This is a global register and context save/restored as part of power context image.

Preemptable Commands	Source
MI_ARB_CHECK 3D_PRIMITIVE GPGPU_WALKER MEDIA_STATE_FLUSH	RenderCS

BB_PREEMPT_ADDR - Batch Buffer Head Pointer Preemption Register

PIPE_CONTROL (Only in GPGPU mode of pipeline selection) MI_ATOMIC (Post Sync Operation set in GPGPU mode of pipeline selection) MI_SEMAPHORE_SIGNAL (Post Sync Operation set in GPGPU mode of pipeline selection)	

Preemptable Commands	Source
MI_ARB_CHECK	BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS

Programming Notes

Programming Restriction: This register should NEVER be programmed by driver, this is for HW internal use only.

DWord	Bit	Description
0	31:2	**Batch Buffer Head Pointer** Format: GraphicsAddress[31:2] This field specifies the DWord-aligned Graphics Memory Address of the PREEMPTABLE commandin a batch buffer where the Preemption has occurred.
	1:0	**Reserved** Access: RO Format: MBZ

Batch Buffer Head Pointer Register

BB_ADDR - Batch Buffer Head Pointer Register	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02140h-02143h
Name:	Batch Buffer Head Pointer Register
ShortName:	BB_ADDR_RCSUNIT
Address:	18140h-18143h
Name:	Batch Buffer Head Pointer Register
ShortName:	BB_ADDR_POCSUNIT
Address:	22140h-22143h
Name:	Batch Buffer Head Pointer Register
ShortName:	BB_ADDR_BCSUNIT
Address:	1C0140h-1C0143h
Name:	Batch Buffer Head Pointer Register
ShortName:	BB_ADDR_VCSUNIT0
Address:	1C4140h-1C4143h
Name:	Batch Buffer Head Pointer Register
ShortName:	BB_ADDR_VCSUNIT1
Address:	1C8140h-1C8143h
Name:	Batch Buffer Head Pointer Register
ShortName:	BB_ADDR_VECSUNIT0
Address:	1D0140h-1D0143h
Name:	Batch Buffer Head Pointer Register
ShortName:	BB_ADDR_VCSUNIT2
Address:	1D4140h-1D4143h
Name:	Batch Buffer Head Pointer Register
ShortName:	BB_ADDR_VCSUNIT3
Address:	1D8140h-1D8143h
Name:	Batch Buffer Head Pointer Register
ShortName:	BB_ADDR_VECSUNIT1
Address:	1E0140h-1E0143h
Name:	Batch Buffer Head Pointer Register

BB_ADDR - Batch Buffer Head Pointer Register

ShortName:	BB_ADDR_VCSUNIT4
Address:	1E4140h-1E4143h
Name:	Batch Buffer Head Pointer Register
ShortName:	BB_ADDR_VCSUNIT5
Address:	1E8140h-1E8143h
Name:	Batch Buffer Head Pointer Register
ShortName:	BB_ADDR_VECSUNIT2
Address:	1F0140h-1F0143h
Name:	Batch Buffer Head Pointer Register
ShortName:	BB_ADDR_VCSUNIT6
Address:	1F4140h-1F4143h
Name:	Batch Buffer Head Pointer Register
ShortName:	BB_ADDR_VCSUNIT7
Address:	1F8140h-1F8143h
Name:	Batch Buffer Head Pointer Register
ShortName:	BB_ADDR_VECSUNIT3
Address:	1A140h-1A143h
Name:	Batch Buffer Head Pointer Register
ShortName:	BB_ADDR_CCSUNIT0

Description

This field specifies the DWord-aligned Graphics Memory Address of commands being fetched for the most recently initiated batch buffer. This register has valid values only when the Valid bit is set to0. Level of the batch buffer is indicated based on the Batch Buffer Stack Pointer value in BB_STATE register.

- Stack Pointer holding a value 0 indicates First Level batch buffer.
- Stack Pointer holding a value 1 indicates Second Level batch buffer.
- Stack Pointer holding a value 2 indicates Third Level batch buffer.

Programming Notes

Programming Restriction: This register should NEVER be programmed by driver. This is for HW internal use only.

DWord	Bit	Description
0	31:2	**Batch Buffer Head Pointer** Format: GraphicsAddress[31:2] This field specifies the DWord-aligned Graphics Memory Address of commands being fetched for the most recently initiated batch buffer. This register have valid values only when the Valid bit is set to0. Level of the batch buffer is indicated based on the Batch Buffer Stack Pointer value in BB_STATE register.

BB_ADDR - Batch Buffer Head Pointer Register

- Stack Pointer holding a value 0 indicates First Level batch buffer.
- Stack Pointer holding a value 1 indicates Second Level batch buffer.
- Stack Pointer holding a value 2 indicates Third Level batch buffer.

1 Reserved

Access:	RO
Format:	MBZ

0 Valid

Format:	U1

Value	Name	Description
0h	Invalid **[Default]**	Batch buffer Invalid
1h	Valid	Batch buffer Valid

Batch Buffer Per Context Pointer

BB_PER_CTX_PTR - Batch Buffer Per Context Pointer	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	021C0h-021C3h
Name:	Batch Buffer Per Context Pointer
ShortName:	BB_PER_CTX_PTR_RCSUNIT
Address:	181C0h-181C3h
Name:	Batch Buffer Per Context Pointer
ShortName:	BB_PER_CTX_PTR_POCSUNIT
Address:	221C0h-221C3h
Name:	Batch Buffer Per Context Pointer
ShortName:	BB_PER_CTX_PTR_BCSUNIT
Address:	1C01C0h-1C01C3h
Name:	Batch Buffer Per Context Pointer
ShortName:	BB_PER_CTX_PTR_VCSUNIT0
Address:	1C41C0h-1C41C3h
Name:	Batch Buffer Per Context Pointer
ShortName:	BB_PER_CTX_PTR_VCSUNIT1
Address:	1C81C0h-1C81C3h
Name:	Batch Buffer Per Context Pointer
ShortName:	BB_PER_CTX_PTR_VECSUNIT0
Address:	1D01C0h-1D01C3h
Name:	Batch Buffer Per Context Pointer
ShortName:	BB_PER_CTX_PTR_VCSUNIT2
Address:	1D41C0h-1D41C3h
Name:	Batch Buffer Per Context Pointer
ShortName:	BB_PER_CTX_PTR_VCSUNIT3
Address:	1D81C0h-1D81C3h
Name:	Batch Buffer Per Context Pointer
ShortName:	BB_PER_CTX_PTR_VECSUNIT1
Address:	1E01C0h-1E01C3h
Name:	Batch Buffer Per Context Pointer

BB_PER_CTX_PTR - Batch Buffer Per Context Pointer

ShortName:	BB_PER_CTX_PTR_VCSUNIT4
Address:	1E41C0h-1E41C3h
Name:	Batch Buffer Per Context Pointer
ShortName:	BB_PER_CTX_PTR_VCSUNIT5
Address:	1E81C0h-1E81C3h
Name:	Batch Buffer Per Context Pointer
ShortName:	BB_PER_CTX_PTR_VECSUNIT2
Address:	1F01C0h-1F01C3h
Name:	Batch Buffer Per Context Pointer
ShortName:	BB_PER_CTX_PTR_VCSUNIT6
Address:	1F41C0h-1F41C3h
Name:	Batch Buffer Per Context Pointer
ShortName:	BB_PER_CTX_PTR_VCSUNIT7
Address:	1F81C0h-1F81C3h
Name:	Batch Buffer Per Context Pointer
ShortName:	BB_PER_CTX_PTR_VECSUNIT3
Address:	1A1C0h-1A1C3h
Name:	Batch Buffer Per Context Pointer
ShortName:	BB_PER_CTX_PTR_CCSUNIT0

This register is used to program the batch buffer address to be executed between context restore and execution of ring/execution list if enabled. This will only get executed due to regular context save/restore and not during power restore. This register is part of the execution list context and will be executed per context. Only supported if execution list is enabled. There is no preempting workloads within the Per Context Batch Buffer.

Programming Notes

BlitterCS/VideoCS/VideoCS2/VideoEnhancementCS: This register functionality is not supported and must not be programmed for these command streamers.

Render CS: Per Context Batch Buffer execution must not look at the MI_RS_CONTROL or Wait For Event status that are restored for the corresponding context. Ex: A context with MI_RS_CONTROL status with RS disabled doesn't stop Render CS from triggering Resource Streamer to execute Per Context Batch Buffer when "**RS Enabled Batch Buffer Per Context**" is set.

RenderCS: The following commands are not supported within a Per Context Batch Buffer:

Command Name
MI_WAIT_FOR_EVENT
MI_ARB_CHECK
MI_REPORT_HEAD
MI_URB_ATOMIC_ALLOC
MI_SUSPEND_FLUSH

BB_PER_CTX_PTR - Batch Buffer Per Context Pointer

MI_TOPOLOGY_FILTER
MI_SET_CONTEXT
MI_URB_CLEAR
MI_SEMAPHORE_WAIT (Memory Poll Mode). Note: MI_SEMAPHORE_WAIT in register poll mode is supported.
MI_SEMAPHORE_SIGNAL
MI_BATCH_BUFFER_START
MI_CONDITIONAL_BATCH_BUFFER_END
MEDIA_OBJECT_WALKER
GPGPU_WALKER
3DPRIMITIVE
3DSTATE_BINDING_TABLE_POINTERS_VS
3DSTATE_BINDING_TABLE_POINTERS_HS
3DSTATE_BINDING_TABLE_POINTERS_DS
3DSTATE_BINDING_TABLE_POINTERS_GS
3DSTATE_BINDING_TABLE_POINTERS_PS
3DSTATE_CONSTANT_VS
3DSTATE_CONSTANT_GS
3DSTATE_CONSTANT_PS
3DSTATE_CONSTANT_HS
3DSTATE_CONSTANT_DS
PIPECONTROL

DWord	Bit	Description
0	31:12	**Batch Buffer Per Context Address** Format: U20 Pointer to the Context in memory to be executed as a batch.
	11:3	**Reserved** Access: RO Format: MBZ
	2	**FORCE BB_PER_CTX_PTR** On detecting a context restore (not lite restore) with head pointer equals to tail pointer, command stream optimizes context switch process by not doing engine context restore and context save for the corresponding context. As part of this optimization command stream doesn't execute batch buffer per context pointer (BB_PER_CTX_PTR). Setting this bit allows command stream to execute BB_PER_CTX_PT even on context restore flows with head pointer equals to tail pointer. **Value** / **Name** / **Description** 0 / **[Default]** / Command stream does not execute BB_PER_CTX_PTR on context restore with head pointer equals to tail pointer.

BB_PER_CTX_PTR - Batch Buffer Per Context Pointer

	1	Command stream does execute BB_PER_CTX_PTR on context restore with head pointer equals to tail pointer.

Reserved

Access:	RO
Format:	MBZ

Batch Buffer Per Context Valid

Format:	U1

If set, the command stream will execute the context from the **Batch Buffer Per Context Address** prior to the execution of actual submitted workloads.

Batch Buffer Stack Write Port

BB_STACK_WRITE_PORT - Batch Buffer Stack Write Port	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02588h-0258Bh
Name:	BB_STACK_WRITE_PORT
ShortName:	BB_STACK_WRITE_PORT_RCSUNIT
Address:	18588h-1858Bh
Name:	BB_STACK_WRITE_PORT
ShortName:	BB_STACK_WRITE_PORT_POCSUNIT
Address:	22588h-2258Bh
Name:	BB_STACK_WRITE_PORT
ShortName:	BB_STACK_WRITE_PORT_BCSUNIT
Address:	1C0588h-1C058Bh
Name:	BB_STACK_WRITE_PORT
ShortName:	BB_STACK_WRITE_PORT_VCSUNIT0
Address:	1C4588h-1C458Bh
Name:	BB_STACK_WRITE_PORT
ShortName:	BB_STACK_WRITE_PORT_VCSUNIT1
Address:	1C8588h-1C858Bh
Name:	BB_STACK_WRITE_PORT
ShortName:	BB_STACK_WRITE_PORT_VECSUNIT0
Address:	1D0588h-1D058Bh
Name:	BB_STACK_WRITE_PORT
ShortName:	BB_STACK_WRITE_PORT_VCSUNIT2
Address:	1D4588h-1D458Bh
Name:	BB_STACK_WRITE_PORT
ShortName:	BB_STACK_WRITE_PORT_VCSUNIT3
Address:	1D8588h-1D858Bh
Name:	BB_STACK_WRITE_PORT
ShortName:	BB_STACK_WRITE_PORT_VECSUNIT1
Address:	1E0588h-1E058Bh
Name:	BB_STACK_WRITE_PORT

BB_STACK_WRITE_PORT - Batch Buffer Stack Write Port

ShortName:	BB_STACK_WRITE_PORT_VCSUNIT4
Address:	1E4588h-1E458Bh
Name:	BB_STACK_WRITE_PORT
ShortName:	BB_STACK_WRITE_PORT_VCSUNIT5
Address:	1E8588h-1E858Bh
Name:	BB_STACK_WRITE_PORT
ShortName:	BB_STACK_WRITE_PORT_VECSUNIT2
Address:	1F0588h-1F058Bh
Name:	BB_STACK_WRITE_PORT
ShortName:	BB_STACK_WRITE_PORT_VCSUNIT6
Address:	1F4588h-1F458Bh
Name:	BB_STACK_WRITE_PORT
ShortName:	BB_STACK_WRITE_PORT_VCSUNIT7
Address:	1F8588h-1F858Bh
Name:	BB_STACK_WRITE_PORT
ShortName:	BB_STACK_WRITE_PORT_VECSUNIT3
Address:	1A588h-1A58Bh
Name:	BB_STACK_WRITE_PORT
ShortName:	BB_STACK_WRITE_PORT_CCSUNIT0

DWord	Bit	Description
0	31:0	**Batch Buffer Stack Pointer** Format: U32 This register provides a mechanism to update the entries of the Batch Buffer Stack in hardware. Consecutive writes to this register results in updating consecutive entries of the Batch Buffer Stack, starts fromEntry0(bottom of the stack) of the stack, reaches the last entry of the stack (top of the stack) and wraps around to Entry0. Note that each entry of the stack is a qword and two consecutive MMIO writes are required to update an entry of a stack. This register must not be written by SW and is only meant for hardware internal usage to context save/restore batch buffer stack values. BATCH_BUFFER_STACK_STRUCTURE defines the structure of the batch buffer stack implemented in hardware.

Batch Buffer Start Head Pointer Register

BB_START_ADDR - Batch Buffer Start Head Pointer Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02150h-02153h
Name:	Batch Buffer Start Head Pointer Register
ShortName:	BB_START_ADDR_RCSUNIT
Address:	18150h-18153h
Name:	Batch Buffer Start Head Pointer Register
ShortName:	BB_START_ADDR_POCSUNIT
Address:	22150h-22153h
Name:	Batch Buffer Start Head Pointer Register
ShortName:	BB_START_ADDR_BCSUNIT
Address:	1C0150h-1C0153h
Name:	Batch Buffer Start Head Pointer Register
ShortName:	BB_START_ADDR_VCSUNIT0
Address:	1C4150h-1C4153h
Name:	Batch Buffer Start Head Pointer Register
ShortName:	BB_START_ADDR_VCSUNIT1
Address:	1C8150h-1C8153h
Name:	Batch Buffer Start Head Pointer Register
ShortName:	BB_START_ADDR_VECSUNIT0
Address:	1D0150h-1D0153h
Name:	Batch Buffer Start Head Pointer Register
ShortName:	BB_START_ADDR_VCSUNIT2
Address:	1D4150h-1D4153h
Name:	Batch Buffer Start Head Pointer Register
ShortName:	BB_START_ADDR_VCSUNIT3
Address:	1D8150h-1D8153h
Name:	Batch Buffer Start Head Pointer Register
ShortName:	BB_START_ADDR_VECSUNIT1
Address:	1E0150h-1E0153h
Name:	Batch Buffer Start Head Pointer Register

BB_START_ADDR - Batch Buffer Start Head Pointer Register

ShortName:	BB_START_ADDR_VCSUNIT4
Address:	1E4150h-1E4153h
Name:	Batch Buffer Start Head Pointer Register
ShortName:	BB_START_ADDR_VCSUNIT5
Address:	1E8150h-1E8153h
Name:	Batch Buffer Start Head Pointer Register
ShortName:	BB_START_ADDR_VECSUNIT2
Address:	1F0150h-1F0153h
Name:	Batch Buffer Start Head Pointer Register
ShortName:	BB_START_ADDR_VCSUNIT6
Address:	1F4150h-1F4153h
Name:	Batch Buffer Start Head Pointer Register
ShortName:	BB_START_ADDR_VCSUNIT7
Address:	1F8150h-1F8153h
Name:	Batch Buffer Start Head Pointer Register
ShortName:	BB_START_ADDR_VECSUNIT3
Address:	1A150h-1A153h
Name:	Batch Buffer Start Head Pointer Register
ShortName:	BB_START_ADDR_CCSUNIT0

This register contains the address specified in the last MI_BATCH_BUFFER_START command executed for the first level batch buffer or chained first level batch buffer.

Programming Notes

Programming Restriction: This register should NEVER be programmed by driver, this is for HW internal use only.

DWord	Bit	Description
0	31:2	**Batch Buffer Start Head Pointer** Format: GraphicsAddress[31:2] This field specifies the DWord-aligned Graphics Memory Address where the last initiated Batch Buffer starting address.
	1:0	**Reserved** Access: RO Format: MBZ

Batch Buffer Start Upper Head Pointer Register

BB_START_ADDR_UDW - Batch Buffer Start Upper Head Pointer Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02170h-02173h
Name:	Batch Buffer Start Upper Head Pointer Register
ShortName:	BB_START_ADDR_UDW_RCSUNIT
Address:	18170h-18173h
Name:	Batch Buffer Start Upper Head Pointer Register
ShortName:	BB_START_ADDR_UDW_POCSUNIT
Address:	22170h-22173h
Name:	Batch Buffer Start Upper Head Pointer Register
ShortName:	BB_START_ADDR_UDW_BCSUNIT
Address:	1C0170h-1C0173h
Name:	Batch Buffer Start Upper Head Pointer Register
ShortName:	BB_START_ADDR_UDW_VCSUNIT0
Address:	1C4170h-1C4173h
Name:	Batch Buffer Start Upper Head Pointer Register
ShortName:	BB_START_ADDR_UDW_VCSUNIT1
Address:	1C8170h-1C8173h
Name:	Batch Buffer Start Upper Head Pointer Register
ShortName:	BB_START_ADDR_UDW_VECSUNIT0
Address:	1D0170h-1D0173h
Name:	Batch Buffer Start Upper Head Pointer Register
ShortName:	BB_START_ADDR_UDW_VCSUNIT2
Address:	1D4170h-1D4173h
Name:	Batch Buffer Start Upper Head Pointer Register
ShortName:	BB_START_ADDR_UDW_VCSUNIT3
Address:	1D8170h-1D8173h
Name:	Batch Buffer Start Upper Head Pointer Register
ShortName:	BB_START_ADDR_UDW_VECSUNIT1
Address:	1E0170h-1E0173h

BB_START_ADDR_UDW - Batch Buffer Start Upper Head Pointer Register

Name:	Batch Buffer Start Upper Head Pointer Register
ShortName:	BB_START_ADDR_UDW_VCSUNIT4

Address:	1E4170h-1E4173h
Name:	Batch Buffer Start Upper Head Pointer Register
ShortName:	BB_START_ADDR_UDW_VCSUNIT5

Address:	1E8170h-1E8173h
Name:	Batch Buffer Start Upper Head Pointer Register
ShortName:	BB_START_ADDR_UDW_VECSUNIT2

Address:	1F0170h-1F0173h
Name:	Batch Buffer Start Upper Head Pointer Register
ShortName:	BB_START_ADDR_UDW_VCSUNIT6

Address:	1F4170h-1F4173h
Name:	Batch Buffer Start Upper Head Pointer Register
ShortName:	BB_START_ADDR_UDW_VCSUNIT7

Address:	1F8170h-1F8173h
Name:	Batch Buffer Start Upper Head Pointer Register
ShortName:	BB_START_ADDR_UDW_VECSUNIT3

Address:	1A170h-1A173h
Name:	Batch Buffer Start Upper Head Pointer Register
ShortName:	BB_START_ADDR_UDW_CCSUNIT0

This register contains the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space specified in the last MI_START_BATCH_BUFFER command.

Programming Notes

Programming Restriction: This register should NEVER be programmed by driver, this is for HW internal use only.

DWord	Bit	Description
0	31:25	**Reserved** Access: RO Format: MBZ
	24:16	**Reserved** Access: RO Format: MBZ
	15:0	**Head Pointer Upper DWORD** Format: GraphicsAddress[47:32]

Batch Buffer State Register

BB_STATE - Batch Buffer State Register	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02110h-02113h
Name:	Batch Buffer State Register
ShortName:	BB_STATE_RCSUNIT
Address:	18110h-18113h
Name:	Batch Buffer State Register
ShortName:	BB_STATE_POCSUNIT
Address:	22110h-22113h
Name:	Batch Buffer State Register
ShortName:	BB_STATE_BCSUNIT
Address:	1C0110h-1C0113h
Name:	Batch Buffer State Register
ShortName:	BB_STATE_VCSUNIT0
Address:	1C4110h-1C4113h
Name:	Batch Buffer State Register
ShortName:	BB_STATE_VCSUNIT1
Address:	1C8110h-1C8113h
Name:	Batch Buffer State Register
ShortName:	BB_STATE_VECSUNIT0
Address:	1D0110h-1D0113h
Name:	Batch Buffer State Register
ShortName:	BB_STATE_VCSUNIT2
Address:	1D4110h-1D4113h
Name:	Batch Buffer State Register
ShortName:	BB_STATE_VCSUNIT3
Address:	1D8110h-1D8113h
Name:	Batch Buffer State Register
ShortName:	BB_STATE_VECSUNIT1
Address:	1E0110h-1E0113h
Name:	Batch Buffer State Register

BB_STATE - Batch Buffer State Register

ShortName:	BB_STATE_VCSUNIT4
Address:	1E4110h-1E4113h
Name:	Batch Buffer State Register
ShortName:	BB_STATE_VCSUNIT5
Address:	1E8110h-1E8113h
Name:	Batch Buffer State Register
ShortName:	BB_STATE_VECSUNIT2
Address:	1F0110h-1F0113h
Name:	Batch Buffer State Register
ShortName:	BB_STATE_VCSUNIT6
Address:	1F4110h-1F4113h
Name:	Batch Buffer State Register
ShortName:	BB_STATE_VCSUNIT7
Address:	1F8110h-1F8113h
Name:	Batch Buffer State Register
ShortName:	BB_STATE_VECSUNIT3
Address:	1A110h-1A113h
Name:	Batch Buffer State Register
ShortName:	BB_STATE_CCSUNIT0

Description

This register specifies the state of the most recently executed batch buffer. Contents of this register are only valid when Valid bit in BB_ADDR register is set.
Level of the batch buffer is indicated by the Batch Buffer Stack Pointer value in BB_STATE register.
Stack Pointer holding a value 0 indicates First Level batch buffer.
Stack Pointer holding a value 1 indicates Second Level batch buffer.
Stack Pointer holding a value 2 indicates Third Level batch buffer.

This register should not be written by software. These fields should only get written by a context restore. Software should always set these fields via the MI_BATCH_BUFFER_START command when initiating a batch buffer. This register is saved and restored with context.

Programming Notes

Contents of this register are valid only when "Valid" bit in BB_ADDR register is set.

DWord	Bit	Description
0	31:10	**Reserved** Access: RO Format: MBZ

BB_STATE - Batch Buffer State Register

9 — POSH Start

Exists If:	//RCS, POCS

This bit reflects the POSH Start value programmed by the active first level MI_BATCH_BUFFER_START command.

8 — POSH Enable

Exists If:	//RCS, POCS

This bit reflects the POSH Enable value programmed by the active first level MI_BATCH_BUFFER_START command.

7 — Reserved

Access:	RO
Format:	MBZ

6 — Clear Command Buffer Enable

Source:	RenderCS
Format:	U1

5 — Address Space Indicator

Note: This field reflects the effective address space indicator security level and may not be the same as the Address Space Indicator written using MI_BATCH_BUFFER_START.

Value	Name	Description
0h	GGTT **[Default]**	This Batch buffer is located in GGTT memory and is privileged
1h	PPGTT	This Batch buffer is located in PPGTT memory and is non-privileged.

4 — Reserved

3:2 — Reserved

Access:	RO
Format:	MBZ

1:0 — Batch Buffer Stack Pointer

Format:	U2

This field holds the index of the stack entry which got recently updated on executing a next level batch buffer (stack push). This index points to the entry on top of the stack. Stack pointer having a value of '00b indicates stack empty and a value of '11b indicates stack full.

Value	Name	Description
0h		Stack has no data and is empty.
1h		Stack has one valid entry and have first level batch buffer details.
2h		Stack has two valid entries and have first and second level batch buffer details.
3h		Stack has three valid entries and have first, second and third level batch buffer details.

Batch Buffer Upper Head Pointer Preemption Register

BB_PREEMPT_ADDR_UDW - Batch Buffer Upper Head Pointer Preemption Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0216Ch-0216Fh
Name:	Batch Buffer Upper Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_UDW_RCSUNIT
Address:	1816Ch-1816Fh
Name:	Batch Buffer Upper Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_UDW_POCSUNIT
Address:	2216Ch-2216Fh
Name:	Batch Buffer Upper Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_UDW_BCSUNIT
Address:	1C016Ch-1C016Fh
Name:	Batch Buffer Upper Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_UDW_VCSUNIT0
Address:	1C416Ch-1C416Fh
Name:	Batch Buffer Upper Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_UDW_VCSUNIT1
Address:	1C816Ch-1C816Fh
Name:	Batch Buffer Upper Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_UDW_VECSUNIT0
Address:	1D016Ch-1D016Fh
Name:	Batch Buffer Upper Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_UDW_VCSUNIT2
Address:	1D416Ch-1D416Fh
Name:	Batch Buffer Upper Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_UDW_VCSUNIT3
Address:	1D816Ch-1D816Fh
Name:	Batch Buffer Upper Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_UDW_VECSUNIT1
Address:	1E016Ch-1E016Fh

BB_PREEMPT_ADDR_UDW - Batch Buffer Upper Head Pointer Preemption Register

Name:	Batch Buffer Upper Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_UDW_VCSUNIT4
Address:	1E416Ch-1E416Fh
Name:	Batch Buffer Upper Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_UDW_VCSUNIT5
Address:	1E816Ch-1E816Fh
Name:	Batch Buffer Upper Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_UDW_VECSUNIT2
Address:	1F016Ch-1F016Fh
Name:	Batch Buffer Upper Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_UDW_VCSUNIT6
Address:	1F416Ch-1F416Fh
Name:	Batch Buffer Upper Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_UDW_VCSUNIT7
Address:	1F816Ch-1F816Fh
Name:	Batch Buffer Upper Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_UDW_VECSUNIT3
Address:	1A16Ch-1A16Fh
Name:	Batch Buffer Upper Head Pointer Preemption Register
ShortName:	BB_PREEMPT_ADDR_UDW_CCSUNIT0

This register contains the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space of the last preempted batch buffer. This register follows the same rules as the BB_PREEMPT_ADDR register.

Programming Notes

Programming Restriction: This register should NEVER be programmed by driver, this is for HW internal use only.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Batch Buffer Head Pointer Upper DWORD** Format: GraphicsAddress[47:32] This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space of the last preempted batch buffer.

Batch Buffer Upper Head Pointer Register

BB_ADDR_UDW - Batch Buffer Upper Head Pointer Register	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02168h-0216Bh
Name:	Batch Buffer Upper Head Pointer Register
ShortName:	BB_ADDR_UDW_RCSUNIT
Address:	18168h-1816Bh
Name:	Batch Buffer Upper Head Pointer Register
ShortName:	BB_ADDR_UDW_POCSUNIT
Address:	22168h-2216Bh
Name:	Batch Buffer Upper Head Pointer Register
ShortName:	BB_ADDR_UDW_BCSUNIT
Address:	1C0168h-1C016Bh
Name:	Batch Buffer Upper Head Pointer Register
ShortName:	BB_ADDR_UDW_VCSUNIT0
Address:	1C4168h-1C416Bh
Name:	Batch Buffer Upper Head Pointer Register
ShortName:	BB_ADDR_UDW_VCSUNIT1
Address:	1C8168h-1C816Bh
Name:	Batch Buffer Upper Head Pointer Register
ShortName:	BB_ADDR_UDW_VECSUNIT0
Address:	1D0168h-1D016Bh
Name:	Batch Buffer Upper Head Pointer Register
ShortName:	BB_ADDR_UDW_VCSUNIT2
Address:	1D4168h-1D416Bh
Name:	Batch Buffer Upper Head Pointer Register
ShortName:	BB_ADDR_UDW_VCSUNIT3
Address:	1D8168h-1D816Bh
Name:	Batch Buffer Upper Head Pointer Register
ShortName:	BB_ADDR_UDW_VECSUNIT1
Address:	1E0168h-1E016Bh
Name:	Batch Buffer Upper Head Pointer Register

BB_ADDR_UDW - Batch Buffer Upper Head Pointer Register

ShortName:	BB_ADDR_UDW_VCSUNIT4
Address:	1E4168h-1E416Bh
Name:	Batch Buffer Upper Head Pointer Register
ShortName:	BB_ADDR_UDW_VCSUNIT5
Address:	1E8168h-1E816Bh
Name:	Batch Buffer Upper Head Pointer Register
ShortName:	BB_ADDR_UDW_VECSUNIT2
Address:	1F0168h-1F016Bh
Name:	Batch Buffer Upper Head Pointer Register
ShortName:	BB_ADDR_UDW_VCSUNIT6
Address:	1F4168h-1F416Bh
Name:	Batch Buffer Upper Head Pointer Register
ShortName:	BB_ADDR_UDW_VCSUNIT7
Address:	1F8168h-1F816Bh
Name:	Batch Buffer Upper Head Pointer Register
ShortName:	BB_ADDR_UDW_VECSUNIT3
Address:	1A168h-1A16Bh
Name:	Batch Buffer Upper Head Pointer Register
ShortName:	BB_ADDR_UDW_CCSUNIT0

Description

This register specifies the upper 32 bits of the 4GB aligned base address, within the 64-bit host virtual address space of the commands being fetched from the most recently initiated batch buffer. This register have valid values only when the Valid bit in BB_ADDR register is set to1. Level of the batch buffer is indicated based on the Batch Buffer Stack Pointer value in BB_STATE register. GraphicsAddress is 64-bit value [63:0], but only a portion of it is used by hardware. The uppermost reserved bits are ignored and MBZ.
Stack Pointer holding a value 0 indicates First Level batch buffer.
Stack Pointer holding a value 1 indicates Second Level batch buffer.
Stack Pointer holding a value 2 indicates Third Level batch buffer.

Programming Notes

This register should NEVER be programmed by driver. This is for HW internal use only.

DWord	Bit	Description
0	31:25	**Reserved** Access: RO Format: MBZ
	24:16	**Reserved** Access: RO Format: MBZ

BB_ADDR_UDW - Batch Buffer Upper Head Pointer Register		
	15:0	**Batch Buffer Head Pointer Upper DWORD** Format: GraphicsAddress[47:32]

Batch Offset Register

BB_OFFSET - Batch Offset Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02158h-0215Bh
Name:	Batch Offset Register
ShortName:	BB_OFFSET_RCSUNIT
Address:	18158h-1815Bh
Name:	Batch Offset Register
ShortName:	BB_OFFSET_POCSUNIT
Address:	22158h-2215Bh
Name:	Batch Offset Register
ShortName:	BB_OFFSET_BCSUNIT
Address:	1C0158h-1C015Bh
Name:	Batch Offset Register
ShortName:	BB_OFFSET_VCSUNIT0
Address:	1C4158h-1C415Bh
Name:	Batch Offset Register
ShortName:	BB_OFFSET_VCSUNIT1
Address:	1C8158h-1C815Bh
Name:	Batch Offset Register
ShortName:	BB_OFFSET_VECSUNIT0
Address:	1D0158h-1D015Bh
Name:	Batch Offset Register
ShortName:	BB_OFFSET_VCSUNIT2
Address:	1D4158h-1D415Bh
Name:	Batch Offset Register
ShortName:	BB_OFFSET_VCSUNIT3
Address:	1D8158h-1D815Bh
Name:	Batch Offset Register
ShortName:	BB_OFFSET_VECSUNIT1
Address:	1E0158h-1E015Bh
Name:	Batch Offset Register

BB_OFFSET - Batch Offset Register

ShortName:	BB_OFFSET_VCSUNIT4
Address:	1E4158h-1E415Bh
Name:	Batch Offset Register
ShortName:	BB_OFFSET_VCSUNIT5
Address:	1E8158h-1E815Bh
Name:	Batch Offset Register
ShortName:	BB_OFFSET_VECSUNIT2
Address:	1F0158h-1F015Bh
Name:	Batch Offset Register
ShortName:	BB_OFFSET_VCSUNIT6
Address:	1F4158h-1F415Bh
Name:	Batch Offset Register
ShortName:	BB_OFFSET_VCSUNIT7
Address:	1F8158h-1F815Bh
Name:	Batch Offset Register
ShortName:	BB_OFFSET_VECSUNIT3
Address:	1A158h-1A15Bh
Name:	Batch Offset Register
ShortName:	BB_OFFSET_CCSUNIT0

This register contains the offset value to be added to the Batch Buffer Start Address in the MI_BATCH_BUFFER_START command when the Enable Offset bit in MI_BATCH_BUFFER_START command is set.

Preemptable Commands	Source
• MI_ARB_CHECK • 3D_PRIMITIVE • GPGPU_WALKER • MEDIA_STATE_FLUSH • PIPE_CONTROL (Only in GPGPU mode of pipeline selection) • MI_ATOMIC (Post Sync Operation set in GPGPU mode of pipeline selection) • MI_SEMAPHORE_SIGNAL (Post Sync Operation set in GPGPU mode of pipeline selection)	RenderCS

Preemptable Commands	Source
MI_ARB_CHECK	BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS

Programming Notes

On preemption occurring within a primary/chain batch buffer this register is loaded with the offset value of the preempted command header from the batch start address when the Enable Load is set. Preemption of 3D or

BB_OFFSET - Batch Offset Register

GP_GPU workloads can only occur on preemptable commands. Batch buffer offset always points to the preemptable command if preempted on preemption or the immediate command following it if not preempted on preemption. EX: Preemption occurs on 3D_PRIMITVE command

- If the 3D_PRIMTIVE command is completely processed by render pipe then the BB_OFFSET points to the command following 3D_PRIMITIVE
- If the 3D_PRIMTIVE command is not completely processed by render pipe then the BB_OFFSET points to the 3D_PRIMITIVE command.

DWord	Bit	Description
0	31:2	**Batch Buffer Offset** Format: GraphicsAddress[31:2] This field specifies the DWord-aligned offset between the starting address of the batch buffer and where the last initiated Batch Buffer is currently fetching commands.
	1	**Reserved** Access: RO Format: MBZ
	0	**Enable Load** Default Value: 1 Format: Enable If this bit is set then the Batch Buffer Offset is loaded with the preempted command offset or the following command whenever a batch buffer is ended due to a Preemptable command.

BCS Context Sizes

BCS_CXT_SIZE - BCS Context Sizes			
Register Space:	MMIO: 0/2/0		
Access:	Read/32 bit Write Only		
Size (in bits):	32		
Address:	221A8h		
DWord	**Bit**	**Description**	
0	31:13	**Reserved**	
		Access:	RO
		Format:	MBZ
	12:8	**BCS Context Size**	
		Format:	U5
	7:5	**Reserved**	
		Access:	RO
		Format:	MBZ
	4:0	**Execlist Context Size**	
		Format:	U5

BCS CSB

BCS_CSB - BCS CSB		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
SW reads this offset to read the Context Status Buffer entry at the top of the CSB FIFO. Reads must occur in pairs to obtain a single 64 bit CSB entry. The second read of a pair pops the CSB entry off the CSB fifo.		
DWord	**Bit**	**Description**
0	31:0	**Context Status Buffer DW** Access: RO This DW holds CSB bits[31:0] for the first read and CSB bits[63:32] for the second read.

BCS CSB Fifo Status Register

BCS_CSB_FSR - BCS CSB Fifo Status Register

Register Space: MMIO: 0/2/0

Size (in bits): 32

This RO register holds status of the CSB fifo.

DWord	Bit	Description
0	31	**Not Empty** Access: RO
	30:16	**Reserved** Access: RO Format: MBZ
	15:8	**FIFO Maximum Occupancy Count** This field is a read-only field. It reflects the depth of the FIFO, which is a static value for each product.
	7:0	**FIFO Occupancy Count** Access: RO This is a second bit field in a register template

BCS Ring Buffer Next Context ID Register

BCS_RNCID - BCS Ring Buffer Next Context ID Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
Address:	22198h-2219Fh

This register contains the *next* ring context ID associated with the ring buffer.

Programming Notes

The current context (RCCID) register can be updated indirectly from this register on a context switch event. Note that the only time a context switch can occur is when MI_ARB_CHECK enables preemption or the current context runs dry (head pointer becomes equal to tail pointer).

DWord	Bit	Description
0	63:0	**Unnamed** See Context Descriptor for BCS

BCS SW Control

BCS_SWCTRL - BCS SW Control

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	BUS
Address:	22200h

DWord	Bit	Description
0	31:16	**Mask** Access: WO Format: Mask _Custom_GTIReset: BUS
	15:4	**Reserved** Access: RO Format: MBZ
	3	**Shrink Blitter Cache** Format: U1 _Custom_GTIReset: BUS This bit is primarily used for validation purposes to speed up the test time. The full cache depth of 128 CLs should be used for production. This bit is part of the context save/restore. This bit only applies to the XY_FAST_COPY_BLT command. **Value** / **Name** / **Description** 0 / **[Default]** / Blitter/BCS flush will flush and invalidate all cachelines in the Blitter/BLB cache (default). 1 / / Blitter Cache depth will be shortened from 128 CLs to 16 CLs.
	2	**Not Invalidate Blitter Cache on BCS Flush** Format: U1 _Custom_GTIReset: BUS Programming this bit allows optimal/maximal cache hit usage, when the destination surface of a Fast Copy Blit, is to be used as the Source for a follow on Fast Copy blit, even if the destination surface is flushed out for Display coherency reasons (where the destination surface is also needed to be Displayed). Such a flush with clean cacheline state is suggested when the intermediate blit operation results are being required to maintain memory coherency. The legacy method of cache invalidation on flush can be still pursued at the end of all blit operations or when switching happens due to other prescribed legacy reasons, or when switching from the new Fast Copy Engine blit, to legacy Engine blits. This bit should be programmed set only when used with Fast Copy Blit commands. This bit is part of the context save/restore. This bit only applies to the XY_FAST_COPY_BLT command.

BCS_SWCTRL - BCS SW Control

Value	Name	Description
0	**[Default]**	Blitter/BLB Cache will be 128 cache lines in depth (default).
1		BCS flush will put all dirty CL in the Blitter cache in the clean state. Any CL already in the clean state will remain clean.

1 **Tile Y Destination**

Description
Programming this bit makes the HW treat all destination surfaces as Tile Y. This bit over-rides the setting of the destination format in the packet provided to the blitter command streamer. SW is required to flush the HW before changing the polarity of this bit. This bit is part of the context save/restore. This bit does not impact the operations of the XY_FAST_COPY_BLT command.

0 **Tile Y Source**

Description
Programming this bit makes the HW treat all source surfaces as Tile Y. This bit over-rides the setting of the source format in the packet provided to the blitter command streamer. SW is required to flush the HW before changing the polarity of this bit. This bit is part of the context save/restore. This bit does not impact the operations of the XY_FAST_COPY_BLT command.

BIOS2DRIVER Scratch0

B2D_SCRATCH0 - BIOS2DRIVER Scratch0

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	102000h

This register is used by Software and for software purposes only. Preliminary definition. The bit definitions may change later as the driver team has better clarity on what information is required and how best to format. Any updates will have no effect on the hardware.

DWord	Bit	Description
0	31	**Display HD Audio Disable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	30	**Spare_16** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	29	**Spare_15** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	28	**Spare_14** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	27	**Spare_13** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	26	**Spare_12** Default Value: 0b Access: R/W _Custom_GTIReset: BUS

B2D_SCRATCH0 - BIOS2DRIVER Scratch0

	25	**ECC Disable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	24	**Force DRAM ECC Enabled** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	23	**Spare_11** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	22	**Spare_10** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	21	**Spare_9** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	20:19	**DDR Size** Default Value: 00b Access: R/W _Custom_GTIReset: BUS
	18	**Bclk overclocking disable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	17	**Disable 1N Mode** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	16	**Spare_8** Default Value: 0b Access: R/W _Custom_GTIReset: BUS

B2D_SCRATCH0 - BIOS2DRIVER Scratch0

	15	**Spare_7** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	14	**2 DIMMS per Channel Disable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	13	**Spare_6** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	12	**Performance Dual Channel Disable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	11	**Spare_5** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	10	**Spare_4** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	9:8	**Spare_3** Default Value: 00b Access: R/W _Custom_GTIReset: BUS
	7:4	**Spare_2** Default Value: 0000b Access: R/W _Custom_GTIReset: BUS
	3	**DDR Overclocking** Default Value: 0b Access: R/W _Custom_GTIReset: BUS

B2D_SCRATCH0 - BIOS2DRIVER Scratch0		
	2	**IA Overclocking Enabled b** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	1	**Spare_1** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	0	**DDR3L Enable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS

BIOS2DRIVER Scratch1

B2D_SCRATCH1 - BIOS2DRIVER Scratch1

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	102004h

This register is used by Software and for software purposes only. Preliminary definition. The bit definitions may change later as the driver team has better clarity on what information is required and how best to format. Any updates will have no effect on the hardware.

DWord	Bit	Description	
0	31	**Spare_14**	
		Default Value:	0b
		Access:	R/W
		_Custom_GTIReset:	BUS
	30	**IA Overclocking DSKU Control Disable**	
		Default Value:	0b
		Access:	R/W
		_Custom_GTIReset:	BUS
	29	**IA Overclocking Enable**	
		Default Value:	0b
		Access:	R/W
		_Custom_GTIReset:	BUS
	28	**Spare_13**	
		Default Value:	0b
		Access:	R/W
		_Custom_GTIReset:	BUS
	27:25	**Cache Size Capability**	
		Default Value:	000b
		Access:	R/W
		_Custom_GTIReset:	BUS
	24	**Spare_12**	
		Default Value:	0b
		Access:	R/W
		_Custom_GTIReset:	BUS

B2D_SCRATCH1 - BIOS2DRIVER Scratch1

	23:21	**DDR3 Maximum Frequency Capability with 100 Memory** Default Value: 000b Access: R/W _Custom_GTIReset: BUS
	20	**Spare_11** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	19	**Spare_10** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	18	**Spare_9** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	17	**Spare_8** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	16	**Spare_7** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	15:12	**Spare_6** Default Value: 0000b Access: R/W _Custom_GTIReset: BUS
	11	**Reserved**
	10:8	**Spare_5** Default Value: 000b Access: R/W _Custom_GTIReset: BUS

B2D_SCRATCH1 - BIOS2DRIVER Scratch1

	7	**Spare_4** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	6:4	**DDR3 Maximum Frequency Capability** Default Value: 000b Access: R/W _Custom_GTIReset: BUS
	3	**Spare_3** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	2	**DDR4 DSKU Enable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	1	**Spare_2** Default Value: 0b Access: R/W _Custom_GTIReset: BUS
	0	**Spare_1** Default Value: 0b Access: R/W _Custom_GTIReset: BUS

BIOS2DRIVER Scratch2

B2D_SCRATCH2 - BIOS2DRIVER Scratch2

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	102008h

Programming Notes

This register is used by Software and for software purposes only. Preliminary definition. The bit definitions may change later as the driver team has better clarity on what information is required and how best to format. Any updates will have no effect on the hardware.

DWord	Bit	Description
0	31:0	**Spare** _Custom_GTIReset: BUS Reserved.

BIOS2DRIVER Scratch3

B2D_SCRATCH3 - BIOS2DRIVER Scratch3

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	10200Ch

Programming Notes

This register is used by Software and for software purposes only. Preliminary definition. The bit definitions may change later as the driver team has better clarity on what information is required and how best to format. Any updates will have no effect on the hardware.

DWord	Bit	Description
0	31:0	**Spare** _Custom_GTIReset: BUS Reserved.

BIOS2DRIVER Scratch4

B2D_SCRATCH4 - BIOS2DRIVER Scratch4

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	102010h

Programming Notes

This register is used by Software and for software purposes only. Preliminary definition. The bit definitions may change later as the driver team has better clarity on what information is required and how best to format. Any updates will have no effect on the hardware.

DWord	Bit	Description
0	31:0	**Spare** _Custom_GTIReset: BUS Reserved.

BIOS2DRIVER Scratch5

B2D_SCRATCH5 - BIOS2DRIVER Scratch5

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	102014h

Programming Notes

This register is used by Software and for software purposes only. Preliminary definition. The bit definitions may change later as the driver team has better clarity on what information is required and how best to format. Any updates will have no effect on the hardware.

DWord	Bit	Description
0	31:0	**Spare** _Custom_GTIReset: BUS Reserved.

BIOS2DRIVER Scratch6

B2D_SCRATCH6 - BIOS2DRIVER Scratch6

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	102018h

Programming Notes

This register is used by Software and for software purposes only. Preliminary definition. The bit definitions may change later as the driver team has better clarity on what information is required and how best to format. Any updates will have no effect on the hardware.

DWord	Bit	Description
0	31:0	**Spare** _Custom_GTIReset: BUS Reserved.

BIOS2DRIVER Scratch7

B2D_SCRATCH7 - BIOS2DRIVER Scratch7

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	10201Ch

Programming Notes

This register is used by Software and for software purposes only. Preliminary definition. The bit definitions may change later as the driver team has better clarity on what information is required and how best to format. Any updates will have no effect on the hardware.

DWord	Bit	Description
0	31:0	**Spare** _Custom_GTIReset: BUS Reserved.

Bitstream Output Bit Count for the last Syntax Element Report Register

MFC_BITSTREAM_SE_BITCOUNT_SLICE - Bitstream Output Bit Count for the last Syntax Element Report Register	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	128D4h
Name:	SE Output Bit Count
ShortName:	SE_Output_Bit_Count

This register stores the count of number of bits in the bitstream for the last syntax element before padding. The bit count is before the byte-aligned alignment padding insertion, but includes the stop-one-bit. This register is part of the context save and restore.

DWord	Bit	Description
0	31:0	**MFC Bitstream Syntax Element Bit Count** Total number of bits in the bitstream output before padding. This count is updated each time the internal counter is incremented.

Bitstream Output Byte Count Per Slice Report Register

MFC_BITSTREAM_BYTECOUNT_SLICE - Bitstream Output Byte Count Per Slice Report Register		
Register Space:	MMIO: 0/2/0	
Access:	RO	
Size (in bits):	32	
Address:	128D0h	
This register stores the count of bytes of the bitstream output. This register is part of the context save and restore.		
DWord	**Bit**	**Description**
0	31:0	**MFC Bitstream Byte Count** Total number of bytes in the bitstream output from the encoder. This count is updated for every time the internal bitstream counter is incremented.

Bitstream Output Minimal Size Padding Count Report Register

MFC_AVC_MINSIZE_PADDING_COUNT - Bitstream Output Minimal Size Padding Count Report Register	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	12814h
Name:	Minimal Size Padding
ShortName:	Minimal_Size_Padding

This register stores the count in bytes of **minimal size padding insertion. It is primarily provided for statistical data gathering.** This register is part of the context save and restore.

DWord	Bit	Description
0	31:0	**MFC AVC MinSize Padding Count** Total number of bytes in the bitstream output contributing to minimal size padding operation. This count is updated each time when the padding count is incremented.

BLC_PWM_CTL

BLC_PWM_CTL

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	48250h-48253h
Name:	Backlight PWM Control
ShortName:	BLC_PWM_CTL
Reset:	soft

This register controls the backlight PWM logic going to the display utility pin on the CPU.

DWord	Bit	Description
0	31	**PWM Enable** This bit enables the PWM logic.

Value	Name	Description
0b	Disable	PWM disabled
1b	Enable	PWM enabled

Restriction
The display utility pin must be configured correctly to output the PWM. Program the frequency and duty cycle before enabling PWM.

DWord	Bit	Description
	30:29	**Pipe Select** This field selects which vertical blank will be used for backlight blinking.

Value	Name	Description
00b	Pipe A	Use Pipe A
01b	Pipe B	Use Pipe B
10b	Pipe C	Use Pipe C
11b	Pipe D	

DWord	Bit	Description
	28	**Blinking Enable** This bit enables backlight blinking. When enabled, the backlight will be driven on at the programmed brightness during vertical blank and driven off during vertical active.

Value	Name
0b	Disable
1b	Enable

BLC_PWM_CTL

	27	**PWM Granularity** This field controls the granularity (minimum increment) of the PWM backlight control counter. **Value** / **Name** / **Description** 0b / 128 / PWM frequency adjustment on 128 clock increments 1b / 8 / PWM frequency adjustment on 8 clock increments
	26:0	**Reserved** Access: RO Format: MBZ

BLC_PWM_DATA

BLC_PWM_DATA		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	48254h-48257h	
Name:	Backlight PWM Data	
ShortName:	BLC_PWM_DATA	
Reset:	soft	

DWord	Bit	Description
0	31:16	**Backlight Frequency** This field determines the number of time base events in total for a complete cycle of modulated backlight control. This field is programmed based on the frequency of the clock that is being used and the desired PWM frequency. This value represents the period of the PWM stream in CD clocks multiplied by 128 (default increment) or 8 (alternate increment selected by BLC_PWM_CTL PWM_Granularity).
	15:0	**Backlight Duty Cycle** This field determines the number of time base events for the active portion of the PWM backlight control. A value of zero will turn the backlight off. A value equal to the backlight modulation frequency field will be full on. Updates will take affect at the end of the current PWM cycle. This value represents the active time of the PWM stream in CD clock periods multiplied by 128 (default increment) or 8 (alternate increment selected by BLC_PWM_CTL PWM_Granularity). **Restriction** This should never be larger than the frequency field.

Blitter Cache Control Register

BLIT_CCTL - Blitter Cache Control Register

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	BUS
Address:	22204h

DWord	Bit	Description
0	31:23	**Reserved** Access: RO Format: MBZ
	22:16	**Pattern MOCS** Format: MEMORY_OBJECT_CONTROL_STATE MOCS for blitter pattern operands.
	15	**Pattern MOCS Select** This bit is used to select what MOCS bits (pattern or source) are driven on the shared pattern/source MOCS bus. Value 0b: **[Default]** Drive source MOCS bits on the shared MOCS bus. Value 1b: Drive pattern MOCS bits on the shared MOCS bus.
	14:8	**Destination MOCS** Format: MEMORY_OBJECT_CONTROL_STATE MOCS for blitter destination operands.
	7	**Reserved** Access: RO Format: MBZ
	6:0	**Source MOCS** Format: MEMORY_OBJECT_CONTROL_STATE MOCS for blitter source operands.

Boot Hash Check Status

BOOT_HASH_CHK - Boot Hash Check Status

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32

Programming Notes

This register is saved in the power context

DWord	Bit	Description
0	31	**Valid uKernel Loaded** Whether there is a valid uKernel loaded into the SRAM. During boot this bit is set when the ukernel is moved from graphics memory to WOPCM & SRAM. During RC6 resume this bit is set when the uKernel is moved from WOPCM to SRAM. On initial Boot: BootROM code shall check this bit during the process of bringing up MinuteIA to determine when the uKernel is in-place in the SRAM before transferring control to it. On RC6 exit: If C068[4] is 0 (Early Jump): Bootrom code does not poll on C010[31] before jumping to ukernel in SRAM. Any reads/writes to SRAM area not yet restored by DMA are stalled based on "CL data present" bit for that cacheline. If C068[4] is 1 (Skip Early Jump): C010[31] is used same as in initial boot case.
	30:9	**Reserved** Access: RO Format: MBZ
	8	**Reserved**
	7:3	**Reserved** Access: RO Format: MBZ
	2	**Reserved**
	1	**Reserved**
	0	**uKernel Hash Ready** Indicates HW has completed SHA computation and loaded the value into the uKernel hash register

BOOT VECTOR

BOOTMSG - BOOT VECTOR

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	08504h

Boot Message Register
This register gets locked by the Hardware once written and is cleared only during the reset. This is extra protection given against Illegal Programming.

DWord	Bit	Description
0	31:0	**Boot Vector Message** Access: R/W Lock _Custom_GTIReset: BUS Boot vector is pass through. MBC gets the boot message from GPMunit and forwards it to MSQC. Breakdown of message is done in MSQC. Details: if b[26] = 1 C6SliceA = b[20:17]; C6SliceB= d[13:10] C6Way = 0 C6Area = 0 if b[26] = 0 C6Way = b[25:21], C6Slice = d[20:17]; C6Area = d[17:10] Context Restore = b[7] Reset Type = b[6:5] Ring Stop ID = b[4:0]

Built In Self Test

BIST_0_2_0_PCI - Built In Self Test		
Register Space:	PCI: 0/2/0	
Size (in bits):	8	
Address:	0000Fh	
This register is used for control and status of Built In Self Test (BIST).		
DWord	**Bit**	**Description**
0	7	**BIST Supported** Default Value: 0b Access: RO _Custom_GTIReset: BUS BIST is not supported. This bit is hardwired to 0.
	6:0	**Reserved** Access: RO Format: MBZ

BW_BUDDY_CTL

BW_BUDDY_CTL

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

Address:	45140h-45143h
Name:	Bandwidth Buddy1 Control
ShortName:	BW_BUDDY1_CTL
Reset:	soft

Address:	45150h-45153h
Name:	Bandwidth Buddy2 Control
ShortName:	BW_BUDDY2_CTL
Reset:	soft

DWord	Bit	Description
0	31	**BW Buddy Disable** Access: R/W This field indicates if the address buddy logic is disabled. 0b: Enabled **[Default]** 1b: Disabled
	30	**Reserved**
	29	**Reserved** Access: RO Format: MBZ
	28:23	**Plane Request Timer** Access: R/W This is the timer to pick when a tracker gets allocated by a regular HP plane Request and starts to wait for its buddy (based on the mask) to come in. 010000b: 16 **[Default]** [1,63]
	22	**Reserved** Access: RO Format: MBZ

BW_BUDDY_CTL

	21:16	**TLB Request Timer** Access: R/W This is the timer to pick when a tracker gets allocated by a TLB Request and starts to wait for its buddy (based on the mask) to come in. Value: 0010000b — Name: 8 **[Default]** Value: [1,63] — Name:
	15	**Reserved**
	14:0	**Reserved** Access: RO Format: MBZ

BW_BUDDY_PAGE_MASK

BW_BUDDY_PAGE_MASK			
Register Space:	MMIO: 0/2/0		
Access:	R/W		
Size (in bits):	32		
Address:	45144h-45147h		
Name:	Bandwidth Buddy1 Page Mask		
ShortName:	BW_BUDDY1_PAGE_MASK		
Reset:	soft		
Address:	45154h-45157h		
Name:	Bandwidth Buddy2 Page Mask		
ShortName:	BW_BUDDY2_PAGE_MASK		
Reset:	soft		
DWord	**Bit**	**Description**	
0	31:28	**Reserved**	
		Access:	RO
		Format:	MBZ
	27:0	**BW Buddy Page Mask**	
		Default Value:	0000000h All address bits are not Masked
		Access:	R/W
		When set, bits in this register will cause Address bits to be excluded from the buddy address comparison. Mask bit[0] is associated with address bit[9], mask bit[1] is associated with address bit[10] and so on. For example, if bit [0] of the mask register is set, then address bit[9] is not used in the buddy address comparison. The default is to compare all address bits.	
		Programming Notes	
		Optimal programming will depend on the memory configuration used. See **Arbiter** page for details.	

C6 Entry latency

C6_ENTRY_LATENCY - C6 Entry latency

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 00D80h
Name: C6 Entry latency
ShortName: C6_ENTRY_LATENCY
_Custom_GTIContextMappedUnit: mgsr

The C6 Entry Latency written by GPMunit

DWord	Bit	Description
0	31:0	**Count for C6 entry Latency** Access: R/W _Custom_GTIReset: BUS C6 Entry Latency written by GPM Measure latency of C6 entry in terms usec pulses

Cache Line Size

CLS_0_2_0_PCI - Cache Line Size		
Register Space:	PCI: 0/2/0	
Size (in bits):	8	
Address:	0000Ch	
DWord	**Bit**	**Description**
0	7:0	**Cache Line Size Value** Default Value: 00000000b Access: R/W _Custom_GTIReset: BUS This field is implemented by PCI Express devices as a read-write field for legacy compatibility purposes but has no effect on any PCI Express device behavior.

Cache Mode Register 0

CACHE_MODE_0 - Cache Mode Register 0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV

Address:	07000h
Name:	Cache Mode Register 0
ShortName:	CACHE_MODE_0

This register is used to control the operation of the Render and Sampler L2 Caches. All reserved bits are implemented as read/write.
Before changing the value of this register, GFX pipeline must be idle i.e. full flush is required.
This Register is saved and restored as part of Context.

DWord	Bit	Description
0	31:16	**Mask** Access: WO Format: Mask _Custom_GTIReset: DEV A 1 in a bit in this field allows the modification of the corresponding bit in Bits 15:0.
	15	**Reserved** Access: R/W Format: PBC _Custom_GTIReset: DEV
	14:12	**MSAA Compression Plane Number Threshold for eLLC** Access: R/W _Custom_GTIReset: DEV

Value	Name	Description
0h	threshold0 **[Default]**	Cache only planeID = 0 in eLLC.
1h	threshold1	Cache only planeID = 0, 1 in eLLC.
2h	threshold2	Cache only planeID = 0..2 in eLLC.
3h	threshold3	Cache only planeID = 0..3 in eLLC.
4h	threshold4	Cache only planeID = 0..4 in eLLC.
5h	threshold5	Cache only planeID = 0..5 in eLLC.
6h	threshold6	Cache only planeID = 0..6 in eLLC.
7h	threshold7	Cache only planeID = 0..7 in eLLC.

CACHE_MODE_0 - Cache Mode Register 0

Programming Notes
This bit-field is programmed based on MSAA. When MSAA compression is enabled, these settingsaffect HW, else it is ignored. For 16X MSAA only lower 8 planes can be cached in eLLC.

11 Reserved

Access:	R/W
Format:	PBC
_Custom_GTIReset:	DEV

10 RCZ PMA Not-Promoted Allocation stall optimization Disable due to change in depth parameters

Access:	R/W
Format:	Disable
_Custom_GTIReset:	DEV

Setting this bit will force the RCZ cache to stall at the allocation of a CL if any of the values in {Depth-mode, DTE, DWE, DTF} are different between the old and new requests to the same CL. The default is a smart stall depending on the New requests depth-test and depth write fields.

Value	Name	Description
0h	**[Default]**	Optimization is enabled
1h		Optimization is disabled

9 Sampler L2 TLB Prefetch Enable

Access:	R/W
_Custom_GTIReset:	DEV

Value	Name	Description
0h	**[Default]**	TLB Prefetch Disabled
1h		TLB Prefetch Enabled

8 Reserved

7 Reserved

Access:	R/W
Format:	PBC
_Custom_GTIReset:	DEV

6:5 Reserved

Access:	R/W
Format:	PBC
_Custom_GTIReset:	DEV

CACHE_MODE_0 - Cache Mode Register 0

4 — RCC Eviction Policy

Access:	R/W
Format:	Disable
_Custom_GTIReset:	DEV

If this bit is set, RCCunit will have LRA as replacement policy. The default value i.e.(when this bit is reset) indicates that non-LRA eviction policy. This bit must be reset. LRA replacement policy is not supported.

3 — Reserved

2 — Hierarchical Z RAW Stall Optimization Disable

Access:	R/W

Description
The Hierarchical Z RAW Stall Optimization allows non-overlapping polygons in the same 8x4 pixel/sample area to be processed without stalling waiting for the earlier ones to write to Hierarchical Z buffer.

Value	Name	Description
0h	Enable **[Default]**	Enables the hierarchical Z RAW Stall Optimization.
1h	Disable	Disables the hierarchical Z RAW Stall Optimization.

Programming Notes
This bit must be set to 0 to enable the Hierarchical Z RAW stall optimization.

1 — Disable clock gating in the pixel backend

Access:	R/W
Format:	Disable
_Custom_GTIReset:	DEV

MCL related clock gating is disabled in the pixel backend. Before setting this bit to 1,the instruction/state caches must be invalidated. [DevGT:{WKA}]

0 — Disable Byte sharing for 3D TYF LOD1 surfaces for 32/64/128 bpp

Access:	R/W
_Custom_GTIReset:	DEV

Value	Name	Description
1		Enable byte sharing for 3D TYF LOD1 surfaces - 32/64/128 bpp
0	**[Default]**	Disable Byte Sharing for 3D TYF surfaces LOD1 , 32/64/128 bpp

Cache Mode Register 1

CACHE_MODE_1 - Cache Mode Register 1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Reset:	DEV
Address:	07004h
Name:	Cache Mode Register 1
ShortName:	CACHE_MODE_1

RegisterType: MMIO_SVL

Before changing the value of this register, GFX pipeline must be idle; i.e., full flush is required. This Register is saved and restored as part of Context.

DWord	Bit	Description
0	31:16	**Mask** Access: WO Mask: MASK Format: Mask _Custom_GTIReset: DEV Must be set to modify corresponding data bit. Reads to this field returns zero.
	15	**Color Compression Disable** Access: R/W _Custom_GTIReset: DEV Setting this bit causes Lossless Render Target Color Compression to be disabled in Classic Clear (1x) Mode of Operation. Default value, i.e. resetting this bit, Enables Color Compression in Classic Clear Mode (1x) when CCS is Enabled.

Value	Name	Description
0h	**[Default]**	Enables Color Compression in Classic Clear Mode (1x) when CCS is Enabled.
1h		Causes Lossless Render Target Color Compression to be disabled in Classic Clear (1x) Mode of Operation

Programming Notes

The Below programming forces Color Compression to be disabled for MSAA modes explicitly as a HW WA. When switching from 1x ==> MSAA. Program this bit to 1 When switching from MSAA ==> 1x. Program this bit to 0

CACHE_MODE_1 - Cache Mode Register 1

14 Render Target 64B Read Disabled by RCC

Access:	R/W
Format:	U1
_Custom_GTIReset:	DEV

Setting this bit causes RCC to disable 64B reads and switch to legacy 128B Reads per RCC CL

Value	Name
1h	
0h	**[Default]**

13 NP EARLY Z FAILS DISABLE

Access:	R/W
_Custom_GTIReset:	DEV

Value	Name	Description
0h	**[Default]**	IZ does conservatively fail any NP/R pixels.
1h		Disables IZ to conservatively fail pixels.

12 Reserved

Access:	R/W
Format:	PBC
_Custom_GTIReset:	DEV

11 Reserved

Access:	R/W
Format:	PBC
_Custom_GTIReset:	DEV

10 Reserved

Access:	R/W
Format:	PBC
_Custom_GTIReset:	DEV

9 MSC RAW Hazard Avoidance Bit

Access:	R/W
Format:	Enable
_Custom_GTIReset:	DEV

When this field is set, MSC will enable RAW Hazard prevention mechanism, when lossless compression is enabled.

Value	Name	Programming Notes
0h	**[Default]**	
1h		This field should be programmed to 1 only if need arise to avoid RAW hazard when lossless compression is enabled

CACHE_MODE_1 - Cache Mode Register 1

8 Reserved

Access:	R/W
Format:	PBC
_Custom_GTIReset:	DEV

7 Reserved

_Custom_GTIReset:	DEV

6 Shader Independent Evaluate Enable

When this bit is enabled, any shader stage can send evaluate messages. Hardware ignores AMFS state programming.

Value	Name	Description
0	Shader Independent Evaluate Disabled **[Default]**	Only Pixel Shaders can do evaluate
1		ALl shader stages can do evaluate

5 MCS Cache Disable

Access:	R/W
Format:	Disable
_Custom_GTIReset:	DEV

For Programming restrictions please refer to the 3D Pipeline.

Value	Name	Description
0h	**[Default]**	MCS cache enabled. It allows RTs with MCS buffer enabled to be rendered using either MSAA compression for MSRT OR with color clear feature for non MSRT.
1h		MCS cache is disabled. Hence no MSAA compression for MSRT and no color clear for non-MSRT.

4 Reserved

Access:	R/W
Format:	PBC
_Custom_GTIReset:	DEV

3 RCZ PMA Promoted 2 Not-Promoted Allocation stall optimization Disable

Access:	R/W
_Custom_GTIReset:	DEV

Setting this bit will force the RCZ cache to stall at the allocation of a CL until the old Promoted writes retire in the RCZ$. Default mode is to stall at the IZ-read point until promoted writes are complete.

Value	Name	Description
0h	**[Default]**	Optimization is enabled
1h		Optimization is disabled

CACHE_MODE_1 - Cache Mode Register 1

2 **AMFS Disable Overshading**

Access:	R/W
_Custom_GTIReset:	DEV

Value	Name
0	**[Default]**
1	

Programming Notes
SW must set this bit to disable Overshading in AMFS control cache If this bit is set to 1, every eval message is allocated in the Control Cache. If this bit to set to 0, eval messages bypass control cache and all texels in the cacheline are shaded

1 **YCoCg DIsable**

Access:	R/W
Format:	Disable
_Custom_GTIReset:	DEV

Value	Name	Description
0h	**[Default]**	YCoCg will be enabled by Defualt
1h		Setting this bit to 1 will disable YCoCg Compression and only compress using legacy RGB color space

0 **Disable Lossless Compression of partial Evictions on Previous Uncompressed Cache line**

Access:	R/W
Format:	PBC
_Custom_GTIReset:	DEV

Value	Name	Description
0h	**[Default]**	Lossless Compression of partial Evictions on Previous Uncompressed Cache line is Enabled
1h		Lossless Compression of partial Evictions on Previous Uncompressed Cache line is Disabled

Cache Mode Subslice Register

CACHE_MODE_SS - Cache Mode Subslice Register

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0E420h

DWord	Bit	Description
0	31:16	**Mask Bits** Format: Mask Must be set to modify corresponding bit in Bits 15:0. (All implemented bits)
	15:13	**Reserved** Access: RO Format: MBZ
	12	**Reserved** Access: RO Format: MBZ
	11	**Reserved** Format: PBC
	10:5	**Reserved** Access: RO Format: MBZ
	4	**Reserved** Format: PBC
	3:2	**Reserved** Access: RO Format: MBZ
	1	**Instruction Level 1 Cache and In-Flight Queue Disable** Format: Disable Value / Name / Description: 0h / **[Default]** / Cache is enabled. 1h / / Reserved

CACHE_MODE_SS - Cache Mode Subslice Register

	0	**Instruction Level 1 Cache Disable** Format: Disable **Value** / **Name** / **Description** 0h / **[Default]** / Cache is enabled. 1h / / Reserved

Capabilities A

CAPID0_A_0_2_0_PCI - Capabilities A

Register Space:	PCI: 0/2/0
Size (in bits):	32
Address:	00044h

Populated by pulling relevant fuses.

DWord	Bit	Description
0	31:25	**Spare Fuses 1** Default Value: 00h Access: RO Variant _Custom_GTIReset: BUS Spare Fuse
	24	**Display FuSa disable** Default Value: 0b Access: RO Variant _Custom_GTIReset: BUS Fuse to disable FuSa
	23:4	**Spare Fuses 2** Default Value: 00000h Access: RO Variant _Custom_GTIReset: BUS Spare Fuses
	3	**VGT Enable Fuse** Default Value: 0b Access: RO Variant _Custom_GTIReset: BUS
	2	**Use Dedicated Memory Path** Default Value: 0b Access: RO Variant _Custom_GTIReset: BUS This bit indicates if the direct memory path is enabled or not. 1'b0 (Default) - Dedicated memory path is disabled 1'b1 - Dedicated memory path is enabled.

CAPID0_A_0_2_0_PCI - Capabilities A			
	1	**SVM Disable Fuse**	
		Default Value:	0b
		Access:	RO Variant
		_Custom_GTIReset:	BUS
	0	**Vtd Disable Fuse**	
		Default Value:	0b
		Access:	RO Variant
		_Custom_GTIReset:	BUS

Capabilities B

CAPID0_B_0_2_0_PCI - Capabilities B		
Register Space: PCI: 0/2/0 Size (in bits): 32		
Address: 00048h		
Populated by pulling relevant fuses.		
DWord	**Bit**	**Description**
0	31:0	**Reserved Fuses** Default Value: 0b Access: RO _Custom_GTIReset: BUS

Capabilities Control

CAPCTRL0_0_2_0_PCI - Capabilities Control		
Register Space:	PCI: 0/2/0	
Size (in bits):	16	
Address:	00042h	
DWord	**Bit**	**Description**
0	15:12	**Reserved** Access: RO Format: MBZ
	11:8	**CAPID Version** Default Value: 0001b Access: RO _Custom_GTIReset: BUS This field is hardwired to the value 1h to identify the first revision of the CAPID register definition.
	7:0	**CAPID Length** Default Value: 00001100b Access: RO _Custom_GTIReset: BUS This field is hardwired to the value 0Ch to indicate the structure length (12 bytes).

Capabilities Pointer

CAPPOINT_0_2_0_PCI - Capabilities Pointer		
Register Space:	PCI: 0/2/0	
Size (in bits):	8	
Address:	00034h	
This register points to a linked list of capabilities implemented by this device.		
DWord	**Bit**	**Description**
0	7:0	**Capabilities Pointer Value** Default Value: 01000000b Access: RO _Custom_GTIReset: BUS This field contains an offset into the function's PCI Configuration Space for the first item in the New Capabilities Linked List, the CAPID0 register at offset 40h.

Capability Identifier

CAPID0_0_2_0_PCI - Capability Identifier

Register Space:	PCI: 0/2/0
Size (in bits):	16
Address:	00040h

DWord	Bit	Description
0	15:8	**Next Capability Pointer** Default Value: 01110000b Access: RO _Custom_GTIReset: BUS This field is hardwired to point to the next PCI Capability structure, the PCIe Capabilities structure at 70h.
	7:0	**Capability Identifier** Default Value: 00001001b Access: RO _Custom_GTIReset: BUS This field is hardwired to the value 09h to identify the CAP_ID assigned by the PCI SIG for vendor dependent capability pointers.

CAPTURE_0_DSSM0

CAPTURE_0_DSSM0 - CAPTURE_0_DSSM0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Capture Buffer DW 0		
DWord	**Bit**	**Description**
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_0_DSSM1

CAPTURE_0_DSSM1 - CAPTURE_0_DSSM1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Capture Buffer DW 0		
DWord	**Bit**	**Description**
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_0_L3

CAPTURE_0_L3 - CAPTURE_0_L3		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Capture Buffer DW 0		
DWord	**Bit**	**Description**
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_1_DSSM0

CAPTURE_1_DSSM0 - CAPTURE_1_DSSM0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Capture Buffer DW 1		
DWord	**Bit**	**Description**
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_1_DSSM1

CAPTURE_1_DSSM1 - CAPTURE_1_DSSM1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Capture Buffer DW 0		
DWord	**Bit**	**Description**
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_1_L3

CAPTURE_1_L3 - CAPTURE_1_L3

Register Space: MMIO: 0/2/0

Size (in bits): 32

Capture Buffer DW 1

DWord	Bit	Description
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_2_DSSM0

CAPTURE_2_DSSM0 - CAPTURE_2_DSSM0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Capture Buffer DW 2		
DWord	**Bit**	**Description**
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_2_DSSM1

CAPTURE_2_DSSM1 - CAPTURE_2_DSSM1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Capture Buffer DW 2		
DWord	**Bit**	**Description**
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_2_L3

CAPTURE_2_L3 - CAPTURE_2_L3		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Capture Buffer DW 2		
DWord	**Bit**	**Description**
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_3_DSSM0

CAPTURE_3_DSSM0 - CAPTURE_3_DSSM0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Capture Buffer DW 3		
DWord	**Bit**	**Description**
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_3_DSSM1

CAPTURE_3_DSSM1 - CAPTURE_3_DSSM1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Capture Buffer DW 3		
DWord	**Bit**	**Description**
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_3_L3

CAPTURE_3_L3 - CAPTURE_3_L3		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Capture Buffer DW 3		
DWord	**Bit**	**Description**
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_4_DSSM0

CAPTURE_4_DSSM0 - CAPTURE_4_DSSM0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Capture Buffer DW 4		
DWord	**Bit**	**Description**
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_4_DSSM1

CAPTURE_4_DSSM1 - CAPTURE_4_DSSM1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Capture Buffer DW 4		
DWord	**Bit**	**Description**
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_4_L3

CAPTURE_4_L3 - CAPTURE_4_L3

Register Space: MMIO: 0/2/0

Size (in bits): 32

Capture Buffer DW 4

DWord	Bit	Description
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_5_DSSM0

CAPTURE_5_DSSM0 - CAPTURE_5_DSSM0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Capture Buffer DW 5		
DWord	**Bit**	**Description**
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_5_DSSM1

CAPTURE_5_DSSM1 - CAPTURE_5_DSSM1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Capture Buffer DW 5		
DWord	**Bit**	**Description**
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_5_L3

CAPTURE_5_L3 - CAPTURE_5_L3

Register Space: MMIO: 0/2/0

Size (in bits): 32

Capture Buffer DW 5

DWord	Bit	Description
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_6_DSSM0

CAPTURE_6_DSSM0 - CAPTURE_6_DSSM0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Capture Buffer DW 6		
DWord	**Bit**	**Description**
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_6_DSSM1

CAPTURE_6_DSSM1 - CAPTURE_6_DSSM1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Capture Buffer DW 6		
DWord	**Bit**	**Description**
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_6_L3

CAPTURE_6_L3 - CAPTURE_6_L3		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Capture Buffer DW 6		
DWord	**Bit**	**Description**
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_7_DSSM0

CAPTURE_7_DSSM0 - CAPTURE_7_DSSM0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Capture Buffer DW 7		
DWord	**Bit**	**Description**
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_7_DSSM1

CAPTURE_7_DSSM1 - CAPTURE_7_DSSM1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Capture Buffer DW 7		
DWord	**Bit**	**Description**
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

CAPTURE_7_L3

CAPTURE_7_L3 - CAPTURE_7_L3

Register Space: MMIO: 0/2/0

Size (in bits): 32

Capture Buffer DW 7

DWord	Bit	Description
0	31:24	**Byte_3** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	23:16	**Byte_2** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	15:8	**Byte1** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS
	7:0	**Byte_0** Default Value: 00000000b default Access: RO _Custom_GTIReset: BUS

Catastrophic Event FIFO Status

CT_FIFO_STATUS - Catastrophic Event FIFO Status

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

Status of the catastrophic event FIFO.

Programming Notes

When servicing page response interrupts is done, a '1' will be written to the "Done Reading" field. Note that "Done Reading" can be set before "FIFO Occupancy Count" becomes 0 (i.e. the interrupt handler may choose to not service all of the pending catastrophic events). If "Done Reading" is set prior to the FIFO going empty, then there will be a catastrophic event interrupt arbitration in the IOMMU to produce another interrupt (since catastrophic events have the highest priority).The Interrupt-Routing bit is sampled by HW when a 1 is written by SW to the Done-Reading bit. As a result, SW shall program Interrupt-Routing bit appropriately to maintain the routing of the interrupts (Host shall write 0) when updating the Done-Reading bit. This register is saved in the power context.

DWord	Bit	Description
0	31:25	**Reserved** Access: RO Format: MBZ
	24	**Done Reading** Set by host CPU at the end of servicing the IOMMU page response event interrupt. Setting this field will lead to a new catastrophic event interrupt if the FIFO has at least one valid entry. "Done Reading" is cleared by hardware when the IOMMU interrupt is accepted. Note that there is a programming requirement for the Interrupt-routing bit when this bit is set by SW. 0h – Not done – Not done reading the event FIFO. 1h – Done **[Default]** – Done reading the event FIFO, so a new interrupt can be generated if the FIFO is not empty.
	23:18	**Reserved** Access: RO Format: MBZ
	17	**Mask** This bit, when set, causes HW to not report an interrupt to SW even when valid entries exist. HW continues to accumulate FIFO data when this bit is set. An interrupt is generated when this bit is cleared. SW must periodically clear bit and consume data to prevent HW stalls resulting from lack of credits. This bit being set when valid data is present will prevent RC6 entry.

CT_FIFO_STATUS - Catastrophic Event FIFO Status

16 — Interrupt Routing

Where interrupts will be routed. This field can be read or written by the host CPU.

Value	Name	Description
0h	Host **[Default]**	Interrupts will be routed to the host CPU.
1h	Reserved	Reserved

15:9 — Reserved

Access:	RO
Format:	MBZ

8 — Not Empty

Whether the catastrophic event FIFO has at least one valid entry. This field is written by the graphics hardware and can be read by the host CPU.

Value	Name
0h	FIFO empty **[Default]**
1h	FIFO not empty

7:2 — Reserved

Access:	RO
Format:	MBZ

1:0 — FIFO Occupancy Count

The number of valid entries in the catastrophic event FIFO. This field is updated every time the host CPU reads the FIFO or an IOMMU catastrophic event message write is received from the IOMMU/GAM. It is written by the graphics hardware and can be read by the host CPU.

Value	Name
[0,2]	Number of valid FIFO entries

Catastrophic Interrupt Context ID

CT_INTR_CTX_ID - Catastrophic Interrupt Context ID

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32

Reports the context ID at the top of the catastrophic interrupt FIFO. It is written by the graphics hardware and is read by the microkernel and host CPU.

Programming Notes

Catastrophic interrupt FIFO output's a qword data consisting of CTX_INTR_ERR_ID and CT_INTR_CTX_ID registers.
As part of the CAT Error interrupt service routine, SW/FW must first read to CT_INTR_ERR_ID register prior to reading this register.
Each read of this register results in a pop of the catastrophic interrupt FIFO.

DWord	Bit	Description
0	31:0	**Context ID** The context ID at the top of the catastrophic interrupt FIFO. This FIFO collects information about contexts that caused catastrophic faults.

CCS CSB

CCS_CSB - CCS CSB		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
SW reads this offset to read the CSB entry at the top of the CSB fifo. Reads must occur in pairs to obtain a single 64 bit entry. The second read pops the entry off the CSB fifo.		
DWord	**Bit**	**Description**
0	31:0	**Context Status Buffer DW** Access: RO This DW holds CSB bits[31:0] for the first read and CSB bits[63:32] for the second read.

CCS CSB Fifo Status Register

CCS_CSB_FSR - CCS CSB Fifo Status Register		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
This RO register holds status of the CSB fifo.		
DWord	**Bit**	**Description**
0	31	**Not Empty** Access: RO
	30:16	**Reserved** Access: RO Format: MBZ
	15:8	**FIFO Maximum Occupancy Count** This field is a read-only field. It reflects the depth of the FIFO, which is a static value for each product.
	7:0	**FIFO Occupancy Count** Access: RO

CDCLK_CTL

CDCLK_CTL

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	46000h-46003h
Name:	CD Clock Control
ShortName:	CDCLK_CTL
Reset:	global

This register is not reset by the device 2 FLR.

Restriction
These fields should only be changed as part of the Display Sequences for Changing CD Clock Frequency.

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23:22	**CD2X Divider Select** Access: Double Buffered _Custom_Display_DoubleBufferUpdatePoint: Pipe off or start of vertical blank This field selects how the CDCLK PLL output is divided before driving the display CD2X clock. This field is double buffered to align with the pipe from CD2X Pipe Select. It will update at the start of vertical blank of the selected pipe, or immediately if the selected pipe is disabled or no pipe is selected. Value: 00b – Name: Divide by 1 Value: 10b – Name: Divide by 2 **[Default]** **Restriction** CD2X Divider Select must not be changed while more than one pipe is enabled. When one pipe is enabled, the CD2X Pipe Select must be set to that pipe before changing CD2X Divider Select.
	21:19	**CD2X Pipe Select** This field selects the pipe enable and vertical blank to be used for double buffering the CD2X Divider Select. Value: 000b – Name: Pipe A Value: 010b – Name: Pipe B

100b	Pipe C	
110b	Pipe D	
111b	None	Double buffer enable is tied to 1 so that writes to the CD2X Divider Select will take effect immediately.

18	**Reserved**
17	**Reserved**
16	**SSA Precharge Enable** This field is unused.

Value	Name
0b	Disable

15	**Reserved**
14:11	**Reserved**

Access:	RO
Format:	MBZ

10:0 **CD Frequency Decimal**

Format:	U10.1

This field selects the decimal value of the frequency for CD clock, which is used to generate divided down clocks for some display engine timers. This value is represented in a 10.1 format with 10 integer bits and 1 fractional bit.
Many possible values are listed here, and not all valid values are included. To find which values are valid for a project, refer to the clocks page for that project.

Program this field to select the pre-defined value that matches the CD frequency chosen by the CDCLK PLL and CD2X Divider. If no value is defined, program this field with the CD frequency, rounded to the closest 0.5, then minus one.

Value	Name	Description
00 1010 0111 0b	168 MHz CD **[Default]**	This value is default, but not valid.
00101011000b	172.8 MHz CD	
00101100100b	179.2 MHz CD	
00101100110b	180 MHz CD	
00101111110b	192 MHz CD	
01001100100b	307.2 MHz CD	
01 0011 0111 0b	312 MHz CD	
01010000110b	324 MHz CD	
01010001011b	326.4MHz CD	
01110111110b	480MHz CD	
10 0010 0111 0b	552 MHz CD	

CDCLK_CTL				
		10 0010 1100 0b	556.8 MHz CD	
		10 1000 0111 0b	648 MHz CD	
		10 1000 1100 0b	652.8 MHz CD	

CDCLK_PLL_ENABLE

CDCLK_PLL_ENABLE

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
CrashLogSaved:	true
CrashLogPriority:	2
CrashLogVisibility:	public
ExternalLongName:	DE CDCLK PLL Enable
ExternalDescription:	Display engine core PLL enable

Address:	46070h-46073h
Name:	CDCLK PLL Enable
ShortName:	CDCLK_PLL_ENABLE
Reset:	soft

This register is used to control the CDCLK PLL.

DWord	Bit	Description
0	31	**PLL Enable** This field enables or disables the CDCLK PLL. Value: 0b – Disable; 1b – Enable
	30	**PLL Lock** Access: RO This fields indicates the status of the CDCLK PLL Lock. Value: 0b – Not locked or not enabled; 1b – Locked
	29:28	**Reserved** Access: RO Format: MBZ
	27	**Slow Clock Enable** This field enables or disables the slow clock that can be used when the CDCLK PLL is disabled. If the CDCLK PLL is disabled, and slow clock is disabled or not locked, then the crystal clock will be used instead. Value: 0b – Disable; 1b – Enable

CDCLK_PLL_ENABLE

26 — Slow Clock Lock

Access:	RO

This fields indicates the status of the slow clock lock.

Value	Name
0b	Not locked or not enabled
1b	Locked

25:24 — Reserved

Access:	RO
Format:	MBZ

23:22 — Reserved

Access:	RO
Format:	MBZ

21:12 — Reserved

Access:	RO
Format:	MBZ

11 — Reserved

Access:	RO
Format:	MBZ

10:8 — Reserved

Access:	RO
Format:	MBZ

7:0 — PLL Ratio

This field selects the CDCLK PLL divider ratio, controlling the output frequency.
Refer to the Clocks page for valid ratios to program.

Value	Name	Description
1Ch	28 default **[Default]**	Default value. Refer to the Clocks page for valid ratios to program.

Restriction
This field must be configured before enabling CDCLK PLL and not changed while it is enabled.

CGE_CTRL

CGE_CTRL	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank

Address:	49080h-49083h
Name:	Pipe Color Gamut Enhancement Control
ShortName:	CGE_CTRL_A
Reset:	soft

Address:	49180h-49183h
Name:	Pipe Color Gamut Enhancement Control
ShortName:	CGE_CTRL_B
Reset:	soft

Address:	49280h-49283h
Name:	Pipe Color Gamut Enhancement Control
ShortName:	CGE_CTRL_C
Reset:	soft

Address:	49380h-49383h
Name:	Pipe Color Gamut Enhancement Control
ShortName:	CGE_CTRL_D
Reset:	soft

DWord	Bit	Description
0	31	**CGE Enable** This bit enables the Color Gamut Enhancement logic. Value: 0b – Disable; 1b – Enable
	30	**Allow Double Buffer Update Disable** Access: R/W This field controls whether double buffer updates are allowed to be disabled for the CGE registers that are double buffered. The DOUBLE_BUFFER_CTL register can be configured to globally disable double buffer updates for those resources that allow them to be disabled. Value: 0b – Not Allowed; 1b – Allowed **[Default]**

CGE_CTRL		
	29:0	**Reserved** Access: RO Format: MBZ

CGE_WEIGHT

CGE_WEIGHT	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	160
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank OR pipe disabled
Address:	49090h-490A3h
Name:	Pipe Color Gamut Enhancement Weights
ShortName:	CGE_WEIGHT_A
Reset:	soft
Address:	49190h-491A3h
Name:	Pipe Color Gamut Enhancement Weights
ShortName:	CGE_WEIGHT_B
Reset:	soft
Address:	49290h-492A3h
Name:	Pipe Color Gamut Enhancement Weights
ShortName:	CGE_WEIGHT_C
Reset:	soft
Address:	49390h-493A3h
Name:	Pipe Color Gamut Enhancement Weights
ShortName:	CGE_WEIGHT_D
Reset:	soft

These are the weights contained in the lookup up table (LUT) used in pipe color gamut enhancement. LUT index 0 contains the weight for the least saturated colors, and LUT index 16 contains the weight for the most saturated colors. Weight values can range from 00000b (100% of the enhanced output color is from the pipe gamma and CSC output corrected color) to 100000b (100% of the enhanced output color is from the pipe gamma and CSC input color).

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29:24	**CGE Weight Index 3** This is the weight value for this color gamut enhancement LUT index.
	23:22	**Reserved** Access: RO Format: MBZ

CGE_WEIGHT

	21:16	**CGE Weight Index 2** This is the weight value for this color gamut enhancement LUT index.
	15:14	**Reserved** Access: RO Format: MBZ
	13:8	**CGE Weight Index 1** This is the weight value for this color gamut enhancement LUT index.
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**CGE Weight Index 0** This is the weight value for this color gamut enhancement LUT index.
1	31:30	**Reserved** Access: RO Format: MBZ
	29:24	**CGE Weight Index 7** This is the weight value for this color gamut enhancement LUT index.
	23:22	**Reserved** Access: RO Format: MBZ
	21:16	**CGE Weight Index 6** This is the weight value for this color gamut enhancement LUT index.
	15:14	**Reserved** Access: RO Format: MBZ
	13:8	**CGE Weight Index 5** This is the weight value for this color gamut enhancement LUT index.
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**CGE Weight Index 4** This is the weight value for this color gamut enhancement LUT index.
2	31:30	**Reserved** Access: RO Format: MBZ
	29:24	**CGE Weight Index 11** This is the weight value for this color gamut enhancement LUT index.

CGE_WEIGHT		
	23:22	**Reserved** Access: RO Format: MBZ
	21:16	**CGE Weight Index 10** This is the weight value for this color gamut enhancement LUT index.
	15:14	**Reserved** Access: RO Format: MBZ
	13:8	**CGE Weight Index 9** This is the weight value for this color gamut enhancement LUT index.
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**CGE Weight Index 8** This is the weight value for this color gamut enhancement LUT index.
3	31:30	**Reserved** Access: RO Format: MBZ
	29:24	**CGE Weight Index 15** This is the weight value for this color gamut enhancement LUT index.
	23:22	**Reserved** Access: RO Format: MBZ
	21:16	**CGE Weight Index 14** This is the weight value for this color gamut enhancement LUT index.
	15:14	**Reserved** Access: RO Format: MBZ
	13:8	**CGE Weight Index 13** This is the weight value for this color gamut enhancement LUT index.
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**CGE Weight Index 12** This is the weight value for this color gamut enhancement LUT index.
4	31:6	**Reserved** Access: RO Format: MBZ

CGE_WEIGHT		
	5:0	**CGE Weight Index 16** This is the weight value for this color gamut enhancement LUT index.

Clipper Invocation Counter

CL_INVOCATION_COUNT - Clipper Invocation Counter	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02338h-0233Fh
Name:	Clipper Invocation Counter
ShortName:	CL_INVOCATION_COUNT_RCSUNIT_BE_GEOMETRY
Address:	18338h-1833Fh
Name:	Clipper Invocation Counter
ShortName:	CL_INVOCATION_COUNT_POCSUNIT_BE_GEOMETRY

This register stores the count of objects entering the Clipper stage. This register is part of the context save and restore.

DWord	Bit	Description
0	63:32	**CL Invocation Count Report UDW** Number of objects entering the clipper stage. Updated only when Statistics Enable is set in CLIP_STATE (see the Clipper Chapter in the 3D Volume.)
	31:0	**CL Invocation Count Report LDW** Number of objects entering the clipper stage. Updated only when Statistics Enable is set in CLIP_STATE (see the Clipper Chapter in the 3D Volume.)

Clipper Primitives Counter

CL_PRIMITIVES_COUNT - Clipper Primitives Counter	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02340h-02347h
Name:	Clipper Primitives Counter
ShortName:	CL_PRIMITIVES_COUNT_RCSUNIT_BE_GEOMETRY
Address:	18340h-18347h
Name:	Clipper Primitives Counter
ShortName:	CL_PRIMITIVES_COUNT_POCSUNIT_BE_GEOMETRY

This register reflects the total number of primitives that have been output by the clipper. This register is part of the context save and restore.

DWord	Bit	Description
0	63:32	**Clipped Primitives Output Count UDW** Total number of primitives output by the clipper stage. This count is updated for every primitive output by the clipper stage, as long as Statistics Enable is set in SF_STATE (see the Clipper and SF Chapters in the 3D Volume.)
	31:0	**Clipped Primitives Output Count LDW** Total number of primitives output by the clipper stage. This count is updated for every primitive output by the clipper stage, as long as Statistics Enable is set in SF_STATE (see the Clipper and SF Chapters in the 3D Volume.)

Command Buffer Caching Control Register

CMD_BUF_CCTL - Command Buffer Caching Control Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02084h-02087h
Name:	CMD_BUF_CCTL
ShortName:	CMD_BUF_CCTL_RCSUNIT
Address:	18084h-18087h
Name:	CMD_BUF_CCTL
ShortName:	CMD_BUF_CCTL_POCSUNIT
Address:	22084h-22087h
Name:	CMD_BUF_CCTL
ShortName:	CMD_BUF_CCTL_BCSUNIT
Address:	1C0084h-1C0087h
Name:	CMD_BUF_CCTL
ShortName:	CMD_BUF_CCTL_VCSUNIT0
Address:	1C4084h-1C4087h
Name:	CMD_BUF_CCTL
ShortName:	CMD_BUF_CCTL_VCSUNIT1
Address:	1C8084h-1C8087h
Name:	CMD_BUF_CCTL
ShortName:	CMD_BUF_CCTL_VECSUNIT0
Address:	1D0084h-1D0087h
Name:	CMD_BUF_CCTL
ShortName:	CMD_BUF_CCTL_VCSUNIT2
Address:	1D4084h-1D4087h
Name:	CMD_BUF_CCTL
ShortName:	CMD_BUF_CCTL_VCSUNIT3
Address:	1D8084h-1D8087h
Name:	CMD_BUF_CCTL
ShortName:	CMD_BUF_CCTL_VECSUNIT1
Address:	1E0084h-1E0087h
Name:	CMD_BUF_CCTL

CMD_BUF_CCTL - Command Buffer Caching Control Register

ShortName:	CMD_BUF_CCTL_VCSUNIT4
Address:	1E4084h-1E4087h
Name:	CMD_BUF_CCTL
ShortName:	CMD_BUF_CCTL_VCSUNIT5
Address:	1E8084h-1E8087h
Name:	CMD_BUF_CCTL
ShortName:	CMD_BUF_CCTL_VECSUNIT2
Address:	1F0084h-1F0087h
Name:	CMD_BUF_CCTL
ShortName:	CMD_BUF_CCTL_VCSUNIT6
Address:	1F4084h-1F4087h
Name:	CMD_BUF_CCTL
ShortName:	CMD_BUF_CCTL_VCSUNIT7
Address:	1F8084h-1F8087h
Name:	CMD_BUF_CCTL
ShortName:	CMD_BUF_CCTL_VECSUNIT3
Address:	1A084h-1A087h
Name:	CMD_BUF_CCTL
ShortName:	CMD_BUF_CCTL_CCSUNIT0

This register informs the size of the command buffer cache allocated in L3 for DMA requests from Command Streamer. This register also defines the MOCS index that need be send on command buffer read request to be cached in L3.

Programming Notes

This register only functionally impacts the render command streamer as the **Enable Command Cache** bit in PIPE_CONTROL is only supported for the render command streamer. Only render command streamer enables this register a non-privilege.

DWord	Bit	Description
0	31:16	**Mask** Access: WO Format: Mask Mask bits act as write enables for the bits in the lower bits of this register
	15:14	**Reserved** Access: RO Format: MBZ

CMD_BUF_CCTL - Command Buffer Caching Control Register

13:12 **Reserved**

Access:	RO
Format:	MBZ

11:8 **Command Buffer Cache Size**

Format:	U4

The value programmed to this field should be less than or equal to the actual physical cache space reserved in L3 for command buffercaching through L3 configuration. Command buffer DMA engine uses the cache size programmed in this field to limit the read requests of a cacheable batch buffer to be cached in L3. DMA engine does this by tracking the amount of read requests made cacheable and stops caching when the read requested data size equals to the size of the command cache allocated. DMA engine resets the command caching tracker on events listed below. This avoids thrashing of cache for Batch Buffers larger in size compared to the command buffer cache allocated in L3.

- On an End Of Tile in PTRBR/POSH mode of operation.
- On a context save of a context.
- On command cache invalidation through PIPE_CONTROL.

Value	Name	Description
1h	Cache Size 16 KB **[Default]**	Size of the command buffer cache allocated in L3 is 16KB
2h	Cache Size 32 KB	Size of the command buffer cache allocated in L3 is 32KB
3h	Cache Size 64 KB	Size of the command buffer cache allocated in L3 is 64KB
4h	Cache Size 128 KB	Size of the command buffer cache allocated in L3 is 128KB
5h	Cache Size 256 KB	Size of the command buffer cache allocated in L3 is 256KB
6h	Cache Size 512 KB	Size of the command buffer cache allocated in L3 is 512KB
0h,7h,8h,9h, Ah, Bh, Ch, Dh, Eh, Fh	Reserved	Reserved.

7 **Reserved**

Access:	RO
Format:	MBZ

6:0 **Memory Object Control State MOCS for Command Buffer Caching**

Format:	MEMORY_OBJECT_CONTROL_STATE

This field has the standard format defined for MOCS globally. Index to MOCS Tables field of MOCS is used for L3 and System cache memory properties.
Index to MOCS Tables attribute of this field is used for defining the caching properties for requests made for batch buffer command fetches.

Comp Ctx TLB Invalidation Register

COMPCTX_TLB_INV_CR - Comp Ctx TLB Invalidation Register

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0CF04h

DWord	Bit	Description
0	31:16	**Mask Bits** Default Value: 0000000000000000b Access: R/W
	15:8	**Reserved** Access: RO Format: MBZ
	7	**Invalidate Compute Ctx TLBs bit7** Default Value: 0b Access: R/W SW writes 1 to invalidate the TLBs for the associated Compute Context and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's (compute context's) HW pipeline is flushed and cleared from all its memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs. Bit[7:0] invalidates Compute Context[7:0] TLBs. No Effect on non-existent Compute Contexts. These bits self clear.
	6	**Invalidate Compute Ctx TLBs bit6** Default Value: 0b Access: R/W SW writes 1 to invalidate the TLBs for the associated Compute Context and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's (compute context's) HW pipeline is flushed and cleared from all its memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs. Bit[7:0] invalidates Compute Context[7:0] TLBs. No Effect on non-existent Compute Contexts. These bits self clear.
	5	**Invalidate Compute Ctx TLBs bit5** Default Value: 0b Access: R/W SW writes 1 to invalidate the TLBs for the associated Compute Context and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's (compute context's) HW pipeline is flushed and cleared from all its memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs. Bit[7:0] invalidates Compute Context[7:0] TLBs. No Effect on non-existent Compute Contexts. These bits self clear.

COMPCTX_TLB_INV_CR - Comp Ctx TLB Invalidation Register

	4	**Invalidate Compute Ctx TLBs bit4** Default Value: 0b Access: R/W SW writes 1 to invalidate the TLBs for the associated Compute Context and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's (compute context's) HW pipeline is flushed and cleared from all its memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs. Bit[7:0] invalidates Compute Context[7:0] TLBs. No Effect on non-existent Compute Contexts. These bits self clear.
	3	**Invalidate Compute Ctx TLBs bit3** Default Value: 0b Access: R/W SW writes 1 to invalidate the TLBs for the associated Compute Context and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's (compute context's) HW pipeline is flushed and cleared from all its memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs. Bit[7:0] invalidates Compute Context[7:0] TLBs. No Effect on non-existent Compute Contexts. These bits self clear.
	2	**Invalidate Compute Ctx TLBs bit2** Default Value: 0b Access: R/W SW writes 1 to invalidate the TLBs for the associated Compute Context and HW clears the bitwhen invalidation is complete. To ensure proper invalidation of the TLBs, SW has toensure the corresponding engine's (compute context's) HW pipeline is flushed and cleared from all itsmemory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs.Bit[7:0] invalidates Compute Context[7:0] TLBs. No Effect on non-existent Compute Contexts. These bits self clear.
	1	**Invalidate Compute Ctx TLBs bit1** Default Value: 0b Access: R/W SW writes 1 to invalidate the TLBs for the associated Compute Context and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's (compute context's) HW pipeline is flushed and cleared from all its memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs. Bit[7:0] invalidates Compute Context[7:0] TLBs. No Effect on non-existent Compute Contexts. These bits self clear.
	0	**Invalidate Compute Ctx TLBs bit0** Default Value: 0b Access: R/W SW writes 1 to invalidate the TLBs for the associated Compute Context and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's (compute context's) HW pipeline is flushed and cleared from all its

COMPCTX_TLB_INV_CR - Comp Ctx TLB Invalidation Register		
		memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs. Bit[7:0] invalidates Compute Context[7:0] TLBs. No Effect on non-existent Compute Contexts. These bits self clear.

Compute Engine DSS Reservation

CE_DSS_RSERVATION - Compute Engine DSS Reservation

Register Space: MMIO: 0/2/0

Size (in bits): 32

This register indicates the number of DSSs to be reserved for compute engine. This register is privileged and power context save/restored by HW.

DWord	Bit	Description
0	31:16	**MASK** Access: RO Format: Mask Must be set to modify corresponding bit in Bits 15:0. (All implemented bits)
	15:8	**Reserved** Format: PBC
	7:0	**Number ComputeEngine DSS Reservation** Format: U8 This field indicates the number of DSSs to be reserved for compute engine. These DSSs will be exclusively reserved for compute engine and will never be allocated for render engine. **Programming Notes** Value programmed must be less than the maximum number of DSS's available on a given GT SKU.

Config to MCI HI

CFGTOMCIDFTHI - Config to MCI HI		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	098A0h	
Config to MCI and DFT Ring		
DWord	**Bit**	**Description**
0	31	**CFG to MCI HI dispatch** Access: R/WC _Custom_GTIReset: BUS
	30	**CFG to MCI HI error clear** Access: R/WC _Custom_GTIReset: BUS
	29:20	**RSVD_29_20** Access: R/WC _Custom_GTIReset: BUS
	19:0	**BitField: MCI DFT Ring Write data[39:20]. When dispatch is hi, then [39:20] is taken from here and [19:0] is taken from CFGTOMCIDFTLO** Access: R/WC _Custom_GTIReset: BUS

Config to MCI LO

CFGTOMCIDFTLO - Config to MCI LO		
Register Space: MMIO: 0/2/0 Size (in bits): 32		
Address: 0989Ch		
Config to MCI and DFT Ring		
DWord	**Bit**	**Description**
0	31	**CFG to MCI LO dispatch** Access: R/WC _Custom_GTIReset: BUS
	30:20	**MCI DFT Ring Write data[30:20]. When dispatch is lo, then [30:20] is taken from here and [39:31] is taken from [19:11] of CFGTOMCIDFTHI** Access: R/WC _Custom_GTIReset: BUS
	19:0	**MCI DFT Ring Write data[19:0]** Access: R/WC _Custom_GTIReset: BUS

Config to MCI STATUS1

CFGTOMCIDFTSTATUS1 - Config to MCI STATUS1		
Register Space: MMIO: 0/2/0 Size (in bits): 32		
Address: 098A4h		
Config to MCI and DFT Ring		
DWord	**Bit**	**Description**
0	31:5	**RSVD_31_5** Access: RO _Custom_GTIReset: BUS
	4	**report fifo empty** Access: RO _Custom_GTIReset: BUS
	3	**Reserved**
	2	**Reserved**
	1	**mci fifo overflow** Access: RO _Custom_GTIReset: BUS
	0	**report fifo overflow** Access: RO _Custom_GTIReset: BUS

Configuration Register0 for RPMunit

CONFIG0 - Configuration Register0 for RPMunit

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00D00h

Lock bit LOCK applies to all RW/L fields in this register. Lock is overridden during context restore.

DWord	Bit	Description
0	31	**Lock for RW/L Fields in this Register** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of CONFIG0 register are R/W. 1 = All bits of CONFIG0 register are RO (including this lock bit). Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). Lock is reset on a restore after context is captured.
	30:7	**Placeholder Bits** Access: R/W Lock _Custom_GTIReset: BUS
	6	**Reserved**
	5:3	**Crystal Clock Freq Selector** Access: RO _Custom_GTIReset: BUS This indicates the crystal clock frequency. BIOS will read this bit and then program the shift parameters, below, appropriately

Value	Name
0b	Crystal clock is at 24 MHz **[Default]**
1b	Crystal clock is at 19.2 MHz
10b	Crystal clock is at 38.4 MHz
11b	Crystal clock is at 25 MHz
100b	Rsvd for future use

	2:1	**CTC SHIFT parameter** Default Value: 10b Access: R/W Lock _Custom_GTIReset: BUS 00 - use bit 7 as microsecond, bit 3 as lowest *CS timestamp 01 - use bit 6 as microsecond, bit 2 as lowest *CS timestamp 10 - use bit 5 as microsecond, bit 1 as lowest *CS timestamp (default) 11 - use bit 4 as microsecond, bit 0 as lowest *CS timestamp

CONFIG0 - Configuration Register0 for RPMunit

	0	**Disable TSC Synchronization** Access: R/W Lock _Custom_GTIReset: BUS 'b0 - TSC synchronization enabled in GT (default) 'b1 - TSC synchronization DISABLED in GT

Configuration Register1 for RPMunit

CONFIG1 - Configuration Register1 for RPMunit

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00D04h

Lock bit LOCK applies to all RW/L fields in this register. Lock is overridden during context restore.

DWord	Bit	Description
0	31	**Lock for RW/L Fields in this Register** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of CONFIG0 register are R/W. 1 = All bits of CONFIG0 register are RO (including this lock bit). Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). Lock is reset on a restore after context is captured.
	30:10	**Placeholder Bits** Access: R/W Lock _Custom_GTIReset: BUS
	9	**Reserved**
	8:0	**RCP L3 FREQ DETECT** Default Value: 000011110b Access: R/W Lock _Custom_GTIReset: BUS This is control signal needed from clock unit that can be set at 1 when 2X clock frequency is less than or equal to 1GHz. It needs to be at 0 when 2X clock frequency is >1GHz. Without this circuit changes, the GT L3 cache will not be functional at lower frequency due to Vcc is less than Vccmin of 0.9V for the array. default value: low2xthresh=0x01Eh corresponding to a 1x frequency of 500Mhz.

CONTEXT_SCHEDULING_ATTRIBUTES

CONTEXT_SCHEDULING_ATTRIBUTES - CONTEXT_SCHEDULING_ATTRIBUTES	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	025A8h-025ABh
Name:	CONTEXT_SCHEDULING_ATTRIBUTES
ShortName:	CONTEXT_SCHEDULING_ATTRIBUTES_RCSUNIT
Address:	185A8h-185ABh
Name:	CONTEXT_SCHEDULING_ATTRIBUTES
ShortName:	CONTEXT_SCHEDULING_ATTRIBUTES_POCSUNIT
Address:	225A8h-225ABh
Name:	CONTEXT_SCHEDULING_ATTRIBUTES
ShortName:	CONTEXT_SCHEDULING_ATTRIBUTES_BCSUNIT
Address:	1C05A8h-1C05ABh
Name:	CONTEXT_SCHEDULING_ATTRIBUTES
ShortName:	CONTEXT_SCHEDULING_ATTRIBUTES_VCSUNIT0
Address:	1C45A8h-1C45ABh
Name:	CONTEXT_SCHEDULING_ATTRIBUTES
ShortName:	CONTEXT_SCHEDULING_ATTRIBUTES_VCSUNIT1
Address:	1C85A8h-1C85ABh
Name:	CONTEXT_SCHEDULING_ATTRIBUTES
ShortName:	CONTEXT_SCHEDULING_ATTRIBUTES_VECSUNIT0
Address:	1D05A8h-1D05ABh
Name:	CONTEXT_SCHEDULING_ATTRIBUTES
ShortName:	CONTEXT_SCHEDULING_ATTRIBUTES_VCSUNIT2
Address:	1D45A8h-1D45ABh
Name:	CONTEXT_SCHEDULING_ATTRIBUTES
ShortName:	CONTEXT_SCHEDULING_ATTRIBUTES_VCSUNIT3
Address:	1D85A8h-1D85ABh
Name:	CONTEXT_SCHEDULING_ATTRIBUTES
ShortName:	CONTEXT_SCHEDULING_ATTRIBUTES_VECSUNIT1
Address:	1E05A8h-1E05ABh

CONTEXT_SCHEDULING_ATTRIBUTES - CONTEXT_SCHEDULING_ATTRIBUTES

Name:	CONTEXT_SCHEDULING_ATTRIBUTES
ShortName:	CONTEXT_SCHEDULING_ATTRIBUTES_VCSUNIT4
Address:	1E45A8h-1E45ABh
Name:	CONTEXT_SCHEDULING_ATTRIBUTES
ShortName:	CONTEXT_SCHEDULING_ATTRIBUTES_VCSUNIT5
Address:	1E85A8h-1E85ABh
Name:	CONTEXT_SCHEDULING_ATTRIBUTES
ShortName:	CONTEXT_SCHEDULING_ATTRIBUTES_VECSUNIT2
Address:	1F05A8h-1F05ABh
Name:	CONTEXT_SCHEDULING_ATTRIBUTES
ShortName:	CONTEXT_SCHEDULING_ATTRIBUTES_VCSUNIT6
Address:	1F45A8h-1F45ABh
Name:	CONTEXT_SCHEDULING_ATTRIBUTES
ShortName:	CONTEXT_SCHEDULING_ATTRIBUTES_VCSUNIT7
Address:	1F85A8h-1F85ABh
Name:	CONTEXT_SCHEDULING_ATTRIBUTES
ShortName:	CONTEXT_SCHEDULING_ATTRIBUTES_VECSUNIT3
Address:	1A5A8h-1A5ABh
Name:	CONTEXT_SCHEDULING_ATTRIBUTES
ShortName:	CONTEXT_SCHEDULING_ATTRIBUTES_CCSUNIT0

Programming Notes

This register is only functional on RenderCS and ComputeCS and must be only programmed on RenderCS and ComputeCS. This register is context save/restored on a context switch.

DWord	Bit	Description
0	31:16	**Mask** Access: WO Format: Mask Mask bits act as write enables for the bits in the lower bits of this register
	15:2	**Reserved** Format: PBC
	1:0	**Reserved** Format: PBC

Context Info of Per-process GTT Space HDW

CTXT_INFO_HDW - Context Info of Per-process GTT Space HDW	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register matches the format required by GAM in the Element Descriptor register. HW stores the programmed values of the Per Process GTT registers across RC6 and re-sends them to the GAM on a RC6 resume, causing GAM to attempt to reload per process table pointers from memory. To avoid loading Per Process structures un-necessarily and to avoid issues with stale Per Process loads (if a application or context that used a per process table terminates), SW shall program the Per Process registers when needed and disable right after use. This register is saved in the power context.

DWord	Bit	Description
0	31:0	**Context ID** This field is available for SW to set to any value it wishes. It can be used to differentiate different instances of the same context or to serve as a lookup value in a context structure so SW does not have to search for a context by LRCA. This ID will also appear in the status writes and can be used to associate status with submitted contexts.

Context Info of Per-process GTT Space LDW

CTXT_INFO_LDW - Context Info of Per-process GTT Space LDW

Register Space: MMIO: 0/2/0

Access: R/W

Size (in bits): 32

This register matches the format required by GAM in the Element Descriptor register. HW stores the programmed values of the Per Process GTT registers across RC6 and re-sends them to the GAM on a RC6 resume, causing GAM to attempt to reload per process table pointers from memory. To avoid loading Per Process structures un-necessarily and to avoid issues with stale Per Process loads (if a application or context that used a per process table terminates), SW shall program the Per Process registers when needed and disable right after use. This register is saved in the power context.

DWord	Bit	Description
0	31:12	**Logical Ring Context Address (LRCA)** Format: GraphicsAddress[31:12] This field contains the 4 KB-aligned address of the Logical Ring Context associated with this execlist element.
	11:9	**Function Number** GFX device is considered to be on Bus0 with device number of 2. Function number is normally assigned as "0" however for gfx virtualization; there would be different function numbers which needs to be attached to context.
	8	**Privileged Context / GGTT vs PPGTT mode** Differs in legacy vs advanced context modes: **In Legacy Context**: Defines the page tables to be used. This is how page walker come to know PPGTT vs GGTT selection for the entire context. **In Advanced Context**: Defines the privilege level for the context

Value	Name	Exists If
0h	Use Global GTT	[Context Type] == 'Legacy Context'
1h	Use Per-Process GTT	[Context Type] == 'Legacy Context'
0h	User Mode Context	[Context Type] == 'Advanced Context'
1h	Supervisor Mode Context	[Context Type] == 'Advanced Context'

Fault Handling

Value	Name
00b	Fault & Hang
01b	Fault & Halt/Wait
10b	Fault & Stream & Switch
11b	Reserved

Deeper IA coherency Support

In Advanced Context: Defines the level of IA coherency

Value	Name	Description
0h		IA coherency is provided at LLC level for all streams of GPU (i.e. gen7.5 like mode)

CTXT_INFO_LDW - Context Info of Per-process GTT Space LDW

	1h		IA coherency is provided at L3 level for EU data accesses of GPU

4 — A&D Support / 32&64b Address Support

Differs in legacy vs advanced context modes: **In Legacy Context**: Defines 32b vs 64b (48b canonical) addressing format **In Advanced Context**: Defines A&D bit support

Value	Name	Exists If
0h	32b addressing format	[Context Type] == 'Legacy Context'
1h	64b (48b canonical) addressing format	[Context Type] == 'Legacy Context'
0h	A&D bit management in page tables is NOT supported	[Context Type] == 'Advanced Context'
1h	A&D bit management in page tables is supported.	[Context Type] == 'Advanced Context'

3 — Context Type

Defines the context type. *Note that: Bits [8:4] differs in functions when legacy vs advanced context modes are selected.*

Value	Name	Description
0h	Advanced Context	Defines the rest of the advanced capabilities (i.e. OS page table support, fault models). Note that advanced context is not bounded to GPGPU.
1h	Legacy Context	Defines the context as legacy mode which is similar to prior

2 — Force Restore

Setting this bit will force a context restore operation when switching to this context even if the LRCA in the CCID register (normally the LRCA of the last context from the prior execlist) matches this one. Note that it is legal (and likely desirable) for the **Render Context Restore Inhibit** bit (part of the CTXT_SR_CTL register) in the context image being restored to also be set. The "ring" context is being forced to be restored from a newly initialized context despite a possible LRCA match; however, the render context for such a newly initialized context will likely be uninitialized and so should not be restored

1 — Scheduling Mode

Value	Name	Description
0h		Not valid
1h	**[Default]**	Indicates Execlist mode of scheduling.

0 — Valid

Indicates that element descriptor is valid. If GAM is programmed with an invalid descriptor, it will continue but flag an error.

Context Restore Request To TDL

TDL_CONTEXT_RESTORE - Context Restore Request To TDL

Register Space:	MMIO: 0/2/0
Access:	WO
Size (in bits):	32
Address:	0E418h

DWord	Bit	Description
0	31:17	**Reserved** Access: RO Format: MBZ
	16	**Context Restore Mask** Value: 1 — Name: — Description: Bit 0 and bit 16 both need to be 1 for Context restore request
	15:1	**Reserved** Access: RO Format: MBZ
	0	**Context Restore** Value: 1 — Name: — Description: Bit 0 and bit 16 both need to be 1 for Context restore request

Context Sizes

CXT_SIZE - Context Sizes

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	021A8h

The actual size of a logical rendering context is the amount of data stored/restored during a context switch and is measured in 64B cache lines.

This register will be power context save/restored. Note that this register will default to the correct value, so software should not have to modify it.

DWord	Bit	Description
0	31:28	**Reserved** Access: RO Format: MBZ
	27:16	**Render Engine Context Size** This field indicates the size of the render engine context data that needs to be save/restored when extended mode is not enabled for a context; this excludes URB context size. Note that this excludes the ring context image size and the engine context saved by CSFE. Value: 15Eh — Name: **[Default]**
	15:8	**SOL Context Offset** This field indicates the cacheline aligned offset of the SOL context in the render context image starting from Ring Context. Note that in execlist of scheduling Ring context itself is at 4KBoffset from LRCA. Value: 66h — Name: **[Default]**
	7:0	**Reserved** Access: RO Format: MBZ

Context Status Buffer1 Contents

CTXT_ST_BUF1 - Context Status Buffer1 Contents	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	384
_Custom_GTIReset:	DEV
Address:	023C0h-023EFh
Name:	Context Status Buffer1 Contents
ShortName:	CTXT_ST_BUF1_RCSUNIT
Address:	183C0h-183EFh
Name:	Context Status Buffer1 Contents
ShortName:	CTXT_ST_BUF1_POCSUNIT
Address:	223C0h-223EFh
Name:	Context Status Buffer1 Contents
ShortName:	CTXT_ST_BUF1_BCSUNIT
Address:	1C03C0h-1C03EFh
Name:	Context Status Buffer1 Contents
ShortName:	CTXT_ST_BUF1_VCSUNIT0
Address:	1C43C0h-1C43EFh
Name:	Context Status Buffer1 Contents
ShortName:	CTXT_ST_BUF1_VCSUNIT1
Address:	1C83C0h-1C83EFh
Name:	Context Status Buffer1 Contents
ShortName:	CTXT_ST_BUF1_VECSUNIT0
Address:	1D03C0h-1D03EFh
Name:	Context Status Buffer1 Contents
ShortName:	CTXT_ST_BUF1_VCSUNIT2
Address:	1D43C0h-1D43EFh
Name:	Context Status Buffer1 Contents
ShortName:	CTXT_ST_BUF1_VCSUNIT3
Address:	1D83C0h-1D83EFh
Name:	Context Status Buffer1 Contents
ShortName:	CTXT_ST_BUF1_VECSUNIT1
Address:	1E03C0h-1E03EFh
Name:	Context Status Buffer1 Contents

CTXT_ST_BUF1 - Context Status Buffer1 Contents

ShortName:	CTXT_ST_BUF1_VCSUNIT4

Address:	1E43C0h-1E43EFh
Name:	Context Status Buffer1 Contents
ShortName:	CTXT_ST_BUF1_VCSUNIT5

Address:	1E83C0h-1E83EFh
Name:	Context Status Buffer1 Contents
ShortName:	CTXT_ST_BUF1_VECSUNIT2

Address:	1F03C0h-1F03EFh
Name:	Context Status Buffer1 Contents
ShortName:	CTXT_ST_BUF1_VCSUNIT6

Address:	1F43C0h-1F43EFh
Name:	Context Status Buffer1 Contents
ShortName:	CTXT_ST_BUF1_VCSUNIT7

Address:	1F83C0h-1F83EFh
Name:	Context Status Buffer1 Contents
ShortName:	CTXT_ST_BUF1_VECSUNIT3

Address:	1A3C0h-1A3EFh
Name:	Context Status Buffer1 Contents
ShortName:	CTXT_ST_BUF1_CCSUNIT0

All "Context Status* LDW" have the format of the Bits[31:0] of the "Context Status" definition.
All "Context Status* UDW" have the format of the Bits[61:32] of the "Context Status" definition.

Programming Notes

This structure contains the Context Switch status locations Context Status 6 to Context Status 11.

DWord	Bit	Description
0	63:32	**Context Status 6 UDW** Format: U32
	31:0	**Context Status 6 LDW** Format: U32
1	63:32	**Context Status 7 UDW** Format: U32
	31:0	**Context Status 7 LDW** Format: U32
2	63:32	**Context Status 8 UDW** Format: U32

CTXT_ST_BUF1 - Context Status Buffer1 Contents		
	31:0	**Context Status 8 LDW** Format: U32
3	63:32	**Context Status 9 UDW** Format: U32
	31:0	**Context Status 9 LDW** Format: U32
4	63:32	**Context Status 10 UDW** Format: U32
	31:0	**Context Status 10 LDW** Format: U32
5	63:32	**Context Status 11 UDW** Format: U32
	31:0	**Context Status 11 LDW** Format: U32

Context Status Buffer Contents

CTXT_ST_BUF - Context Status Buffer Contents	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	384
_Custom_GTIReset:	DEV
Address:	02370h-0239Fh
Name:	Context Status Buffer Contents
ShortName:	CTXT_ST_BUF_RCSUNIT
Address:	18370h-1839Fh
Name:	Context Status Buffer Contents
ShortName:	CTXT_ST_BUF_POCSUNIT
Address:	22370h-2239Fh
Name:	Context Status Buffer Contents
ShortName:	CTXT_ST_BUF_BCSUNIT
Address:	1C0370h-1C039Fh
Name:	Context Status Buffer Contents
ShortName:	CTXT_ST_BUF_VCSUNIT0
Address:	1C4370h-1C439Fh
Name:	Context Status Buffer Contents
ShortName:	CTXT_ST_BUF_VCSUNIT1
Address:	1C8370h-1C839Fh
Name:	Context Status Buffer Contents
ShortName:	CTXT_ST_BUF_VECSUNIT0
Address:	1D0370h-1D039Fh
Name:	Context Status Buffer Contents
ShortName:	CTXT_ST_BUF_VCSUNIT2
Address:	1D4370h-1D439Fh
Name:	Context Status Buffer Contents
ShortName:	CTXT_ST_BUF_VCSUNIT3
Address:	1D8370h-1D839Fh
Name:	Context Status Buffer Contents
ShortName:	CTXT_ST_BUF_VECSUNIT1
Address:	1E0370h-1E039Fh
Name:	Context Status Buffer Contents

CTXT_ST_BUF - Context Status Buffer Contents

ShortName:	CTXT_ST_BUF_VCSUNIT4
Address:	1E4370h-1E439Fh
Name:	Context Status Buffer Contents
ShortName:	CTXT_ST_BUF_VCSUNIT5
Address:	1E8370h-1E839Fh
Name:	Context Status Buffer Contents
ShortName:	CTXT_ST_BUF_VECSUNIT2
Address:	1F0370h-1F039Fh
Name:	Context Status Buffer Contents
ShortName:	CTXT_ST_BUF_VCSUNIT6
Address:	1F4370h-1F439Fh
Name:	Context Status Buffer Contents
ShortName:	CTXT_ST_BUF_VCSUNIT7
Address:	1F8370h-1F839Fh
Name:	Context Status Buffer Contents
ShortName:	CTXT_ST_BUF_VECSUNIT3
Address:	1A370h-1A39Fh
Name:	Context Status Buffer Contents
ShortName:	CTXT_ST_BUF_CCSUNIT0

Contents of the Execlist 0 in HW.

All "Context Status* LDW" have the format of the Bits[31:0] of the "Context Status" definition.
All "Context Status* UDW" have the format of the Bits[61:32] of the "Context Status" definition.

Programming Notes

This structure contains the Context Switch status locations Context Status 0 to Context Status 5.

This register functionality is not supported and must not be programmed for Position command streamer.

DWord	Bit	Description
0	63:32	**Context Status 0 UDW** Format: U32
	31:0	**Context Status 0 LDW** Format: U32
1	63:32	**Context Status 1 UDW** Format: U32
	31:0	**Context Status 1 LDW** Format: U32

CTXT_ST_BUF - Context Status Buffer Contents		
2	63:32	**Context Status 2 UDW** Format: U32
	31:0	**Context Status 2 LDW** Format: U32
3	63:32	**Context Status 3 UDW** Format: U32
	31:0	**Context Status 3 LDW** Format: U32
4	63:32	**Context Status 4 UDW** Format: U32
	31:0	**Context Status 4 LDW** Format: U32
5	63:32	**Context Status 5 UDW** Format: U32
	31:0	**Context Status 5 LDW** Format: U32

Context Status Buffer Interrupt Mask Register

CSB_INTERRUPT_MASK - Context Status Buffer Interrupt Mask Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02218h-0221Bh
Name:	CSB_INTERRUPT_MASK
ShortName:	CSB_INTERRUPT_MASK_RCSUNIT
Address:	18218h-1821Bh
Name:	CSB_INTERRUPT_MASK
ShortName:	CSB_INTERRUPT_MASK_POCSUNIT
Address:	22218h-2221Bh
Name:	CSB_INTERRUPT_MASK
ShortName:	CSB_INTERRUPT_MASK_BCSUNIT
Address:	1C0218h-1C021Bh
Name:	CSB_INTERRUPT_MASK
ShortName:	CSB_INTERRUPT_MASK_VCSUNIT0
Address:	1C4218h-1C421Bh
Name:	CSB_INTERRUPT_MASK
ShortName:	CSB_INTERRUPT_MASK_VCSUNIT1
Address:	1C8218h-1C821Bh
Name:	CSB_INTERRUPT_MASK
ShortName:	CSB_INTERRUPT_MASK_VECSUNIT0
Address:	1D0218h-1D021Bh
Name:	CSB_INTERRUPT_MASK
ShortName:	CSB_INTERRUPT_MASK_VCSUNIT2
Address:	1D4218h-1D421Bh
Name:	CSB_INTERRUPT_MASK
ShortName:	CSB_INTERRUPT_MASK_VCSUNIT3
Address:	1D8218h-1D821Bh
Name:	CSB_INTERRUPT_MASK
ShortName:	CSB_INTERRUPT_MASK_VECSUNIT1
Address:	1E0218h-1E021Bh

CSB_INTERRUPT_MASK - Context Status Buffer Interrupt Mask Register

Name:	CSB_INTERRUPT_MASK
ShortName:	CSB_INTERRUPT_MASK_VCSUNIT4
Address:	1E4218h-1E421Bh
Name:	CSB_INTERRUPT_MASK
ShortName:	CSB_INTERRUPT_MASK_VCSUNIT5
Address:	1E8218h-1E821Bh
Name:	CSB_INTERRUPT_MASK
ShortName:	CSB_INTERRUPT_MASK_VECSUNIT2
Address:	1F0218h-1F021Bh
Name:	CSB_INTERRUPT_MASK
ShortName:	CSB_INTERRUPT_MASK_VCSUNIT6
Address:	1F4218h-1F421Bh
Name:	CSB_INTERRUPT_MASK
ShortName:	CSB_INTERRUPT_MASK_VCSUNIT7
Address:	1F8218h-1F821Bh
Name:	CSB_INTERRUPT_MASK
ShortName:	CSB_INTERRUPT_MASK_VECSUNIT3
Address:	1A218h-1A21Bh
Name:	CSB_INTERRUPT_MASK
ShortName:	CSB_INTERRUPT_MASK_CCSUNIT0

Hardware generates context switch interrupt and the associated context switch status report for the context switch reasons unmasked in this register. By default the context switch interrupts for all context switch reasons are un-masked. This register is privileged and global across all contexts and power context save/restored by hardware.
Note that on a context switch status report even the status of the masked context switch reasons are reported.

Programming Notes

Software must program this register through direct MMIO when hardware is idle and not processing any contexts.

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**Active to Idle** This mask bit controls the context switch interrupt generation and the associated context switch status report on a context switch leading hardware to go idle. Active-to-Idle is a special case of element switch due to ring done or un-successful semaphore wait or un-successful display wait

for event following which hardware goes idle.

Value	Name	Description
0	**[Default]**	Context switch interrupt and associated context switch status report is generated on a context switch leading hardware to go idle.
1		Context switch interrupt and associated context switch status report is not generated on a context switch leading hardware to go idle.

6 **Preempt to Idle**

This mask bit controls the context switch interrupt generation and the associated context switch status report on a context switch due to Preempt-to-Idle.

Value	Name	Description
0	**[Default]**	Context switch interrupt and associated context switch status report is generated on a context switch due to Preempt-to-Idle.
1		Context switch interrupt and associated context switch status report is not generated on a context switch due to Preempt-to-Idle.

5 **Lite Restore**

This mask bit controls the context switch interrupt generation and the associated context switch status report on a context switch due to lite restore.

Value	Name	Description
0	**[Default]**	Context switch interrupt and associated context switch status report is generated on a context switch due to lite restore.
1		Context switch interrupt and associated context switch status report is not generated on a context switch due to lite restore.

4 **Preemption**

This mask bit controls the context switch interrupt generation and the associated context switch status report on a context switch due to preemption. Preemption of an ongoing context is triggered due to loading of submission queue to execution queue on a "Load".

Value	Name	Description
0	**[Default]**	Context switch interrupt and associated context switch status report is generated on a context switch due to preemption.
1		Context switch interrupt and associated context switch status report is not generated on a context switch due to preemption.

3 **Display Wait For Event**

This mask bit controls the context switch interrupt generation and the associated context switch status report on Ring-Done context switch.

Value	Name	Description
0	**[Default]**	Context switch interrupt and associated context switch status report is generated on a context switch due to un-successful display wait for event. Context switch on un-successful display wait for even wait is a part of element switch.

CSB_INTERRUPT_MASK - Context Status Buffer Interrupt Mask Register

		1		Context switch interrupt and associated context switch status report is not generated on a context switch due to un-successful display wait for event.
	2	**Semaphore Wait** This mask bit controls the context switch interrupt generation and the associated context switch status report on Ring-Done context switch.		
		Value	**Name**	**Description**
		0	**[Default]**	Context switch interrupt and associated context switch status report is generated on a context switch due to un-successful sempahore wait. Context switch on un-successful semaphore wait is a part of element switch.
		1		Context switch interrupt and associated context switch status report is not generated on a context switch un-successful semaphore wait.
	1	**Ring Done** This mask bit controls the context switch interrupt generation and the associated context switch status report on Ring-Done context switch.		
		Value	**Name**	**Description**
		0	**[Default]**	Context switch interrupt and associated context switch status report is generated on a context switch reason due to Ring-Done. Context switch on ring done is a part of element switch.
		1		Context switch interrupt and associated context switch status report is not generated on a context switch due to Ring-Done.
	0	**Idle To Active** This mask bit controls the context switch interrupt generation and the associated context switch status report on Idle-to-Active context switch.		
		Value	**Name**	**Description**
		0	**[Default]**	Context switch interrupt and associated context switch status report is generated on a context switch due to Idle-to-Active. Idle2Active is a special case of submission queue acceptance by hardware.
		1		Context switch interrupt and associated context switch status report is not generated on a context switch due to Idle-to-Active.

Context Status Buffer Read Register

CSB_STATUS - Context Status Buffer Read Register	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	023A4h-023A7h
Name:	Context Status Buffer Read Register
ShortName:	CSB_STATUS_RCSUNIT
Address:	183A4h-183A7h
Name:	Context Status Buffer Read Register
ShortName:	CSB_STATUS_POCSUNIT
Address:	223A4h-223A7h
Name:	Context Status Buffer Read Register
ShortName:	CSB_STATUS_BCSUNIT
Address:	1C03A4h-1C03A7h
Name:	Context Status Buffer Read Register
ShortName:	CSB_STATUS_VCSUNIT0
Address:	1C43A4h-1C43A7h
Name:	Context Status Buffer Read Register
ShortName:	CSB_STATUS_VCSUNIT1
Address:	1C83A4h-1C83A7h
Name:	Context Status Buffer Read Register
ShortName:	CSB_STATUS_VECSUNIT0
Address:	1D03A4h-1D03A7h
Name:	Context Status Buffer Read Register
ShortName:	CSB_STATUS_VCSUNIT2
Address:	1D43A4h-1D43A7h
Name:	Context Status Buffer Read Register
ShortName:	CSB_STATUS_VCSUNIT3
Address:	1D83A4h-1D83A7h
Name:	Context Status Buffer Read Register
ShortName:	CSB_STATUS_VECSUNIT1
Address:	1E03A4h-1E03A7h
Name:	Context Status Buffer Read Register
ShortName:	CSB_STATUS_VCSUNIT4
Address:	1E43A4h-1E43A7h

CSB_STATUS - Context Status Buffer Read Register

Name:	Context Status Buffer Read Register
ShortName:	CSB_STATUS_VCSUNIT5
Address:	1E83A4h-1E83A7h
Name:	Context Status Buffer Read Register
ShortName:	CSB_STATUS_VECSUNIT2
Address:	1F03A4h-1F03A7h
Name:	Context Status Buffer Read Register
ShortName:	CSB_STATUS_VCSUNIT6
Address:	1F43A4h-1F43A7h
Name:	Context Status Buffer Read Register
ShortName:	CSB_STATUS_VCSUNIT7
Address:	1F83A4h-1F83A7h
Name:	Context Status Buffer Read Register
ShortName:	CSB_STATUS_VECSUNIT3
Address:	1A3A4h-1A3A7h
Name:	Context Status Buffer Read Register
ShortName:	CSB_STATUS_CCSUNIT0

This 32 bit address contains the value of the context status that is next to be read by scheduler.

Programming Notes

This register functionality is not supported and must not be programmed for Position command streamer.

DWord	Bit	Description
0	31:0	**Context Status** Access: RO This field contains the value of the Context Status depending on the value of **Context Status Buffer Read Pointer** in the CTXT_ST_PTR. Read Pointer value of zero will point to context status zero, read pointer value of one will pointer to context status one, and so on. The scheduler must read this register twice to get the full 64b of context status. The first read returns the lower DW of the status and the second read returns the upper DW of the status.

Context Timestamp Count

CTX_TIMESTAMP - Context Timestamp Count	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	023A8h-023ABh
Name:	Context Timestamp Count
ShortName:	CTX_TIMESTAMP_RCSUNIT
Address:	183A8h-183ABh
Name:	Context Timestamp Count
ShortName:	CTX_TIMESTAMP_POCSUNIT
Address:	223A8h-223ABh
Name:	Context Timestamp Count
ShortName:	CTX_TIMESTAMP_BCSUNIT
Address:	1C03A8h-1C03ABh
Name:	Context Timestamp Count
ShortName:	CTX_TIMESTAMP_VCSUNIT0
Address:	1C43A8h-1C43ABh
Name:	Context Timestamp Count
ShortName:	CTX_TIMESTAMP_VCSUNIT1
Address:	1C83A8h-1C83ABh
Name:	Context Timestamp Count
ShortName:	CTX_TIMESTAMP_VECSUNIT0
Address:	1D03A8h-1D03ABh
Name:	Context Timestamp Count
ShortName:	CTX_TIMESTAMP_VCSUNIT2
Address:	1D43A8h-1D43ABh
Name:	Context Timestamp Count
ShortName:	CTX_TIMESTAMP_VCSUNIT3
Address:	1D83A8h-1D83ABh
Name:	Context Timestamp Count
ShortName:	CTX_TIMESTAMP_VECSUNIT1
Address:	1E03A8h-1E03ABh
Name:	Context Timestamp Count

CTX_TIMESTAMP - Context Timestamp Count

ShortName:	CTX_TIMESTAMP_VCSUNIT4
Address:	1E43A8h-1E43ABh
Name:	Context Timestamp Count
ShortName:	CTX_TIMESTAMP_VCSUNIT5
Address:	1E83A8h-1E83ABh
Name:	Context Timestamp Count
ShortName:	CTX_TIMESTAMP_VECSUNIT2
Address:	1F03A8h-1F03ABh
Name:	Context Timestamp Count
ShortName:	CTX_TIMESTAMP_VCSUNIT6
Address:	1F43A8h-1F43ABh
Name:	Context Timestamp Count
ShortName:	CTX_TIMESTAMP_VCSUNIT7
Address:	1F83A8h-1F83ABh
Name:	Context Timestamp Count
ShortName:	CTX_TIMESTAMP_VECSUNIT3
Address:	1A3A8h-1A3ABh
Name:	Context Timestamp Count
ShortName:	CTX_TIMESTAMP_CCSUNIT0

This register provides a mechanism to obtain cumulative run time of a GPU context on HW. This register gets context save/restored on a context switch. SW must reset this register on very first submission of a context to HW, then afterwards gets context save/restored maintaining the cumulative run time of the corresponding context. This register (effectively) counts at a constant frequency by adjusting the increment amount according to the actual reference clock frequency. SW therefore does not need to know the reference clock frequency. This register gets reset on an engine reset.

This register is context save restore on a context switch.

DWord	Bit	Description
0	31:0	**Timestamp Value** Format: U32 The granularity of this toggle is at the rate of the bit 3 in the "Reported Timestamp Count" register(0x2358).. The toggle will be 8 times slower that "Reported Timestamp Count". The granularity of the time stamp base unit for "Reported Timestamp Count" is defined in the Timestamp Bases subsection in Power Management chapter.

Control Register for Power Management

WAIT_FOR_RC6_EXIT - Control Register for Power Management	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	020CCh-020CFh
Name:	Control Register for Power Management
ShortName:	WAIT_FOR_RC6_EXIT_RCSUNIT
Address:	180CCh-180CFh
Name:	Control Register for Power Management
ShortName:	WAIT_FOR_RC6_EXIT_POCSUNIT
Address:	220CCh-220CFh
Name:	Control Register for Power Management
ShortName:	WAIT_FOR_RC6_EXIT_BCSUNIT
Address:	1C00CCh-1C00CFh
Name:	Control Register for Power Management
ShortName:	WAIT_FOR_RC6_EXIT_VCSUNIT0
Address:	1C40CCh-1C40CFh
Name:	Control Register for Power Management
ShortName:	WAIT_FOR_RC6_EXIT_VCSUNIT1
Address:	1C80CCh-1C80CFh
Name:	Control Register for Power Management
ShortName:	WAIT_FOR_RC6_EXIT_VECSUNIT0
Address:	1D00CCh-1D00CFh
Name:	Control Register for Power Management
ShortName:	WAIT_FOR_RC6_EXIT_VCSUNIT2
Address:	1D40CCh-1D40CFh
Name:	Control Register for Power Management
ShortName:	WAIT_FOR_RC6_EXIT_VCSUNIT3
Address:	1D80CCh-1D80CFh
Name:	Control Register for Power Management
ShortName:	WAIT_FOR_RC6_EXIT_VECSUNIT1
Address:	1E00CCh-1E00CFh
Name:	Control Register for Power Management

WAIT_FOR_RC6_EXIT - Control Register for Power Management

ShortName:	WAIT_FOR_RC6_EXIT_VCSUNIT4
Address:	1E40CCh-1E40CFh
Name:	Control Register for Power Management
ShortName:	WAIT_FOR_RC6_EXIT_VCSUNIT5
Address:	1E80CCh-1E80CFh
Name:	Control Register for Power Management
ShortName:	WAIT_FOR_RC6_EXIT_VECSUNIT2
Address:	1F00CCh-1F00CFh
Name:	Control Register for Power Management
ShortName:	WAIT_FOR_RC6_EXIT_VCSUNIT6
Address:	1F40CCh-1F40CFh
Name:	Control Register for Power Management
ShortName:	WAIT_FOR_RC6_EXIT_VCSUNIT7
Address:	1F80CCh-1F80CFh
Name:	Control Register for Power Management
ShortName:	WAIT_FOR_RC6_EXIT_VECSUNIT3
Address:	1A0CCh-1A0CFh
Name:	Control Register for Power Management
ShortName:	WAIT_FOR_RC6_EXIT_CCSUNIT0

This register gets power context save/restored. Bit[0] contents of this register does't get save/restored.Note: Even though this register exists in BlitterCS, VideoCS, VidoeEnhancmentCS, individual bit driven functionality is not supported.

Programming Notes
This register is functional for RenderCS only and must not be programmed for other command streamers.
This register functionality is not supported and must not be programmed for Position command streamer.

DWord	Bit	Description
0	31:16	**Mask** Access: WO Format: Mask Must be set to modify corresponding bit in Bits 15:0. (All implemented bits)
	15	**Selective Read Addressing Enable** Source: RenderCS, ComputeCS This field controls the outbound read request originating from Render Command Streamer. This field enables to read the MMIO register from selected unit in a given slice and sub-slice instead of multicasting the read cycle to all slices/sub-slices.

WAIT_FOR_RC6_EXIT - Control Register for Power Management

Value	Name	Description
0h	**[Default]**	Lowest Slice and Lowest Sub-Slice Enabled. Ex: Slice-0, Sub-Slice-0 are the lowest in GT.
1h	Selective Unit Enabled	Unit selected based on **Selective Read Slice Select** and **Selective Read Sub-Slice Select**.

14 **Reserved**

Source:	RenderCS, ComputeCS
Format:	PBC

13:12 **Selective Read Slice Select**

Source:	RenderCS, ComputeCS

This field selects the slice from which the read return data value has to be considered when **Selective Read Addressing Enable** is set. Below value must be programmed with a legal value supported by the GT configuration, one must not program this to a slice that is disabled or not supported by GT.

Value	Name
0h	Slice-0
1h	Slice-1
2h	Slice-2
3h	Slice-3

11:9 **Selective Read Sub-Slice Select**

Source:	RenderCS, ComputeCS

This field selects the sub-slice from which the read return data value has to be considered when **Selective Read Addressing Enable** is set. Below value must be programmed with a legal value supported by the GT configuration, one must not program this to a sub-slice that is disabled or not supported by GT.

Value	Name
000b	Sub Slice-0
001b	Sub Slice-1
010b	Sub Slice-2
011b	Sub Slice-3
100b	Sub Slice-4
101b	Sub Slice-5
110b	Sub Slice-6
111b	Sub Slice-7

8 **Selective Write Addressing Enable**

Source:	RenderCS, ComputeCS

This field controls the outbound write request on message channel originating from Render Command Streamer on executing LRI, LRR and LRM commands. Setting this field doesn't affect the execution of LRI commands from context image during context restore. This field enables to

select the unit in given slice and sub-slice instead of multicasting the write cycle to all slices/half-slices.

Value	Name	Description
0h	Multi Cast **[Default]**	
1h	Selective Unit Enabled	Unit selected based on **Selective Write Slice Select** and **Selective Write Sub-Slice Select**.

Programming Notes
This field is used to implement per-slice performance counting by limiting which slices receive Flex EU programming.

7 **Reserved**

Source:	RenderCS, ComputeCS
Format:	PBC

6:5 **Selective Write Slice Select**

Source:	RenderCS, ComputeCS

This field selects the slice to which the write has to be done when **Selective Write Addressing Enable** is set. Below value must be programmed with a legal value supported by the GT configuration, one must not program this to a slice that is disabled or not supported by GT.

Value	Name
000b	Slice-0
001b	Slice-1
010b	Slice-2
011b	Slice-3

Programming Notes
This field is used to implement per-slice performance counting by limiting which slices receive Flex EU programming. Please refer to Flex EU event for more details.

4:2 **Selective Write Sub-Slice Select**

Source:	RenderCS, ComputeCS

This field selects the Sub-Slice to which the write has to be done when **Selective Write Addressing Enable** is set. Below value must be programmed with a legal value supported by the GT configuration, one must not program this to a sub-slice that is disabled or not supported by GT.

Value	Name
000b	Sub Slice-0
001b	Sub Slice-1
010b	Sub Slice-2
011b	Sub Slice-3

WAIT_FOR_RC6_EXIT - Control Register for Power Management

		100b: Sub Slice-4 101b: Sub Slice-5 110b: Sub Slice-6 111b: Sub Slice-7
	1	**Reserved** Source: RenderCS, ComputeCS Format: PBC
	0	**Reserved** Source: RenderCS, ComputeCS Format: PBC

Count Active Channels Dispatched

TS_GPGPU_THREADS_DISPATCHED - Count Active Channels Dispatched

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV

Address:	02290h-02297h
Name:	Count Active Channels Dispatched
ShortName:	TS_GPGPU_THREADS_DISPATCHED_RCSUNIT_BE_COMPUTE

Address:	1A290h-1A297h
Name:	Count Active Channels Dispatched
ShortName:	TS_GPGPU_THREADS_DISPATCHED_CCSUNIT_BE_COMPUTE0

This register is used to count the number of active channels that TS sends for dispatch. For each dispatch the active bits in the execution mask are summed and added to this register. This register is reset when a write occurs to 2290h

DWord	Bit	Description
0..1	63:32	**GPGPU_THREADS_DISPATCHED UDW** Format: U32 This count is increased by the number of active bits in the execution mask each time the TS sends a GPGPU dispatch.
	31:0	**GPGPU_THREADS_DISPATCHED LDW** Format: U32 This count is increased by the number of active bits in the execution mask each time the TS sends a GPGPU dispatch.

Count of C6 Latency maximum

C6_EXIT_LATENCY_MAX - Count of C6 Latency maximum		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	0A620h	
Name:	Count of C6 Latency maximum	
ShortName:	C6_EXIT_LATENCY_MAX	
This register keeps track of the maximum Latency taken for C6 exit i.e from Boot evt to go to DILE		
DWord	**Bit**	**Description**
0	31:0	**C6 Latency Maximum** Access: RO _Custom_GTIReset: DEV Maximum C6 Latency

Count of Media License latency

CNT_MED_LIC_LAT - Count of Media License latency	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	0A0ECh
Name:	Count of Media License Latency
ShortName:	CNT_MED_LIC_LAT
This register keeps track of the Latency taken for Media License	

DWord	Bit	Description
0	31:0	**Media License Latency Count** Access: RO _Custom_GTIReset: BUS Count of Media License Latency

Count of Media License latency Maximum

CNT_MED_LIC_LAT_MAX - Count of Media License latency Maximum

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	0A0F8h
Name:	Count of Media License Latency Maximum
ShortName:	CNT_MED_LIC_LAT_MAX

This register keeps track of the Maximum Latency taken for Media License

DWord	Bit	Description
0	31:0	**Media License Latency Maximum Count** Access: RO _Custom_GTIReset: BUS Count of Media License Latency Maximum

Count of Media License Request

CNT_MED_LIC_REQ - Count of Media License Request	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	0A0E0h
Name:	Count of Media License Request
ShortName:	CNT_MED_LIC_REQ

This register keeps track of the number of Media License requests that have been sent

DWord	Bit	Description
0	31:0	**Media License Request Count** Access: RO _Custom_GTIReset: BUS Count of Media License Request

Count of Media Sampler License latency

CNT_MSAMP_LIC_LAT - Count of Media Sampler License latency	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	0A104h
Name:	Count of Media Sampler License latency
ShortName:	CNT_MSAMP_LIC_LAT

This register keeps track of the Latency taken for Media Sampler License

DWord	Bit	Description
0	31:0	**Media Sampler License Latency** Default Value: 000000000000000b Access: RO _Custom_GTIReset: BUS

Count of Media Sampler License latency Maximum

CNT_MSAMP_LIC_LAT_MAX - Count of Media Sampler License latency Maximum

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	0A108h
Name:	Count of Media Sampler License latency Maximum
ShortName:	CNT_MSAMP_LIC_LAT_MAX

This register keeps track of the Maximum Latency taken for Media Sampler License

DWord	Bit	Description
0	31:0	**Maximun Latency for Media Sampler Licensing Request** Default Value: 0000000000000000b Access: RO _Custom_GTIReset: BUS

Count of Media Sampler License Request

CNT_MSAMP_LIC_REQ - Count of Media Sampler License Request	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	0A100h
Name:	Count of Media Sampler License Request
ShortName:	CNT_MSAMP_LIC_REQ

This register keeps track of how many license requests have been sent out for Media Sampler

DWord	Bit	Description
0	31:0	**Number of Media Sampler License Request** Default Value: 000000000000000b Access: RO _Custom_GTIReset: BUS

Count of Render License latency

CNT_REN_LIC_LAT - Count of Render License latency		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	0A0E8h	
Name:	Count of Render License Latency	
ShortName:	CNT_REN_LIC_LAT	
This register keeps track of the Latency taken for Render License		
DWord	**Bit**	**Description**
0	31:0	**Render License Latency Count** Access: RO _Custom_GTIReset: BUS Count of Render License Latency

Count of Render License latency Maximum

CNT_REN_LIC_LAT_MAX - Count of Render License latency Maximum	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	0A0F4h
Name:	Count of Render License Latency Maximum
ShortName:	CNT_REN_LIC_LAT_MAX
This register keeps track of the Maximum Latency taken for Render License	

DWord	Bit	Description
0	31:0	**Render License Latency Maximum Value** Access: RO _Custom_GTIReset: BUS Count of Render License Latency Maximum

Count of Render License Request

CNT_REN_LIC_REQ - Count of Render License Request		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	0A0DCh	
Name:	Count of Render License Request	
ShortName:	CNT_REN_LIC_REQ	
This register keeps track of the number of Render License Count		
DWord	**Bit**	**Description**
0	31:0	**Render License Request Count** Access: RO _Custom_GTIReset: BUS Count of Render License Request

Count of spec License Request

CNT_SPEC_LIC_REQ - Count of spec License Request		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	0A0E4h	
Name:	Count of spec License Request	
ShortName:	CNT_SPEC_LIC_REQ	
DWord	**Bit**	**Description**
0	31:0	**Spec License Request Count** Access: RO _Custom_GTIReset: BUS Count of spec License Request

Count of Speculative License latency

CNT_SPEC_LIC_LAT - Count of Speculative License latency		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	0A0F0h	
Name:	Count of Speculative License Latency	
ShortName:	CNT_SPEC_LIC_LAT	
DWord	**Bit**	**Description**
0	31:0	**Speculative License Latency Value** Access: RO _Custom_GTIReset: BUS Count of Speculative License Latency

Count of Speculative License latency Maximum

CNT_SPEC_LIC_LAT_MAX - Count of Speculative License latency Maximum

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	0A0FCh
Name:	Count of Speculative License Latency Maximum
ShortName:	CNT_SPEC_LIC_LAT_MAX

This register keeps track of the Maximum Latency taken for Speculative License

DWord	Bit	Description
0	31:0	**Speculative License Latency Maximum Value** Access: RO _Custom_GTIReset: BUS Count of Speculative License Latency Maximum

CPS Invocation Counter

CPS_INVOCATION_COUNT - CPS Invocation Counter	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02478h
Name:	CPS Invocation Counter
ShortName:	CPS_INVOCATION_COUNT

This register stores the value of the coarse pixel count shaded. This register is part of the context save and restore.

DWord	Bit	Description
0..1	63:32	**CPS Invocation Count Report UDW** Number of coarse pixels that are dispatched as threads by the PS Stage. Updated only when **Statistics Enable** is set in **3DSTATE_CPS**.
	31:0	**CPS Invocation Count Report LDW** Number of coarse pixels that are dispatched as threads by the PS Stage. Updated only when **Statistics Enable** is set in **3DSTATE_CPS**.

CSC_COEFF

CSC_COEFF	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	192
_Custom_Display_DoubleBufferArmedBy:	Write to CSC_MODE
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank after armed

Address:	49010h-49027h
Name:	Pipe CSC Coefficients
ShortName:	CSC_COEFF_A
Reset:	soft

Address:	49110h-49127h
Name:	Pipe CSC Coefficients
ShortName:	CSC_COEFF_B
Reset:	soft

Address:	49210h-49227h
Name:	Pipe CSC Coefficients
ShortName:	CSC_COEFF_C
Reset:	soft

Address:	49310h-49327h
Name:	Pipe CSC Coefficients
ShortName:	CSC_COEFF_D
Reset:	soft

DWord	Bit	Description
0	31:16	**RY** Format: CSC COEFFICIENT FORMAT
	15:0	**GY** Format: CSC COEFFICIENT FORMAT
1	31:16	**BY** Format: CSC COEFFICIENT FORMAT
	15:0	**Reserved** Access: RO Format: MBZ
2	31:16	**RU** Format: CSC COEFFICIENT FORMAT

CSC_COEFF		
	15:0	**GU** Format: CSC COEFFICIENT FORMAT
3	31:16	**BU** Format: CSC COEFFICIENT FORMAT
	15:0	**Reserved** Access: RO Format: MBZ
4	31:16	**RV** Format: CSC COEFFICIENT FORMAT
	15:0	**GV** Format: CSC COEFFICIENT FORMAT
5	31:16	**BV** Format: CSC COEFFICIENT FORMAT
	15:0	**Reserved** Access: RO Format: MBZ

CSC_MODE

CSC_MODE	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank OR pipe disabled
Address:	49028h-4902Bh
Name:	Pipe CSC Mode
ShortName:	CSC_MODE_A
Reset:	soft
Address:	49128h-4912Bh
Name:	Pipe CSC Mode
ShortName:	CSC_MODE_B
Reset:	soft
Address:	49228h-4922Bh
Name:	Pipe CSC Mode
ShortName:	CSC_MODE_C
Reset:	soft
Address:	49328h-4932Bh
Name:	Pipe CSC Mode
ShortName:	CSC_MODE_D
Reset:	soft

Writes to this register arm CSC registers for this pipe.

DWord	Bit	Description
0	31	**Pipe CSC Enable** This bit enables the pipe color space conversion.
	30	**Pipe Output CSC Enable** This bit enables the pipe output color space conversion.
	29	**Allow Double Buffer Update Disable** Access: R/W This field controls whether double buffer updates are allowed to be disabled for the Color Space Conversionregisters that are double buffered. The DOUBLE_BUFFER_CTL register can be configured to globally disable double buffer updates for those resources that allow them to be disabled. Value: 0b — Not Allowed Value: 1b — Allowed **[Default]**

CSC_MODE		
	28	**Reserved** Access: RO Format: MBZ
	27:0	**Reserved** Access: RO Format: MBZ

CSC_POSTOFF

CSC_POSTOFF	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	96
_Custom_Display_DoubleBufferArmedBy:	Write to CSC_MODE
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank after armed

Address:	49040h-4904Bh
Name:	Pipe CSC Post-Offsets
ShortName:	CSC_POSTOFF_A
Reset:	soft

Address:	49140h-4914Bh
Name:	Pipe CSC Post-Offsets
ShortName:	CSC_POSTOFF_B
Reset:	soft

Address:	49240h-4924Bh
Name:	Pipe CSC Post-Offsets
ShortName:	CSC_POSTOFF_C
Reset:	soft

Address:	49340h-4934Bh
Name:	Pipe CSC Post-Offsets
ShortName:	CSC_POSTOFF_D
Reset:	soft

The post-offset is intended to add an offset from 0 on the Y or RGB channels and to convert UV channels from 2's complement to excess 0.5 as they exit pipe color space conversion (CSC).

DWord	Bit	Description
0	31:13	**Reserved** Access: RO Format: MBZ
	12:0	**PostCSC High Offset** This value is used to give an offset to the high color channel as it exits CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).
1	31:13	**Reserved** Access: RO Format: MBZ

CSC_POSTOFF		
	12:0	**PostCSC Medium Offset** This value is used to give an offset to the medium color channel as it exits CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).
2	31:13	**Reserved** Access: RO Format: MBZ
	12:0	**PostCSC Low Offset** This value is used to give an offset to the low color channel as it exits CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).

CSC_PREOFF

CSC_PREOFF	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	96
_Custom_Display_DoubleBufferArmedBy:	Write to CSC_MODE
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank after armed

Address:	49030h-4903Bh
Name:	Pipe CSC Pre-Offsets
ShortName:	CSC_PREOFF_A
Reset:	soft

Address:	49130h-4913Bh
Name:	Pipe CSC Pre-Offsets
ShortName:	CSC_PREOFF_B
Reset:	soft

Address:	49230h-4923Bh
Name:	Pipe CSC Pre-Offsets
ShortName:	CSC_PREOFF_C
Reset:	soft

Address:	49330h-4933Bh
Name:	Pipe CSC Pre-Offsets
ShortName:	CSC_PREOFF_D
Reset:	soft

The pre-offset is intended to remove an offset from 0 on the Y or RGB channels and to convert UV channels from excess 0.5 to 2's complement as they enter pipe color space conversion (CSC).

DWord	Bit	Description
0	31:13	**Reserved** Access: RO Format: MBZ
	12:0	**PreCSC High Offset** This value is used to give an offset to the high color channel as it enters CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).
1	31:13	**Reserved** Access: RO Format: MBZ

CSC_PREOFF		
	12:0	**PreCSC Medium Offset** This value is used to give an offset to the medium color channel as it enters CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).
2	31:13	**Reserved** Access: RO Format: MBZ
	12:0	**PreCSC Low Offset** This value is used to give an offset to the low color channel as it enters CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).

CS CSB

CS_CSB - CS CSB		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
SW reads this offset to read the Context Status Buffer entry at the top of the CSB fifo. Reads must occur in pairs to obtain a single 64bit CSB entry. The second read pops the CSB entry off the FIFO.		
DWord	**Bit**	**Description**
0	31:0	**Context Status Buffer DW** Access: RO This DW holds CSB bits[31:0] for the first read and CSB bits[63:32] for the second read.

CS CSB _FSR

CS_CSB_FSR - CS CSB _FSR		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
This RO register holds status of the CSB fifo.		
DWord	**Bit**	**Description**
0	31	**Not Empty** Access: RO
	30:16	**Reserved** Access: RO Format: MBZ
	15:8	**FIFO Maximum Occupancy Count** This field is a read-only field. It reflects the depth of the FIFO, which is a static value for each product.
	7:0	**FIFO Occupancy Count**

CSFE FSM2

CSFEFSM2 - CSFE FSM2	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	022A4h-022A7h
Name:	CSFE FSM2
ShortName:	CSFEFSM2_RCSUNIT
Address:	182A4h-182A7h
Name:	CSFE FSM2
ShortName:	CSFEFSM2_POCSUNIT
Address:	222A4h-222A7h
Name:	CSFE FSM2
ShortName:	CSFEFSM2_BCSUNIT
Address:	1C02A4h-1C02A7h
Name:	CSFE FSM2
ShortName:	CSFEFSM2_VCSUNIT0
Address:	1C42A4h-1C42A7h
Name:	CSFE FSM2
ShortName:	CSFEFSM2_VCSUNIT1
Address:	1C82A4h-1C82A7h
Name:	CSFE FSM2
ShortName:	CSFEFSM2_VECSUNIT0
Address:	1D02A4h-1D02A7h
Name:	CSFE FSM2
ShortName:	CSFEFSM2_VCSUNIT2
Address:	1D42A4h-1D42A7h
Name:	CSFE FSM2
ShortName:	CSFEFSM2_VCSUNIT3
Address:	1D82A4h-1D82A7h
Name:	CSFE FSM2
ShortName:	CSFEFSM2_VECSUNIT1
Address:	1E02A4h-1E02A7h
Name:	CSFE FSM2

CSFEFSM2 - CSFE FSM2

ShortName:	CSFEFSM2_VCSUNIT4
Address:	1E42A4h-1E42A7h
Name:	CSFE FSM2
ShortName:	CSFEFSM2_VCSUNIT5
Address:	1E82A4h-1E82A7h
Name:	CSFE FSM2
ShortName:	CSFEFSM2_VECSUNIT2
Address:	1F02A4h-1F02A7h
Name:	CSFE FSM2
ShortName:	CSFEFSM2_VCSUNIT6
Address:	1F42A4h-1F42A7h
Name:	CSFE FSM2
ShortName:	CSFEFSM2_VCSUNIT7
Address:	1F82A4h-1F82A7h
Name:	CSFE FSM2
ShortName:	CSFEFSM2_VECSUNIT3
Address:	1A2A4h-1A2A7h
Name:	CSFE FSM2
ShortName:	CSFEFSM2_CCSUNIT0

DWord	Bit	Description
0	31:30	**POCSLITERST** Format: MBZ
	29:27	**POCSELSUB** Format: MBZ
	26:23	**CTXSWM** Format: MBZ
	22:20	**CSRL_PREEMPT** Format: MBZ
	19:16	**CSCTXSW_CTXSEQ** Format: MBZ
	15:13	**CSPREP4SWITCH** Format: MBZ
	12:11	**SVWR** Format: MBZ

CSFEFSM2 - CSFE FSM2		
	10:9	**RRCRD** Format: MBZ
	8:6	**RENDCTX** Format: MBZ
	5:3	**CTXSEQ** Format: MBZ
	2:0	**RCURSTSEQ**

CS Indirect Base

CS_INDIRECT_BASE - CS Indirect Base

Register Space: MMIO: 0/2/0

Size (in bits): 32

This is a general-purpose register to hold data. This register is power context save/restored across idle sequences.
Data in this register is used as a base address in below mentioned MI commands upon ADD_INDIRECT_BASE field set.

- MI_LOAD_REGISTER_IMM
- MI_STORE_REGISTER_MEM
- MI_LOAD_REGISTER_MEM
- MI_LOAD_REGISTER_REG

DWord	Bit	Description
0	31:0	**Indirect Base** Access: R/W Format: U32

CS-LTISEQ Flush Message Register

CS_LTISEQ_FLSH_MSG - CS-LTISEQ Flush Message Register

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B464h

This register handles flush messaging between CS and LTISEQ.
Valid values are :
1. 0x00010001 : Tile Cache Flush Request
2. 0x00020002 : RCS Fabric Flush Request
2. 0x00040004 : CCS Fabric Flush Request

DWord	Bit	Description
0	31:3	**Reserved** Access: RO Format: MBZ
	2	**CCS Fabric Flush Request Message Bit** Access: R/W Hardware Clear _Custom_GTIReset: DEV Message Bit of Fabric flush request from CCS to LTISEQ. Bit[2] must be set to "1", along with Bit[18]=1, for Valid Fabric Flush Requests from Compute CS to LTISEQ.
	1	**Fabric Flush Request Message Bit** Access: R/W Hardware Clear _Custom_GTIReset: DEV Message Bit of Fabric flush request from CS to LTISEQ. Bit[1] must be set to "1", along with Bit[17]=1, for Valid Fabric Flush Requests from Render CS to LTISEQ.
	0	**Tile Cache Flush Request Message Bit** Access: R/W Hardware Clear _Custom_GTIReset: DEV Message Bit of Tile Cache flush request from CS to LTISEQ. Bit[0] must be set to "1", along with Bit[16]=1, for Valid Tile Cache Flush Requests from CS to LTISEQ.

CSPREEMPT

CSPREEMPT - CSPREEMPT

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

Address:	024B0h
Name:	CSPREEMPT
ShortName:	CSPREEMPT

Address:	224B0h
Name:	BCSPREEMPT
ShortName:	BCSPREEMPT

Address:	124B0h
Name:	VCSPREEMPT
ShortName:	VCSPREEMPT

Address:	1A4B0h
Name:	VECSPREEMPT
ShortName:	VECSPREEMPT

Programming Notes

This is for HW internal usage and must not be written by SW.

DWord	Bit	Description
0	31:16	**Mask Bits** Format: Mask[15:0] Must be set to modify corresponding bit in Bits 15:0. (All implemented bits)
	15:1	**Reserved** Access: RO Format: MBZ
	0	**Unnamed** Format: Disable This is a message bit written by the cross CS in case of GT4-CBR/SFR mode of operation. To set this bit both bit[0] and bit[16] (mask) needs to be set. This bit set indicates CS in other GT has reached a preemption point. This bit gets reset by CS when preemption takes place.

CUR_BASE

CUR_BASE	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled
Address:	70084h-70087h
Name:	Cursor Base Address
ShortName:	CUR_BASE_A
Reset:	soft
Address:	71084h-71087h
Name:	Cursor Base Address
ShortName:	CUR_BASE_B
Reset:	soft
Address:	72084h-72087h
Name:	Cursor Base Address
ShortName:	CUR_BASE_C
Reset:	soft
Address:	73084h-73087h
Name:	Cursor Base Address
ShortName:	CUR_BASE_D
Reset:	soft

Writes to this register arm cursor registers for this pipe.

DWord	Bit	Description
0	31:12	**Cursor Base 31 12** Format: GraphicsAddress[31:12] This field specifies bits 31:12 of the graphics address of the base of the cursor for hi-res mode. When performing 180 degree rotation, this address does not need to change, hardware will internally offset to start from the last pixel of the last line of the cursor. **Workaround** To prevent false VT-d type 6 errors, use 64KB address alignment and allocate an extra 2 Page Table Entries (PTEs) beyond the end of the displayed surface. Only the PTEs will be used, not the pages themselves. **Restriction** The cursor surface address must be 4K byte aligned. The cursor must be in linear memory, it cannot be tiled.

CUR_BASE		
	11	**Reserved**
	10:7	**Reserved** Access: RO Format: MBZ
	6:4	**Reserved**
	3	**Reserved** Access: RO Format: MBZ
	2	**Reserved**
	1:0	**Reserved** Access: RO Format: MBZ

CUR_COLOR_CTL

CUR_COLOR_CTL	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to CUR_BASE or cursor not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled; after armed

Address:	700C0h-700C3h
Name:	Cursor Color Control
ShortName:	CUR_COLOR_CTL_A
Reset:	soft

Address:	710C0h-710C3h
Name:	Cursor Color Control
ShortName:	CUR_COLOR_CTL_B
Reset:	soft

Address:	720C0h-720C3h
Name:	Cursor Color Control
ShortName:	CUR_COLOR_CTL_C
Reset:	soft

Address:	730C0h-730C3h
Name:	Cursor Color Control
ShortName:	CUR_COLOR_CTL_D
Reset:	soft

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15	**Tone Mapping Enable** This field enables the tone mapping of cursor pixels using the programmed tone mapping factor. 1b – Enable 0b – Disable
	14:10	**Reserved** Access: RO Format: MBZ

CUR_COLOR_CTL		
	9:0	**Tone Mapping Factor** This field specifies the tone mapping factor. Each color component gets corrected with this programmed 10 bit fractional value.

CUR_CSC_COEFF

CUR_CSC_COEFF	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	192
_Custom_Display_DoubleBufferArmedBy:	Write to CUR_CTL
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank after armed

Address:	700D0h-700E7h
Name:	Cursor CSC Coefficients
ShortName:	CUR_CSC_COEFF_A
Reset:	soft

Address:	710D0h-710E7h
Name:	Cursor CSC Coefficients
ShortName:	CUR_CSC_COEFF_B
Reset:	soft

Address:	720D0h-720E7h
Name:	Cursor CSC Coefficients
ShortName:	CUR_CSC_COEFF_C
Reset:	soft

Address:	730D0h-730E7h
Name:	Cursor CSC Coefficients
ShortName:	CUR_CSC_COEFF_D
Reset:	soft

DWord	Bit	Description
0	31:16	**RY** Format: CSC COEFFICIENT FORMAT
	15:0	**GY** Format: CSC COEFFICIENT FORMAT
1	31:16	**BY** Format: CSC COEFFICIENT FORMAT
	15:0	**Reserved** Access: RO Format: MBZ
2	31:16	**RU** Format: CSC COEFFICIENT FORMAT

CUR_CSC_COEFF		
	15:0	**GU** Format: CSC COEFFICIENT FORMAT
3	31:16	**BU** Format: CSC COEFFICIENT FORMAT
	15:0	**Reserved** Access: RO Format: MBZ
4	31:16	**RV** Format: CSC COEFFICIENT FORMAT
	15:0	**GV** Format: CSC COEFFICIENT FORMAT
5	31:16	**BV** Format: CSC COEFFICIENT FORMAT
	15:0	**Reserved** Access: RO Format: MBZ

CUR_CTL

CUR_CTL	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to CUR_BASE or cursor not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled; after armed

Address:	70080h-70083h
Name:	Cursor Control
ShortName:	CUR_CTL_A
Reset:	soft

Address:	71080h-71083h
Name:	Cursor Control
ShortName:	CUR_CTL_B
Reset:	soft

Address:	72080h-72083h
Name:	Cursor Control
ShortName:	CUR_CTL_C
Reset:	soft

Address:	73080h-73083h
Name:	Cursor Control
ShortName:	CUR_CTL_D
Reset:	soft

Address:	70880h-70883h
Name:	Selective Fetch Cursor Control
ShortName:	SEL_FETCH_CUR_CTL_A
Reset:	soft

The cursor is enabled by programming a valid cursor mode in the cursor mode select fields. The cursor is disabled by programming all 0s in the cursor mode select fields.

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30:28	**Pipe Slice Arbitration Slots** This field specifies the number of slots allocated to cursor in pipe slice request arbitration. This field is ignored when the 'PIPE_SLICE_ARBITRATION_CTL->Use Programmed Slots' is not set. This field is zero based; a programmed value of 0 results in 1 slot allocation.

CUR_CTL

27 Reserved

Access:	RO
Format:	MBZ

26 Gamma Enable

This bit enables pipe gamma correction for the cursor pixel data. In VGA pop-up operation, the cursor data will always bypass gamma.

This field is deprecated.

Value	Name
0b	Disable
1b	Enable

25 Reserved

Access:	RO
Format:	MBZ

24 Pipe CSC Enable

This bit enables pipe color space conversion for the cursor pixel data.

Use CSC_MODE.Pipe CSC Enable, GAMMA_MODE.*_GAMMA_ENABLE for enabling pipe color space conversion and gamma respectively across all pixels from all planes. Cursor CSC must be used for cursor specific color space conversion.

This field is deprecated.

Value	Name
0b	Disable
1b	Enable

23 Allow Double Buffer Update Disable

Access:	R/W

This field controls whether double buffer updates are allowed to be disabled for this cursor. The DOUBLE_BUFFER_CTL register can be configured to globally disable double buffer updates for resources that allow them to be disabled.

Value	Name
0b	Not Allowed
1b	Allowed

22:19 Reserved

Access:	RO
Format:	MBZ

CUR_CTL

18 Pre CSC Gamma Enable

This bit enables the cursor pre-CSC gamma for the cursor pixel data. This is generally used with HDR to de-gamma the sRGB cursor pixel data before the RGB2020 conversion.

Value	Name
0b	Disable
1b	Enable

17 Reserved

Access:	RO
Format:	MBZ

16 CSC Enable

This bit enables the cursor color space conversion for the cursor pixel data. Hardware uses the coefficients programmed in the CUR_CSC_COEFF registers to perform the color space conversion.

Value	Name
0b	Disable
1b	Enable

15 180 Rotation

This mode causes the cursor image to be rotated 180 degrees. In addition to setting this bit, the cursor position must be adjusted to match the physical orientation of the display.

Value	Name
0b	No rotation
1b	180 degree rotation

Restriction
Only 32 bits per pixel cursors can be rotated. This field must be zero when the cursor format is 2 bits per pixel.

14:12 Reserved

Access:	RO
Format:	MBZ

11:10 Force Alpha Plane Select

This field selects which planes the cursor alpha value will be forced for. It is used together with the Force Alpha Value field.

Value	Name	Description
00b	Disable	Disable alpha forcing
01b	Pipe CSC Enabled	Enable alpha forcing where cursor overlaps a plane that has enabled pipe CSC
10b	Pipe CSC Disabled	Enable alpha forcing where cursor overlaps plane that has disabled pipe CSC
11b	Reserved	Reserved

CUR_CTL

9:8 Force Alpha Value

This field controls the behavior of cursor when alpha blending onto certain plane pixels. It is used together with the Force Alpha Plane Select field.

Value	Name	Description
00b	Disable	Cursor pixels alpha blend normally over any plane.
01b	50	Cursor pixels with alpha >= 50% are made fully opaque where they overlap the selected plane(s). Cursor pixels with alpha < 50% are made fully transparent where they overlap the selected plane(s).
10b	75	Cursor pixels with alpha >= 75% are made fully opaque where they overlap the selected plane(s). Cursor pixels with alpha < 75% are made fully transparent where they overlap the selected plane(s).
11b	100	Cursor pixels with alpha = 100% are made fully opaque where they overlap the selected plane(s). Cursor pixels with alpha < 100% are made fully transparent where they overlap the selected plane(s).

Restriction
Force Alpha is only for use with ARGB cursor formats.

7:6 Reserved

Access:	RO
Format:	MBZ

5:0 Cursor Mode Select

This field selects the cursor mode. Cursor is disabled when the selection is 000000b and enabled when the selection is any other value. The cursor vertical size can be overriden by the size reduction mode.

Color channels should be pre-multiplied with alpha by software.

Value	Name	Description
000000b	Disable	Cursor is disabled
000010b	128x128 32bpp AND/INV	128x128 32bpp AND/INVERT
000011b	256x256 32bpp AND/INV	256x256 32bpp AND/INVERT
000100b	64x64 2bpp 3-color	64x64 2bpp Indexed 3-color and transparency
000101b	64x64 2bpp 2-color	64x64 2bpp Indexed AND/XOR 2-color
000110b	64x64 2bpp 4-color	64x64 2bpp Indexed 4-color
000111b	64x64 32bpp AND/INV	64x64 32bpp AND/INVERT
100010b	128x128 32bpp ARGB	128x128 32bpp ARGB (8:8:8:8 MSB-A:R:G:B)
100011b	256x256 32bpp ARGB	256x256 32bpp ARGB (8:8:8:8 MSB-A:R:G:B)
100100b	64x64 32bpp AND/XOR	64x64 32bpp AND/XOR
100101b	128x128 32bpp AND/XOR	128x128 32bpp AND/XOR
100110b	256x256 32bpp AND/XOR	256x256 32bpp AND/XOR
100111b	64x64 32bpp ARGB	64x64 32bpp ARGB (8:8:8:8 MSB-A:R:G:B)

Others	Reserved	Reserved

Programming Notes
INVERT, XOR, and alpha blends may not look as expected when the plane underlying the cursor is YUV or extended range RGB. Out of range RGB values will be clamped prior to alpha blending, INVERT, or XOR with cursor. It is recommended to use Force Alpha when cursor is alpha blending onto a plane of a different color space or extended gamut.
The AND/INVERT format uses the most significant byte (MSB) to control the color. If MSB is 0xFF: Cursor is opaque. Show cursor color from three least significant bytes. If MSB is 0x00: Cursor is transparent. Three least significant bytes must be zero. If MSB is not 0x00 or 0xFF: Cursor inverts the color of the surface underneath.
The AND/XOR format uses the most significant byte (MSB) to control the color. If MSB is 0xFF: Cursor is opaque. Show cursor color from three least significant bytes. If MSB is 0x00: Cursor is transparent. Three least significant bytes must be zero. If MSB is not 0x00 or 0xFF: The three least significant bytes are XOR'd with the color of the surface underneath.

CUR_FBC_CTL

CUR_FBC_CTL	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to CUR_BASE or cursor not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled; after armed

Address:	700A0h-700A3h
Name:	Cursor FBC Control
ShortName:	CUR_FBC_CTL_A
Reset:	soft

Address:	710A0h-710A3h
Name:	Cursor FBC Control
ShortName:	CUR_FBC_CTL_B
Reset:	soft

Address:	720A0h-720A3h
Name:	Cursor FBC Control
ShortName:	CUR_FBC_CTL_C
Reset:	soft

Address:	730A0h-730A3h
Name:	Cursor FBC Control
ShortName:	CUR_FBC_CTL_D
Reset:	soft

DWord	Bit	Description
0	31	**Size Reduction Enable** This enables cursor size reduction logic. The cursor engine will fetch and display the programmed reduced number of lines, then go transparent for the rest of the frame.

Value	Name
0b	Disable
1b	Enable

Restriction
Cursor size reduction is not allowed with 2bpp cursor formats or cursor 180 degree rotation. The reduced scan lines field must be programmed with a valid value when cursor size reduction is enabled.

<table>
<tr><th colspan="3">CUR_FBC_CTL</th></tr>
<tr><td></td><td>30:8</td><td>Reserved<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>7:0</td><td>Reduced Scan Lines
This specifies the number of scan lines of cursor data to fetch and display when cursor size reduction is enabled. The value programmed is the size minus one.<table><tr><th>Restriction</th></tr><tr><td>The minimum size is 8 lines, programmed as 07h.The maximum size cannot be greater than the normal size when size reduction is not enabled.</td></tr></table></td></tr>
</table>

CUR_PAL

CUR_PAL	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled
Address:	70090h-70093h
Name:	Cursor Palette
ShortName:	CUR_PAL_0_A
Reset:	soft
Address:	70094h-70097h
Name:	Cursor Palette
ShortName:	CUR_PAL_1_A
Reset:	soft
Address:	70098h-7009Bh
Name:	Cursor Palette
ShortName:	CUR_PAL_2_A
Reset:	soft
Address:	7009Ch-7009Fh
Name:	Cursor Palette
ShortName:	CUR_PAL_3_A
Reset:	soft
Address:	71090h-71093h
Name:	Cursor Palette
ShortName:	CUR_PAL_0_B
Reset:	soft
Address:	71094h-71097h
Name:	Cursor Palette
ShortName:	CUR_PAL_1_B
Reset:	soft
Address:	71098h-7109Bh
Name:	Cursor Palette
ShortName:	CUR_PAL_2_B
Reset:	soft
Address:	7109Ch-7109Fh

CUR_PAL	
Name:	Cursor Palette
ShortName:	CUR_PAL_3_B
Reset:	soft
Address:	72090h-72093h
Name:	Cursor Palette
ShortName:	CUR_PAL_0_C
Reset:	soft
Address:	72094h-72097h
Name:	Cursor Palette
ShortName:	CUR_PAL_1_C
Reset:	soft
Address:	72098h-7209Bh
Name:	Cursor Palette
ShortName:	CUR_PAL_2_C
Reset:	soft
Address:	7209Ch-7209Fh
Name:	Cursor Palette
ShortName:	CUR_PAL_3_C
Reset:	soft
Address:	73090h-73093h
Name:	Cursor Palette
ShortName:	CUR_PAL_0_D
Reset:	soft
Address:	73094h-73097h
Name:	Cursor Palette
ShortName:	CUR_PAL_1_D
Reset:	soft
Address:	73098h-7309Bh
Name:	Cursor Palette
ShortName:	CUR_PAL_2_D
Reset:	soft
Address:	7309Ch-7309Fh
Name:	Cursor Palette
ShortName:	CUR_PAL_3_D
Reset:	soft
The cursor palette provides color information when using the indexed cursor modes. There are 4 instances of	

CUR_PAL

this register format per cursor. The table below describes how the cursor mode and index value will select between the cursor palette colors, AND/XOR, transparency, and destination invert.

Index Value	2 color mode	3 color mode	4 color mode
00	CUR_PAL 0	CUR_PAL 0	CUR_PAL 0
01	CUR_PAL 1	CUR_PAL 1	CUR_PAL 1
10	Transparent	Transparent	CUR_PAL 2
11	Invert Destination	CUR_PAL 3	CUR_PAL 3

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**Palette Red** This field is the cursor palette red value
	15:8	**Palette Green** This field is the cursor palette green value.
	7:0	**Palette Blue** This field is the cursor palette blue value.

CUR_POS

CUR_POS	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled
Address:	70088h-7008Bh
Name:	Cursor Position
ShortName:	CUR_POS_A
Reset:	soft
Address:	71088h-7108Bh
Name:	Cursor Position
ShortName:	CUR_POS_B
Reset:	soft
Address:	72088h-7208Bh
Name:	Cursor Position
ShortName:	CUR_POS_C
Reset:	soft
Address:	73088h-7308Bh
Name:	Cursor Position
ShortName:	CUR_POS_D
Reset:	soft

This register specifies the screen position of the cursor. The origin of the cursor position is always the upper left corner of the display pipe source image area. When performing 180 degree rotation, the cursor image is rotated by hardware, but the position is not, so it should be adjusted if it is desired to maintain the same apparent position on a physically rotated display.

Restriction
The cursor must have at least a single pixel positioned over the pipe source area. The cursor must not overlap both the left and right sides of the pipe source area.

DWord	Bit	Description
0	31	**Y Position Sign** This specifies the sign of the vertical position of the cursor upper left corner.
	30:29	**Reserved** Access: RO Format: MBZ

CUR_POS		
	28:16	**Y Position Magnitude** This specifies the magnitude of the vertical position of the cursor upper left corner in lines.
	15	**X Position Sign** This specifies the sign of the horizontal position of the cursor upper left corner.
	14:13	**Reserved** Access: RO Format: MBZ
	12:0	**X Position Magnitude** This specifies the magnitude of the horizontal position of the cursor upper left corner in pixels.

CUR_PRE_CSC_GAMC_DATA

CUR_PRE_CSC_GAMC_DATA	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	700B4h-700B7h
Name:	Cursor Pre CSC Gamma Data
ShortName:	CUR_PRE_CSC_GAMMA_DATA_A
Reset:	soft
Address:	710B4h-710B7h
Name:	Cursor Pre CSC Gamma Data
ShortName:	CUR_PRE_CSC_GAMMA_DATA_B
Reset:	soft
Address:	720B4h-720B7h
Name:	Cursor Pre CSC Gamma Data
ShortName:	CUR_PRE_CSC_GAMMA_DATA_C
Reset:	soft
Address:	730B4h-730B7h
Name:	Cursor Pre CSC Gamma Data
ShortName:	CUR_PRE_CSC_GAMMA_DATA_D
Reset:	soft

CUR_PRE_CSC_GAMC_INDEX and CUR_PRE_CSC_GAMC_DATA registers are used to program the values that determine the characteristics of the gamma correction for the cursor pixel data before Color Space Conversion. Additional gamma correction can be done after the Color Space Conversion if desired.
The gamma correction curve is represented by specifying a set of gamma entry reference points spaced equally along the curve for values between -1 and 1. For extended values there are extended gamma entry reference points. All input values are clamped to the greater than -7.0 and less than 7.0 range before the gamma calculation.
For input values greater than or equal to 0 and less than 1.0, the input value is used to linearly interpolate between two adjacent points of the first 33 gamma entries to create the result value. The first 32 entries are stored as 16 bits per color in an unsigned 0.16 format with 0 integer and 16 fractional. The 33rd, 34th and 35th entries are stored as 19 bits per color in an unsigned 3.16 format with 3 integer and 16 fractional bits.
For input values greater than or equal to 1.0 and less than 3.0, the input value is used to linearly interpolate between the 33rd and 34th gamma entries to create the result value.
For input values greater than or equal to 3.0 and less than 7.0, the input value is used to linearly interpolate between the 34th and 35th gamma entries to create the result value.
For negative input values, gamma is mirrored along the X-axis, giving the same result as positive input values, except for a negative sign. When gamma input may be negative, the first gamma point should be programmed to a value of 0.0 in order to have a symmetric mirroring.
Pre-CSC Gamma correction gets enabled or disabled based on the "Pre CSC Enable" bit in the CUR_CTL register.

CUR_PRE_CSC_GAMC_DATA

Programming Notes
To program the gamma correction entries, calculate the desired gamma curve for inputs from 0 to 7.0. For inputs of 0 to 1.0, multiply the input value by 32 to find the gamma entry number, then store the desired gamma result in that entry. For inputs greater than 1.0 and less than or equal to 3.0, store the result for an input of 3.0 in the 34th gamma entry. For inputs greater than 3.0 and less than or equal to 7.0, store the result for an input of 7.0 in the 35th gamma entry.

Restriction
The gamma curve must be flat or increasing, never decreasing. The gamma correction registers should only be updated when the plane is off, otherwise screen artifacts may show temporarily.

DWord	Bit	Description
0	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**Gamma Value** Default Value: 00000000000000000b Format: U3.16

CUR_PRE_CSC_GAMC_INDEX

CUR_PRE_CSC_GAMC_INDEX

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	700B0h-700B3h
Name:	Cursor Pre CSC Gamma Index
ShortName:	CUR_PRE_CSC_GAMMA_INDEX_A
Reset:	soft
Address:	710B0h-710B3h
Name:	Cursor Pre CSC Gamma Index
ShortName:	CUR_PRE_CSC_GAMMA_INDEX_B
Reset:	soft
Address:	720B0h-720B3h
Name:	Cursor Pre CSC Gamma Index
ShortName:	CUR_PRE_CSC_GAMMA_INDEX_C
Reset:	soft
Address:	730B0h-730B3h
Name:	Cursor Pre CSC Gamma Index
ShortName:	CUR_PRE_CSC_GAMMA_INDEX_D
Reset:	soft

DWord	Bit	Description
0	31:11	**Reserved** Access: RO Format: MBZ
	10	**Index Auto Increment** This field enables the index auto increment. Value 0b: No Increment — Do not automatically increment the index value. Value 1b: Auto Increment **[Default]** — Increment the index value with each read or write to the data register.
	9:6	**Reserved** Access: RO Format: MBZ

CUR_PRE_CSC_GAMC_INDEX

	5:0	**Index Value** Access: Write/Read Status This index controls access to the array of pipe pre color space conversion gamma values. This value can be automatically incremented by a read or a write to the data register if the index auto increment bit is set. When automatically incrementing, the index will roll over to 0 after writing or reading the entire allowed range. While in auto increment mode, after performing reads or writes to only part of the range, the auto increment bit must be cleared before resetting the index value. **Value** / **Name** [0,34]

CUR_SURFLIVE

CUR_SURFLIVE	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	700ACh-700AFh
Name:	Cursor Live Base Address
ShortName:	CUR_SURFLIVE_A
Reset:	soft
Address:	710ACh-710AFh
Name:	Cursor Live Base Address
ShortName:	CUR_SURFLIVE_B
Reset:	soft
Address:	720ACh-720AFh
Name:	Cursor Live Base Address
ShortName:	CUR_SURFLIVE_C
Reset:	soft
Address:	730ACh-730AFh
Name:	Cursor Live Base Address
ShortName:	CUR_SURFLIVE_D
Reset:	soft
There is one instance of this register for each pipe.	

DWord	Bit	Description
0	31:12	**Live Surface Base Address** This gives the live value of the surface base address as being currently used for the cursor.
	11:0	**Reserved** Access: RO Format: MBZ

Customizable Event Creation 5-0

CEC5_0 - Customizable Event Creation 5-0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02798h

Description
This register is used to define custom counter event 5, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.
This register can be used to customize counters for events from both unslice and slice units.

DWord	Bit	Description
0	31:21	**Negate** Format: U11 _Custom_GTIReset: DEV The 11-bit array defined in this field allows input bus bits [10:0] to be individually negated in order to facilitate more complicated custom event creation (e.g. A & (!B | !C)). Note that LSB of this field affects bit 0 of the selected input bus. 0b – Pass-through – Input bit is passed through to comparator as is 1b – Negated – Input bit is negated before passing to comparator
	20:19	**Source Select** Format: U2 _Custom_GTIReset: DEV Selects the input signals to the Boolean event definition logic (see block diagram in the Custom Event Counters section). 01b – Prev Event – Selects the conditioned/flopped input from the previous CEC block as the input bus to this CEC block 11b – Reserved –
	18:3	**Compare Value** Format: U16 _Custom_GTIReset: DEV The value in this field is compared the 16-bit conditioned input bus that are fed into the comparator(see block diagram in the Custom Event Counters section). The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the custom event is asserted. This signal in turn can be counted by the B0 performance counter or fed into other CEC blocks.

CEC5_0 - Customizable Event Creation 5-0

2:0 **Compare Function**

Format:	U3
_Custom_GTIReset:	DEV

This field selects the function used by the CEC comparator when comparing the compare value to the value active on the CEC conditioned input bus (see block diagram in the Custom Event Counters section).

Value	Name	Description
000b	Any Are Equal	Compare and assert output if any are equal (Can be used as OR function)
001b	Greater Than	Compare and assert output if greater than
010b	Equal	Compare and assert output if equal to (Can also be used as AND function)
011b	Greater Than or Equal	Compare and assert output if greater than or equal
100b	Less Than	Compare and assert output if less than
101b	Not Equal	Compare and assert output if not equal
110b	Less Than or Equal	Compare and assert output if less than or equal
111b	Reserved	

DATAM

DATAM	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60030h-60033h
Name:	Transcoder Data M Value 1
ShortName:	TRANS_DATAM1_A
Reset:	soft
Address:	61030h-61033h
Name:	Transcoder Data M Value 1
ShortName:	TRANS_DATAM1_B
Reset:	soft
Address:	62030h-62033h
Name:	Transcoder Data M Value 1
ShortName:	TRANS_DATAM1_C
Reset:	soft
Address:	63030h-63033h
Name:	Transcoder Data M Value 1
ShortName:	TRANS_DATAM1_D
Reset:	soft

This register is double buffered to update on the next MSA after LINKN is written.

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30:25	**TU or VCpayload Size** In DisplayPort SST mode this field is the size of the transfer unit, minus one. Typically it is programmed with a value of 63 for TU size of 64. In DisplayPort MST mode this field is the Virtual Channel payload size, minus one. **Restriction** In DisplayPort MST mode the Virtual Channel payload size must not be programmed greater than 62 (resulting payload size of 63). In DisplayPort MST mode the Virtual Channel payload size must not be changed while the Virtual Channel is enabled, even after a transcoder has been disabled.

DATAM		
	24	**Reserved** Access: RO Format: MBZ
	23:0	**Data M value** This field is the data M value for internal use.

DATAN

DATAN	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60034h-60037h
Name:	Transcoder Data N Value 1
ShortName:	TRANS_DATAN1_A
Reset:	soft
Address:	61034h-61037h
Name:	Transcoder Data N Value 1
ShortName:	TRANS_DATAN1_B
Reset:	soft
Address:	62034h-62037h
Name:	Transcoder Data N Value 1
ShortName:	TRANS_DATAN1_C
Reset:	soft
Address:	63034h-63037h
Name:	Transcoder Data N Value 1
ShortName:	TRANS_DATAN1_D
Reset:	soft
This register is double buffered to update on the next MSA after LINKN is written.	

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23:0	**Data N value** This field is the data N value for internal use.

DBUF_CTL

DBUF_CTL		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	45008h-4500Bh	
Name:	DBUF Slice 1 Control	
ShortName:	DBUF_CTL_S1	
Reset:	soft	
Address:	44FE8h-44FEBh	
Name:	DBUF Slice 2 Control	
ShortName:	DBUF_CTL_S2	
Reset:	soft	

DWord	Bit	Description
0	31	**DBUF Power Request** Access: R/W This field requests DBUF power to enable or disable. 0b – Disable 1b – Enable **Programming Notes** DBUF power must be enabled prior to using internal display engine features. Enable power by programming the power request to 1, then wait for the power state to indicate it is enabled.
	30	**DBUF Power State** Access: RO This field indicates the status of DBUF power. 0b – Disabled 1b – Enabled
	29:28	**Reserved** Access: RO Format: MBZ
	27	**Reserved**

DBUF_CTL

	26	**Reserved** Access: RO Format: MBZ
	25:24	**Reserved**
	23:19	**Tracker State Service** Default Value: 1100b This field sets the maximum number of clocks before servicing tracker state.
	18:16	**Reserved** Access: RO Format: MBZ
	15:12	**CC Block Valid State Service** Default Value: 1100b This field sets the maximum number of clocks before servicing CC block valid state.
	11:10	**Reserved** Access: RO Format: MBZ
	9:8	**Reserved** Access: RO Format: MBZ
	7	**Reserved**
	6	**Reserved** Access: RO Format: MBZ
	5:4	**Reserved**
	3:0	**Reserved** Access: RO Format: MBZ

DBUF_CTL2

DBUF_CTL2

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	44FFCh-44FFFh
Name:	DBUF Slice 1 Control 2
ShortName:	DBUF_CTL2_S1
Reset:	soft
Address:	44FE4h-44FE7h
Name:	DBUF Slice 2 Control 2
ShortName:	DBUF_CTL2_S2
Reset:	soft

DWord	Bit	Description
0	31:13	**Reserved** Access: RO Format: MBZ
	12:8	**HPUT Service Count** Forces an Hput to be serviced every number of programmed clocks. Value: 01010b — Name: **[Default]** Value: [1,31] — Name:
	7:4	**APUT Service Count** The number of time slots devoted to APUTs. Value: 0001b — Name: **[Default]** Value: [1,15] — Name:
	3:0	**Bypass Service Count** The number of time slots devoted to BYPASS Puts. Value: 0001b — Name: **[Default]** Value: [1,15] — Name:

DBUF_ECC_STAT

DBUF_ECC_STAT	
Register Space:	MMIO: 0/2/0
Access:	R/WC
Size (in bits):	32
Address:	45010h-45013h
Name:	DBUF Slice 1 ECC Status
ShortName:	DBUF_ECC_STAT_S1
Reset:	soft
Address:	44FF0h-44FF3h
Name:	DBUF Slice 2 ECC Status
ShortName:	DBUF_ECC_STAT_S2
Reset:	soft

Each of these fields is a sticky bit that gives the ECC error status for a particular memory bank. A '1' in a bit indicates that ECC detected an error. A write of '1' to a bit will clear the bit.Single errors are corrected by ECC. Double errors are not correctable.

DWord	Bit	Description
0	31	**Double Error Bank 15**
	30	**Double Error Bank 14**
	29	**Double Error Bank 13**
	28	**Double Error Bank 12**
	27	**Double Error Bank 11**
	26	**Double Error Bank 10**
	25	**Double Error Bank 9**
	24	**Double Error Bank 8**
	23	**Double Error Bank 7**
	22	**Double Error Bank 6**
	21	**Double Error Bank 5**
	20	**Double Error Bank 4**
	19	**Double Error Bank 3**
	18	**Double Error Bank 2**
	17	**Double Error Bank 1**
	16	**Double Error Bank 0**
	15	**Single Error Bank 15**
	14	**Single Error Bank 14**
	13	**Single Error Bank 13**

DBUF_ECC_STAT		
	12	**Single Error Bank 12**
	11	**Single Error Bank 11**
	10	**Single Error Bank 10**
	9	**Single Error Bank 9**
	8	**Single Error Bank 8**
	7	**Single Error Bank 7**
	6	**Single Error Bank 6**
	5	**Single Error Bank 5**
	4	**Single Error Bank 4**
	3	**Single Error Bank 3**
	2	**Single Error Bank 2**
	1	**Single Error Bank 1**
	0	**Single Error Bank 0**

DBUF_STATUS

DBUF_STATUS		
Register Space:	MMIO: 0/2/0	
Access:	R/WC	
Size (in bits):	32	
Address:	4500Ch-4500Fh	
Name:	DBUF Slice 1 Status	
ShortName:	DBUF_STATUS_S1	
Reset:	soft	
Address:	44FECh-44FEFh	
Name:	DBUF Slice 2 Status	
ShortName:	DBUF_STATUS_S2	
Reset:	soft	

DWord	Bit	Description
0	31	**Spare31**
	30	**Dataout FIFO overrun** Access: R/WC A '1' indicates the DBUF out FIFO overrun. Sticky bit cleared by a write of '1'.
	29	**Spare29** Access: R/WC
	28	**EAck FIFO Overrun** A '1' indicates that the EAckFIFO overflowed. Sticky bit cleared by a write of '1'.
	27	**DROB Rsvd6** Access: R/WC
	26	**DROB Rsvd5** Access: R/WC
	25	**DROB Rsvd4** Access: R/WC
	24	**DROB Rsvd3** Access: R/WC
	23	**DROB Track Num FIFO Overrun** Access: R/WC A '1' indicates that the DROB Track Num FIFO overflowed. Sticky bit cleared by a write of '1'.
	22	**Spare22** Access: R/WC

DBUF_STATUS		
	21	**Tracker Over Allocated** Access: R/WC A '1' indicates that the Arbiter allocated more than 16 trackers. Sticky bit cleared by a write of '1'.
	20	**Eput Fifo Overrun** Access: R/WC Indication that the bypass fifo in coming into dbuf has overrun.
	19	**Spare19** Access: R/WC
	18	**Spare18** Access: R/WC
	17	**Spare17** Access: R/WC
	16	**Spare16** Access: R/WC
	15:0	**Reserved**

DC_STATE_EN

DC_STATE_EN

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
CrashLogSaved:	true
CrashLogPriority:	1
CrashLogVisibility:	public
ExternalLongName:	DE DC State Enable
ExternalDescription:	Display Engine C-state enable

Address:	45504h-45507h
Name:	Display C State Enable
ShortName:	DC_STATE_EN
Reset:	soft

DWord	Bit	Description
0	31	**MODE SET in Progress** This bit indicates that Mode set is in progress and DCPR will not generate any CSR_Start to DMC when set. Software needs to program this bit when mode set is started and software should reset it when mode set is done. Value: 0b – CSR_start generation not gated **[Default]** Value: 1b – CSR_start generation is gated
	30	**Display Clock Off Enable** This bit indicates that the DC*clock off is allowed. Driver must program this bit to 1 to allow the DMC to go the DC*CO states. Value: 0b – DC*CO is disallowed **[Default]** Value: 1b – DC*CO is allowed
	29	**Display DC*CO State Status** This bit indicates that the DMC DC*CO exit has completed and driver has to write a 0 to clear this bit. Value: 0b – **[Default]** Value: 1b – DMC DC*CO exit completed
	28	**DSI PLLs turn off disallowed** This bit indicates that when set, both the DSI PLLs will not be allowed to turnoff in the DC*CO state. Driver needs to set this bit if it does not want the DSI PLL to turn off in DC*CO states. Value: 0b – DSI PLLs turn off allowed **[Default]**

	1b	DSI PLLs turn off disallowed

27:24 Reserved

Access:	RO
Format:	MBZ

23:22 Reserved

Access:	RO
Format:	MBZ

21:20 Reserved

Access:	RO
Format:	MBZ

19:17 Reserved

Access:	RO
Format:	MBZ

16 Reserved

Access:	RO
Format:	MBZ

15:11 Reserved

Access:	RO
Format:	MBZ

10 Reserved

Access:	RO
Format:	MBZ

9 In CSR Flow

Value	Name
0b	Not In CSR
1b	In CSR

Restriction
This field is used for hardware communication. Software must not change this field.

8 Block Outbound Traffic

Access is read/write, but hardware can also clear the value based on the PM Request.

Value	Name
0b	Do Not Block
1b	Block

Restriction
This field is used for hardware communication. Software must not change this field.

DC_STATE_EN

	7:6	**Reserved** Access: RO Format: MBZ
	5	**Reserved** Access: RO Format: MBZ
	4	**Mask Poke** This field masks the poke signal that would otherwise be generated by a write to the DC_STATE_SEL register. Value: 0b – Unmask; 1b – Mask Restriction: This field is used for hardware communication. Software must not change this field.
	3	**DC9 Allow** This field indicates software allows Display C9. When allowed, the PCU can save the display PCI Config context and power down display Value: 0b – Do not allow; 1b – Allow
	2	**Reserved** Access: RO Format: MBZ
	1:0	**Dynamic DC State Enable** This field enables hardware to dynamically enter and exit Display C states. Value: 00b – Disable; 01b – Enable up to DC5; 10b – Enable up to DC6 Restriction: The Display CSR code must be loaded before this field is enabled.

DCPR_STATUS_1

DCPR_STATUS_1	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	46440h-46443h
Name:	DCPR Status 1
ShortName:	DCPR_STATUS_1
Reset:	global

This register indicates whether display is demoting Isoch requests.

DWord	Bit	Description
0	31	**Reserved**
	30	**Reserved**
	29	**Reserved**
	28	**Reserved**
	27	**Reserved**
	26	**Reserved** Access: RO Format: MBZ
	25:0	**Reserved** Access: RO Format: MBZ

DDI_AUX_CTL

DDI_AUX_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	64310h-64313h
Name:	DDI AUX Channel Control
ShortName:	DDI_AUX_CTL_USBC1
Reset:	soft
Address:	64410h-64413h
Name:	DDI AUX Channel Control
ShortName:	DDI_AUX_CTL_USBC2
Reset:	soft
Address:	64510h-64513h
Name:	DDI AUX Channel Control
ShortName:	DDI_AUX_CTL_USBC3
Reset:	soft
Address:	64610h-64613h
Name:	DDI AUX Channel Control
ShortName:	DDI_AUX_CTL_USBC4
Reset:	soft
Address:	64710h-64713h
Name:	DDI AUX Channel Control
ShortName:	DDI_AUX_CTL_USBC5
Reset:	soft
Address:	64810h-64813h
Name:	DDI AUX Channel Control
ShortName:	DDI_AUX_CTL_USBC6
Reset:	soft
Address:	64010h-64013h
Name:	DDI AUX Channel Control
ShortName:	DDI_AUX_CTL_A
Reset:	soft
Address:	64110h-64113h
Name:	DDI AUX Channel Control

DDI_AUX_CTL

ShortName:	DDI_AUX_CTL_B
Reset:	soft
Address:	64210h-64213h
Name:	DDI AUX Channel Control
ShortName:	DDI_AUX_CTL_C
Reset:	soft

DWord	Bit	Description
0	31	**Send Busy** Access: R/W Set Writing this bit with 1b initiates the transaction, when read this bit will be a 1b until the transmission completes. The transaction is completed when the response is received or when a timeout occurs. This is a sticky bit. Write a 1b to this bit to set it and initiate the transaction. Hardware will clear it when the transaction completes. **Programming Notes** Aux IO power must be enabled in PWR_WELL_CTL prior to starting an Aux transaction. **Restriction** Do not change any fields while Send Busy is asserted. Do not write a 1b again until transaction completes.
	30	**Done** Access: R/WC A sticky bit that indicates the transaction has completed. Write a 1 to this bit to clear the event 0b – Not done 1b – Done
	29	**Interrupt on Done** Access: R/W Enable an interrupt when the transaction completes or times out. 0b – Disable 1b – Enable
	28	**Time out error** Access: R/WC A sticky bit that indicates the transaction has timed out. Write a 1 to this bit to clear the event. 0b – Not error 1b – Error

DDI_AUX_CTL

27:26 Time out timer value

Access:	R/W

Used to determine how long to wait for receiver response before timing out.

Value	Name
01b	600us **[Default]**
10b	800us
11b	4000us

25 Receive error

Access:	R/WC

A sticky bit that indicates that the data received was corrupted, not in multiples of a full byte, or more than 20 bytes. Write a 1 to this bit to clear the event.

Value	Name
0b	Not Error
1b	Error

24:20 Message Size

Access:	Write/Read Status

The value written to this field indicates the total number bytes to transmit (including the header). The value read from this field indicates the number of bytes received, including the header, in the last transaction. Sync/Stop are not part of the message or the message size. Reads of this field will give the response message size. The read value will not be valid while Send/Busy bit 31 is asserted.

Restriction
Message sizes of 0 or >20 are not allowed. Reads and writes are valid only when the done bit is set and timeout or receive error has not occurred.

19:16 Reserved

Access:	RO
Format:	MBZ

15 Reserved

14 Reserved

13 Reserved

12 Reserved

11 IO Select

Access:	R/W

This field selects which IO will be used for the Aux transaction. It must not be switched while a transaction is in progress.

Value	Name	Description
1b	TBT	Use Thunderbolt IO
0b	Legacy	Use legacy IO. Either typeC or regular DDI, depending on project and SKU

DDI_AUX_CTL

	10	**Reserved** Access: RO Format: MBZ
	9:5	**Fast Wake Sync Pulse Count** Default Value: 1 0001b 18 pulses Access: R/W This field determines the total number of SYNC pulses sent during the SYNC phase of a fast wake transaction. The value programmed is the number of pulses minus 1. When this is field is set to "n" the aux controller will send "n+1" SYNC pulses before transmitting the STOP pattern. **SW must always program this field to the value of decimal 7.**
	4:0	**Sync Pulse Count** Default Value: 1 1111b 32 pulses Access: R/W This field determines the total number of SYNC pulses sent during the SYNC phase of a standard transaction. The value programmed is the number of pulses minus 1. When this is field is set to "n" the aux controller will send "n+1" SYNC pulses before transmitting the STOP pattern. **Restriction** This field must be programmed to at least 25 decimals to send the minimum amount of pulses required for a standard transaction.

DDI_AUX_DATA

DDI_AUX_DATA	
Register Space:	MMIO: 0/2/0
Access:	Write/Read Status
Size (in bits):	32
Address:	64314h-64317h
Name:	DDI AUX Channel Data 0
ShortName:	DDI_AUX_DATA_0_USBC1
Reset:	soft
Address:	64318h-6431Bh
Name:	DDI AUX Channel Data 1
ShortName:	DDI_AUX_DATA_1_USBC1
Reset:	soft
Address:	6431Ch-6431Fh
Name:	DDI AUX Channel Data 2
ShortName:	DDI_AUX_DATA_2_USBC1
Reset:	soft
Address:	64320h-64323h
Name:	DDI AUX Channel Data 3
ShortName:	DDI_AUX_DATA_3_USBC1
Reset:	soft
Address:	64324h-64327h
Name:	DDI AUX Channel Data 4
ShortName:	DDI_AUX_DATA_4_USBC1
Reset:	soft
Address:	64414h-64417h
Name:	DDI AUX Channel Data 0
ShortName:	DDI_AUX_DATA_0_USBC2
Reset:	soft
Address:	64418h-6441Bh
Name:	DDI AUX Channel Data 1
ShortName:	DDI_AUX_DATA_1_USBC2
Reset:	soft
Address:	6441Ch-6441Fh
Name:	DDI AUX Channel Data 2

DDI_AUX_DATA	
ShortName:	DDI_AUX_DATA_2_USBC2
Reset:	soft
Address:	64420h-64423h
Name:	DDI AUX Channel Data 3
ShortName:	DDI_AUX_DATA_3_USBC2
Reset:	soft
Address:	64424h-64427h
Name:	DDI AUX Channel Data 4
ShortName:	DDI_AUX_DATA_4_USBC2
Reset:	soft
Address:	64514h-64517h
Name:	DDI AUX Channel Data 0
ShortName:	DDI_AUX_DATA_0_USBC3
Reset:	soft
Address:	64518h-6451Bh
Name:	DDI AUX Channel Data 1
ShortName:	DDI_AUX_DATA_1_USBC3
Reset:	soft
Address:	6451Ch-6451Fh
Name:	DDI AUX Channel Data 2
ShortName:	DDI_AUX_DATA_2_USBC3
Reset:	soft
Address:	64520h-64523h
Name:	DDI AUX Channel Data 3
ShortName:	DDI_AUX_DATA_3_USBC3
Reset:	soft
Address:	64524h-64527h
Name:	DDI AUX Channel Data 4
ShortName:	DDI_AUX_DATA_4_USBC3
Reset:	soft
Address:	64614h-64617h
Name:	DDI AUX Channel Data 0
ShortName:	DDI_AUX_DATA_0_USBC4
Reset:	soft
Address:	64618h-6461Bh
Name:	DDI AUX Channel Data 1

DDI_AUX_DATA	
ShortName:	DDI_AUX_DATA_1_USBC4
Reset:	soft
Address:	6461Ch-6461Fh
Name:	DDI AUX Channel Data 2
ShortName:	DDI_AUX_DATA_2_USBC4
Reset:	soft
Address:	64620h-64623h
Name:	DDI AUX Channel Data 3
ShortName:	DDI_AUX_DATA_3_USBC4
Reset:	soft
Address:	64624h-64627h
Name:	DDI AUX Channel Data 4
ShortName:	DDI_AUX_DATA_4_USBC4
Reset:	soft
Address:	64714h-64717h
Name:	DDI AUX Channel Data 0
ShortName:	DDI_AUX_DATA_0_USBC5
Reset:	soft
Address:	64718h-6471Bh
Name:	DDI AUX Channel Data 1
ShortName:	DDI_AUX_DATA_1_USBC5
Reset:	soft
Address:	6471Ch-6471Fh
Name:	DDI AUX Channel Data 2
ShortName:	DDI_AUX_DATA_2_USBC5
Reset:	soft
Address:	64720h-64723h
Name:	DDI AUX Channel Data 3
ShortName:	DDI_AUX_DATA_3_USBC5
Reset:	soft
Address:	64724h-64727h
Name:	DDI AUX Channel Data 4
ShortName:	DDI_AUX_DATA_4_USBC5
Reset:	soft
Address:	64814h-64817h
Name:	DDI AUX Channel Data 0

DDI_AUX_DATA	
ShortName:	DDI_AUX_DATA_0_USBC6
Reset:	soft
Address:	64818h-6481Bh
Name:	DDI AUX Channel Data 1
ShortName:	DDI_AUX_DATA_1_USBC6
Reset:	soft
Address:	6481Ch-6481Fh
Name:	DDI AUX Channel Data 2
ShortName:	DDI_AUX_DATA_2_USBC6
Reset:	soft
Address:	64820h-64823h
Name:	DDI AUX Channel Data 3
ShortName:	DDI_AUX_DATA_3_USBC6
Reset:	soft
Address:	64824h-64827h
Name:	DDI AUX Channel Data 4
ShortName:	DDI_AUX_DATA_4_USBC6
Reset:	soft
Address:	64014h-64017h
Name:	DDI AUX Channel Data 0
ShortName:	DDI_AUX_DATA_0_A
Reset:	soft
Address:	64018h-6401Bh
Name:	DDI AUX Channel Data 1
ShortName:	DDI_AUX_DATA_1_A
Reset:	soft
Address:	6401Ch-6401Fh
Name:	DDI AUX Channel Data 2
ShortName:	DDI_AUX_DATA_2_A
Reset:	soft
Address:	64020h-64023h
Name:	DDI AUX Channel Data 3
ShortName:	DDI_AUX_DATA_3_A
Reset:	soft
Address:	64024h-64027h
Name:	DDI AUX Channel Data 4

DDI_AUX_DATA	
ShortName:	DDI_AUX_DATA_4_A
Reset:	soft
Address:	64114h-64117h
Name:	DDI AUX Channel Data 0
ShortName:	DDI_AUX_DATA_0_B
Reset:	soft
Address:	64118h-6411Bh
Name:	DDI AUX Channel Data 1
ShortName:	DDI_AUX_DATA_1_B
Reset:	soft
Address:	6411Ch-6411Fh
Name:	DDI AUX Channel Data 2
ShortName:	DDI_AUX_DATA_2_B
Reset:	soft
Address:	64120h-64123h
Name:	DDI AUX Channel Data 3
ShortName:	DDI_AUX_DATA_3_B
Reset:	soft
Address:	64124h-64127h
Name:	DDI AUX Channel Data 4
ShortName:	DDI_AUX_DATA_4_B
Reset:	soft
Address:	64214h-64217h
Name:	DDI AUX Channel Data 0
ShortName:	DDI_AUX_DATA_0_C
Reset:	soft
Address:	64218h-6421Bh
Name:	DDI AUX Channel Data 1
ShortName:	DDI_AUX_DATA_1_C
Reset:	soft
Address:	6421Ch-6421Fh
Name:	DDI AUX Channel Data 2
ShortName:	DDI_AUX_DATA_2_C
Reset:	soft
Address:	64220h-64223h
Name:	DDI AUX Channel Data 3

DDI_AUX_DATA

ShortName:	DDI_AUX_DATA_3_C
Reset:	soft

Address:	64224h-64227h
Name:	DDI AUX Channel Data 4
ShortName:	DDI_AUX_DATA_4_C
Reset:	soft

There are 5 DWords of this register format per instance.

DWord	Bit	Description
0	31:0	**AUX CH DATA** This field contains a DWord of the AUX message. Writes to this register give the data to transmit during the transaction. The MSbyte is transmitted first. Reads to this register will give the response data after transaction complete. The read value will not be valid while the Aux Channel Control Register Send/Busy bit is asserted

DDI_BUF_CTL

DDI_BUF_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	64300h-64303h
Name:	DDI Buffer Control
ShortName:	DDI_BUF_CTL_USBC1
Reset:	soft
Address:	64400h-64403h
Name:	DDI Buffer Control
ShortName:	DDI_BUF_CTL_USBC2
Reset:	soft
Address:	64500h-64503h
Name:	DDI Buffer Control
ShortName:	DDI_BUF_CTL_USBC3
Reset:	soft
Address:	64600h-64603h
Name:	DDI Buffer Control
ShortName:	DDI_BUF_CTL_USBC4
Reset:	soft
Address:	64700h-64703h
Name:	DDI Buffer Control
ShortName:	DDI_BUF_CTL_USBC5
Reset:	soft
Address:	64800h-64803h
Name:	DDI Buffer Control
ShortName:	DDI_BUF_CTL_USBC6
Reset:	soft
Address:	64000h-64003h
Name:	DDI Buffer Control
ShortName:	DDI_BUF_CTL_A
Reset:	soft
Address:	64100h-64103h
Name:	DDI Buffer Control

DDI_BUF_CTL

ShortName:	DDI_BUF_CTL_B
Reset:	soft

Address:	64200h-64203h
Name:	DDI Buffer Control
ShortName:	DDI_BUF_CTL_C
Reset:	soft

Do not read or write the register when the associated power well is disabled.

DWord	Bit	Description
0	31	**DDI Buffer Enable** This bit enables the DDI buffer. Value: 0b = Disable; 1b = Enable
	30	**Reserved** Access: RO Format: MBZ
	29	**Override Training Enable** This field enables the override on the training enable signal that tells the DDI I/O to pick up any DDI voltage swing and pre-emphasis changes. Value: 1b = Enable Override; 0b = Disable Override
	28	**Phy Param Adjust** Enables adjustment of Phy parameters such as voltage swing and pre emphasis outside Inik training process. This field is conditioned on "override training enable" (DDI_BUF_CTL[29]). Value: 1b = Enable; 0b = Disable
	27:24	**Reserved** Access: RO Format: MBZ
	23:18	**Reserved** Access: RO Format: MBZ

DDI_BUF_CTL

17 Reserved

Access:	RO
Format:	MBZ

16 Port Reversal

This field enables lane reversal within the port. Lane reversal swaps the data on the lanes as they are output from the port.

Value	Name
0b	Not reversed
1b	Reversed

Programming Notes

Type-C/TBT dynamic connections:
The DDIs going to thunderbolt or USB-C DP alternate mode should not be reversed here. The reversal is taken care of in the FIA.
Static/fixed connections (DP/HDMI) through FIA:
In the case of static connections such as "No pin assignment (Non Type-C DP)", DDIs will use this lane reversal bit.
All other connections:
DDIs will use this lane reversal bit.

Restriction

This field must not be changed while the DDI is enabled.

15:8 USB Type-C DP Lane Staggering Delay

Specifies the number of symbol clocks delay used to stagger assertion/deassertion of the port lane enables. The target time recommended by circuit team is 100ns or greater. The delay should be programmed based on link clock frequency. This staggering delay is ONLY required when the port is used in USB Type C mode. Otherwise the default delay is zero which means no staggering. Example: 270MHz link clock = 1/270MHz = 3.7ns. (100ns/3.7ns)=27.02 symbols. Round up to 28.

7 DDI Idle Status

Access:	RO

This bit indicates when the DDI buffer is idle.

Value	Name
0b	Buffer Not Idle
1b	Buffer Idle

6 Reserved

Access:	RO
Format:	MBZ

5:4 Reserved

Access:	RO
Format:	MBZ

DDI_BUF_CTL

	3:1	**DP Port Width Selection** This bit selects the number of lanes to be enabled on the DDI link for DisplayPort.

Value	Name	Description
000b	x1	x1 Mode
001b	x2	x2 Mode
011b	x4	x4 Mode
Others	Reserved	Reserved

Restriction
When in DisplayPort mode the value selected here must match the value selected in TRANS_DDI_FUNC_CTL attached to this DDI.
This field must not be changed while the DDI is enabled.

	0	**Init Display Detected**

Access:	RO

Strap indicating whether a display was detected on this port during initialization. It signifies the level of the port detect pin at boot. This bit is only informative. It does not prevent this port from being enabled in hardware. This field only indicates the DDIA detection.

Value	Name	Description
0b	Not Detected	Digital display not detected during initialization
1b	Detected	Digital display detected during initialization

DDI_CLK_SEL

DDI_CLK_SEL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	4610Ch-4610Fh
Name:	DDI USBC1 Clock Select
ShortName:	DDI_CLK_SEL_USBC1
Reset:	soft
Address:	46110h-46113h
Name:	DDI USBC2 Clock Select
ShortName:	DDI_CLK_SEL_USBC2
Reset:	soft
Address:	46114h-46117h
Name:	DDI USBC3 Clock Select
ShortName:	DDI_CLK_SEL_USBC3
Reset:	soft
Address:	46118h-4611Bh
Name:	DDI USBC4 Clock Select
ShortName:	DDI_CLK_SEL_USBC4
Reset:	soft
Address:	4611Ch-4611Fh
Name:	DDI USBC5 Clock Select
ShortName:	DDI_CLK_SEL_USBC5
Reset:	soft
Address:	46120h-46123h
Name:	DDI USBC6 Clock Select
ShortName:	DDI_CLK_SEL_USBC6
Reset:	soft

DWord	Bit	Description
0	31:28	**Clock Select** Select which clock to use for this DDI.

Value	Name	Description
0000b	None	Nothing selected. Clock is disabled for this DDI.
1000b	MG	Type-C PHY PLL output
1100b	TBT 162	Thunderbolt 162 MHz

		1101b	TBT 270	Thunderbolt 270 MHz
		1110b	TBT 540	Thunderbolt 540 MHz
		1111b	TBT 810	Thunderbolt 810 MHz

Restriction

This must not be changed while the DDI is enabled or any transcoder directed to the DDI is enabled.

27:0 **Reserved**

Access:	RO
Format:	MBZ

DE_FUSA_IOSF_PARITY_CNTRL

DE_FUSA_IOSF_PARITY_CNTRL - DE_FUSA_IOSF_PARITY_CNTRL

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	100140h

This register controls the parity generation, checking, and error insertion logic in the DE IOSF end points : PSF 0 and PSF DIP

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29	**PSF DIP Cmd Parity Gen Dis** Default Value: 0h Access: R/W _Custom_GTIReset: BUS When 1 command parity generation is disabled
	28	**Reserved** Access: RO Format: MBZ
	27:26	**Reserved** Access: RO Format: MBZ
	25	**PSF DIP Cmd Parity Chk En** Default Value: 0h Access: R/W _Custom_GTIReset: BUS When 1 checking of command parity is enabled
	24	**PSF DIP Data Parity Chk En** Default Value: 0h Access: R/W _Custom_GTIReset: BUS When 1 checking of data parity is enabled

DE_FUSA_IOSF_PARITY_CNTRL - DE_FUSA_IOSF_PARITY_CNTRL

23	**Reserved** Access: RO Format: MBZ
22	**PSF DIP Cmd Parity Err Inj** Default Value: 0h Access: R/W _Custom_GTIReset: BUS 0: No error injection 1: Invert mcparity Once set the next command mcparity is corrupted and then the bit is cleared by HW.
21:20	**Reserved** Access: RO Format: MBZ
19:14	**Reserved** Access: RO Format: MBZ
13	**PSF 0 Cmd Parity Gen Dis** Default Value: 0h Access: R/W _Custom_GTIReset: BUS When 1 command parity generation is disabled
12	**PSF 0 Data Parity Gen Dis** Default Value: 0h Access: R/W _Custom_GTIReset: BUS When 1 data parity generation is disabled
11:10	**Reserved** Access: RO Format: MBZ
9	**PSF 0 Cmd Parity Chk En** Default Value: 0h Access: R/W _Custom_GTIReset: BUS When 1 checking of command parity is enabled

DE_FUSA_IOSF_PARITY_CNTRL - DE_FUSA_IOSF_PARITY_CNTRL

8	**PSF 0 Data Parity Chk En** Default Value: 0h Access: R/W _Custom_GTIReset: BUS When 1 checking of data parity is enabled
7	**Reserved** Access: RO Format: MBZ
6	**PSF 0 Cmd Parity Err Inj** Default Value: 0h Access: R/W _Custom_GTIReset: BUS 0: No error injection 1: Invert mcparity Once set the next command mcparityis corrupted and then the bit is cleared by HW.
5:4	**PSF 0 Data Parity Error Inj** Default Value: 0h Access: R/W _Custom_GTIReset: BUS 00: No error injected 01:Invert mdparity[0] 10:Invert mdparity[1] 11: Invert mdparity[1:0] Once set the next command with data is corrupted and then the bit is cleared by HW. Note: mdparity[1] is only present forIOSF data widths of 512
3:1	**Reserved** Access: RO Format: MBZ
0	**DE_FUSA_IOSF_PARITY_CNTRL_LOCK** Default Value: 000b _Custom_GTIReset: BUS Writing 1 to this bit will lock the register from further updates

DE_PIPE_INTERRUPT

DE_PIPE_INTERRUPT	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	44400h-4440Fh
Name:	Display Engine Pipe A Interrupts
ShortName:	DE_PIPE_INTERRUPT_A
Reset:	soft
Address:	44410h-4441Fh
Name:	Display Engine Pipe B Interrupts
ShortName:	DE_PIPE_INTERRUPT_B
Reset:	soft
Address:	44420h-4442Fh
Name:	Display Engine Pipe C Interrupts
ShortName:	DE_PIPE_INTERRUPT_C
Reset:	soft
Address:	44430h-4443Fh
Name:	Display Engine Pipe D Interrupts
ShortName:	DE_PIPE_INTERRUPT_D
Reset:	soft

This table indicates which events are mapped to each bit of the Display Engine Pipe Interrupt registers. The IER enabled Display Engine Pipe Interrupt IIR (sticky) bits are ORed together to generate the DE_Pipe Interrupts Pending bit in the next level up interrupt registers. There is one full set of Display Engine Pipe interrupts per display pipe. The STEREO3D_EVENT_MASK selects between left eye and right eye reporting of vertical blank, vertical sync, and scanline events in stereo 3D modes.

0x44400 = ISR A, 0x44410 = ISR B, 0x44420 = ISR C, 0x44430 = ISR D
0x44404 = IMR A, 0x44414 = IMR B, 0x44424 = IMR C, 0x44434 = IMR D
0x44408 = IIR A, 0x44418 = IIR B, 0x44428 = IIR C, 0x44438 = IIR D
0x4440C = IER A, 0x4441C = IER B, 0x4442C = IER C, 0x4443C = IER D

DWord	Bit	Description
0	31	**Underrun** Description: The ISR is an active high pulse when there is an underrun on the transcoder attached to this pipe.
	30	**VRR Double Buffer Update** The ISR is an active high pulse on the eDP/DP Variable Refresh Rate double buffer update event on this pipe.
	29	**Reserved**

DE_PIPE_INTERRUPT

	28	**Reserved**
	27	**Isoch Ack** The ISR is an active high pulse when the IsocReq for this pipe receives an Ack.
	26	**PIPEDMC_Interrupt** The ISR is an active high pulse when the PIPE DMC has an interrupt.
	25	**PIPEDMC_gtt_fault_status** The ISR is an active high pulse when the PIPE DMC gtt fault occurs.
	24	**Unused_Int_24** These interrupts are currently unused.
	23	**LACE Fast Access Interrupt** The ISR is an active high level indicating an interrupt is set in DPLC_FA_STATUS.
	22	**Plane7_GTT_Fault_Status** The ISR is an active high pulse when a GTT fault is detected for plane 7 on this pipe.
	21	**Plane6_GTT_Fault_Status** The ISR is an active high pulse when a GTT fault is detected for plane 6 on this pipe.
	20	**Plane5_GTT_Fault_Status** Description The ISR is an active high pulse when a GTT fault is detected for plane 5 on this pipe.
	19	**Vblank unmodified** The ISR is an active high level for the duration of the vertical blank of the transcoder attached to this pipe, the unmodified vertical blank, as opposed to the modified vertical blank that the pipe units use. The transcoder vertical blank always begins at the end of transcoder vertical active (unmodified). When the transcoder vertical blank start is programmed later than transcoder vertical active, the pipe vertical blank will start later than the transcoder vertical blank (modified).
	18	**Plane7_Flip_Done** The ISR is an active high pulse when the flip is done for plane 7 on this pipe.
	17	**Plane6_Flip_Done** The ISR is an active high pulse when the flip is done for plane 6 on this pipe.
	16	**Plane5_Flip_Done** The ISR is an active high pulse when the flip is done for plane 5 on this pipe.

DE_PIPE_INTERRUPT

	15	**DSB_2_Interrupt** The ISR is an active high pulse when there is interrupt from DSB 2. SW must read the DSB interrupt registers to check what is caused interrupt in DSB.
	14	**DSB_1_Interrupt** The ISR is an active high pulse when there is interrupt from DSB 1. SW must read the DSB interrupt registers to check what is caused interrupt in DSB..
	13	**DSB_0_Interrupt** The ISR is an active high pulse when there is interrupt from DSB 0. SW must read the DSB interrupt registers to check what is caused interrupt in DSB..
	12	**DPST_Histogram_event** The ISR is an active high pulse on the DPST Histogram event on this pipe.
	11	**Cursor_GTT_Fault_Status** **Description** The ISR is an active high pulse when a GTT fault is detected for the cursor on this pipe.
	10	**Plane4_GTT_Fault_Status** **Description** The ISR is an active high pulse when a GTT fault is detected for plane 4 on this pipe.
	9	**Plane3_GTT_Fault_Status** **Description** The ISR is an active high pulse when a GTT fault is detected for plane 3 on this pipe.
	8	**Plane2_GTT_Fault_Status** **Description** The ISR is an active high pulse when a GTT fault is detected for plane 2 on this pipe.
	7	**Plane1_GTT_Fault_Status** **Description** The ISR is an active high pulse when a GTT fault is detected for plane 1 on this pipe.
	6	**Plane4_Flip_Done** The ISR is an active high pulse when the flip is done for plane 4 on this pipe.
	5	**Plane3_Flip_Done** The ISR is an active high pulse when the flip is done for plane 3 on this pipe.
	4	**Plane2_Flip_Done** The ISR is an active high pulse when the flip is done for plane 2 on this pipe.
	3	**Plane1_Flip_Done** The ISR is an active high pulse when the flip is done for plane 1 on this pipe.
	2	**Scan_Line_Event** The ISR is an active high pulse on the scan line event of the transcoder attached to this pipe.

DE_PIPE_INTERRUPT		
	1	**Vsync** The ISR is an active high level for the duration of the vertical sync of the transcoder attached to this pipe.
	0	**Vblank** **Description** The ISR is an active high level for the duration of the vertical blank of the transcoder attached to this pipe. This represents the vertical blank observed by units within the pipe and used for updating double-buffered registers in the pipe. The transcoder vertical blank always begins at the end of transcoder vertical active (unmodified). When the transcoder vertical blank start is programmed later than transcoder vertical active, the pipe vertical blank will start later than the transcoder vertical blank (modified).

DE_POWER1

DE_POWER1	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	42400h-42403h
Name:	Display Engine Power 1
ShortName:	DE_POWER1
Reset:	global

DWord	Bit	Description
0	31	**Power Well 2 State** This field indicates the status of display power well 2. Value: 0b = Off; 1b = On
	30	**Display Pipes Enabled** This field indicates if any display pipes are enabled. Value: 0b = Disabled (All display pipes disabled); 1b = Enabled (One or more display pipes enabled)
	29	**Reserved** Access: RO Format: MBZ
	28	**Power Well 1 State** This field indicates the status of display power well 1. Value: 0b = Off; 1b = On
	27:26	**SRD Status** This field indicates the live status of the SRD link on transcoder A. Value: 00b = Full Off (Link is fully off. DDI lanes are disabled and most memory reads are disabled.); 01b = Full On (Link is fully on. Normal operation.); 11b = Reserved (Reserved)
	25	**KVM Session Status** This field indicates the status of KVM session.

DE_POWER1

Value	Name	Description
0b	Disabled	KVM session disabled
1b	Enabled	KVM session enabled

24:20	**Transmit Lanes Enabled** The total number of DDI lanes enabled.
19:14	**Reserved** Access: RO Format: MBZ
13:10	**Enabled Pipe Scalers** Indicates total usage of the Scaler EBBs.
9:8	**Enabled DEPLLs** The total number of display CCU PLLs enabled.
7	**Reserved** Access: RO Format: MBZ
6:4	**Reserved** Access: RO Format: MBZ
3	**Enabled CDPLLs** Indicates if CD PLL is enabled.
2:0	**Enabled MGPLLs** The total number of type-C PLLs enabled by display.

DE_POWER2

DE_POWER2	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	42404h-42407h
Name:	Display Engine Power 2
ShortName:	DE_POWER2
Reset:	global

DWord	Bit	Description
0	31:0	**DE bandwidth counter** This counter increments on every cache line put arriving at the DE. The bandwidth is estimated by taking the difference between two reads at a known interval. Access is actually a read/write variant. Writes to this register will load the write data into the counter. **Restriction** This register will measure the memory Bandwidth of the Low Priority channel to memory. It is associated with ABOX. It does not measure BW associated with ABOX1 or ABOX2.

DE_RR_DEST

DE_RR_DEST

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	44058h-4405Bh
Name:	Render Response Destination
ShortName:	DE_RR_DEST
Reset:	soft

This register selects the destination of certain render responses that may go to CS, BCS, or both. In order for a response to be sent to a particular destination, the event most occur, the event must be unmasked, and that destination must be selected.

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7:6	**Pipe D Vertical Blank Destination** This field selects the destination for the render response sent on pipe D start of vertical blank. 00b: CS 01b: BCS 10b,11b: Both CS and BCS
	5:4	**Pipe C Vertical Blank Destination** This field selects the destination for the render response sent on pipe C start of vertical blank. 00b: CS 01b: BCS 10b,11b: Both CS and BCS
	3:2	**Pipe B Vertical Blank Destination** This field selects the destination for the render response sent on pipe B start of vertical blank. 00b: CS 01b: BCS 10b,11b: Both CS and BCS

<table>
<tr><th colspan="3">DE_RR_DEST</th></tr>
<tr><td></td><td>1:0</td><td>Pipe A Vertical Blank Destination
This field selects the destination for the render response sent on pipe A start of vertical blank.
<table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>CS</td></tr><tr><td>01b</td><td>BCS</td></tr><tr><td>10b,11b</td><td>Both CS and BCS</td></tr></table></td></tr>
</table>

DE_RRMR

DE_RRMR	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	44050h-44053h
Name:	Render Response Mask
ShortName:	DE_RRMR
Reset:	soft

This register contains a bit mask which selects which events cause and are reported in the render response message.
See the render response message definition table to find the source event for each bit.
The render response message is sent from the display engine to the render command streamer (CS) or blitter command streamer (BCS). The message is used to inform CS and BCS of certain display events.
This register is used to control which render response message bits are masked or unmasked.
Unmasked bits will cause a render response message to be sent and will be reported in that message.
Masked bits will not be reported and will not cause a render response message to be sent.
Vertical blank events occur periodically while the associated display pipe timing generator is running and will be reported in a render response to CS or BCS (depending on DE_RR_DEST destination selection) if un-masked here.
Scanline events occur after they have been initiated through MMIO writes or LRI to the Display Load Scan Lines register. A scanline done event will be reported in a render response to CS if un-masked here and the Display Load Scanline source is CS. A scanline done event will be reported in a render response to BCS if un-masked here and the Display Load Scanline source is BCS.
Flip done events occur after they have been initiated through MI_DISPLAY_FLIP or MMIO write to plane surface address registers. A flip event will be reported in a render response to CS if un-masked here and the flip source is CS. A flip event will be reported in a render response to BCS if un-masked here and the flip source is BCS.

This register defines the DWord 0 of the render response bit mask. DWord 1and DWord 2 are defined inDE_RRMR_DW1 and DE_RRMR_DW2 registers.

Programming Notes

Programming this register can be done through MMIO or a command streamer LOAD_REGISTER_IMMEDIATE (LRI) command.
When using LRI, care must be taken to follow all the programming rules for LRI targeting the display engine.
Unmasked events will wake GT as they occur, so for improved power savings it is recommended to only unmask events while they are required.

Restriction

Events must be unmasked prior to waiting for them with a MI_WAIT_FOR_EVENT ring command, or in the case of flips or scanlines, prior to starting the flip or loading the scanline.

DWord	Bit	Description
0	31	**Mask 31**

DE_RRMR

		Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	30	**Mask 30** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	29	**Mask 29** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	28	**Mask 28** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	27	**Mask 27** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	26	**Mask 26** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	25	**Mask 25** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	24	**Mask 24** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	23	**Mask 23** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**

DE_RRMR

	22	**Mask 22** **Value** / **Name** 0b – Not Masked 1b – Masked **[Default]**
	21	**Mask 21** **Value** / **Name** 0b – Not Masked 1b – Masked **[Default]**
	20	**Mask 20** **Value** / **Name** 0b – Not Masked 1b – Masked **[Default]**
	19	**Mask 19** **Value** / **Name** 0b – Not Masked 1b – Masked **[Default]**
	18	**Mask 18** **Value** / **Name** 0b – Not Masked 1b – Masked **[Default]**
	17	**Mask 17** **Value** / **Name** 0b – Not Masked 1b – Masked **[Default]**
	16	**Mask 16** **Value** / **Name** 0b – Not Masked 1b – Masked **[Default]**
	15	**Mask 15** **Value** / **Name** 0b – Not Masked 1b – Masked **[Default]**
	14	**Mask 14** **Value** / **Name** 0b – Not Masked 1b – Masked **[Default]**

DE_RRMR

	13	**Mask 13** **Value** / **Name** 0b – Not Masked 1b – Masked **[Default]**
	12	**Mask 12** **Value** / **Name** 0b – Not Masked 1b – Masked **[Default]**
	11	**Mask 11** **Value** / **Name** 0b – Not Masked 1b – Masked **[Default]**
	10	**Mask 10** **Value** / **Name** 0b – Not Masked 1b – Masked **[Default]**
	9	**Mask 9** **Value** / **Name** 0b – Not Masked 1b – Masked **[Default]**
	8	**Mask 8** **Value** / **Name** 0b – Not Masked 1b – Masked **[Default]**
	7	**Mask 7** **Value** / **Name** 0b – Not Masked 1b – Masked **[Default]**
	6	**Mask 6** **Value** / **Name** 0b – Not Masked 1b – Masked **[Default]**
	5	**Mask 5** **Value** / **Name** 0b – Not Masked 1b – Masked **[Default]**

DE_RRMR

	4	**Mask 4** Value: 0b — Not Masked Value: 1b — Masked **[Default]**
	3	**Mask 3** Value: 0b — Not Masked Value: 1b — Masked **[Default]**
	2	**Mask 2** Value: 0b — Not Masked Value: 1b — Masked **[Default]**
	1	**Mask 1** Value: 0b — Not Masked Value: 1b — Masked **[Default]**
	0	**Mask 0** Value: 0b — Not Masked Value: 1b — Masked **[Default]**

DE_RRMR_DW1

DE_RRMR_DW1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	44048h-4404Bh
Name:	Render Response Mask DW 1
ShortName:	DE_RRMR_DW1
Reset:	soft

This register contains the Dword 1 of the CS/BCS render response bit mask. For more details refer to DE_RRMR.

DWord	Bit	Description
0	31	**Mask 31** Value: 0b – Not Masked Value: 1b – Masked **[Default]**
	30	**Mask 30** Value: 0b – Not Masked Value: 1b – Masked **[Default]**
	29	**Mask 29** Value: 0b – Not Masked Value: 1b – Masked **[Default]**
	28	**Mask 28** Value: 0b – Not Masked Value: 1b – Masked **[Default]**
	27	**Mask 27** Value: 0b – Not Masked Value: 1b – Masked **[Default]**
	26	**Mask 26** Value: 0b – Not Masked Value: 1b – Masked **[Default]**

DE_RRMR_DW1

	25	**Mask 25** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	24	**Mask 24** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	23	**Mask 23** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	22	**Mask 22** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	21	**Mask 21** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	20	**Mask 20** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	19	**Mask 19** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	18	**Mask 18** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	17	**Mask 17** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**

DE_RRMR_DW1

	16	**Mask 16** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	15	**Mask 15** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	14	**Mask 14** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	13	**Mask 13** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	12	**Mask 12** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	11	**Mask 11** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	10	**Mask 10** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	9	**Mask 9** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	8	**Mask 8** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**

DE_RRMR_DW1

	7	**Mask 7** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	6	**Mask 6** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	5	**Mask 5** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	4	**Mask 4** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	3	**Mask 3** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	2	**Mask 2** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	1	**Mask 1** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	0	**Mask 0** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**

DE_RRMR_DW2

DE_RRMR_DW2

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	4404Ch-4404Fh
Name:	Render Response Mask DW 2
ShortName:	DE_RRMR_DW2
Reset:	soft

This register contains the DWord 2 of the CS/BCS render response bit mask. For more details refer to DE_RRMR.

DWord	Bit	Description
0	31	**Mask 31** Value: 0b – Not Masked Value: 1b – Masked **[Default]**
	30	**Mask 30** Value: 0b – Not Masked Value: 1b – Masked **[Default]**
	29	**Mask 29** Value: 0b – Not Masked Value: 1b – Masked **[Default]**
	28	**Mask 28** Value: 0b – Not Masked Value: 1b – Masked **[Default]**
	27	**Mask 27** Value: 0b – Not Masked Value: 1b – Masked **[Default]**
	26	**Mask 26** Value: 0b – Not Masked Value: 1b – Masked **[Default]**

DE_RRMR_DW2

	25	**Mask 25** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	24	**Mask 24** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	23	**Mask 23** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	22	**Mask 22** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	21	**Mask 21** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	20	**Mask 20** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	19	**Mask 19** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	18	**Mask 18** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	17	**Mask 17** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**

DE_RRMR_DW2

	16	**Mask 16** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	15	**Mask 15** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	14	**Mask 14** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	13	**Mask 13** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	12	**Mask 12** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	11	**Mask 11** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	10	**Mask 10** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	9	**Mask 9** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	8	**Mask 8** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**

DE_RRMR_DW2

	7	**Mask 7** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	6	**Mask 6** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	5	**Mask 5** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	4	**Mask 4** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	3	**Mask 3** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	2	**Mask 2** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	1	**Mask 1** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	0	**Mask 0** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**

Decouple Register 0 DW0

DECOUPREG0DW0 - Decouple Register 0 DW0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	00F00h-00F03h	
DWord	**Bit**	**Description**
0	31:0	**DecoupReg0DW0Data** Access: R/W _Custom_GTIReset: BUS Decouple Register 0 DW0 Data.

Decouple Register 0 DW1

DECOUPREG0DW1 - Decouple Register 0 DW1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00F04h-00F07h	
DWord	**Bit**	**Description**
0	31	**GO** Access: R/W Lock Decouple Register Go/Status Bit: Software sets this bit along with attributes (full DW write) to initiate a DCR request. Hardware clears this bit to 0 when the command is complete. This bit serves as a lock for the register pair preventing software from updating the status/values once set.
	30:28	**OP** Access: R/W Lock Decouple Register Opcode: 3'b000 = Read, 3'b001 = Write; All others undefined (Request ignored and go/status cleared).
	27:24	**BE_B** Access: R/W Lock Decouple Register Active Low Byte Enables. Byte enables affect data merging into MGSR shadow storage however GT only supports full dword accesses.
	23:0	**Addr** Access: R/W Lock Decouple Register Address.

Decouple Register 1 DW0

DECOUPREG1DW0 - Decouple Register 1 DW0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	00F08h-00F0Bh	
DWord	**Bit**	**Description**
0	31:0	**DecoupReg1DW0Data** Access: R/W _Custom_GTIReset: BUS Decouple Register 1 DW0 Data.

Decouple Register 1 DW1

DECOUPREG1DW1 - Decouple Register 1 DW1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00F0Ch-00F0Fh	
DWord	**Bit**	**Description**
0	31	**GO** Access: R/W Lock Decouple Register Go/Status Bit: Software sets this bit along with attributes (full DW write) to initiate a DCR request. Hardware clears this bit to 0 when the command is complete. This bit serves as a lock for the register pair preventing software from updating the status/values once set.
	30:28	**OP** Access: R/W Lock Decouple Register Opcode: 3'b000 = Read, 3'b001 = Write; All others undefined (Request ignored and go/status cleared).
	27:24	**BE_B** Access: R/W Lock Decouple Register Active Low Byte Enables. Byte enables affect data merging into MGSR shadow storage however GT only supports full dword accesses.
	23:0	**Addr** Access: R/W Lock Decouple Register Address.

Decouple Register 2 DW0

DECOUPREG2DW0 - Decouple Register 2 DW0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	00F10h-00F13h	
DWord	**Bit**	**Description**
0	31:0	**DecoupReg2DW0Data** Access: R/W _Custom_GTIReset: BUS Decouple Register 2 DW0 Data.

Decouple Register 2 DW1

DECOUPREG2DW1 - Decouple Register 2 DW1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00F14h-00F17h	
DWord	**Bit**	**Description**
0	31	**GO** Access: R/W Lock Decouple Register Go/Status Bit: Software sets this bit along with attributes (full DW write) to initiate a DCR request. Hardware clears this bit to 0 when the command is complete. This bit serves as a lock for the register pair preventing software from updating the status/values once set.
	30:28	**OP** Access: R/W Lock Decouple Register Opcode: 3'b000 = Read, 3'b001 = Write; All others undefined (Request ignored and go/status cleared).
	27:24	**BE_B** Access: R/W Lock Decouple Register Active Low Byte Enables. Byte enables affect data merging into MGSR shadow storage however GT only supports full dword accesses.
	23:0	**Addr** Access: R/W Lock Decouple Register Address.

Decouple Register 3 DW0

DECOUPREG3DW0 - Decouple Register 3 DW0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	00F18h-00F1Bh	
DWord	**Bit**	**Description**
0	31:0	**DecoupReg3DW0Data** Access: R/W _Custom_GTIReset: BUS Decouple Register 3 DW0 Data.

Decouple Register 3 DW1

DECOUPREG3DW1 - Decouple Register 3 DW1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00F1Ch-00F1Fh	
DWord	**Bit**	**Description**
0	31	**GO** Access: R/W Lock Decouple Register Go/Status Bit: Software sets this bit along with attributes (full DW write) to initiate a DCR request. Hardware clears this bit to 0 when the command is complete. This bit serves as a lock for the register pair preventing software from updating the status/values once set.
	30:28	**OP** Access: R/W Lock Decouple Register Opcode: 3'b000 = Read, 3'b001 = Write; All others undefined (Request ignored and go/status cleared).
	27:24	**BE_B** Access: R/W Lock Decouple Register Active Low Byte Enables. Byte enables affect data merging into MGSR shadow storage however GT only supports full dword accesses.
	23:0	**Addr** Access: R/W Lock Decouple Register Address.

Decouple Register 4 DW0

DECOUPREG4DW0 - Decouple Register 4 DW0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	00F20h-00F23h	
DWord	**Bit**	**Description**
0	31:0	**DecoupReg4DW0Data** Access: R/W _Custom_GTIReset: BUS Decouple Register 4 DW0 Data. DCR requests on the NP path will be blocked during CPD

Decouple Register 4 DW1

DECOUPREG4DW1 - Decouple Register 4 DW1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00F24h-00F27h	
DWord	**Bit**	**Description**
0	31	**GO** Access: R/W Lock Decouple Register Go/Status Bit: Software sets this bit along with attributes (full DW write) to initiate a DCR request. Hardware clears this bit to 0 when the command is complete. This bit serves as a lock for the register pair preventing software from updating the status/values once set.
	30:28	**OP** Access: R/W Lock Decouple Register Opcode: 3'b000 = Read, 3'b001 = Write; All others undefined (Request ignored and go/status cleared).
	27:24	**BE_B** Access: R/W Lock Decouple Register Active Low Byte Enables. Byte enables affect data merging into MGSR shadow storage however GT only supports full dword accesses.
	23:0	**Addr** Access: R/W Lock Decouple Register Address.

Decouple Register 5 DW0

DECOUPREG5DW0 - Decouple Register 5 DW0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	00F28h-00F2Bh	
DWord	**Bit**	**Description**
0	31:0	**DecoupReg5DW0Data** Access: R/W _Custom_GTIReset: BUS Decouple Register 5 DW0 Data.

Decouple Register 5 DW1

DECOUPREG5DW1 - Decouple Register 5 DW1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00F2Ch-00F2Fh	
DWord	**Bit**	**Description**
0	31	**GO** Access: R/W Lock Decouple Register Go/Status Bit: Software sets this bit along with attributes (full DW write) to initiate a DCR request. Hardware clears this bit to 0 when the command is complete. This bit serves as a lock for the register pair preventing software from updating the status/values once set.
	30:28	**OP** Access: R/W Lock Decouple Register Opcode: 3'b000 = Read, 3'b001 = Write; All others undefined (Request ignored and go/status cleared).
	27:24	**BE_B** Access: R/W Lock Decouple Register Active Low Byte Enables. Byte enables affect data merging into MGSR shadow storage however GT only supports full dword accesses.
	23:0	**Addr** Access: R/W Lock Decouple Register Address.

Decouple Register 6 DW0

DECOUPREG6DW0 - Decouple Register 6 DW0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	00F30h-00F33h	
DWord	**Bit**	**Description**
0	31:0	**DecoupReg6DW0Data** Access: R/W _Custom_GTIReset: BUS Decouple Register 6 DW0 Data.

Decouple Register 6 DW1

DECOUPREG6DW1 - Decouple Register 6 DW1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00F34h-00F37h	
DWord	**Bit**	**Description**
0	31	**GO** Access: R/W Lock Decouple Register Go/Status Bit: Software sets this bit along with attributes (full DW write) to initiate a DCR request. Hardware clears this bit to 0 when the command is complete. This bit serves as a lock for the register pair preventing software from updating the status/values once set.
	30:28	**OP** Access: R/W Lock Decouple Register Opcode: 3'b000 = Read, 3'b001 = Write; All others undefined (Request ignored and go/status cleared).
	27:24	**BE_B** Access: R/W Lock Decouple Register Active Low Byte Enables. Byte enables affect data merging into MGSR shadow storage however GT only supports full dword accesses.
	23:0	**Addr** Access: R/W Lock Decouple Register Address.

Decouple Register 7 DW0

DECOUPREG7DW0 - Decouple Register 7 DW0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	00F38h-00F3Bh	
DWord	**Bit**	**Description**
0	31:0	**DecoupReg7DW0Data** Access: R/W _Custom_GTIReset: BUS Decouple Register 7 DW0 Data.

Decouple Register 7 DW1

DECOUPREG7DW1 - Decouple Register 7 DW1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00F3Ch-00F3Fh	
DWord	**Bit**	**Description**
0	31	**GO** Access: R/W Lock Decouple Register Go/Status Bit: Software sets this bit along with attributes (full DW write) to initiate a DCR request. Hardware clears this bit to 0 when the command is complete. This bit serves as a lock for the register pair preventing software from updating the status/values once set.
	30:28	**OP** Access: R/W Lock Decouple Register Opcode: 3'b000 = Read, 3'b001 = Write; All others undefined (Request ignored and go/status cleared).
	27:24	**BE_B** Access: R/W Lock Decouple Register Active Low Byte Enables. Byte enables affect data merging into MGSR shadow storage however GT only supports full dword accesses.
	23:0	**Addr** Access: R/W Lock Decouple Register Address.

Decouple Register 8 DW0

DECOUPREG8DW0 - Decouple Register 8 DW0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	00F40h-00F43h	
DWord	**Bit**	**Description**
0	31:0	**DecoupReg8DW0Data** Access: R/W _Custom_GTIReset: BUS Decouple Register 8 DW0 Data.

Decouple Register 8 DW1

DECOUPREG8DW1 - Decouple Register 8 DW1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00F44h-00F47h	
DWord	**Bit**	**Description**
0	31	**GO** Access: R/W Lock Decouple Register Go/Status Bit: Software sets this bit along with attributes (full DW write) to initiate a DCR request. Hardware clears this bit to 0 when the command is complete. This bit serves as a lock for the register pair preventing software from updating the status/values once set.
	30:28	**OP** Access: R/W Lock Decouple Register Opcode: 3'b000 = Read, 3'b001 = Write; All others undefined (Request ignored and go/status cleared).
	27:24	**BE_B** Access: R/W Lock Decouple Register Active Low Byte Enables. Byte enables affect data merging into MGSR shadow storage however GT only supports full dword accesses.
	23:0	**Addr** Access: R/W Lock Decouple Register Address.

Decouple Register 9 DW0

DECOUPREG9DW0 - Decouple Register 9 DW0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	00F48h-00F4Bh	
DWord	**Bit**	**Description**
0	31:0	**DecoupReg9DW0Data** Access: R/W _Custom_GTIReset: BUS Decouple Register 9 DW0 Data.

Decouple Register 9 DW1

DECOUPREG9DW1 - Decouple Register 9 DW1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00F4Ch-00F4Fh	
DWord	**Bit**	**Description**
0	31	**GO** Access: R/W Lock Decouple Register Go/Status Bit: Software sets this bit along with attributes (full DW write) to initiate a DCR request. Hardware clears this bit to 0 when the command is complete. This bit serves as a lock for the register pair preventing software from updating the status/values once set.
	30:28	**OP** Access: R/W Lock Decouple Register Opcode: 3'b000 = Read, 3'b001 = Write; All others undefined (Request ignored and go/status cleared).
	27:24	**BE_B** Access: R/W Lock Decouple Register Active Low Byte Enables. Byte enables affect data merging into MGSR shadow storage however GT only supports full dword accesses.
	23:0	**Addr** Access: R/W Lock Decouple Register Address.

Decouple Register 10 DW0

DECOUPREG10DW0 - Decouple Register 10 DW0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	00F50h-00F53h	
DWord	**Bit**	**Description**
0	31:0	**DecoupReg10DW0Data** Access: R/W _Custom_GTIReset: BUS Decouple Register 10 DW0 Data.

Decouple Register 10 DW1

DECOUPREG10DW1 - Decouple Register 10 DW1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00F54h-00F57h	
DWord	**Bit**	**Description**
0	31	**GO** Access: R/W Lock Decouple Register Go/Status Bit: Software sets this bit along with attributes (full DW write) to initiate a DCR request. Hardware clears this bit to 0 when the command is complete. This bit serves as a lock for the register pair preventing software from updating the status/values once set.
	30:28	**OP** Access: R/W Lock Decouple Register Opcode: 3'b000 = Read, 3'b001 = Write; All others undefined (Request ignored and go/status cleared).
	27:24	**BE_B** Access: R/W Lock Decouple Register Active Low Byte Enables. Byte enables affect data merging into MGSR shadow storage however GT only supports full dword accesses.
	23:0	**Addr** Access: R/W Lock Decouple Register Address.

Decouple Register 11 DW0

DECOUPREG11DW0 - Decouple Register 11 DW0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	00F58h-00F5Bh	
DWord	**Bit**	**Description**
0	31:0	**DecoupReg11DW0Data** Access: R/W _Custom_GTIReset: BUS Decouple Register 11 DW0 Data.

Decouple Register 11 DW1

DECOUPREG11DW1 - Decouple Register 11 DW1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00F5Ch-00F5Fh	
DWord	**Bit**	**Description**
0	31	**GO** Access: R/W Lock Decouple Register Go/Status Bit: Software sets this bit along with attributes (full DW write) to initiate a DCR request. Hardware clears this bit to 0 when the command is complete. This bit serves as a lock for the register pair preventing software from updating the status/values once set.
	30:28	**OP** Access: R/W Lock Decouple Register Opcode: 3'b000 = Read, 3'b001 = Write; All others undefined (Request ignored and go/status cleared).
	27:24	**BE_B** Access: R/W Lock Decouple Register Active Low Byte Enables. Byte enables affect data merging into MGSR shadow storage however GT only supports full dword accesses.
	23:0	**Addr** Access: R/W Lock Decouple Register Address.

Decouple Register 12 DW0

DECOUPREG12DW0 - Decouple Register 12 DW0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	00F60h-00F63h	
DWord	**Bit**	**Description**
0	31:0	**DecoupReg12DW0Data** Access: R/W _Custom_GTIReset: BUS Decouple Register 12 DW0 Data.

Decouple Register 12 DW1

DECOUPREG12DW1 - Decouple Register 12 DW1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00F64h-00F67h	
DWord	**Bit**	**Description**
0	31	**GO** Access: R/W Lock Decouple Register Go/Status Bit: Software sets this bit along with attributes (full DW write) to initiate a DCR request. Hardware clears this bit to 0 when the command is complete. This bit serves as a lock for the register pair preventing software from updating the status/values once set.
	30:28	**OP** Access: R/W Lock Decouple Register Opcode: 3'b000 = Read, 3'b001 = Write; All others undefined (Request ignored and go/status cleared).
	27:24	**BE_B** Access: R/W Lock Decouple Register Active Low Byte Enables. Byte enables affect data merging into MGSR shadow storage however GT only supports full dword accesses.
	23:0	**Addr** Access: R/W Lock Decouple Register Address.

Decouple Register 13 DW0

DECOUPREG13DW0 - Decouple Register 13 DW0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	00F68h-00F6Bh	
DWord	**Bit**	**Description**
0	31:0	**DecoupReg13DW0Data** Access: R/W _Custom_GTIReset: BUS Decouple Register 13 DW0 Data.

Decouple Register 13 DW1

DECOUPREG13DW1 - Decouple Register 13 DW1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00F6Ch-00F6Fh	
DWord	**Bit**	**Description**
0	31	**GO** Access: R/W Lock Decouple Register Go/Status Bit: Software sets this bit along with attributes (full DW write) to initiate a DCR request. Hardware clears this bit to 0 when the command is complete. This bit serves as a lock for the register pair preventing software from updating the status/values once set.
	30:28	**OP** Access: R/W Lock Decouple Register Opcode: 3'b000 = Read, 3'b001 = Write; All others undefined (Request ignored and go/status cleared).
	27:24	**BE_B** Access: R/W Lock Decouple Register Active Low Byte Enables. Byte enables affect data merging into MGSR shadow storage however GT only supports full dword accesses.
	23:0	**Addr** Access: R/W Lock Decouple Register Address.

Decouple Register 14 DW0

DECOUPREG14DW0 - Decouple Register 14 DW0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	00F70h-00F73h	
DWord	**Bit**	**Description**
0	31:0	**DecoupReg14DW0Data** Access: R/W _Custom_GTIReset: BUS Decouple Register 14 DW0 Data.

Decouple Register 14 DW1

DECOUPREG14DW1 - Decouple Register 14 DW1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00F74h-00F77h	
DWord	**Bit**	**Description**
0	31	**GO** Access: R/W Lock Decouple Register Go/Status Bit: Software sets this bit along with attributes (full DW write) to initiate a DCR request. Hardware clears this bit to 0 when the command is complete. This bit serves as a lock for the register pair preventing software from updating the status/values once set.
	30:28	**OP** Access: R/W Lock Decouple Register Opcode: 3'b000 = Read, 3'b001 = Write; All others undefined (Request ignored and go/status cleared).
	27:24	**BE_B** Access: R/W Lock Decouple Register Active Low Byte Enables. Byte enables affect data merging into MGSR shadow storage however GT only supports full dword accesses.
	23:0	**Addr** Access: R/W Lock Decouple Register Address.

Decouple Register 15 DW0

DECOUPREG15DW0 - Decouple Register 15 DW0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	00F78h-00F7Bh	
DWord	**Bit**	**Description**
0	31:0	**DecoupReg15DW0Data** Access: R/W _Custom_GTIReset: BUS Decouple Register 15 DW0 Data.

Decouple Register 15 DW1

DECOUPREG15DW1 - Decouple Register 15 DW1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00F7Ch-00F7Fh	
DWord	**Bit**	**Description**
0	31	**GO** Access: R/W Lock Decouple Register Go/Status Bit: Software sets this bit along with attributes (full DW write) to initiate a DCR request. Hardware clears this bit to 0 when the command is complete. This bit serves as a lock for the register pair preventing software from updating the status/values once set.
	30:28	**OP** Access: R/W Lock Decouple Register Opcode: 3'b000 = Read, 3'b001 = Write; All others undefined (Request ignored and go/status cleared).
	27:24	**BE_B** Access: R/W Lock Decouple Register Active Low Byte Enables. Byte enables affect data merging into MGSR shadow storage however GT only supports full dword accesses.
	23:0	**Addr** Access: R/W Lock Decouple Register Address.

Dedicated Path Arbiter Credits

DEDICATED_PATH_ARB_CREDITS - Dedicated Path Arbiter Credits

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	100180h

This register holds the credits for arbitrating between different ABOXs for streaming HP traffic to the dedicated memory path

DWord	Bit	Description
0	31:15	**Spare_0** Default Value: 0000h Access: R/W _Custom_GTIReset: BUS Spare bits
	14:8	**ABOX1 Credits** Default Value: 00h Access: R/W _Custom_GTIReset: BUS Number of Credits for ABOX1
	7	**Spare_1** Default Value: 0b Access: R/W _Custom_GTIReset: BUS Spare bit
	6:0	**ABOX0 Credits** Default Value: 00h Access: R/W _Custom_GTIReset: BUS Number of Credits for ABOX0

DE HPD Interrupt Definition

DE HPD Interrupt Definition	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	44470h-4447Fh
Name:	Display Engine HPD Interrupts
ShortName:	DE_HPD_INTERRUPT
Reset:	soft

This table indicates which events are mapped to each bit of the Display Engine HPD Interrupt registers.
0x44470 = ISR
0x44474 = IMR
0x44478 = IIR
0x4447C = IER
The HPD bits use a modified interrupt structure where the live value goes to the ISR and a processed event goes to the IMR and IIR path.

DWord	Bit	Description
0	31	**Unused 31**
	30	**Unused 30**
	29	**Unused 29**
	28	**Unused 28**
	27	**Unused 27**
	26	**Unused 26**
	25	**Unused 25**
	24	**Unused 24**
	23	**TC8 Hotplug** The ISR gives the live state of the HPD for typeC DP alternate mode. The IIR is set if a short or long pulse is detected when HPD input is enabled.
	22	**TC7 Hotplug** The ISR gives the live state of the HPD for typeC DP alternate mode. The IIR is set if a short or long pulse is detected when HPD input is enabled.
	21	**TC6 Hotplug** The ISR gives the live state of the HPD for typeC DP alternate mode. The IIR is set if a short or long pulse is detected when HPD input is enabled.
	20	**TC5 Hotplug** The ISR gives the live state of the HPD for typeC DP alternate mode. The IIR is set if a short or long pulse is detected when HPD input is enabled.
	19	**TC4 Hotplug** The ISR gives the live state of the HPD for typeC DP alternate mode. The IIR is set if a short or long pulse is detected when HPD input is enabled.

DE HPD Interrupt Definition		
	18	**TC3 Hotplug** The ISR gives the live state of the HPD for typeC DP alternate mode. The IIR is set if a short or long pulse is detected when HPD input is enabled.
	17	**TC2 Hotplug** The ISR gives the live state of the HPD for typeC DP alternate mode. The IIR is set if a short or long pulse is detected when HPD input is enabled.
	16	**TC1 Hotplug** The ISR gives the live state of the HPD for typeC DP alternate mode. The IIR is set if a short or long pulse is detected when HPD input is enabled.
	15	**Unused 15**
	14	**Unused 14**
	13	**Unused 13**
	12	**Unused 12**
	11	**Unused 11**
	10	**Unused 10**
	9	**Unused 9**
	8	**Unused 8**
	7	**TBT8 Hotplug** The ISR gives the live state of the HPD for thunderbolt. The IIR is set if a short or long pulse is detected when HPD input is enabled.
	6	**TBT7 Hotplug** The ISR gives the live state of the HPD for thunderbolt. The IIR is set if a short or long pulse is detected when HPD input is enabled.
	5	**TBT6 Hotplug** The ISR gives the live state of the HPD for thunderbolt. The IIR is set if a short or long pulse is detected when HPD input is enabled.
	4	**TBT5 Hotplug** The ISR gives the live state of the HPD for thunderbolt. The IIR is set if a short or long pulse is detected when HPD input is enabled.
	3	**TBT4 Hotplug** The ISR gives the live state of the HPD for thunderbolt. The IIR is set if a short or long pulse is detected when HPD input is enabled.
	2	**TBT3 Hotplug** The ISR gives the live state of the HPD for thunderbolt. The IIR is set if a short or long pulse is detected when HPD input is enabled.

DE HPD Interrupt Definition		
	1	**TBT2 Hotplug** The ISR gives the live state of the HPD for thunderbolt. The IIR is set if a short or long pulse is detected when HPD input is enabled.
	0	**TBT1 Hotplug** The ISR gives the live state of the HPD for thunderbolt. The IIR is set if a short or long pulse is detected when HPD input is enabled.

DE Misc Interrupt Definition

DE Misc Interrupt Definition

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	44460h-4446Fh
Name:	Display Engine Miscellaneous Interrupts
ShortName:	DE_MISC_INTERRUPT
Reset:	soft

This table indicates which events are mapped to each bit of the Display Engine Miscellaneous Interrupt registers.
0x44460 = ISR
0x44464 = IMR
0x44468 = IIR
0x4446C = IER

DWord	Bit	Description
0	31	**Poison** The ISR is an active high pulse on receiving the poison response to a memory transaction.
	30	**ECC_Double_Error** The ISR is an active high level while any of the ECC Double Error status bits are set.
	29	**Reserved** Access: RO Format: MBZ
	28	**Isoch_msg_error** The ISR is an active high pulse on the Isoch messages error.
	27	**Isoch_rsptimeout_error** The ISR is an active high pulse on the Isoch response timeout error.
	26	**Reserved** Access: RO Format: MBZ
	25	**Reserved**
	24	**Reserved**
	23	**WD0_Interrupts_Combined** The ISR is an active high level while any of the WD0_IIR bits are set.
	22:20	**Reserved** Access: RO Format: MBZ

DE Misc Interrupt Definition

	19	**SRD_Interrupts_Combined** The ISR is an active high level while any of the SRD_IIR bits are set.
	18	**WD1_Interrupts_Combined** The ISR is an active high level while any of the WD1_IIR bits are set.
	17:16	**Reserved** Access: RO Format: MBZ
	15	**GTC_Interrupts_Combined** The ISR is an active high level while any of the GTC_IIR bits are set.
	14:8	**Reserved** Access: RO Format: MBZ
	7	**Reserved**
	6	**Reserved**
	5	**Reserved**
	4	**Reserved**
	3	**NonPipe Isoch Ack** The ISR is an active high pulse when the Non-Pipe IsocReq receives an Ack.
	2	**Reserved** Access: RO Format: MBZ
	1:0	**Reserved** Access: RO Format: MBZ

DE Port Interrupt Definition

DE Port Interrupt Definition

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	44440h-4444Fh
Name:	Display Engine Port Interrupts
ShortName:	DE_PORT_INTERRUPT
Reset:	soft

This table indicates which events are mapped to each bit of the Display Engine Port Interrupt registers.
0x44440 = ISR
0x44444 = IMR
0x44448 = IIR
0x4444C = IER

DWord	Bit	Description
0	31	**DSI1** The ISR is an active high level indicating a non-TE interrupt is set in DSI_INTER_IDENT_REG_1.
	30	**DSI0** The ISR is an active high level indicating a non-TE interrupt is set in DSI_INTER_IDENT_REG_0.
	29:25	**Reserved** Access: RO Format: MBZ
	24	**DSI1 TE** The ISR is an active high level indicating a TE interrupt is set in DSI_INTER_IDENT_REG_1.
	23	**DSI0 TE** The ISR is an active high level indicating a TE interrupt is set in DSI_INTER_IDENT_REG_0.
	22:21	**Reserved** Access: RO Format: MBZ
	20:14	**Reserved** Access: RO Format: MBZ
	13	**AUX USBC6** The ISR is an active high pulse on the AUX done event. This event will not occur for HW triggered AUX transactions.

DE Port Interrupt Definition

	12	**AUX USBC5** The ISR is an active high pulse on the AUX done event. This event will not occur for HW triggered AUX transactions.
	11	**AUX USBC4** The ISR is an active high pulse on the AUX done event. This event will not occur for HW triggered AUX transactions.
	10	**AUX USBC3** The ISR is an active high pulse on the AUX done event. This event will not occur for HW triggered AUX transactions.
	9	**AUX USBC2** The ISR is an active high pulse on the AUX done event. This event will not occur for HW triggered AUX transactions.
	8	**AUX USBC1** The ISR is an active high pulse on the AUX done event. This event will not occur for HW triggered AUX transactions.
	7:3	**Reserved** Access: RO Format: MBZ
	2	**AUX DDIC** The ISR is an active high pulse on the AUX done event. This event will not occur for HW triggered AUX transactions.
	1	**AUX DDIB** The ISR is an active high pulse on the AUX done event. This event will not occur for HW triggered AUX transactions.
	0	**AUX DDIA** The ISR is an active high pulse on the AUX done event. This event will not occur for HW triggered AUX transactions.

Device 2 Control

DEV2CTL_0_2_0_PCI - Device 2 Control			
Register Space:	PCI: 0/2/0		
Size (in bits):	8		
Address:	00058h		
This register implements a control bit to disable and hide the IOBAR register in systems that do not require legacy IOBAR access to Gfx MMIO registers.			
DWord	**Bit**	**Description**	
0	7:1	**Reserved**	
		Access:	RO
		Format:	MBZ
	0	**Reserved**	

Device Capabilities

DEVICECAP_0_2_0_PCI - Device Capabilities

Register Space:	PCI: 0/2/0
Size (in bits):	32
Address:	00074h

PCI Express Device Capabilities

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28	**Functional Level Reset Capability** Default Value: 1b Access: RO _Custom_GTIReset: BUS Hardwired to 1b to indicate the Function supports the optional Function Level Reset mechanism.
	27:26	**Captured Slot Power Limit Scale** Default Value: 00b Access: RO _Custom_GTIReset: BUS Not applicable for a Root Complex Integrated Endpoint with no Link or Slot. Hardwired to 00b
	25:18	**Captured Slot Power Limit Value** Default Value: 00000000h Access: RO _Custom_GTIReset: BUS Not applicable for a Root Complex Integrated Endpoint with no Link or Slot. Hardwired to 00h
	17:16	**Reserved** Access: RO Format: MBZ
	15	**Role-Based Error Reporting** Default Value: 1b Access: RO _Custom_GTIReset: BUS When Set, this bit indicates that the Function implements the functionality originally defined in the Error Reporting ECN for PCI Express Base Specification, Revision 1.0a, and later incorporated

DEVICECAP_0_2_0_PCI - Device Capabilities

		into PCI Express Base Specification, Revision 1.1.Hardwired to 1b as this bit must be Set by all Functions conforming to the ECN, PCI Express Base Specification, Revision 1.1, or subsequent PCI Express Base Specification revisions.
	14:12	**Reserved** Access: RO Format: MBZ
	11:9	**Endpoint L1 Acceptable Latency** Default Value: 000b Access: RO _Custom_GTIReset: BUS This field indicates the acceptable total latency that an Endpoint can withstand due to the transition from the L1 state to the L0 state. This does not apply to the integrated graphics device, so it is hardwired to 000b (Maximum of 1 us).
	8:6	**Endpoint L0s Acceptable Latency** Default Value: 000b Access: RO _Custom_GTIReset: BUS This field indicates the acceptable total latency that an Endpoint can withstand due to the transition from the L0s state to the L0 state. This does not apply to the integrated graphics device, so it is hardwired to 000b (Maximum of 64 ns).
	5	**Extended Tag Field Supported** Default Value: 0b Access: RO _Custom_GTIReset: BUS This bit indicates the maximum supported size of the Tag field as a Requester.This does not apply to the integrated graphics device, so it is hardwired to 0b (5-bit Tag field supported).
	4:3	**Phantom Functions Supported** Default Value: 00b Access: RO _Custom_GTIReset: BUS This field indicates the support for use of unclaimed Function Numbers to extend the number of outstanding transactions for PCIe devices.This does not apply to the integrated graphics device, so it is hardwired to 00b to indicate no Function Number bits are used for Phantom Functions.

DEVICECAP_0_2_0_PCI - Device Capabilities

	2:0	**Max Payload Size Supported** Default Value: 000b Access: RO _Custom_GTIReset: BUS This field indicates the maximum payload size that the Function can support for TLPs. Hardwired to 000b to represents 128 bytes, the minimum allowed value.

Device Enable

DEVEN_0_0_0_PCI - Device Enable

Register Space: PCI: 0/0/0
Size (in bits): 32

Address: 00054h

Allows for enabling/disabling of PCI devices and functions that are within the CPU package.

DWord	Bit	Description
0	31:15	**Reserved** Access: RO Format: MBZ
	14	**Chap Enable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS Unused - Bit field not relevant for the current project
	13	**Device 6 Enable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS Unused - Bit field not relevant for the current project
	12:11	**Reserved** Access: RO Format: MBZ
	10	**Device 5 Enable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS Unused - Bit field not relevant for the current project
	9:8	**Reserved** Access: RO Format: MBZ

DEVEN_0_0_0_PCI - Device Enable

	7	**Device 4 Enable** Default Value: 1b Access: R/W _Custom_GTIReset: BUS Unused - Bit field not relevant for the current project
	6	**Reserved** Access: RO Format: MBZ
	5	**Device 3 enable for Display HD Audio** Default Value: 1b Access: R/W _Custom_GTIReset: BUS
	4	**Internal Graphics Engine** Default Value: 1b Access: R/W _Custom_GTIReset: BUS 0: Bus 0 Device 2 is disabled and hidden1: Bus 0 Device 2 is enabled and visible This bit will be set to 0b and remain 0b if Device 2 capability is disabled.
	3	**PEG10 Enable** Default Value: 1b Access: R/W _Custom_GTIReset: BUS Unused - Bit field not relevant for the current project
	2	**PEG11 Enable** Default Value: 1b Access: R/W _Custom_GTIReset: BUS Unused - Bit field not relevant for the current project
	1	**PEG12 Enable** Default Value: 1b Access: R/W _Custom_GTIReset: BUS

DEVEN_0_0_0_PCI - Device Enable		
	0	**Host Bridge** Default Value: 1b Access: RO _Custom_GTIReset: BUS

Device Identification

DID2_0_2_0_PCI - Device Identification

Register Space:	PCI: 0/2/0
Size (in bits):	16
Address:	00002h

This register combined with the Vendor Identification register uniquely identifies any PCI device.

DWord	Bit	Description
0	15:7	**Device Identification Number MSB** Default Value: 100110100b Access: R/W Variant _Custom_GTIReset: BUS All 16 bits of Device ID is acquired through fuse pull as per Chassis 2.1 updates.
	6:0	**Device Identification Number SKU** Default Value: 1000000b Access: RO Variant _Custom_GTIReset: BUS All 16 bits of Device ID is acquired through fuse pull as per Chassis 2.1 updates.

DFSDONE

DFSDONE	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	51080h-51083h
Name:	Display Fuse Done
ShortName:	DFSDONE
Reset:	global

This register is not reset by FLR.

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**Download Done** This field indicates when fuse download is complete. 0b – Note Done 1b – Done

DFSM

DFSM

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	51000h-51003h
Name:	Display Fuse
ShortName:	DFSM
Reset:	global

This register contains fuse and strap settings for display. This register is not reset by FLR.

DWord	Bit	Description
0	31	**Reserved**
	30	**Display PipeA Disable** This bit indicates whether the display pipe A (first pipe) capability is disabled. Value / Name / Description: 0b / Enable / Pipe A Capability Enabled 1b / Disable / Pipe A Capability Disabled
	29	**Reserved**
	28	**Display PipeC Disable** This bit indicates whether the display pipe C (third pipe) capability is disabled. Value / Name / Description: 0b / Enable / Pipe C Capability Enabled 1b / Disable / Pipe C Capability Disabled
	27	**Display PM Disable** This bit indicates whether the display power management FBC and DPST capabilities are disabled. Value / Name / Description: 0b / Enable / PM Capability Enabled 1b / Disable / PM Capability Disabled
	26	**Display eDP Disable** Description: This fuse indicates that all combo PHY ports are disabled by the SoC and cannot be used. Value / Name / Description: 0b / Enable / eDP Capability Enabled 1b / Disable / eDP Capability Disabled
	25	**Reserved**
	24	**Reserved**

23	**Reserved**
22	**Display PipeD Disable** This bit indicates whether the display pipe D (fourth pipe) capability is disabled. **Value** / **Name** 0b – Enable 1b – Disable
21	**Display PipeB Disable** This bit indicates whether the display pipe B (second pipe) capability is disabled. **Value** / **Name** 0b – Pipe B Capability Enabled 1b – Pipe B Capability Disabled
20	**Display WD Disable** This bit indicates whether the display WD capability is disabled. **Value** / **Name** / **Description** 0b – Enable – WD Capability Enabled 1b – Disable – WD Capability Disabled
19	**Reserved**
18	**Reserved**
17	**Reserved**
16	**Isolated Decode Disable** This field indicates whether the Isolated Decode feature is disabled. **Value** / **Name** 0b – Isolated Decode Capability Enabled 1b – Isolated Decode Capability Disabled
15:8	**Audio Codec ID** This field indicates the lower 8 bits of the audio codec device ID. See the root node F00 verb for the device IDs on each project. **Value** / **Name** / **Description** 0Bh – Audio Codec ID 280Bh **[Default]** – Default value is N/A. Fuse download will override with correct value for this project.
7	**Display DSC Disable** This field indicates whether the DSC (port Display Stream Compression) feature is disabled. **Value** / **Name** 0b – DSC Capability Enabled 1b – DSC Capability Disabled
6	**Display RSB Enable** This bit indicates whether the remote screen blanking feature is enabled in the display engine.

DFSM

		Value / **Name** / **Description** 0b / Disable / RSB Capability Disabled 1b / Enable / RSB Capability Enabled
	5	**Reserved**
	4	**Reserved**
	3	**Reserved**
	2	**Reserved**
	1	**Reserved**
	0	**Display Audio Codec Disable** This bit indicates whether the display audio codec capability is disabled. **Value** / **Name** / **Description** 0b / Enable / Audio Codec Capability Enabled 1b / Disable / Audio Codec Capability Disabled

DG_CLKREQ_POLICY

DG_CLKREQ_POLICY - DG_CLKREQ_POLICY

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	101038h

Various bits that control clock req functionality in DG. This register should be programmed as part of graphics initialization.

DWord	Bit	Description
0	31:24	**SPARE2** Default Value: 00000000b Access: R/W _Custom_GTIReset: BUS Reserved
	23:16	**CLKREQ_HYST_CNTR** Default Value: 00000000b Access: R/W _Custom_GTIReset: BUS Register to store the parameter used to counting IDLE cycles during Hysteris state. Deassert clkreq when DG is idle for the number of cycles this register is programmed to.
	15:2	**SPARE1** Default Value: 00000000000000b Access: R/W _Custom_GTIReset: BUS Reserved
	1	**Reserved**
	0	**Reserved**

Discard counter status register

L3DISCARDCNTR - Discard counter status register		
Register Space:	MMIO: 0/2/0	
Access:	RO	
Size (in bits):	32	
_Custom_GTIReset:	DEV	
Address:	0B130h	
DWord	**Bit**	**Description**
0	31:16	**Discard Counter** Counter indicating Total evictions (in multiple of 64 events) dropped due to discard mechanism
	15:0	**C/Z Active Tile Eviction Counter** Counter indicating total number of eviction (in multiple of 64 events) from C/Z (or unified tile cache) ways when the tile it belongs to is active

Discard Enables for Z streams

Z_DISCARD_EN - Discard Enables for Z streams

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	07040h
ShortName:	Z_DISCARD_EN

Under Posh Based Tiled Rendering (aka PTBR), at the end of tile, certain streams can be discarded from the on-chip caches i.e. without evicting to memory. SW programs LOAD_REG_MEMORY command in the RCS command buffer to program this register. HW uses this value live from the register
This register defines the discard bits for Z streams.
Setting the discard enable bit for a stream, allows HW to drop writing back dirty cachelines from on-chip caches to memory.

Programming Notes

This register value must not change during the render pass (aka tile pass).

DWord	Bit	Description
0	31:2	**Reserved** Access: RO Format: MBZ
	1	**STC Discard Enable** When this bit is set, Stencil (STC) stream can be discarded at the end of tile in the PTBR mode. 0 – Disable **[Default]** 1 – Enable
	0	**Z Discard Enable** Access: R/W **Description** When this bit is set, all Z streams i.e. HiZ and Z can be discarded at the end of tile in the PTBR mode. 0 – Disable **[Default]** 1 – Enable

DISMBASE_LSB

DISMBASE_LSB - DISMBASE_LSB		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
SOC_Consumer:	BIOS	
Address:	108410h	
32 bit register that defines LSB of Display Stolen Memory base.		
DWord	**Bit**	**Description**
0	31:24	**DISM_BASE_LSB** Default Value: 00000000b Access: R/W _Custom_GTIReset: BUS
	23:0	**Reserved** Access: RO Format: MBZ

DISMBASE_MSB

DISMBASE_MSB - DISMBASE_MSB		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
SOC_Consumer:	BIOS	
Address:	108414h	
32 bit register that defines MSB of Display Stolen memory base.		
DWord	**Bit**	**Description**
0	31:0	**DISM_BASE_MSB** Default Value: 00000000000000000000000000000000b Access: R/W _Custom_GTIReset: BUS

DISMLIMIT_LSB

DISMLIMIT_LSB - DISMLIMIT_LSB		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
SOC_Consumer:	BIOS	
Address:	108418h	
32 bit register that defines LSB of Display Stolen Memory limit.		
DWord	**Bit**	**Description**
0	31:24	**DISM_LIMIT_LSB** Default Value: 00000000b Access: R/W _Custom_GTIReset: BUS
	23:0	**Reserved** Access: RO Format: MBZ

DISMLIMIT_MSB

DISMLIMIT_MSB - DISMLIMIT_MSB		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
SOC_Consumer:	BIOS	
Address:	10841Ch	
32 bit register that defines MSB of Display Stolen Memory limit.		
DWord	**Bit**	**Description**
0	31:0	**DISM_LIMIT_MSB** Default Value: 00000000000000000000000000000000b Access: R/W _Custom_GTIReset: BUS

DISPLAY_INT_CTL

DISPLAY_INT_CTL

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

Address:	44200h-44203h
Name:	Display Interrupt Control
ShortName:	DISPLAY_INT_CTL
Reset:	soft

This register has the primary enable for display interrupts and gives an overview of what interrupts are pending. An interrupt pending bit will read 1b while one or more interrupts of that category are set (IIR) and enabled (IER). All Pending Interrupts are ORed together to generate the combined interrupt.
The combined interrupt is ANDed with the Display Interrupt enable to create the display enabled interrupt.
The display enabled interrupt goes to graphics interrupt processing.

DWord	Bit	Description
0	31	**Display Interrupt Enable** Access: R/W This is the ultimate control for display interrupts. This must be enabled for any of these interrupts to propagate to graphics interrupt processing. 0b: Disable 1b: Enable
	30:26	**Reserved** Access: RO Format: MBZ
	25	**Direct PICA Interrupts Pending** Access: RO This field indicates if PICA interrupts are pending. This is use on SoCs that connect the PICA interrupt to North DE. Other SoCs connect the PICA interrupt to south DE because of partitioning and power domains. The PICA interrupt is configured through the PICA interrupt registers.
	24	**Audio Codec Interrupts Pending** Access: RO This field indicates if audio codec interrupts are pending.
	23	**DE PCH Interrupts Pending** Access: RO This field indicates if South (PCH) display interrupts are pending. The South Display interrupt is configured through the SDE interrupt registers.

DISPLAY_INT_CTL		
	22	**DE Misc Interrupts Pending** Access: RO This field indicates if DE Misc interrupts are pending.
	21	**DE HPD Interrupts Pending** Access: RO This field indicates if North DE HPD interrupts are pending.
	20	**DE Port Interrupts Pending** Access: RO This field indicates if Port interrupts are pending.
	19	**DE Pipe D Interrupts Pending** Access: RO This field indicates if Pipe D interrupts are pending.
	18	**DE Pipe C Interrupts Pending** Access: RO This field indicates if Pipe C interrupts are pending.
	17	**DE Pipe B Interrupts Pending** Access: RO This field indicates if Pipe B interrupts are pending.
	16	**DE Pipe A Interrupts Pending** Access: RO This field indicates if Pipe A interrupts are pending.
	15:0	**Reserved** Access: RO Format: MBZ

DKL_BIAS

DKL_BIAS	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168B3Ch-168B3Fh
Name:	DKL_BIAS_NULL
ShortName:	DKL_BIAS_NULL
Reset:	global

This register is not reset by the device 2 FLR.
This register is associated with the display PLL instance (PLL1).

DWord	Bit	Description
0	31	**i_tdc_fine_res** Default Value: 1b TDC fine resolution select 0: Coarse resolution / 8 1: Coarse resolution / 4
	30	**i_fracnen_h** Default Value: 1b Enables fractional modulator. For SSC, this bit needs to be set to '1', even though it starts with integer division ratio.
	29:24	**i_fbdiv_frac_21_16** Default Value: 1Eh Fractional modulator settings.
	23:16	**i_fbdiv_frac_15_8** Default Value: 00h Fractional modulator settings.
	15:8	**i_fbdiv_frac_7_0** Default Value: 00h Fractional modulator settings.
	7:0	**i_sscinj_stepsize_7_0** Default Value: 00h SSC injection step size.

DKL_CLKTOP2_CORECLKCTL1

DKL_CLKTOP2_CORECLKCTL1 - DKL_CLKTOP2_CORECLKCTL1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168B34h-168B37h
Name:	DKL_CLKTOP2_CORECLKCTL1_NULL
ShortName:	DKL_CLKTOP2_CORECLKCTL1_NULL
Reset:	global

This register is not reset by device 2 FLR.

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29	**od_clktop2_coreclkd_bypass** bypass enable of coreclkd to take input refclk.
	28	**od_clktop2_coreclkd_divretimeren_h** Default Value: 1b retimer enable: 0 for odd division ratio, 1 for even division ratio
	27	**Reserved** Access: RO Format: MBZ
	26	**od_clktop2_coreclkc_bypass** bypass enable of coreclkc to take input refclk
	25	**od_clktop2_coreclkc_divretimeren_h** retimer enable: 0 for odd division ratio, 1 for even division ratio
	24	**Reserved** Access: RO Format: MBZ
	23:16	**od_clktop2_coreclkb_divratio** Default Value: 8h divider ratio for coreclkb divider 8'h00: div/256 8'h01: div/257 8'h02 - 8'hff: div/2 - div/255 8'h08: PCIe4/PCIE3
	15:8	**od_clktop2_coreclka_divratio** Default Value: 5h divider ratio for coreclka divider 8'h00: div/256 8'h01: div/257 8'h02 - 8'hff: div/2 - div/255 8'h05: HBR3/HBR2 8'h0A: HBR/RBR

DKL_CLKTOP2_CORECLKCTL1 - DKL_CLKTOP2_CORECLKCTL1		
	7:6	**Reserved** Access: RO Format: MBZ
	5	**od_clktop2_coreclkb_bypass** bypass enable of coreclkb to take input refclk
	4	**od_clktop2_coreclkb_divretimeren_h** Default Value: 1b retimer enable: 0 for odd division ratio, 1 for even division ratio
	3	**Reserved** Access: RO Format: MBZ
	2	**od_clktop2_coreclka_bypass** bypass enable of coreclka to take input refclk
	1	**od_clktop2_coreclka_divretimeren_h** retimer enable: 0 for odd division ratio, 1 for even division ratio
	0	**Reserved** Access: RO Format: MBZ

DKL_CLKTOP2_HSCLKCTL

DKL_CLKTOP2_HSCLKCTL

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168B30h-168B33h
Name:	DKL_CLKTOP2_HSCLKCTL_NULL
ShortName:	DKL_CLKTOP2_HSCLKCTL_NULL
Reset:	global

This register is not reset by the device 2 FLR.

DWord	Bit	Description
0	31:27	**Reserved** Access: RO Format: MBZ
	26:25	**od_clktop2_clkobs_inputsel** mux select for oc_dfx_ck_clk2obs[0] digobs output 00b: hsdiv output clock 01b: iclk_bypass input from other clktop 10b: dsdiv output clock 11b: non-divided pll clock
	24	**od_clktop2_clk2obs_en_h** enable of oc_dfx_ck_clk2obs[0] digobs output 0b: Disable 1b: Enable
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Reserved**
	19	**Reserved** Access: RO Format: MBZ
	18	**od_clktop2_outclk_bypassen_h** enable of bypass clock output to the other clktop

DKL_CLKTOP2_HSCLKCTL

Value	Name
0b	Disable
1b	Enable

17 **Reserved**

Access:	RO
Format:	MBZ

16 **od_clktop2_coreclk_inputsel**

mux select for input clock to coreclk divhub

Value	Name
0b	hsdiv output
1b	dsdiv output

15:14 **od_clktop2_tlinedrv_clksel**

mux select for non-dedicated tlinedrv clock

Value	Name
00b	hsclkdiv output
01b	iclk_bypass input from other clktop
10b	dsdiv output clock
11b	non-divided pll clock

13:12 **od_clktop2_hsdiv_divratio**

Divider ratio for high speed divider. Div1

Value	Name
00b	Divide by 2
01b	Divide by 3
10b	Divide by 5
11b	Divide by 7

11:8 **od_clktop2_dsdiv_divratio**

Divider ratio settings for programmable divider. Div2

Value	Name
0000b	No Division
0001b	No Div **[Default]**
0010b	Divide by 2
0011b	Divide by 3
0100b	Divide by 4
0101b	Divide by 5
0110b	Divide by 6
0111b	Divide by 7
1000b	Divide by 8

DKL_CLKTOP2_HSCLKCTL

	1001b	Divide by 9
	1010b	Divide by 10

7 — od_clktop2_tlinedrv_overrideen

override enable for following 4 tlinedrv enables

Value	Name
0b	Disable
1b	Enable

6 — od_clktop2_tlinedrv_enleft_ded_h_ovrd

enable of left side dedicated tlinedrv to output full-rate clock to left lanes

Value	Name
0b	Disable
1b	Enable

5 — od_clktop2_tlinedrv_enright_ded_h_ovrd

enable of right side dedicated tlinedrv to output full-rate clock to right lanes

Value	Name
0b	Disable
1b	Enable

4 — od_clktop2_tlinedrv_enleft_h_ovrd

enable of left side tlinedrv to output divided clock to left lanes

Value	Name
0b	Disable
1b	Enable **[Default]**

3 — od_clktop2_tlinedrv_enright_h_ovrd

enable of right side tlinedrv to output divided clock to right lanes

Value	Name
0b	Disable
1b	Enable **[Default]**

2 — od_clktop2_dsdiv_en_h

Enable dsdiv clock divider. Div2.

Value	Name
0b	Disable
1b	Enable **[Default]**

1 — Reserved

Access:	RO
Format:	MBZ

DKL_CLKTOP2_HSCLKCTL		
	0	**od_clktop2_hsdiv_en_h** Enable high speed clock divider. Div1 **Value** / **Name** 0b – Disable 1b – Enable **[Default]**

DKL_CMN_ANA_DWORD28

DKL_CMN_ANA_DWORD28	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168B98h-168B9Bh
Name:	DKL_CMN_ANA_DWORD28_NULL
ShortName:	DKL_CMN_ANA_DWORD28_NULL
Reset:	global

This register is not reset by the device 2 FLR.

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28	**clktop2_id_vga_chpmp_div_en_h** Default Value: 1
	27:25	**clktop2_id_vga_chpmp_ck_divratio** Default Value: 0100b
	24:21	**Reserved** Access: RO Format: MBZ
	20	**clktop1_id_vga_chpmp_div_en_h** Default Value: 1
	19:16	**clktop1_id_vga_chpmp_ck_divratio** Default Value: 0101b
	15	**refclkin2_refclk_dlane_en** Default Value: 1
	14	**refclkin2_refclk_sel**
	13:12	**refclkin2_refclk_dlane_sel**
	11	**refclkin1_refclk_dlane_en** Default Value: 1
	10	**refclkin1_refclk_sel**
	9:8	**refclkin1_refclk_dlane_sel**
	7	**clktop2_vga_clk2dl_en** Default Value: 1

DKL_CMN_ANA_DWORD28		
	6	**clktop2_vga_clk_sel**
	5	**clktop2_divby2clk_bypass_en**
	4	**clktop2_plldivby2_2dmon_en_h**
	3	**clktop1_vga_clk2dl_en** Default Value: 1
	2	**clktop1_vga_clk_sel**
	1	**clktop1_divby2clk_bypass_en**
	0	**clktop1_plldivby2_2dmon_en_h**

DKL_CMN_DIG_PLL_MISC

DKL_CMN_DIG_PLL_MISC	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168B94h-168B97h
Name:	DKL_CMN_DIG_PLL_MISC_NULL
ShortName:	DKL_CMN_DIG_PLL_MISC_NULL
Reset:	global

This register is not reset by the device 2 FLR.

DWord	Bit	Description
0	31:18	**Reserved**
	17	**od_cri_cascaded_pll2_enable**
	16	**od_cri_cascaded_pll1_enable**
	15:1	**Reserved**
	0	**od_cri_pll2_pcie_enable**

DKL_CMN_UC_DW27

DKL_CMN_UC_DW27 - DKL_CMN_UC_DW27

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	168B50h
Name:	DKL_CMN_UC_DW27
ShortName:	DKL_CMN_UC_DW27

This register is not reset by device 2 FLR.

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28	**Spare28** Default Value: 0b Access: RO
	27	**Spare27** Default Value: 0b Access: RO
	26	**Spare26** Default Value: 0b Access: RO
	25	**Spare25** Default Value: 0b Access: RO
	24	**Spare24** Default Value: 0b Access: RO
	23	**Spare23** Default Value: 0b Access: RO
	22	**Spare22** Default Value: 0b Access: RO

DKL_CMN_UC_DW27 - DKL_CMN_UC_DW27

21 Spare21

Default Value:	0b
Access:	RO

20 Spare20

Default Value:	0b
Access:	RO

19 Spare19

Default Value:	0b
Access:	RO

18 Spare18

Default Value:	0b
Access:	RO

17 Spare17

Default Value:	0b
Access:	RO

16 Spare16

Default Value:	0b
Access:	RO

15 uC health

Access:	R/W

- PHY uC health bit default
  - FW not ready.
- uC mode
  - PHY will set this bit to 1 after FW download is complete. Display polls this bit to '1'.
- In uC mode, direct IOSF transactions are supported.

Value	Name	Description
0b	**[Default]**	uC mode: FW not ready. uC bypass mode: Display can force this bit to '1' to select uC bypass (direct IOSF) mode.
1b		uC mode: FW ready. uC bypass mode: direct IOSF mode is set.

14:13 Reserved

Access:	RO
Format:	MBZ

DKL_CMN_UC_DW27 - DKL_CMN_UC_DW27

12	**Spare12**	
	Default Value:	0b
	Access:	R/W
11	**Spare11**	
	Default Value:	0b
	Access:	R/W
10	**Spare10**	
	Default Value:	0b
	Access:	R/W
9	**Spare9**	
	Default Value:	1b
	Access:	R/W
8	**Spare8**	
	Default Value:	0b
	Access:	R/W
7	**Spare7**	
	Default Value:	0b
	Access:	R/W
6	**Spare6**	
	Default Value:	0b
	Access:	R/W
5	**Spare5**	
	Default Value:	1b
	Access:	R/W
4	**Spare4**	
	Default Value:	0b
	Access:	R/W
3	**Spare3**	
	Default Value:	0b
	Access:	R/W
2	**Spare2**	
	Default Value:	1b
	Access:	R/W
1	**Spare1**	
	Default Value:	0b
	Access:	R/W

DKL_CMN_UC_DW27 - DKL_CMN_UC_DW27		
	0	**Spare0** Default Value: 0b Access: R/W

DKL_DP_MODE

DKL_DP_MODE - DKL_DP_MODE		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	168B0Ch-168B0Fh	
Name:	DKL_DP_MODE_LN0_ACU_NULL	
ShortName:	DKL_DP_MODE_LN0_ACU_NULL	
Reset:	global	
This register is not reset by device 2 FLR.		

DWord	Bit	Description
0	31:24	**ldo_powerup_timer** Default Value: 02h Timer to decide when LDO would be up. In case pwr_gate is used instead of LDO, it's configured to wait 20us, i.e 9'd500 by default using 25 MHz susclk.
	23	**cfg_tr2_pwrgate_timer_bypass**
	22	**cfg_tr_pwrgate_timer_bypass**
	21	**cfg_cl_pwrgate_timer_bypass**
	20	**cfg_dig_pwrgate_timer_bypass**
	19	**cfg_crireg_cold_boot_done**
	18	**cfg_vr_pulldwn2gnd_tr2**
	17	**cfg_vr_pulldwn2gnd_tr**
	16	**cfg_ldo_powerup_timer_8**
	15	**cfg_corepwr_ack_with_pcs_pwrreq**
	14	**cfg_cri_digpwr_req**
	13	**cfg_laneclkreq_force** Default Value: 1b When cfg_laneclkreq_gating_ctrl is low, this value will be used for the lane sus_clk request
	12	**cfg_laneclkreq_gating_ctrl** 1'b1 - lanesusclk req gating enabled
	11	**cfg_susclk_gating_ctrl** 1'b1 - susclk gating enabled
	10	**cfg_rawpwr_req_override** Keep rawpwr on when this bit is 1
	9	**cfg_digpwr_req_override** Keep digpwr on when this bit is 1

DKL_DP_MODE - DKL_DP_MODE		
	8	**cfg_rawpwr_gating_ctrl** Power gating enable reg for raw power
	7	**cfg_dp_x2_mode** Indicates x2 mode for DP
	6	**cfg_dp_x1_mode** Indicates x1 mode for DP
	5	**cfg_tr2pwr_gating_ctrl** Power gating enable reg for tr2
	4	**cfg_trpwr_gating_ctrl** Power gating enable reg for tr
	3	**cfg_clnpwr_gating_ctrl** Power gating enable reg for cln
	2	**cfg_digpwr_gating_ctrl** Power gating enable reg for dig
	1	**cfg_gaonpwr_gating_ctrl** Power gating enable reg for gaon
	0	**cfg_suspwr_gating_ctrl** Power gating enable reg for sus

DKL_PLL_DIV0

DKL_PLL_DIV0	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168B00h-168B03h
Name:	DKL_PLL1_DIV0_NULL
ShortName:	DKL_PLL1_DIV0_NULL
Reset:	global

This register is not reset by the device 2 FLR.

DWord	Bit	Description
0	31:30	**i_truelock_criteria_1_0** Default Value: 01b True lock indicator criteria. After early lock generation, external PLL lock indicator asserted high when phase error is less than threshold value 00: 16 consecutive cycles 01: 32 consecutive cycles 10: 48 consecutive cycles 11: 64 consecutive cycles
	29:28	**i_earlylock_criteria_1_0** Default Value: 11b Early lock indicator criteria. Early PLL lock indicator asserted high when phase error is less than threshold value. 00: 16 consecutive cycles 01: 32 consecutive cycles 10: 48 consecutive cycles 11: 64 consecutive cycles
	27:25	**i_afc_startup_2_0** Default Value: 000b This is for AFC start point. 000: fine = 511 001: fine = 639 (+128) 010: fine = 767 (+256) 011: fine = 895 (+384) 100: NA 101: fine = 127 (-384) 110: fine = 255 (-256) 111: fine = 383 (-128).

DKL_PLL_DIV0		
	24	**i_divretimeren** Retiming of feedback clock
	23:21	**i_gainctrl_2_0** Default Value: 001b Adjustable gain for loop filter. Both coefficients shifted right by gainctrl before lock.
	20:16	**i_int_coeff_4_0** Default Value: 07h integral coeff. = 2^(-int_coeff), tageting up to 2^-11.
	15:12	**i_prop_coeff_3_0** Default Value: 0010b proportional coeff. = 2^(-prop_coeff+1).
	11:8	**i_fbprediv_3_0** Default Value: 10b predivider ratio 0000,0001 : reserved 0010: /2 0100: /4 0011: reserved Rest: reserved
	7:0	**i_fbdiv_intgr** Default Value: 69h Feedback divider post division (M2 integer)

DKL_PLL_DIV1

DKL_PLL_DIV1	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168B40h-168B43h
Name:	DKL_PLL1_DIV1_NULL
ShortName:	DKL_PLL1_DIV1_NULL
Reset:	global

This register is not reset by the device 2 FLR.

DWord	Bit	Description
0	31	**i_bw_ampmeas_window** Default Value: 0b 0: 10 modulation cycles of averaging for mplitude measurement 1: 20 modulation cycles of averaging for mplitude measurement
	30:29	**i_bias_calib_stepsize_1_0** Default Value: 00b Bias Calibration Step Size during linear search 00: 1 01: 2 10: 3 11: 4
	28:24	**i_ctrim_4_0** Default Value: 0Ch Cap trimming for irefout. This also has refclock dependency. Current default is for 24MHz.
	23	**i_fastlock_internal_reset** Default Value: 1b Clears internal fastlock memory so that next cold start will do both TDC and AFC calibration instead of fast lock. NOTE: this does not clear the i_fastlock_en_h register bit, clears functional register and self-clears[br]Formerly, i_bbthresh[3] (no longer strap).
	22:21	**i_bias_r_programmability** Default Value: 10b Bias Filter R programmability - note mapped to bias_bonus[1:0].

DKL_PLL_DIV1

	20:16	**i_ireftrim_4_0** Default Value: 1Ch Output current trim . Mirror ratio is changed based on constant current requirement. SInce it is refclock dependent, needs to be reconfigurable i_ireftrim[4:0] - 38.4/19.2 MHz= 5'h1C i_ireftrim[4:0] - 25.0/100.0 MHz = 5'h18 Step size: 20 uA.
	15	**i_biasfilter_en_delay** Default Value: 1b 0: Filter enabled even before pll enabled 1: Filter enable is delay until lock acquisition This bit is sensitive only when i_bias_filter_en is set (bit3)
	14	**i_bias_filter_en** Value: 0b — Name: Disable Value: 1b — Name: Enable **[Default]**
	13	**i_biascal_en_h** Bias Calibration Signal. Bias cal should be disable when override DCO coarse code.
	12	**i_dcodither_config** Whenever we have binary weighted MFC cap, this should be set to 1'b0. Ex: i_dcofine_resolution = 1'b0. For this case, this should be set to 1'b0. 0: No floating dither 1: Floating dither (511+Nobinary - Floating dither).
	11:8	**i_lockthresh_3_0** Default Value: 4h Digital lock detect threshold, the PLL will generate plllockout when the phase error detected by TDC is within lockthresh number of TDC counts for 64 consecutive cycles 5*16 (TDC Code) = 80 (16 is internally hard coded scalar value) - This setting isin middle of coarse range.
	7:0	**i_tdctargetcnt_7_0** Default Value: 22h TDC tristate buffer calibration counter value. Delay line loop oscillation is counted over two refclk cycles. This is used for TDC coarse code calibration.

DKL_PLL_FRAC_LOCK

DKL_PLL_FRAC_LOCK

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168B08h-168B0Bh
Name:	DKL_PLL1_FRAC_LOCK_NULL
ShortName:	DKL_PLL1_FRAC_LOCK_NULL
Reset:	global

This register is not reset by the device 2 FLR.

DWord	Bit	Description
0	31:30	**i_cml2cmosbonus_1_0** Default Value: 10b NOTE: These bits are ported to anatop bit [0] mapped to cml2cmosbonus[1] port - o_plllc_ck_dcocmosclkp_ana disable '1' = o_plllc_ck_dcocmosclkp_ana is disabled (ie for MG B0) '0' = both phases of the cmos clock toggle at the interface bit[1] mapped to cml2cmosbonus[2] port - available.
	29:27	**i_bb_gain2_2_0** Default Value: 000b BB gain for second BB range.
	26:24	**i_bb_gain1_2_0** Default Value: 000b BB gain for first BB range.
	23	**i_fastlock_en_h** Enable FLL based AFC; this replaces binary search based AFC. FLL based AFC faster than binary search. Value: 0b – Disable Value: 1b – Enable
	22:19	**i_fllafc_gain_3_0** Default Value: 1000b Initial FLL gain that decrements down to 1 every refclk cycle in the beginning of FLL based AFC.
	18:16	**i_fllafc_lockcnt_2_0** Default Value: 100b Number of refclk cycles for FLL lock after gain is reduced to 1.

DKL_PLL_FRAC_LOCK		
	15:8	**i_max_cselafc_7_0** Default Value: B5h Max. AFC code for a given DCO.
	7:0	**i_init_cselafc_7_0** Default Value: 6Ah Initial AFC code for FLL AFC; apply approximate AFC, and starting at closer frequency helps fast/accurate calibration.

DKL_PLL_LF

DKL_PLL_LF	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168B04h-168B07h
Name:	DKL_PLL_LF_NULL
ShortName:	DKL_PLL_LF_NULL
Reset:	global

This register is not reset by the device 2 FLR.

DWord	Bit	Description
0	31	**i_afc_divratio** 0: DCO/4 (prediv: 2, Mdithdiv: 2) 1: DCO/8 (prediv: 4 Mdithdiv: 2)
	30:29	**i_plllock_sel_1_0** Default Value: 00b select between lockdetect-based plllock or counter based plllock 11: Sticky lock 10: Counter-based 01: Lock Detection + Counter 00: Lock Detection.
	28:24	**i_bwphase_4_0** Default Value: 00h Phase amplitude for bandwidth measurement
	23:21	**i_ft_mode_sel_2_0** Default Value: 010b ftmodesel[2:0] : 9b vs 10b finetune selection, lsb0 tuning 3'b000 : 10b Nom -> Dither = LSB0, LSB0 = LSB0 3'b001 : 10b+ -> Dither = LSB+, LSB0 = LSB0+ 3'b010 : 10b- -> Dither = LSB-, LSB0 = LSB0- 3'b011 : 10b DNL -> Dither = LSB01, LSB0 = LSB0+ 3'h1xx: 9b -> Dither = LSB1, LSB0 = N/A
	20:19	**i_bw_mode_1_0** Default Value: 00b 00: No measurement 01: BW measurement 10: BW calibration up to +1 direction 11: BW calibration up to +2 direction

DKL_PLL_LF		
	18:16	**i_bw_upperbound_2_0** Default Value: 000b Upper bound setting for BW 000: 0.1MHz 001: 1 MHz 010: 2 MHz 011: 5 MHz 100: 6 MHz 101: 8 MHz 110: 16 MHz]111: 25MHz
	15:13	**i_bw_lowerbound_2_0** Default Value: 000b Lower bound setting for BW 000: 0.1MHz 001: 1 MHz 010: 2 MHz 011: 5 MHz 100: 6 MHz 101: 8 MHz 110: 16 MHz 111: 25MHz
	12:9	**i_dcoamp_3_0** Default Value: 1001b Amplitude override value for DCO, 0000 is min amplitude, 1111 is max amplitude, applied when i_dcoampovrden_h is high. 4bit is left shifted to generate 6bit code (x4).
	8	**i_dcoampovrrden_h** Default Value: 1b DCO amplitude override enable: 0 DCO amplitude set internally (default) 1 DCO amplitude is set by i_dcoamp[3:0]
	7:5	**i_bbthresh2_2_0** Default Value: 0b threshold for second (inner) bang-bang (BB) range.
	4:2	**i_bbthresh1_2_0** Default Value: 0b threshold for first (inner) bang-bang (BB) range.
	1:0	**i_tdc_offset_1_0** Default Value: 0b TDC Offset during lock for integer mode.

DKL_PLL1_CNTR_XXXX_SETTINGS

DKL_PLL1_CNTR_XXXX_SETTINGS

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168B9Ch-168B9Fh
Name:	DKL_PLL1_CNTR_BIST_SETTINGS_NULL
ShortName:	DKL_PLL1_CNTR_BIST_SETTINGS_NULL
Reset:	global

This register is not reset by the device 2 FLR.

DWord	Bit	Description
0	31:28	**RESERVED205** Access: RO
	27:24	**RESERVED204** Access: RO
	23	**RESERVED203** Access: RO
	22:21	**I_DFX_DIV_CKLO_1_0** Access: R/W
	20	**I_M1_LONGLOOP_SEL** Access: R/W
	19:18	**I_DITHER_DIV_1_0** Default Value: 01b Access: R/W
	17	**I_PLLLC_REG_LONGLOOPCLK_SEL** Access: R/W
	16	**AI_PLLLC_REG_FBCLKEXT_SEL** Access: R/W
	15	**RESERVED197** Access: RO
	14:10	**RESERVED196** Access: RO
	9:8	**I_IREFGEN_SETTLING_TIME_RO_STANDBY_1_0** Access: R/W

DKL_PLL1_CNTR_XXXX_SETTINGS		
	7:0	**I_IREFGEN_SETTLING_TIME_CNTR_7_0** Default Value: 0x30 Access: R/W

DKL_PMD_FORCE_PCS_IF_TX1

DKL_PMD_FORCE_PCS_IF_TX1	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168BA4h-168BA7h
Name:	DKL_PMD_FORCE_PCS_IF_TX1_NULL
ShortName:	DKL_PMD_FORCE_PCS_IF_TX1_NULL
Reset:	global

This register is not reset by the device 2 FLR.

DWord	Bit	Description
0	31	**reserved1**
	30	**ecsr_force_val_pmd_pwdn_n_rx** Access: R/W
	29	**ecsr_force_en_pmd_pwdn_n_rx** Access: R/W
	28	**ecsr_force_val_pmd_pwdn_n_tx1** Access: R/W
	27	**ecsr_force_en_pmd_pwdn_n_tx1** Access: R/W
	26	**ecsr_eidle_en_by_tx_pmd_pwdn_n_tx1** Default Value: 1b Access: R/W
	25:22	**force_req_valid_pmd_rx_pwr_mode** Default Value: 0001b Access: R/W
	21:18	**force_val_pmd_rx_pwr_mode** Access: R/W
	17	**force_en_pmd_rx_pwr_mode** Access: R/W
	16:13	**force_req_valid_pmd_tx_pwr_mode_tx1** Default Value: 0001b Access: R/W
	12:9	**force_val_pmd_tx_pwr_mode_tx1** Access: R/W

DKL_PMD_FORCE_PCS_IF_TX1		
	8	**force_en_pmd_tx_pwr_mode_tx1** Access: R/W
	7	**ecsr_force_val_pmd_hard_rst_b_tx1** Access: R/W
	6	**ecsr_force_en_pmd_hard_rst_b_tx1** Access: R/W
	5	**ecsr_force_val_pmd_hard_rst_b_rx** Access: R/W
	4	**ecsr_force_en_pmd_hard_rst_b_rx** Access: R/W
	3	**force_val_pmd_txeidle_tx1** Access: R/W
	2	**force_en_pmd_txeidle_tx1** Access: R/W
	1	**force_val_pmd_pwdn_n_tx1** Access: R/W
	0	**force_en_pmd_pwdn_n_tx1** Access: R/W

DKL_REFCLKIN_CTL

DKL_REFCLKIN_CTL - DKL_REFCLKIN_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168B2Ch-168B2Fh
Name:	DKL_REFCLKIN_CTL_NULL
ShortName:	DKL_REFCLKIN_CTL_NULL
Reset:	global

This register is not reset by device 2 FLR.

DWord	Bit	Description
0	31:12	**Reserved** Access: RO Format: MBZ
	11	**od_refclkin2_refclkinjmux** mux select of external injection refclk inputs: 0 to select amonrefclkinj 1 to select rcomprefclkinj
	10:8	**od_refclkin2_refclkmux** mux select for refclk output 3'b000: external injection refclk 3'b001: xtalinrefclk 3'b010:mgrefclkin 3'b011: socrefclk1 3'b100:socrefclk2 3'b101:socrefclk3 3'b110:socrefclk4 3'b111:socrefclk5
	7:4	**Reserved** Access: RO Format: MBZ
	3	**od_refclkin1_refclkinjmux** mux select of external injection refclk inputs: 0 to select amonrefclkinj 1 to select rcomprefclkinj
	2:0	**od_refclkin1_refclkmux** mux select for refclk output. 3'b000: external injection refclk3'b001: xtalinrefclk 3'b010: mgrefclkin 3'b011: socrefclk13'b100: socrefclk2 3'b101: socrefclk33'b110: socrefclk4 3'b111: socrefclk5

DKL_SSC

DKL_SSC	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168B38h-168B3Bh
Name:	DKL_SSC_NULL
ShortName:	DKL_SSC_NULL
Reset:	global

This register is not reset by the device 2 FLR.

DWord	Bit	Description
0	31:29	**i_iref_ndivratio_2_0** Default Value: 10b SSC injection Adaptive Gain Change Enable
	28:26	**ssc_stepnum_offset_2_0** Default Value: 000b SSC spread step is calculated with two registers, i_sscstepnum and i_sscstepnum_offset. SSC steps = 2^(i_sscstepnum + 1) + 4*i_sscstepnum_offset.
	25	**i_sscinj_adapt_en_h** SSC injection Adaptive Gain Change Enable. Value: 0b – Disable Value: 1b – Enable
	24	**i_sscinj_en_h** SSC inject enable. Value: 0b – Disable Value: 1b – Enable
	23:16	**i_sscsteplength_7_0** Default Value: 13h Number of ref clock cycles in one SSC step.
	15:14	**i_sscfll_update_sel_1_0** Default Value: 0b Select frequency update rate for FLL SSC.

DKL_SSC

	13:11	**i_sscstepnum_2_0** Default Value: 100b Number of SSC steps (=2^sscstepnum).
	10	**i_ssc_openloop_en_h** Default Value: 0b Open loop SSC enable.
	9	**i_sscen_h** Default Value: 1b Dynamic control of SSC modulator.
	8	**i_sscfllen_h** Default Value: 0b Frequency adjustment for FLL SSC.
	7:6	**i_bias_gb_sel_1_0** Default Value: 11b Select guardband after bias calibration.
	5:0	**i_init_dcoamp_5_0** Default Value: 3Fh initial DCOAMP value.

DKL_TDC_COLDST_BIAS

DKL_TDC_COLDST_BIAS	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168B44h-168B47h
Name:	DKL_TDC_COLDST_BIAS_NULL
ShortName:	DKL_TDC_COLDST_BIAS_NULL
Reset:	global

This register is not reset by the device 2 FLR.

DWord	Bit	Description
0	31:29	**i_cloadctrlext_4_2** Default Value: 00h cload control override for H Vernier line. This is applied when i_tdccalexten_h = 1.
	28:24	**i_tribufctrlext_4_0** Default Value: 00h Tribufctrl control override. This is applied when i_tdccalexten_h = 1.
	23:16	**i_dcocoarse_7_0** Default Value: 00h DCO coarse tune frequency value, when dcocoarse_ovrd_h is '1', this input is used to override the value calculated from the Automatic Frequency Calibration (AFC) block.
	15:8	**i_sscstepsize_7_0** Default Value: 0Fh Fractional value for one SSC frequency step.
	7:0	**i_feedfwdgain_7_0** Default Value: 23h Feed forwad gain for fractional mode/SSC mode PLL [br]This setting is needed whenever PLL is configured in fractional mode or configured for SSC clock generation.

DKL_TX_DPCNTL0

DKL_TX_DPCNTL0 - DKL_TX_DPCNTL0	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168B14h-168B17h
Name:	DKL_TX_DPCTRL0_TX1LN0_NULL
ShortName:	DKL_TX_DPCTRL0_TX1LN0_NULL
Reset:	global

This register is not reset by device 2 FLR.

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Trainingen_tx1** Enable set for programming in compliance mode.
	29	**Pipe_select_tx1** Override this bit to 0 to take the BIOS programmed Values and not pipe
	28	**Slow_trim_enable_tx1** Default Value: 1b Enable or Disable the slow trim.
	27:23	**shunt_cm_tx1** Back mode select for the Shunt CM
	22:18	**shunt_cp_tx1** Preshoot Co-efficients
	17:13	**Preshoot Control l0** Pre-shoot coefficients
	12:8	**de_emphasis_control_l0_tx1** De-emphasis co-efficient
	7:3	**Cursor_Control_tx1** Backup mode for shunt on C0
	2:0	**Vswing Control_tx1** Default Value: 111b 3'b000 = 1V + 0dB (Full Swing) 3'b001 = 800 mV + 0dB 3'b100 = 600 mV + 0dB 3'b111 = 400 mV + 0dB

DKL_TX_DPCNTL1

DKL_TX_DPCNTL1 - DKL_TX_DPCNTL1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168B1Ch-168B1Fh
Name:	DKL_TX_DPCNTL1_TX2LN0_NULL
ShortName:	DKL_TX_DPCNTL1_TX2LN0_NULL
Reset:	global

This register is not reset by device 2 FLR.

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Trainingen_tx2** Enable set for programming in compliance mode.
	29	**Pipe_select_tx2** Override this bit to 0 to take the BIOS programmed Values and not pipe
	28	**Slow_trim_enable_tx2** Default Value: 1b Enable or Disable the slow trim.
	27:23	**shunt_cm_tx2** Back mode select for the Shunt CM
	22:18	**shunt_cp_tx2** Preshoot Co-efficients
	17:13	**Preshoot Control l0** Pre-shoot coefficients
	12:8	**de_emphasis_control_l0_tx2** De-emphasis co-efficient
	7:3	**Cursor_Control_tx2** Backup mode for shunt on C0
	2:0	**Vswing Control_tx2** Default Value: 111b 3'b000 = 1V + 0dB (Full Swing) 3'b001 = 800 mV + 0dB 3'b100 = 600 mV + 0dB 3'b111 = 400 mV + 0dB

DKL_TX_DPCNTL2

DKL_TX_DPCNTL2 - DKL_TX_DPCNTL2

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168B24h-168B27h
Name:	DKL_TX_DPCNTL2_TX2LN0_NULL
ShortName:	DKL_TX_DPCNTL2_TX2LN0_NULL
Reset:	global

This register is not reset by device 2 FLR.

DWord	Bit	Description
0	31:23	**Reserved** Access: RO Format: MBZ
	22:10	**reserved_dp3**
	9	**dp_fifo_depth_tx1**
	8	**dp_fifo_depth_tx2**
	7	**dp_2ui_4ui_mode_en** Value: 0 — Name: 4UI Value: 1 — Name: 2UI
	6:5	**loadgenselect_tx2** loadgen select for datapath2ui - Look at the sheet for CFG per Tx
	4:3	**loadgenselect_tx1** loadgen select for datapath2ui - Look at the sheet for CFG per Tx
	2	**dp20bitmode** 20 bit mode support. This will need to be set to 1 if Pipe width does not reflect the 20bit mode.
	1	**rate8boverrideen** 8bit override enable. The pisorate_8b signal is normally decoded from PHYMODE and RATE. When 1, the pisorate_8b signal will take with pisorate8bit_ovrd value.
	0	**rate8boverride** value used for pisorate_8b when pisorate8bit_overden = 1.

DKL_TX_DW17

DKL_TX_DW17 - DKL_TX_DW17	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168B48h-168B4Bh
Name:	DKL_TX_DW17_NULL
ShortName:	DKL_TX_DW17_NULL
Reset:	global

This register is not reset by device 2 FLR.

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29:24	**cri_txdeemph_override11_6** cursor deemph override value when enable bit is set, or cri_txdeemph_override[11:6]
	23	**Reserved** Access: RO Format: MBZ
	22	**cri_txdeemph_override_en** txdeemph override enable bit
	21:16	**cri_txdeemph_override5_0** precursor deemph override value when enable bit is set, or cri_txdeemph_override[5:0]
	15:0	**Reserved** Access: RO Format: MBZ

DKL_TX_DW18

DKL_TX_DW18 - DKL_TX_DW18

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168B4Ch-168B4Fh
Name:	DKL_TX_DW18_NULL
ShortName:	DKL_TX_DW18_NULL
Reset:	global

This register is not reset by device 2 FLR.

DWord	Bit	Description
0	31:6	**Reserved** Access: RO Format: MBZ
	5:0	**cri_txdeemph_override17_12** cursor deemph override value when enable bit is set, or cri_txdeemph_override[17:12]

DKL_TX_FW_CALIB

DKL_TX_FW_CALIB		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	168B58h-168B5Bh	
Name:	DKL_TX_FW_CALIB_NULL	
ShortName:	DKL_TX_FW_CALIB_NULL	
Reset:	global	
This register is not reset by the device 2 FLR.		

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**cfg_disable_wait_init_periodic** This bit has to be set everytime TypeC connection switches from USB to DP ALT mode. Recommendation is to set this bit after PLL programming is completed.
	6	**tx1_dcc_cmp**
	5	**cfg_fw_oneshotcal_req_ctrl_val**
	4	**cfg_tx_fw_oneshotcal_req_ctrl_en**
	3	**cfg_tx_pwr_cal_en_ovrd_val**
	2	**cfg_tx_pwr_cal_en_ovrd_en**
	1	**cfg_rate_cal_en_ovrd_val**
	0	**odkl_tx_rate_ana_cal_en**

DKL_TX_PMD_LANE_SUS_LN0

DKL_TX_PMD_LANE_SUS_LN0	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168B88h-168B8Bh
Name:	DKL_TX_PMD_LANE_SUS_LN0_NULL
ShortName:	DKL_TX_PMD_LANE_SUS_LN0_NULL
Reset:	global

This register is not reset by the device 2 FLR.

DWord	Bit	Description
0	31:0	**dummy** Flush all register bits to '0' at the time Display takes control of this PHY lane.

DKL_TX_PMD_LANE_SUS_LN1

DKL_TX_PMD_LANE_SUS_LN1	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168B8Ch-168B8Fh
Name:	DKL_TX_PMD_LANE_SUS_LN1_NULL
ShortName:	DKL_TX_PMD_LANE_SUS_LN1_NULL
Reset:	global

This register is not reset by the device 2 FLR.

DWord	Bit	Description
0	31:0	**dummy** Flush all register bits to '0' at the time Display takes control of this PHY lane.

DKL_XXX_TDC_CRO

DKL_XXX_TDC_CRO	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	168BA0h-168BA3h
Name:	DKL_XXX_TDC_CRO_NULL
ShortName:	DKL_XXX_TDC_CRO_NULL
Reset:	global

This register is not reset by the device 2 FLR.

DWord	Bit	Description
0	31	**I_PLLLC_REG_FULLCALRESETB** Default Value: 1 Access: R/W
	30	**I_DFX_POSTDIV_DISABLE** Access: R/W
	29	**I_DFX_MDFX_ENABLE** Access: R/W
	28	**I_DFX_MDITH_DISABLE** Access: R/W
	27	**I_DFX_TDC_DISABLE** Access: R/W
	26:25	**I_PLLLC_IREF_CLOCK_SEL_1_0** Access: R/W
	24	**I_PLLLC_IREF_CLOCK_OVRD** Access: R/W
	23	**I_PLLLC_REG_REFCLK_ACK_MODE_CTRL** Access: R/W
	22	**I_PLLLC_REG_REFCLK_ACK** Access: R/W
	21	**I_PLLLC_REG_ACTIVE_STANDBY_MODE_CTRL** Access: R/W
	20	**I_PLLLC_REG_ACTIVE_STANDBY** Access: R/W

DKL_XXX_TDC_CRO			
	19	**I_PLLLC_REG_RESETB_ANA_MODE_CTRL**	
		Access:	R/W
	18	**I_PLLLC_RO_MODE_CTRL**	
		Access:	R/W
	17	**I_PLLLC_RO_REGDISABLE**	
		Access:	R/W
	16	**I_PLLLC_RO_REGEN_H**	
		Access:	R/W
	15:14	**I_PLLLC_EN_MODE_CTRL_1_0**	
		Default Value:	10b
		Access:	R/W
	13	**I_PLLLC_REGEN_H**	
		Access:	R/W
	12	**I_IREF_REFCLK_INV_EN**	
		Access:	R/W
	11	**Reserved**	
	10	**Reserved**	
	9	**I_VGSBUFEN**	
		Access:	R/W
	8	**I_IREFINT_EN**	
		Default Value:	1b
		Access:	R/W
	7	**I_IREFBIAS_STARTUP_PULSE_BYPASS**	
		Access:	R/W
	6:5	**I_IREFBIAS_STARTUP_PULSE_WIDTH_1_0**	
		Default Value:	01b
		Access:	R/W
	4	**I_COLDSTART**	
		Default Value:	1b
		Access:	R/W
	3	**I_BBINLOCK_H**	
		Access:	R/W
	2:1	**I_SWCAP_IREFGEN_CLKMODE_1_0**	
		Access:	R/W
	0	**I_TDCDC_EN_H**	
		Access:	R/W

DKLP_ACU_ACU_DWORD4

DKLP_ACU_ACU_DWORD4		
Register Space: MMIO: 0/2/0 Size (in bits): 32		
RESERVED		
DWord	**Bit**	**Description**
0	31:29	**cfg_reserved513** Default Value: 000b cfg_reserved513_defaultreset Access: RO reserved
	28:24	**cfg_clkhub2_monana1selectb_4_0** Default Value: 1Fh cfg_clkhub2_monana1selectb_4_0_defaultreset Access: R/W DFX control of CLOCK_DIST
	23:21	**cfg_reserved512** Default Value: 000b cfg_reserved512_defaultreset Access: RO reserved
	20:16	**cfg_clkhub2_monana1select_4_0** Default Value: 00h cfg_clkhub2_monana1select_4_0_defaultreset Access: R/W DFX control of CLOCK_DIST
	15:13	**cfg_reserved511** Default Value: 000b cfg_reserved511_defaultreset Access: RO reserved
	12:8	**cfg_clkhub2_monana0selectb_4_0** Default Value: 1Fh cfg_clkhub2_monana0selectb_4_0_defaultreset Access: R/W DFX control of CLOCK_DIST
	7	**cfg_reserved510** Default Value: 0b cfg_reserved510_defaultreset Access: RO reserved

DKLP_ACU_ACU_DWORD4		
	6	**cfg_cfg_reset_phymode_change** Default Value: 0b cfg_cfg_reset_phymode_change_defaultreset Access: R/W cfg bit to reset phymode change feature bit in acu suswell pwrctrl
	5	**cfg_o_cri_cfg_phymode_switch** Default Value: 0b cfg_o_cri_cfg_phymode_switch_defaultreset Access: R/W phymode_switch for pwr_ctrl
	4:0	**cfg_clkhub2_monana0select_4_0** Default Value: 00h cfg_clkhub2_monana0select_4_0_defaultreset Access: R/W DFX control of CLOCK_DIST

DKLP_ACU_ACU_DWORD8

DKLP_ACU_ACU_DWORD8

Register Space: MMIO: 0/2/0
Size (in bits): 32

DP MODE

DWord	Bit	Description
0	31:24	**cfg_ldo_powerup_timer** Default Value: 02h cfg_ldo_powerup_timer_defaultreset Access: R/W ip74pppxdkltypecfamilyew_Timer to decide when LDO would be up, in case pwr_gate is used instead of LDO, it's configured to wait 20us, i.e 9'd500 by default using 25 MHz susclk
	23	**cfg_cfg_tr2_pwrgate_timer_bypass** Default Value: 0b cfg_cfg_tr2_pwrgate_timer_bypass_defaultreset Access: R/W When this bit is 1, bypass the pwrgate ip74pppxdkltypecfamilyew_timer, which is configurable with cfg_ldo_powerup_timer
	22	**cfg_cfg_tr_pwrgate_timer_bypass** Default Value: 0b cfg_cfg_tr_pwrgate_timer_bypass_defaultreset Access: R/W When this bit is 1, bypass the pwrgate ip74pppxdkltypecfamilyew_timer, which is configurable with cfg_ldo_powerup_timer
	21	**cfg_cfg_cl_pwrgate_timer_bypass** Default Value: 0b cfg_cfg_cl_pwrgate_timer_bypass_defaultreset Access: R/W When this bit is 1, bypass the pwrgate ip74pppxdkltypecfamilyew_timer, which is configurable with cfg_ldo_powerup_timer
	20	**cfg_cfg_dig_pwrgate_timer_bypass** Default Value: 0b cfg_cfg_dig_pwrgate_timer_bypass_defaultreset Access: R/W When this bit is 1, bypass the pwrgate ip74pppxdkltypecfamilyew_timer, which is configurable with cfg_ldo_powerup_timer
	19	**cfg_cfg_crireg_cold_boot_done** Default Value: 0b cfg_cfg_crireg_cold_boot_done_defaultreset Access: R/W set cold boot done

DKLP_ACU_ACU_DWORD8

18	**cfg_cfg_vr_pulldwn2gnd_tr2** Default Value: 0b cfg_cfg_vr_pulldwn2gnd_tr2_defaultreset Access: R/W if there is a chance of high voltage from hotplug through diodes. Needed for tr/tr2 LDO
17	**cfg_cfg_vr_pulldwn2gnd_tr** Default Value: 0b cfg_cfg_vr_pulldwn2gnd_tr_defaultreset Access: R/W if there is a chance of high voltage from hotplug through diodes. Needed for tr/tr2 LDO
16	**cfg_cfg_ldo_powerup_timer_8** Default Value: 0b cfg_cfg_ldo_powerup_timer_8_defaultreset Access: R/W ip74pppxdkltypecfamilyew_Timer to decide when LDO would be up, in case pwr_gate is used instead of LDO, it's configured to wait 20us, i.e 9'd500 by default using 25 MHz susclk
15	**cfg_cfg_corepwr_ack_with_pcs_pwrreq** Default Value: 0b cfg_cfg_corepwr_ack_with_pcs_pwrreq_defaultreset Access: R/W if this bit is 1, send corepwr_ack to pcs only when pcs_pwrreq is 1
14	**cfg_cfg_cri_digpwr_req** Default Value: 0b cfg_cfg_cri_digpwr_req_defaultreset Access: R/W Keep dig up during burst CRI transactions, this also bypasses handshake with the susclk domain, greatly reducing the latency.
13	**cfg_cfg_laneclkreq_force** Default Value: 1b cfg_cfg_laneclkreq_force_defaultreset Access: R/W When cfg_laneclkreq_gating_ctrl is low, this value will be used for the lane sus_clk request
12	**cfg_cfg_laneclkreq_gating_ctrl** Default Value: 0b cfg_cfg_laneclkreq_gating_ctrl_defaultreset Access: R/W 1'b1 - lanesusclk req gating enabled
11	**cfg_cfg_susclk_gating_ctrl** Default Value: 0b cfg_cfg_susclk_gating_ctrl_defaultreset Access: R/W 1'b1 - susclk gating enabled

DKLP_ACU_ACU_DWORD8

10	**cfg_cfg_rawpwr_req_override** Default Value: 0b cfg_cfg_rawpwr_req_override_defaultreset Access: R/W Keep rawpwr on when this bit is 1
9	**cfg_cfg_digpwr_req_override** Default Value: 0b cfg_cfg_digpwr_req_override_defaultreset Access: R/W Keep digpwr on when this bit is 1
8	**cfg_cfg_rawpwr_gating_ctrl** Default Value: 0b cfg_cfg_rawpwr_gating_ctrl_defaultreset Access: R/W Power gating enable reg for raw power
7	**cfg_cfg_dp_x2_mode** Default Value: 0b cfg_cfg_dp_x2_mode_defaultreset Access: R/W {cfg_dp_x2_mode,cfg_dp_x1_mode}: 00 -) Only main tx on {cfg_dp_x2_mode,cfg_dp_x1_mode}: 01 -) Only secondary tx on {cfg_dp_x2_mode,cfg_dp_x1_mode}: 1x -) Both tx on
6	**cfg_cfg_dp_x1_mode** Default Value: 0b cfg_cfg_dp_x1_mode_defaultreset Access: R/W {cfg_dp_x2_mode,cfg_dp_x1_mode}: 00 -) Only main tx on {cfg_dp_x2_mode,cfg_dp_x1_mode}: 01 -) Only secondary tx on {cfg_dp_x2_mode,cfg_dp_x1_mode}: 1x -) Both tx on
5	**cfg_cfg_tr2pwr_gating_ctrl** Default Value: 0b cfg_cfg_tr2pwr_gating_ctrl_defaultreset Access: R/W Power gating enable reg for tr2
4	**cfg_cfg_trpwr_gating_ctrl** Default Value: 0b cfg_cfg_trpwr_gating_ctrl_defaultreset Access: R/W Power gating enable reg for tr
3	**cfg_cfg_clnpwr_gating_ctrl** Default Value: 0b cfg_cfg_clnpwr_gating_ctrl_defaultreset Access: R/W Power gating enable reg for cln

DKLP_ACU_ACU_DWORD8		
	2	**cfg_cfg_digpwr_gating_ctrl** Default Value: 0b cfg_cfg_digpwr_gating_ctrl_defaultreset Access: R/W Power gating enable reg for dig
	1	**cfg_cfg_gaonpwr_gating_ctrl** Default Value: 0b cfg_cfg_gaonpwr_gating_ctrl_defaultreset Access: R/W Power gating enable reg for gaon
	0	**cfg_cfg_suspwr_gating_ctrl** Default Value: 0b cfg_cfg_suspwr_gating_ctrl_defaultreset Access: R/W Power gating enable reg for sus

DKLP_ACU_ICL_INDEXED_ACU_INDEXED_DWORD46

DKLP_ACU_ICL_INDEXED_ACU_INDEXED_DWORD46		
Register Space: MMIO: 0/2/0 Size (in bits): 32		
Reserved		
DWord	**Bit**	**Description**
0	31:16	**Reserved** Access: RO Format: MBZ
	15:8	**cfg_reserved580** Default Value: 00h cfg_reserved580_defaultreset Access: RO reserved
	7:0	**cfg_reserved579** Default Value: 00h cfg_reserved579_defaultreset Access: RO reserved

DKLP_CMN_ANA_CMN_ANA_DWORD0

DKLP_CMN_ANA_CMN_ANA_DWORD0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
CLKTOP1_HSCLKCTL		
DWord	**Bit**	**Description**
0	31:27	**cfg_reserved504** Default Value: 00h cfg_reserved504_defaultreset Access: RO Power well:DIG
	26:25	**cfg_od_clktop1_clkobs_muxsel_1_0** Default Value: 00b cfg_od_clktop1_clkobs_muxsel_1_0_defaultreset Access: R/W mux select for oc_dfx_ck_clk2obs[0] digobs output 2'b00: hsdiv output clock 2'b01: iclk_bypass input from the other clktop 2'b10: dsdiv output clock 2'b11: non-dvided pll clock (test mode)
	24	**cfg_od_clktop1_clk2obs_en_h** Default Value: 0b cfg_od_clktop1_clk2obs_en_h_defaultreset Access: R/W enable of oc_dfx_ck_clk2obs[0] digobs output
	23:22	**cfg_reserved501** Default Value: 00b cfg_reserved501_defaultreset Access: RO Power well:DIG
	21:20	**cfg_od_clktop1_clktop1_vhfclk_testen_h_1_0** Default Value: 00b cfg_od_clktop1_clktop1_vhfclk_testen_h_1_0_defaultreset Access: R/W test mode enable of pll clock bit[0]: enable of pll clock to non-dedicated tlinedrv bit[1]: enable of pll clock to dfx ouput
	19	**cfg_reserved502** Default Value: 0b cfg_reserved502_defaultreset Access: RO Power well:DIG
	18	**cfg_od_clktop1_outclk_bypassen_h** Default Value: 0b cfg_od_clktop1_outclk_bypassen_h_defaultreset Access: R/W enable of bypass clock output to the other clktop

DKLP_CMN_ANA_CMN_ANA_DWORD0

	17	**cfg_reserved503** Default Value: 0b cfg_reserved503_defaultreset Access: RO Power well:DIG
	16	**cfg_od_clktop1_coreclk_inputsel** Default Value: 0b cfg_od_clktop1_coreclk_inputsel_defaultreset Access: R/W mux select for input clock to coreclk divhub 1'b0: select hsdiv output;1'b1: select dsdiv output
	15:14	**cfg_od_clktop1_tlinedrv_clksel_1_0** Default Value: 00b cfg_od_clktop1_tlinedrv_clksel_1_0_defaultreset Access: R/W mux select for non-dedicated tlinedrv clocks 2'b00: hsclkdiv output 2'b01: dsdiv output clktop 2'b10: iqdiv2 clock 2'b11: iqdiv2 clock
	13:12	**cfg_od_clktop1_hsdiv_divratio_1_0** Default Value: 00b cfg_od_clktop1_hsdiv_divratio_1_0_defaultreset Access: R/W divider ratio for high speed divider. 2'b00: div/2 2'b01: div/3 2'b10: div/5 2'b11: div/7
	11:8	**cfg_od_clktop1_dsdiv_divratio_3_0** Default Value: Ah cfg_od_clktop1_dsdiv_divratio_3_0_defaultreset Access: R/W divider ratio settings for programmable divider: 4'b0000: no division 4'b0001: no division 4'b0010 - 4'b1010: div/2 - div/10 4'b1011 - 4'b1111: not used (div/10)
	7	**cfg_od_clktop1_tlinedrv_overrideen** Default Value: 0b cfg_od_clktop1_tlinedrv_overrideen_defaultreset Access: R/W override enable for follwing 4 tlinedrv enables
	6	**cfg_od_clktop1_tlinedrv_enleft_ded_h_ovrd** Default Value: 1b cfg_od_clktop1_tlinedrv_enleft_ded_h_ovrd_defaultreset Access: R/W enable of left side dedicated tlinedrv to output full-rate clock to left lanes

DKLP_CMN_ANA_CMN_ANA_DWORD0

	5	**cfg_od_clktop1_tlinedrv_enright_ded_h_ovrd** Default Value: 1b cfg_od_clktop1_tlinedrv_enright_ded_h_ovrd_defaultreset Access: R/W enable of right side dedicated tlinedrv to output full-rate clock to right lanes
	4	**cfg_od_clktop1_tlinedrv_enleft_h_ovrd** Default Value: 0b cfg_od_clktop1_tlinedrv_enleft_h_ovrd_defaultreset Access: R/W enable of left side tlinedrv to output divided clock to left lanes
	3	**cfg_od_clktop1_tlinedrv_enright_h_ovrd** Default Value: 0b cfg_od_clktop1_tlinedrv_enright_h_ovrd_defaultreset Access: R/W enable of right side tlinedrv to output divided clock to right lanes
	2	**cfg_od_clktop1_dsdiv_en_h** Default Value: 1b cfg_od_clktop1_dsdiv_en_h_defaultreset Access: R/W enable of dsdiv clock divider
	1	**cfg_reserved500** Default Value: 0b cfg_reserved500_defaultreset Access: RO Power well:DIG
	0	**cfg_od_clktop1_hsdiv_en_h** Default Value: 1b cfg_od_clktop1_hsdiv_en_h_defaultreset Access: R/W enable of high speed clock divider

DKLP_CMN_ANA_CMN_ANA_DWORD1

DKLP_CMN_ANA_CMN_ANA_DWORD1

Register Space: MMIO: 0/2/0
Size (in bits): 32

CLKTOP1_CORECLKCTL1

DWord	Bit	Description
0	31:30	**cfg_reserved508** Default Value: 00b cfg_reserved508_defaultreset Access: RO Power well:DIG
	29	**cfg_od_clktop1_coreclkd_bypass** Default Value: 0b cfg_od_clktop1_coreclkd_bypass_defaultreset Access: R/W bypass enable of coreclkd to take input refclk
	28	**cfg_od_clktop1_coreclkd_divretimeren_h** Default Value: 1b cfg_od_clktop1_coreclkd_divretimeren_h_defaultreset Access: R/W retimer enable: 0 for odd division ratio, 1 for even division ratio
	27	**cfg_reserved509** Default Value: 0b cfg_reserved509_defaultreset Access: RO Power well:DIG
	26	**cfg_od_clktop1_coreclkc_bypass** Default Value: 0b cfg_od_clktop1_coreclkc_bypass_defaultreset Access: R/W bypass enable of coreclkc to take input refclk
	25	**cfg_od_clktop1_coreclkc_divretimeren_h** Default Value: 1b cfg_od_clktop1_coreclkc_divretimeren_h_defaultreset Access: R/W retimer enable: 0 for odd division ratio, 1 for even division ratio
	24	**cfg_reserved510** Default Value: 0b cfg_reserved510_defaultreset Access: RO Power well:DIG
	23:16	**cfg_od_clktop1_coreclkb_divratio_7_0** Default Value: 08h cfg_od_clktop1_coreclkb_divratio_7_0_defaultreset Access: R/W divider ratio for coreclkb divider 8'h00: div/256 8'h01: div/257 8'h02 - 8'hff: div/2 - div/255

DKLP_CMN_ANA_CMN_ANA_DWORD1

		8'h10: USB3.1 gen2 312.5MHz
	15:8	**cfg_od_clktop1_coreclka_divratio_7_0** Default Value: 14h cfg_od_clktop1_coreclka_divratio_7_0_defaultreset Access: R/W divider ratio for coreclka divider 8'h00: div/256 8'h01: div/257 8'h02 - 8'hff: div/2 - div/255 8'h28: USB3.1 gen1 125MHz
	7	**cfg_on_pll12coreclkd_select** Default Value: 0b cfg_on_pll12coreclkd_select_defaultreset Access: R/W this bit is used to select from pll1_coreclkd and pll2_coreclkd for pll12coreclkd.
	6	**cfg_on_pll12coreclka_select** Default Value: 0b cfg_on_pll12coreclka_select_defaultreset Access: R/W this bit is used to select from pll1_coreclkc and pll2_coreclka for pll12coreclka.
	5	**cfg_od_clktop1_coreclkb_bypass** Default Value: 0b cfg_od_clktop1_coreclkb_bypass_defaultreset Access: R/W bypass enable of coreclkb to take input refclk
	4	**cfg_od_clktop1_coreclkb_divretimeren_h** Default Value: 1b cfg_od_clktop1_coreclkb_divretimeren_h_defaultreset Access: R/W retimer enable: 0 for odd division ratio, 1 for even division ratio
	3	**cfg_reserved506** Default Value: 0b cfg_reserved506_defaultreset Access: RO Power well:DIG
	2	**cfg_od_clktop1_coreclka_bypass** Default Value: 0b cfg_od_clktop1_coreclka_bypass_defaultreset Access: R/W bypass enable of coreclka to take input refclk
	1	**cfg_od_clktop1_coreclka_divretimeren_h** Default Value: 1b cfg_od_clktop1_coreclka_divretimeren_h_defaultreset Access: R/W retimer enable: 0 for odd division ratio, 1 for even division ratio
	0	**cfg_reserved507** Default Value: 0b cfg_reserved507_defaultreset Access: RO Power well:DIG

DKLP_CMN_ANA_CMN_ANA_DWORD5

DKLP_CMN_ANA_CMN_ANA_DWORD5

Register Space: MMIO: 0/2/0

Size (in bits): 32

CLKTOP2_HSCLKCTL

DWord	Bit	Description
0	31:27	**cfg_reserved524** Default Value: 00h cfg_reserved524_defaultreset Access: RO Power well:DIG
	26:25	**cfg_od_clktop2_clkobs_inputsel_1_0** Default Value: 00b cfg_od_clktop2_clkobs_inputsel_1_0_defaultreset Access: R/W mux select for oc_dfx_ck_clk2obs[0] digobs output 2'b00: hsdiv output clock 2'b01: iclk_bypass input from the other clktop 2'b10: dsdiv output clock 2'b11: non-divided pll clock
	24	**cfg_od_clktop2_clk2obs_en_h** Default Value: 0b cfg_od_clktop2_clk2obs_en_h_defaultreset Access: R/W enable of oc_dfx_ck_clk2obs[0] digobs output
	23:22	**cfg_reserved521** Default Value: 00b cfg_reserved521_defaultreset Access: RO Power well:DIG
	21:20	**Reserved**
	19	**cfg_reserved522** Default Value: 0b cfg_reserved522_defaultreset Access: RO Power well:DIG
	18	**cfg_od_clktop2_outclk_bypassen_h** Default Value: 0b cfg_od_clktop2_outclk_bypassen_h_defaultreset Access: R/W enable of div/2 bypass clock output to the other clktop
	17	**cfg_reserved523** Default Value: 0b cfg_reserved523_defaultreset Access: RO Power well:DIG

DKLP_CMN_ANA_CMN_ANA_DWORD5

16	**cfg_od_clktop2_coreclk_inputsel** Default Value: 0b cfg_od_clktop2_coreclk_inputsel_defaultreset Access: R/W mux select for input clock to coreclk divhub. 1'b0: select hsdiv output; 1'b1: select dsdiv output
15:14	**cfg_od_clktop2_tlinedrv_clksel_1_0** Default Value: 00b cfg_od_clktop2_tlinedrv_clksel_1_0_defaultreset Access: R/W mux select for non-dedicated tlinedrv clocks 2'b00: hsclkdiv output 2'b01: dsdiv output clktop 2'b10: iqdiv2 clock 2'b11: iqdiv2clock
13:12	**cfg_od_clktop2_hsdiv_divratio_1_0** Default Value: 00b cfg_od_clktop2_hsdiv_divratio_1_0_defaultreset Access: R/W divider ratio for high speed divider. 2'b00: div/2 HBR3 2'b01: div/3 HBR2/HBR 2'b10: div/5 RBR 2'b11: div/7
11:8	**cfg_od_clktop2_dsdiv_divratio_3_0** Default Value: 1h cfg_od_clktop2_dsdiv_divratio_3_0_defaultreset Access: R/W divider ratio settings for programmable divider: 4'b0000: no division 4'b0001: no division 4'b0010 - 4'b1010: div/2 - div/10 4'b1011 - 4'b1111: not used (div/10)
7	**cfg_od_clktop2_tlinedrv_overrideen** Default Value: 0b cfg_od_clktop2_tlinedrv_overrideen_defaultreset Access: R/W override enable for following 4 tlinedrv enables
6	**cfg_od_clktop2_tlinedrv_enleft_ded_h_ovrd** Default Value: 0b cfg_od_clktop2_tlinedrv_enleft_ded_h_ovrd_defaultreset Access: R/W enable of left side dedicated tlinedrv to output full-rate clock to left lanes
5	**cfg_od_clktop2_tlinedrv_enright_ded_h_ovrd** Default Value: 0b cfg_od_clktop2_tlinedrv_enright_ded_h_ovrd_defaultreset Access: R/W enable of right side dedicated tlinedrv to output full-rate clock to right lanes
4	**cfg_od_clktop2_tlinedrv_enleft_h_ovrd** Default Value: 1b cfg_od_clktop2_tlinedrv_enleft_h_ovrd_defaultreset Access: R/W enable of left side tlinedrv to output divided clock to left lanes
3	**cfg_od_clktop2_tlinedrv_enright_h_ovrd** Default Value: 1b cfg_od_clktop2_tlinedrv_enright_h_ovrd_defaultreset Access: R/W enable of right side tlinedrv to output divided clock to right lanes

DKLP_CMN_ANA_CMN_ANA_DWORD5

	2	**cfg_od_clktop2_dsdiv_en_h** Default Value: 1b cfg_od_clktop2_dsdiv_en_h_defaultreset Access: R/W enable of programmable divider to generate up to div/40 clock working with two iqdivs
	1	**cfg_reserved520** Default Value: 0b cfg_reserved520_defaultreset Access: RO Power well:DIG
	0	**cfg_od_clktop2_hsdiv_en_h** Default Value: 1b cfg_od_clktop2_hsdiv_en_h_defaultreset Access: R/W enable of high speed clock divider

DKLP_CMN_ANA_CMN_ANA_DWORD6

DKLP_CMN_ANA_CMN_ANA_DWORD6

Register Space: MMIO: 0/2/0

Size (in bits): 32

CLKTOP2_CORECLKCTL1

DWord	Bit	Description
0	31:30	**cfg_reserved528** Default Value: 00b cfg_reserved528_defaultreset Access: RO Power well:DIG
	29	**cfg_od_clktop2_coreclkd_bypass** Default Value: 0b cfg_od_clktop2_coreclkd_bypass_defaultreset Access: R/W bypass enable of coreclkd mux to take input as refclk instead of respective coreclk. When bypass_enable is high, refclk becomes the coreclk.
	28	**cfg_od_clktop2_coreclkd_divretimeren_h** Default Value: 1b cfg_od_clktop2_coreclkd_divretimeren_h_defaultreset Access: R/W used to enable or bypass the retimer path for coreclk. 1: enable the retimer path 0: bypass the retimer path (POR)
	27	**cfg_reserved529** Default Value: 0b cfg_reserved529_defaultreset Access: RO Power well:DIG
	26	**cfg_od_clktop2_coreclkc_bypass** Default Value: 0b cfg_od_clktop2_coreclkc_bypass_defaultreset Access: R/W bypass enable of coreclkc mux to take input as refclk instead of respective coreclk. When bypass_enable is high, refclk becomes the coreclk.
	25	**cfg_od_clktop2_coreclkc_divretimeren_h** Default Value: 0b cfg_od_clktop2_coreclkc_divretimeren_h_defaultreset Access: R/W used to enable or bypass the retimer path for coreclk. 1: enable the retimer path 0: bypass the retimer path (POR)

DKLP_CMN_ANA_CMN_ANA_DWORD6

24	**cfg_reserved530** Default Value: 0b cfg_reserved530_defaultreset Access: RO Power well:DIG
23:16	**cfg_od_clktop2_coreclkb_divratio_7_0** Default Value: 08h cfg_od_clktop2_coreclkb_divratio_7_0_defaultreset Access: R/W used to set the divider ratio for coreclkb divider . divider ratio is the decimal equivalent of hex number. 8'h00, 8'h01: div by 1; 8'h02: div by 2; 8'h03: div by3; and so on...
15:8	**cfg_od_clktop2_coreclka_divratio_7_0** Default Value: 05h cfg_od_clktop2_coreclka_divratio_7_0_defaultreset Access: R/W used to set the divider ratio for coreclka divider . divider ratio is the decimal equivalent of hex number. 8'h00, 8'h01: div by 1; 8'h02: div by 2; 8'h03: div by3; and so on...
7:6	**cfg_reserved525** Default Value: 00b cfg_reserved525_defaultreset Access: RO Power well:DIG
5	**cfg_od_clktop2_coreclkb_bypass** Access: R/W bypass enable of coreclkb mux to take input as refclk instead of respective coreclk. When bypass_enable is high, refclk becomes the coreclk.
4	**cfg_od_clktop2_coreclkb_divretimeren_h** Default Value: 0b bypass the retimer path Access: R/W used to enable or bypass the retimer path for coreclk. 1: enable the retimer path 0: bypass the retimer path (POR)

DKLP_CMN_ANA_CMN_ANA_DWORD6

	3	**cfg_reserved526** Default Value: 0b cfg_reserved526_defaultreset Access: RO Power well:DIG
	2	**cfg_od_clktop2_coreclka_bypass** Access: R/W bypass enable of coreclka mux to take input as refclk instead of respective coreclk. When bypass_enable is high, refclk becomes the coreclk.
	1	**cfg_od_clktop2_coreclka_divretimeren_h** Access: R/W used to enable or bypass the retimer path for coreclk. 1: enable the retimer path 0: bypass the retimer path (POR)

Value	Name	Description
0b	bypass the retimer path **[Default]**	This is POR.
1b	enable the retimer path	

	0	**cfg_reserved527** Default Value: 0b cfg_reserved527_defaultreset Access: RO Power well:DIG

DKLP_CMN_ANA_CMN_ANA_DWORD7

DKLP_CMN_ANA_CMN_ANA_DWORD7

Register Space: MMIO: 0/2/0

Size (in bits): 32

CLKTOP2_CORECLKCTL2

DWord	Bit	Description
0	31:24	**cfg_od_clktop2_coreclke_divratio_7_0** Default Value: 32h cfg_od_clktop2_coreclke_divratio_7_0_defaultreset Access: R/W divider ratio for coreclke divider 8'h00: div/256 8'h01: div/257 8'h02 - 8'hff: div/2 - div/255
	23:22	**cfg_reserved531** Default Value: 00b cfg_reserved531_defaultreset Access: RO Power well:DIG
	21	**cfg_od_clktop2_coreclkf_bypass** Default Value: 0b cfg_od_clktop2_coreclkf_bypass_defaultreset Access: R/W bypass enable of coreclkf to take input refclk
	20	**cfg_od_clktop2_coreclkf_divretimeren_h** Default Value: 1b cfg_od_clktop2_coreclkf_divretimeren_h_defaultreset Access: R/W retimer enable: 0 for odd division ratio, 1 for even division ratio
	19	**cfg_reserved532** Default Value: 0b cfg_reserved532_defaultreset Access: RO Power well:DIG
	18	**cfg_od_clktop2_coreclke_bypass** Default Value: 0b cfg_od_clktop2_coreclke_bypass_defaultreset Access: R/W bypass enable of coreclke to take input refclk

DKLP_CMN_ANA_CMN_ANA_DWORD7

	17	**cfg_od_clktop2_coreclke_divretimeren_h** Default Value: 1b cfg_od_clktop2_coreclke_divretimeren_h_defaultreset Access: R/W retimer enable: 0 for odd division ratio, 1 for even division ratio
	16	**cfg_reserved533** Default Value: 0b cfg_reserved533_defaultreset Access: RO Power well:DIG
	15:8	**cfg_od_clktop2_coreclkd_divratio_7_0** Default Value: 32h cfg_od_clktop2_coreclkd_divratio_7_0_defaultreset Access: R/W divider ratio for coreclkd divider 8'h00: div/256 8'h01: div/257 8'h02 - 8'hff: div/2 - div/255
	7:0	**cfg_od_clktop2_coreclkc_divratio_7_0** Default Value: 05h cfg_od_clktop2_coreclkc_divratio_7_0_defaultreset Access: R/W divider ratio for coreclkc divider 8'h00: div/256 8'h01: div/257 8'h02 - 8'hff: div/2 - div/255 8'h05: HD port

DKLP_CMN_ANA_CMN_ANA_DWORD27

DKLP_CMN_ANA_CMN_ANA_DWORD27

Register Space: MMIO: 0/2/0

Size (in bits): 32

REFCLKIN_CTL

DWord	Bit	Description
0	31:24	**cfg_reserved589** Default Value: 00h cfg_reserved589_defaultreset Access: RO Power well:DIG
	23:16	**cfg_reserved588** Default Value: 00h cfg_reserved588_defaultreset Access: RO Power well:DIG
	15:12	**cfg_reserved587** Default Value: 0h cfg_reserved587_defaultreset Access: RO Power well:DIG
	11	**cfg_od_refclkin2_refclkinjmux** Default Value: 0b cfg_od_refclkin2_refclkinjmux_defaultreset Access: R/W mux select of external injection refclk inputs: 0 to select amonrefclkinj, 1 to select rcomprefclkinj
	10:8	**cfg_od_refclkin2_refclkmux_2_0** Default Value: 011b cfg_od_refclkin2_refclkmux_2_0_defaultreset Access: R/W mux select for refclk output. 3'b000: external injection refclk; 3'b001: xtalinrefclk; 3'b010:mgrefclkin; 3'b011: socrefclk1; 3'100:socrefclk2; 3'b101:socrefclk3; 3'b110:socrefclk4; 3'b111:socrefclk5
	7:6	**cfg_reserved586** Default Value: 00b cfg_reserved586_defaultreset Access: RO Power well:DIG

DKLP_CMN_ANA_CMN_ANA_DWORD27

	5	**cfg_od_pll10p3g_refclkin_genlock_refclksel** Default Value: 0b cfg_od_pll10p3g_refclkin_genlock_refclksel_defaultreset Access: R/W pll10p3g select for refclk for genlock feature.
	4	**cfg_od_pll10g_refclkin_genlock_refclksel** Default Value: 0b cfg_od_pll10g_refclkin_genlock_refclksel_defaultreset Access: R/W pll10g select for refclk for genlock feature.
	3	**cfg_od_refclkin1_refclkinjmux** Default Value: 0b cfg_od_refclkin1_refclkinjmux_defaultreset Access: R/W mux select of external injection refclk inputs: 0 to select amonrefclkinj, 1 to select rcomprefclkinj
	2:0	**cfg_od_refclkin1_refclkmux_2_0** Default Value: 001b cfg_od_refclkin1_refclkmux_2_0_defaultreset Access: R/W mux select for refclk output. 3'b000: external injection refclk; 3'b001: xtalinrefclk; 3'b010:mgrefclkin; 3'b011: socrefclk1; 3'100:socrefclk2; 3'b101:socrefclk3; 3'b110:socrefclk4; 3'b111:socrefclk5

DKLP_CMN_DIG_CMN_DIG_DWORD6

DKLP_CMN_DIG_CMN_DIG_DWORD6

Register Space: MMIO: 0/2/0

Size (in bits): 32

MISC_SUS0

DWord	Bit	Description
0	31:29	**cfg_reserved** Default Value: 000b cfg_reserved_defaultreset Access: R/W Power well:SUS
	28:26	**cfg_os_cfg_imb750select** Default Value: 010b cfg_os_cfg_imb750select_defaultreset Access: R/W Power well:SUS
	25:24	**cfg_os_cfg_imbcompsel_2** Default Value: 00b cfg_os_cfg_imbcompsel_2_defaultreset Access: R/W CMN_DIG::cmn_dig_dword6::os_cfg_imbcompsel_2
	23	**cfg_os_cfg_imbcompsel** Default Value: 1b cfg_os_cfg_imbcompsel_defaultreset Access: R/W Power well:SUS
	22	**cfg_os_cfg_ircomp_ovrd_value** Default Value: 0b cfg_os_cfg_ircomp_ovrd_value_defaultreset Access: R/W Power well:SUS
	21	**cfg_os_cfg_ircomp_ovrd_en** Default Value: 0b cfg_os_cfg_ircomp_ovrd_en_defaultreset Access: R/W Power well:SUS

DKLP_CMN_DIG_CMN_DIG_DWORD6

	20	**cfg_os_cfg_invert_ptrim_ircomp_h** Default Value: 0b cfg_os_cfg_invert_ptrim_ircomp_h_defaultreset Access: R/W Power well:SUS
	19	**cfg_os_cfg_invert_ntrim_ircomp_h** Default Value: 0b cfg_os_cfg_invert_ntrim_ircomp_h_defaultreset Access: R/W Power well:SUS
	18	**cfg_os_cfg_invert_ptrim_h** Default Value: 0b cfg_os_cfg_invert_ptrim_h_defaultreset Access: R/W Power Well:SUS
	17	**cfg_os_cfg_invert_ntrim_h** Default Value: 0b cfg_os_cfg_invert_ntrim_h_defaultreset Access: R/W Power well:SUS
	16	**cfg_os_cfg_invert_ircomp_h** Default Value: 0b cfg_os_cfg_invert_ircomp_h_defaultreset Access: R/W Power well:SUS
	15:14	**cfg_os_susclk_dynclkgate_mode_1_0** Default Value: 00b cfg_os_susclk_dynclkgate_mode_1_0_defaultreset Access: R/W Dynamic Susclk Gating Model Select 00: Susclk gating and CLKREQ Forced High. In this mode the susclk will not be gated under any circumstances and the CLKREQ sent to the ip74pppxdkltypecfamilyew_SOC will be statically forced high. (default) 01: Susclk gating disabled, CLKREQ enabled. In this mode the susclk will not be gated under any circumstances. The CLKREQ sent to the ip74pppxdkltypecfamilyew_SOC will ip74pppxdkltypecfamilyew_toggle based on whether the susclk is needed by the MPHY 10: Susclk gating enabled, CLKREQ disabled (Forced High). In this mode the susclk will be gated during PS3-PS7 when no functions are requesting the susclk. The CLRKEQ sent to the ip74pppxdkltypecfamilyew_SOC will be statically forced high. 11: Susclk gating and CLKREQ enabled (POR Mode). In this mode the susclk will be gated during PS3-PS7 when no functions are requesting the susclk. The CLKREQ sent to the ip74pppxdkltypecfamilyew_SOC will ip74pppxdkltypecfamilyew_toggle based on whether the susclk is needed by the MPHY

DKLP_CMN_DIG_CMN_DIG_DWORD6

13	**cfg_cfg_calclk_srcsel** Default Value: 0b cfg_cfg_calclk_srcsel_defaultreset Access: R/W oa_ck_rxcalclk (Hardcoded to zero always)
12	**cfg_os_cfg_tr2pwr_gating_ctrl** Default Value: 0b cfg_os_cfg_tr2pwr_gating_ctrl_defaultreset Access: R/W Dynamic TR2 Power Gating Control Enable When asserted the dynamic power gating is enabled in the common logic. Default it is disable dynamic power gating.
11	**cfg_os_cfg_cl2pwr_gating_ctrl** Default Value: 0b cfg_os_cfg_cl2pwr_gating_ctrl_defaultreset Access: R/W Dynamic CL2 Power Gating Control Enable When asserted the dynamic power gating is enabled in the common logic. Default it is disable dynamic power gating.
10	**cfg_os_cfg_gaonpwr_gating_ctrl** Default Value: 0b cfg_os_cfg_gaonpwr_gating_ctrl_defaultreset Access: R/W Dynamic gatedAON Power Gating Control Enable When asserted the dynamic power gating is enabled in the common logic. Default it is disable dynamic power gating.
9	**cfg_os_cfg_cl2pwr_pll1en_gating_ctrl** Default Value: 1b cfg_os_cfg_cl2pwr_pll1en_gating_ctrl_defaultreset Access: R/W When enabled CL2 pwr will not turn off if pll1 is enabled. When disabled CL2 will ignore pll1 enable.
8	**cfg_cfg_calclkgate_dis** Default Value: 0b cfg_cfg_calclkgate_dis_defaultreset Access: R/W 0 - enable cal clock gating 1 - disable cal clock gating
7	**cfg_os_cfg_trpwr_gating_ctrl** Default Value: 0b cfg_os_cfg_trpwr_gating_ctrl_defaultreset Access: R/W Dynamic TR Power Gating Control Enable When asserted the dynamic power gating is enabled in the common logic. Default it is disable dynamic power gating.

DKLP_CMN_DIG_CMN_DIG_DWORD6

	6	**cfg_os_cfg_cl1pwr_gating_ctrl** Default Value: 0b cfg_os_cfg_cl1pwr_gating_ctrl_defaultreset Access: R/W Dynamic CL1 Power Gating Control Enable When asserted the dynamic power gating is enabled in the common logic. Default it is disable dynamic power gating.
	5	**cfg_os_cfg_dgpwr_gating_ctrl** Default Value: 0b cfg_os_cfg_dgpwr_gating_ctrl_defaultreset Access: R/W Dynamic DG Power Gating Control Enable When asserted the dynamic power gating is enabled in the common logic. Default it is disable dynamic power gating.
	4:0	**cfg_os_cfg_susclk_delay_5_1** Default Value: 10h cfg_os_cfg_susclk_delay_5_1_defaultreset Access: R/W Susclk Gating Cycle Delay ip74pppxdkltypecfamilyew_Timer When dynamic susclk gating is enabled. When all susclk requests are deasserted, this is the number of susclk cycles that the susclk will remain active before gating.

DKLP_CMN_DIG_CMN_DIG_DWORD29

DKLP_CMN_DIG_CMN_DIG_DWORD29

Register Space: MMIO: 0/2/0

Size (in bits): 32

LDO_VREF_ANA

DWord	Bit	Description
0	31:24	**cfg_reserved551** Default Value: 00h cfg_reserved551_defaultreset Access: RO Power well:SUS
	23:16	**cfg_os_cri_vref_rsrv_7_0** Default Value: 00h cfg_os_cri_vref_rsrv_7_0_defaultreset Access: R/W VREF CBB bonus bits
	15:13	**cfg_os_cri_vref_ref_sel_2_0** Default Value: 000b cfg_os_cri_vref_ref_sel_2_0_defaultreset Access: R/W Select between bg and resistor divider reference voltage
	12	**cfg_reserved549** Default Value: 0b cfg_reserved549_defaultreset Access: RO Power well:SUS
	11:10	**cfg_os_cri_vref_amon_sel_1_0** Default Value: 00b cfg_os_cri_vref_amon_sel_1_0_defaultreset Access: R/W Select voltage for monitor
	9	**cfg_reserved550** Default Value: 0b cfg_reserved550_defaultreset Access: RO Power well:SUS

DKLP_CMN_DIG_CMN_DIG_DWORD29

	8	**cfg_os_cfg_uc_handshake_enabled** Default Value: 0b cfg_os_cfg_uc_handshake_enabled_defaultreset Access: R/W Enables the handshake for the uC to disable the common lane DIG well (see bit0) 0: CMN DIG will power gate without intervention from uC 1: CMN DIG will wait for uC before power gating
	7:4	**cfg_os_cri_vref_nsel_3_0** Default Value: 0h cfg_os_cri_vref_nsel_3_0_defaultreset Access: R/W N select for current turning
	3:1	**cfg_os_cri_vref_rsel_2_0** Default Value: 000b cfg_os_cri_vref_rsel_2_0_defaultreset Access: R/W R select for current turning
	0	**cfg_os_cfg_uc_cmndig_pg_ok** Default Value: 0b cfg_os_cfg_uc_cmndig_pg_ok_defaultreset Access: R/W Indication from the uC that is acceptable to power gate the Common Lane DIG well.

DKLP_CMN_DIG_CMN_DIG_DWORD33

DKLP_CMN_DIG_CMN_DIG_DWORD33

Register Space: MMIO: 0/2/0

Size (in bits): 32

CFG_PWRGATE_LDO_MODE

DWord	Bit	Description
0	31:24	**cfg_reserved558** Default Value: 00h cfg_reserved558_defaultreset Access: RO Power well:DIG
	23:16	**cfg_reserved557** Default Value: 00h cfg_reserved557_defaultreset Access: RO Power well:DIG
	15:13	**cfg_reserved556** Default Value: 000b cfg_reserved556_defaultreset Access: RO Power well:SUS
	12	**cfg_os_cfg_tr_powergate_ldo_mode** Default Value: 0b cfg_os_cfg_tr_powergate_ldo_mode_defaultreset Access: R/W cfg powergate ldo mode for tr
	11	**cfg_os_cfg_tr2_powergate_ldo_mode** Default Value: 0b cfg_os_cfg_tr2_powergate_ldo_mode_defaultreset Access: R/W cfg powergate ldo mode for tr2
	10	**cfg_os_cfg_cl1_powergate_ldo_mode** Default Value: 0b cfg_os_cfg_cl1_powergate_ldo_mode_defaultreset Access: R/W cfg powergate ldo mode for cl1

DKLP_CMN_DIG_CMN_DIG_DWORD33		
	9	**cfg_os_cfg_cl2_powergate_ldo_mode** Default Value: 0b cfg_os_cfg_cl2_powergate_ldo_mode_defaultreset Access: R/W cfg powergate ldo mode for cl2
	8	**cfg_os_cfg_dig_powergate_ldo_mode** Default Value: 0b cfg_os_cfg_dig_powergate_ldo_mode_defaultreset Access: R/W cfg powergate ldo mode for dig
	7:0	**cfg_os_cfg_ldo_powerup_timer_7_0** Default Value: 7Dh cfg_os_cfg_ldo_powerup_timer_7_0_defaultreset Access: R/W cfg ldo powerup ip74pppxdkltypecfamilyew_timer

DKLP_CMN_UC_CMN_UC_DWORD27

DKLP_CMN_UC_CMN_UC_DWORD27

Register Space: MMIO: 0/2/0

Size (in bits): 32

CMN_UC::cmn_uc_dword27

DWord	Bit	Description
0	31:29	**cfg_reserved_dw27_2** Default Value: 000b cfg_reserved_dw27_2_defaultreset Access: RO CMN_UC::cmn_uc_dword27::reserved_dw27_2
	28	**cfg_forcepwrpok_req** Default Value: 0b cfg_forcepwrpok_req_defaultreset Access: RO CMN_UC::cmn_uc_dword27::forcepwrpok_req
	27	**cfg_phy_xtensa_ana_restore_done** Default Value: 0b cfg_phy_xtensa_ana_restore_done_defaultreset Access: RO CMN_UC::cmn_uc_dword27::phy_xtensa_ana_restore_done
	26	**cfg_phy_xtensa_unblock_ack** Default Value: 0b cfg_phy_xtensa_unblock_ack_defaultreset Access: RO Reserved
	25	**cfg_phy_xtensa_unblock_req** Default Value: 0b cfg_phy_xtensa_unblock_req_defaultreset Access: RO phy_xtensa_unblock_ack from firmware downloader block.
	24	**cfg_uc_xclkgatedstat** Default Value: 0b cfg_uc_xclkgatedstat_defaultreset Access: RO Phy_xtensa_unblock_request from firmware_downloader block. Can be used by firmware to determine the abort scenario. Cleared on reading this register.

DKLP_CMN_UC_CMN_UC_DWORD27

23	**cfg_phy_xtensa_ana_save_done** Default Value: 0b cfg_phy_xtensa_ana_save_done_defaultreset Access: RO Status register for xclk gated state. Cleared when xclk_gatedstat_clr bit is set.
22:21	**cfg_phy_xtensa_pllclk_change_req** Default Value: 00b cfg_phy_xtensa_pllclk_change_req_defaultreset Access: RO Consolidated version of all DLs ana save done status. CL should also have all the ana save done bits in RO status.
20	**cfg_phy_xtensa_block_req** Default Value: 0b cfg_phy_xtensa_block_req_defaultreset Access: RO PHY to set this register bit based on internal conditions and Xtensa to poll thru TIE. 0: PLL1 1: PLL2 ConfigID value MSB indicate original or shadow register write. Cleared when pllclk_change_ack is high.
19	**cfg_phy_xtensa_cl_corewell_wake** Default Value: 0b cfg_phy_xtensa_cl_corewell_wake_defaultreset Access: RO Set: When PM_block message is decoded for DEKEL10. Reset: When xtensa_phy_block_ack is 1
18	**cfg_phy_xtensa_sb_trigger_ack** Default Value: 0b cfg_phy_xtensa_sb_trigger_ack_defaultreset Access: RO This is set by CL-corewell wake logic (if it was dynamically power gated before). Cleared when Xtensa read this register.
17	**cfg_phy_xtensa_clk_switch_req** Default Value: 0b cfg_phy_xtensa_clk_switch_req_defaultreset Access: RO PHY to set this register If the request from Xtensa is read, then completion will be loaded in to 32-bit CRI-RO register and this bit is set. If request from Xtensa is write, then as soon as CL-hw sends the request to iosf-sb this bit is set. This bit is cleared when trigger_req is 1 -> 0.
16	**cfg_trigger_xtensa_to_start_restore** Default Value: 1b cfg_trigger_xtensa_to_start_restore_defaultreset Access: RO PHY has a need to switch clock source to Xtensa due to external conditions. 1: Indication to uC about its own clock switch. uC to move to idle and respond with ack. Cleared when the clk switching is complete.

DKLP_CMN_UC_CMN_UC_DWORD27

15	**cfg_uc_health** Default Value: 0b cfg_uc_health_defaultreset Access: R/W Set by CL hw after IP_READY is done and restore_b == 0 and internal restore is needed. This acts as trigger for uC to start internal restore process. Set during restore phase. Cleared on PM unblock ack msg.
14:13	**cfg_reseverd_dw27_1** Default Value: 00b cfg_reseverd_dw27_1_defaultreset Access: RO uc_health bit, indicating the firmware is up
12	**cfg_forcepwrpok_ack** Default Value: 0b cfg_forcepwrpok_ack_defaultreset Access: R/W Reserved
11	**cfg_xclkgatedstat_clr** Default Value: 0b cfg_xclkgatedstat_clr_defaultreset Access: R/W xclk gated status bit, used after coming back from clock gating. cleared on falling edge of xclk_gatedstat.
10	**cfg_xtensa_phy_send_block_nak** Default Value: 0b cfg_xtensa_phy_send_block_nak_defaultreset Access: R/W 0: No action 1: CL fsm can send NAK response to PM-REQ block message without delay. This bit can be set by Xtensa in ip74pppxdkltypecfamilyew_SOC where block message requires either ACK or NAK response without delay. ip74pppxdkltypecfamilyew_soc mode this bit is never set thus allowing CL logic/Xtensa to do DRAM save. This can be set by Xtensa if PM-REQ response delay is not tolerated by Punit. Cleared on falling edge of phy_block_req.
9	**cfg_xtensa_phy_block_ack** Default Value: 1b cfg_xtensa_phy_block_ack_defaultreset Access: R/W Once Xtensa takes control, it can SET the bit to ensure all housekeeping is done before PM-REQ is allowed to be responded 1: CL fsm can send block ack based on other internal conditions (like firmware download in progress) 0: CL fsm should look for Xtensa_phy_send_block_NAK to take further action for PM_REQ message in processing. this is cleared after DRAM save is done and phy_xtensa_block_req is TRUE. This is cleared by Xtensa after internal housekeeping is done. Cleared on falling edge of phy_block_req.

DKLP_CMN_UC_CMN_UC_DWORD27

	8	**cfg_xtensa_phy_sb_trigger_req** Default Value: 0b cfg_xtensa_phy_sb_trigger_req_defaultreset Access: R/W
	7:6	**cfg_xtensa_phy_pllclk_change_ack** Default Value: 00b cfg_xtensa_phy_pllclk_change_ack_defaultreset Access: R/W Xtensa has done PLL configuration writes in to main bank or shadow bank (depending on configID value). This bit is cleared after Phy_xtensa_pllclk_change_req is 0. 0: PLL1 1: PLL2
	5	**cfg_xtensa_phy_cl_corewell_pg_ok** Default Value: 1b cfg_xtensa_phy_cl_corewell_pg_ok_defaultreset Access: R/W Xtensa allows CL-sus-fsm to power gate CL-corewell. Other PG-entry conditions are decided by CL-logic. Default is CL-corewell power gating allowed. This bit gets cleared if any interrupt comes. Firmware needs to set it when it is ok to power gate. This bit is regular CRI domain flop and not saved.
	4	**cfg_xtensa_phy_clk_switch_ack** Default Value: 0b cfg_xtensa_phy_clk_switch_ack_defaultreset Access: R/W Xtensa has reduced pending external dependencies and ok for clock source switch. Cleared on falling edge of clk_switch_req.
	3	**cfg_xtensa_int_restore_done** Default Value: 0b cfg_xtensa_int_restore_done_defaultreset Access: R/W This is used in CL-fsm to notify ACU/DL to start internal ANA-restore if needed. Applicable for phy mode and used along with restore_b of 0. 1: Internal restore done for both CL/DL 0: Internal restore not done. This bit is cleared after unblock PM-ACK is sent out to PMC.
	2	**cfg_anasave_at_pm_req** Default Value: 1b cfg_anasave_at_pm_req_defaultreset Access: R/W Abutted to all DLs. Used along with PM_REQ block message. Once Xtensa takes control, it can clear/set this bit to either bypass ANA snapshot trigger or allow. Default is bypass as power gating of DL by ip74pppxdkltypecfamilyew_soc would erase the ANA register contents. 1: DL ACU fsm will trigger ANA-snap shot during save op. 0: DL ACU fsm should bypass ANA-snap shot during external or internal save. Set/Cleared by firmware

DKLP_CMN_UC_CMN_UC_DWORD27

	1	**cfg_xtensa_ok4_pll_disable** Default Value: 0b cfg_xtensa_ok4_pll_disable_defaultreset Access: R/W Set by Xtensa: If PLL is enabled forcefully and ok to be disabled to save power. cleared when the clock is gated.
	0	**cfg_xtensa_ok4_cg** Default Value: 0b cfg_xtensa_ok4_cg_defaultreset Access: R/W If xtensa is idle for X clocks and ok to gate its clock. RUNSTALL is applied prior to clock gate properly followed by explicit clock gating at source. cleared when the clock is gated.

DKLP_CMN_UC_CMN_UC_DWORD30

DKLP_CMN_UC_CMN_UC_DWORD30		
Register Space: MMIO: 0/2/0 Size (in bits): 32		
CMN_UC::cmn_uc_dword30		
DWord	**Bit**	**Description**
0	31:0	**cfg_scratch_reg** Default Value: 00000080h cfg_scratch_reg_defaultreset Access: R/W Used by firmware as a scratch register

DKLP_CMN2_DIG_CMN_DIG_DWORD0

DKLP_CMN2_DIG_CMN_DIG_DWORD0

Register Space: MMIO: 0/2/0

Size (in bits): 32

UC_INTR

DWord	Bit	Description
0	31:18	**cfg_reserved** Default Value: 0000h cfg_reserved_defaultreset Access: RO CMN2_DIG::cmn_dig_dword0::reserved
	17:16	**cfg_i_csr_cmn_pll_mux_select_ovrd_val** Default Value: 00b cfg_i_csr_cmn_pll_mux_select_ovrd_val_defaultreset Access: R/W Power Well:SUS
	15	**cfg_i_csr_cmn_dl_corepwr_ack_mask** Default Value: 1b cfg_i_csr_cmn_dl_corepwr_ack_mask_defaultreset Access: R/W Power Well:SUS
	14	**cfg_i_csr_cmn_pll_mux_select_ovrd_en** Default Value: 0b cfg_i_csr_cmn_pll_mux_select_ovrd_en_defaultreset Access: R/W Power Well:SUS
	13	**cfg_i_csr_cmn_clk_policy_ovrd_en** Default Value: 0b cfg_i_csr_cmn_clk_policy_ovrd_en_defaultreset Access: R/W Power well:SUS
	12:9	**cfg_i_csr_cmn_clk_policy_mask** Default Value: 0h cfg_i_csr_cmn_clk_policy_mask_defaultreset Access: R/W Power well:SUS

DKLP_CMN2_DIG_CMN_DIG_DWORD0

	8	**cfg_id_csr_cmn_uc_sideclk_div_select** Default Value: 0b cfg_id_csr_cmn_uc_sideclk_div_select_defaultreset Access: R/W Power well:SUS
	7	**cfg_os__cmn_cri_intr_trigger** Default Value: 0b cfg_os__cmn_cri_intr_trigger_defaultreset Access: R/W trigger interrupt status
	6	**cfg_os_cmn_cri_intr_rcomp_done** Default Value: 0b cfg_os_cmn_cri_intr_rcomp_done_defaultreset Access: R/W uc interrupt. 1: clear rcomp done. 0: update rcomp done
	5	**cfg_os_cmn_cri_intr_cmn_rst_l** Default Value: 0b cfg_os_cmn_cri_intr_cmn_rst_l_defaultreset Access: R/W uc interrupt. 1: clear cmnrst. 0: update cmnrst
	4	**cfg_os_cmn_cri_intr_procmon_done** Default Value: 0b cfg_os_cmn_cri_intr_procmon_done_defaultreset Access: R/W uc interrupt. 1: clear procmon. 0: update procmon
	3	**cfg_os_cmn_cri_intr_pll1_en** Default Value: 0b cfg_os_cmn_cri_intr_pll1_en_defaultreset Access: R/W uc interrupt, 1: clear pll1 enable. 0: update pll1 enable
	2	**cfg_os_cmn_cri_intr_pll2_en** Default Value: 0b cfg_os_cmn_cri_intr_pll2_en_defaultreset Access: R/W uc interrupt. 1: clear pll2 enable. 0: update pll2 enable
	1	**cfg_os_cmn_cri_intr_pll1_lock** Default Value: 0b cfg_os_cmn_cri_intr_pll1_lock_defaultreset Access: R/W uc interrupt. 1. clear pll1 lock. 0: update pll1 lock

DKLP_CMN2_DIG_CMN_DIG_DWORD0		
	0	**cfg_os_cmn_cri_intr_pll2_lock** Default Value: 0b cfg_os_cmn_cri_intr_pll2_lock_defaultreset Access: R/W uc interrupt. 1: clear pll2 lock. 0: update pll2 lock

DKLP_PCS_GLUE_PMD_IRQ_VEC_EN

DKLP_PCS_GLUE_PMD_IRQ_VEC_EN		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Enable irq functions within lane		
DWord	**Bit**	**Description**
0	31:0	**cfg_cfg_ecsr_irq_en** Default Value: 00080000h cfg_cfg_ecsr_irq_en_defaultreset Access: R/W PCS_Glue::PMD_IRQ_VEC_EN::cfg_ecsr_irq_en

DKLP_PCS_GLUE_RTT_CR_SPARE

DKLP_PCS_GLUE_RTT_CR_SPARE		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Spare registers for RTTLane		
DWord	**Bit**	**Description**
0	31:0	**cfg_rtt_cr_spare** Default Value: 8048000h cfg_rtt_cr_spare_defaultreset Access: R/W PCS_Glue::RTT_CR_SPARE::rtt_cr_spare

DKLP_PCS_GLUE_TX_DPCNTL0

DKLP_PCS_GLUE_TX_DPCNTL0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Dp control 0 register		
DWord	**Bit**	**Description**
0	31	**cfg_reserved_tx1** Default Value: 0b cfg_reserved_tx1_defaultreset Access: RO reserved fields
	30	**cfg_trainingen_tx1** Default Value: 0b cfg_trainingen_tx1_defaultreset Access: R/W Enable set for programming in Compliance mode
	29	**cfg_pipe_select_tx1** Default Value: 0b cfg_pipe_select_tx1_defaultreset Access: R/W Override this bit to 0 to take the BIOS programmed Values and not pipe
	28	**cfg_slow_trim_enable_tx1** Default Value: 1b cfg_slow_trim_enable_tx1_defaultreset Access: R/W Enable or Disable the slow trim
	27:23	**cfg_shunt_cm_tx1** Default Value: 00h cfg_shunt_cm_tx1_defaultreset Access: R/W Back mode select for the Shunt CM
	22:18	**cfg_shunt_cp_tx1** Default Value: 00h cfg_shunt_cp_tx1_defaultreset Access: R/W Back mode select for the Shunt CP

DKLP_PCS_GLUE_TX_DPCNTL0

	17:13	**cfg_preshoot_control_l0_tx1** Default Value: 00h cfg_preshoot_control_l0_tx1_defaultreset Access: R/W Preshoot Co-efficients
	12:8	**cfg_de_emphasis_control_l0_tx1** Default Value: 00h cfg_de_emphasis_control_l0_tx1_defaultreset Access: R/W De-emphasis co-efficient
	7:3	**cfg_cursor_control_tx1** Default Value: 00h cfg_cursor_control_tx1_defaultreset Access: R/W Back Up mode for Shunt on C0
	2:0	**cfg_vswing_control_tx1** Default Value: 111b cfg_vswing_control_tx1_defaultreset Access: R/W 1. 1V + 0dB (Full Swing) - 3b000 2. 800 mV + 0dB - 3b001 3. 600 mV + 0 dB - 3b100 4. 400 mV + 0dB -3b111

DKLP_PCS_GLUE_TX_DPCNTL1

DKLP_PCS_GLUE_TX_DPCNTL1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Dp control 1 register		
DWord	**Bit**	**Description**
0	31	**cfg_reserved_tx2** Default Value: 0b cfg_reserved_tx2_defaultreset Access: RO reserved fields
	30	**cfg_trainingen_tx2** Default Value: 0b cfg_trainingen_tx2_defaultreset Access: R/W Enable set for programming in Compliance mode
	29	**cfg_pipe_select_tx2** Default Value: 0b cfg_pipe_select_tx2_defaultreset Access: R/W Override this bit to 0 to take the BIOS programmed Values and not pipe
	28	**cfg_slow_trim_enable_tx2** Default Value: 1b cfg_slow_trim_enable_tx2_defaultreset Access: R/W Enable or Disable the slow trim
	27:23	**cfg_shunt_cm_tx2** Default Value: 00h cfg_shunt_cm_tx2_defaultreset Access: R/W Back mode select for the Shunt CM
	22:18	**cfg_shunt_cp_tx2** Default Value: 00h cfg_shunt_cp_tx2_defaultreset Access: R/W Back mode select for the Shunt CP

DKLP_PCS_GLUE_TX_DPCNTL1

	17:13	**cfg_preshoot_control_l0_tx2** Default Value: 00h cfg_preshoot_control_l0_tx2_defaultreset Access: R/W Preshoot Co-efficients
	12:8	**cfg_de_emphasis_control_l0_tx2** Default Value: 00h cfg_de_emphasis_control_l0_tx2_defaultreset Access: R/W De-emphasis co-efficient
	7:3	**cfg_cursor_control_tx2** Default Value: 00h cfg_cursor_control_tx2_defaultreset Access: R/W Back Up mode for Shunt on C0
	2:0	**cfg_vswing_control_tx2** Default Value: 111b cfg_vswing_control_tx2_defaultreset Access: R/W 1. 1V + 0dB (Full Swing) - 3b000 2. 800 mV + 0dB - 3b001 3. 600 mV + 0 dB - 3b100 4. 400 mV + 0dB -3b111

DKLP_PCS_GLUE_TX_DPCNTL2

DKLP_PCS_GLUE_TX_DPCNTL2

Register Space: MMIO: 0/2/0

Size (in bits): 32

Dp control 2 register

DWord	Bit	Description
0	31:13	**cfg_reserved_dp3** Default Value: 000000h cfg_reserved_dp3_defaultreset Access: RO PCS_Glue::TX_DPCNTL2::reserved_dp3
	12	**loadgen_sharing_pmd_disable** Access: R/W For DP1p62, HDMI5p94: This bit is used if we want to fall-back to older logic where no loadgen sharing will be there inside PMD logic by setting this bit to 1. For other modes: This bit can be set to 1 for enabling loadgen sharing feature if needed. Internally its used as inversion for these modes. Value: 0b – Disable **[Default]**; 1b – Enable
	11	**cfg_dp_mode_cg_enable** Access: R/W This bit enables the feature of power saving in non-dp modes by clock gating few clocks in Tx2. To disable this feature we need to program this bit to 0. Value: 1b – Enable **[Default]**; 0b – Disable
	10	**usb3_gen1_2ui_mode_en** Access: R/W This bit is to enable 2UI path to counter the lane2lane skew in USB3.2 Gen1 mode. Value: 1b – Enable **[Default]**; 0b – Disable
	9	**cfg_dp_fifo_depth_tx1** Default Value: 0b cfg_dp_fifo_depth_tx1_defaultreset Access: R/W PCS_Glue::TX_DPCNTL2::dp_fifo_depth_tx1

DKLP_PCS_GLUE_TX_DPCNTL2

8	**cfg_dp_fifo_depth_tx2** Default Value: 0b cfg_dp_fifo_depth_tx2_defaultreset Access: R/W PCS_Glue::TX_DPCNTL2::dp_fifo_depth_tx2
7	**cfg_dp_2ui_4ui_mode_en** Default Value: 0b cfg_dp_2ui_4ui_mode_en_defaultreset Access: R/W 1 - 2ui 0- 4ui
6:5	**cfg_loadgenselect_tx2** Default Value: 00b cfg_loadgenselect_tx2_defaultreset Access: R/W loadgen select for datapath2ui - Look at the sheet for CFG per Tx
4:3	**cfg_loadgenselect_tx1** Default Value: 00b cfg_loadgenselect_tx1_defaultreset Access: R/W loadgen select for datapath2ui - Look at the sheet for CFG per Tx
2	**cfg_dp20bitmode** Default Value: 0b cfg_dp20bitmode_defaultreset Access: R/W 20 bit mode supprot. This will need to be set to 1 if Pipe width does not reflect the 20bit mode
1	**cfg_rate8boverride_enable** Default Value: 0b cfg_rate8boverride_enable_defaultreset Access: R/W 8bit override enable. The pisorate_8b signal is normally decoded from PHYMODE and RATE. When 1, the pisorate_8b signal will take with pisorate8bit_ovrd value.
0	**cfg_rate8boverride** Default Value: 0b cfg_rate8boverride_defaultreset Access: R/W value used for pisorate_8b when pisorate8bit_overden = 1

DKLP_PCS_GLUE_TX1_FW_CALIB

DKLP_PCS_GLUE_TX1_FW_CALIB

Register Space: MMIO: 0/2/0

Size (in bits): 32

Tx2 Fw calib support

DWord	Bit	Description
0	31:8	**cfg_reserved_tx1** Default Value: 000000h cfg_reserved_tx1_defaultreset Access: R/W PCS_Glue::TX1_FW_CALIB::reserved_tx1
	7	**cfg_cfg_disable_wait_init_periodic** Default Value: 0b cfg_cfg_disable_wait_init_periodic_defaultreset Access: R/W PCS_Glue::TX1_FW_CALIB::cfg_disable_wait_init_periodic
	6	**cfg_tx1_dcc_cmp** Default Value: 0b cfg_tx1_dcc_cmp_defaultreset Access: RO PCS_Glue::TX1_FW_CALIB::tx1_dcc_cmp
	5	**cfg_cfg_fw_oneshotcal_req_ctrl_val** Default Value: 0b cfg_cfg_fw_oneshotcal_req_ctrl_val_defaultreset Access: R/W PCS_Glue::TX1_FW_CALIB::cfg_fw_oneshotcal_req_ctrl_val
	4	**cfg_cfg_tx_fw_oneshotcal_req_ctrl_en** Default Value: 0b cfg_cfg_tx_fw_oneshotcal_req_ctrl_en_defaultreset Access: R/W PCS_Glue::TX1_FW_CALIB::cfg_tx_fw_oneshotcal_req_ctrl_en
	3	**cfg_cfg_tx_pwr_cal_en_ovrd_val** Default Value: 0b cfg_cfg_tx_pwr_cal_en_ovrd_val_defaultreset Access: R/W PCS_Glue::TX1_FW_CALIB::cfg_tx_pwr_cal_en_ovrd_val

DKLP_PCS_GLUE_TX1_FW_CALIB

	2	**cfg_cfg_tx_pwr_cal_en_ovrd_en** Default Value: 0b cfg_cfg_tx_pwr_cal_en_ovrd_en_defaultreset Access: R/W PCS_Glue::TX1_FW_CALIB::cfg_tx_pwr_cal_en_ovrd_en
	1	**cfg_cfg_rate_cal_en_ovrd_val** Default Value: 0b cfg_cfg_rate_cal_en_ovrd_val_defaultreset Access: R/W PCS_Glue::TX1_FW_CALIB::cfg_rate_cal_en_ovrd_val
	0	**cfg_odkl_tx_rate_ana_cal_en** Default Value: 0b cfg_odkl_tx_rate_ana_cal_en_defaultreset Access: R/W PCS_Glue::TX1_FW_CALIB::odkl_tx_rate_ana_cal_en

DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD0

DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD0

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	168BF0h-168BF3h
Name:	DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD0
ShortName:	DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD0
Reset:	global

DFXMISC 7E

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15	**cri_dfx_marginmode** Default Value: 0b cri_dfx_marginmode_defaultreset Access: R/W Margin mode. Overrides standard setup configurations for use with Eye Width Margining. 0 : Standard Mode (default) 1 : Margin Mode enabled
	14:13	**cri_dfx_evenodd_mask** Default Value: 00b cri_dfx_evenodd_mask_defaultreset Access: R/W Even/odd bit select for comparison. 00 : no masking; 01: select odd path and mask the even path; 10: select even path and mask the odd path; 11: reserved;
	12	**dfx_chk_sel** Default Value: 0b dfx_chk_sel_defaultreset Access: R/W This is to select where the rx data is compared. 0 If compared after Kalgin, and rx data is in 10b domain and running on rxsymbclk; 1 if compared after 10b-)8b, the rx data is 8b domain and running on txsymbclk;

DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD0

11	**cri_dfx_synchdr_ignore_err** Default Value: 0b cri_dfx_synchdr_ignore_err_defaultreset Access: R/W Ignore Corrected Syncheader Errors. When enabled, the BoB LCE will ignore corrected single-bit syncheader errors in USB3 Gen2. 0: include corrected syncheader errors in errcnt 1: ignore corrected syncheader errors, do not add to errcnt. Applies only to BOB LCE.
10:8	**cri_dfx_maxerrcnt_2_0** Default Value: 000b cri_dfx_maxerrcnt_2_0_defaultreset Access: R/W Selectable maximum value that the DFXERRCNT can reach before the Local Compare Engine is automatically stopped. // 000 : Free running (default) // 001 : 2^16 // 010 : 2^10 // 011 : 2^8 // 100 : 2^4 // 101 : 2^2 // 110 : First Error // 111 : 2^32
7	**dfx_lcerx_dp_mode** Default Value: 0b dfx_lcerx_dp_mode_defaultreset Access: R/W 1= enable lcerx dp mode rxdata from upar sent to pipe as rxdata.
6	**cri_dfx_typec_dp_en** Default Value: 0b cri_dfx_typec_dp_en_defaultreset Access: R/W Enable to support Type-C TX1/TX2 data path in Base LCE. 0: ip74pppxdkltypecfamilyew_Normal LCE operating mode 1: LCE would replicate the lower 20/16b of it's output data on its upper 20/16b data. (16b/20b depends on whether encoder/decoder is bypassed or not)
5	**cri_dfx_eieos_sync_en** Default Value: 0b cri_dfx_eieos_sync_en_defaultreset Access: R/W BoB LCE Training Pattern control. 0: use EIEOS/SYNC + ISI pattern blocks 1: use only EIEOS/SYNC blocks

DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD0

4	**cri_dfx_patbuftrainovr** Default Value: 0b cri_dfx_patbuftrainovr_defaultreset Access: R/W Pattern Buffer Training Override. Enables manual training of the Pattern Buffer instead of the automated coordination between the Pattern Checker and Patter Generator. When asserted, the DFXPATBUFTRAIN bit will be used to force when training is active and when it is complete. 0 : Automatic Training (default) 1 : Enable Manual Training
3	**cri_dfx_dword_align_en** Default Value: 0b cri_dfx_dword_align_en_defaultreset Access: R/W Dword align enable. 1: EnableDword 0: No Dword alignment (default)
2	**cri_dfx_patchken** Default Value: 0b cri_dfx_patchken_defaultreset Access: R/W Pattern Checker Enable. 0 : Disable Pattern Checker (default) 1 : Enable Pattern Checker
1	**cri_dfx_patgenen** Default Value: 0b cri_dfx_patgenen_defaultreset Access: R/W Pattern Generator Enable. This will activate the DFx bypass muxes in the PCS Tx path. 0 : Disable Pattern Generator (default) 1: Enable Pattern Generator
0	**lce_en** Default Value: 0b lce_en_defaultreset Access: R/W Enable lce clock gating, must be set to 1'b1 first before lce is used

DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD1

DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD1

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	168BF4h-168BF7h
Name:	DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD1
ShortName:	DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD1
Reset:	global

DFXPATBUFSIZE

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15	**dfx_cri__lcerxtrain** Default Value: 0b dfx_cri__lcerxtrain_defaultreset Access: RO Indicates that the LCE RX has received at least 1 training pattern for BASE in non-raw PRBS and PATBUF modes.
	14:12	**cri_dfx_prbspoly_2_0** Default Value: 000b cri_dfx_prbspoly_2_0_defaultreset Access: R/W PRBS Polynomial. Selects the polynomial to be used by the PRBS in the Pattern Generator and Pattern Checker. For Base: 000 x^16 +x^5 +x^4 +x^3 +1 (USB3/PCIe Scrambler) (8b/16b/32b) 001 x^16 +x^15 +x^13 +x^4 +1 (SATA scrambler) (8b/16b/32b) 010 x^31 +x^28 +1 (8b/16b/32b) 011 x^7 +x^6 +1 (8b/16b/32b) 100 x^7 +x^6 +1 (10b/20b/40b) 101 Reserved 110 Reserved 111 Reserved For BOB: (Only 2 bits 5:4 are used. bit 6 is a don't care): 00 x^23 + x^21 + x^16 + x^8 + x^5 + x^2 + 1 01 x^31 + x^28 + 1 10 x^7 + x^6 + 1 11 Reserved

DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD1

	11	**cri_dfx_patbuftrain** Default Value: 0b cri_dfx_patbuftrain_defaultreset Access: R/W Pattern Buffer Manual Training. When DFXPATBUFTRAINOVR and DFXLCESTART are asserted, this manually controls whether the Pattern Buffers are in training or have completed training. 0 : Send training patterns (default) 1 : Send contents of DFXPATBUF
	10	**cri_dfx_clrerrcnt** Default Value: 0b cri_dfx_clrerrcnt_defaultreset Access: R/W Clear Error ip74pppxdkltypecfamilyew_Counter. Resets the Pattern Checker's error ip74pppxdkltypecfamilyew_counter. 0 : Error ip74pppxdkltypecfamilyew_Counter allowed to run (default) 1 : Error ip74pppxdkltypecfamilyew_Counter held in reset
	9	**cri_dfx_lcereset** Default Value: 0b cri_dfx_lcereset_defaultreset Access: R/W Local Compare Engine Reset. Resets all components of the Local Compare Engine in both the Pattern Generator and the Pattern Checker. 0 : LCE not in reset (default) 1 : LCE in reset
	8	**cri_dfx_lcestart** Default Value: 0b cri_dfx_lcestart_defaultreset Access: R/W Local Compare Engine Start. Controls the starting and stopping of the LCE. Allows for simultaneous or independent start across lanes. Once stopped, a DFXLCERESET is usually required. 0 : LCE stopped (default) 1 : LCE running
	7	**cri_dfx_xor_data_en** Default Value: 0b cri_dfx_xor_data_en_defaultreset Access: R/W Patbuf data scrambling enable. Setting this bit will XOR the patbuf data with the PRBS pattern selected by cri_dfx_prbspoly. Applies to both Base and BoB pattern buffers. 0: Patbuf data is not scrambled by LFSR 1: Patbuf data is scrambled by selected LFSR pattern

DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD1

	6	**cri_dfx_last_sym_en** Default Value: 0b cri_dfx_last_sym_en_defaultreset Access: R/W Base Pattern Buffer Repeat Last Symbol. Enabling this mode will cause the Base patbuf to repeat bits [9:0] an extra 4 times at the end of each patbuf loop. 0: Do not repeat last symbol 1: Repeat last symbol 4 times
	5:4	**cri_dfx_patbufsize_1_0** Default Value: 00b cri_dfx_patbufsize_1_0_defaultreset Access: R/W Pattern Buffer Size. Selects how much of the Pattern Buffer to use, which allows for shorter pattern sequences. Base pattern gen: no effect, Base LCE is always 120 bits, 12 symbols. BoB pattern generator: The MSB will always be bit 127. The LSB will depend on this select. 00 : full 4x32b or 8x16b buffer (default) 01 : 3x32b (while in 32b mode), 7x16b (while in 16b mode) 10 : 2x32b (while in 32b mode), 5x16b (while in 16b mode) 11 : 1x32b (while in 32b mode), 3x16b (while in 16b mode)
	3	**cri_dfx_patbufloop** Default Value: 1b cri_dfx_patbufloop_defaultreset Access: R/W Pattern Buffer Looping enable. Enables looping of the Patter Buffer. By default, the contents of Pattern Buffer will be used once and stop. Alternatively, the Pattern Buffer will continue to loop until the DFXLCESTART is de-asserted. 0 : Run Pattern Buffer only once (default) 1 : Loop Pattern Buffer until stopped
	2	**cri_dfx_patbufdwidth** Default Value: 0b cri_dfx_patbufdwidth_defaultreset Access: R/W Pattern Buffer Data Width. For Base: Selects between 10/20/40 bit (8b/10b encoder on TX bypassed) or 8/16/32 bit data width for the Pattern Buffer. On the RX path, data is checked before K-align Block only for PRBS, when 10/20/40b mode is selected. 0 : 10/20/40b (default) 1 : 8/16/32b For BOB 0 : TX Block Encoder is bypassed/Data is Checked before RX Block Align, which is supported only in PRBS

<table>
<tr><th colspan="3">DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD1</th></tr>
<tr><td></td><td></td><td>1 : TX Block Encoder is used/Data is Checked after RX Block Align</td></tr>
<tr><td></td><td>1:0</td><td>cri_dfx_lcepatsrc
<table><tr><td>Default Value:</td><td>00b cri_dfx_lcepatsrc_defaultreset</td></tr><tr><td>Access:</td><td>R/W</td></tr></table>Local Compare Engine Pattern Source. Selects between using Pattern Buffer or PRBS as source of LCE patterns.
00 : Pattern Buffer; use LCE default training pattern;
01: Pattern Buffer; use the first 40b of pattern buffer for training.
10: PRBS; use LCE default training pattern;
11: PRBS; use the first 40b of pattern buffer for training.</td></tr>
</table>

DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD7

DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD7

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	168BF8h-168BFBh
Name:	DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD7
ShortName:	DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD7
Reset:	global

DFXPRBSSEED1

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:12	**cri_dfx_maxbitcnt_3_0** Default Value: 0h cri_dfx_maxbitcnt_3_0_defaultreset Access: R/W Selectable maximum value that the DFXBITCNT can reach before the Local Compare Engine is automatically stopped. // 000 : Feature is disabled // 1 : 2^13 (4096 words) // 2 : 2^14 (8192 words) // 3 : 2^15 (16384 words) // ... // 13: 2^25 (1.67E7 words) // 14: 2^29 (5.38E8 words) // 15: 2^32 (4.29E9 words)
	11	**cri_dfx_raw_high_16b_sel** Default Value: 0b cri_dfx_raw_high_16b_sel_defaultreset Access: R/W Chooses between upper 16 bits and lower 16bits of the actual and expected data logged to be read out in BOB 32b RX mode. (TBT3) 0: Selects lower 16bits of the expected/actual data in the error logging register 1: Selects upper 16bits of the expected/actual data in the error logging register
	10	**cri_dfx_errlog_freeze** Default Value: 0b cri_dfx_errlog_freeze_defaultreset Access: R/W Error Log register Freeze. This bit will freeze the error logging when in free-running mode so the actual and expected data can be read by software. This bit must be 0 in first-fail mode.

DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD7

		This bit must be cleared to capture subsequent errors. This bit has no effect on the error ip74pppxdkltypecfamilyew_counter and only freezes the log registers. This bit applies to both Base and BoB LCE engines. 0: ip74pppxdkltypecfamilyew_Normal operation, log errors 1: Hold error log register values
	9	**cri_dfx_errlog_index** Default Value: 0b cri_dfx_errlog_index_defaultreset Access: R/W Error Log register bank Index. This bit is used to select which error log register bank is read from the Error Log registers. This bit applies to both Base and BoB LCE engines. 0: select bank 0 1: select bank 1
	8	**cri_dfx_errlog_mode** Default Value: 0b cri_dfx_errlog_mode_defaultreset Access: R/W Error Log storage mode. This bit applies to both Base and BoB LCE engines. 0: Capture first two failing cycles, then stop 1: Capture latest two failing cycles, free-running
	7:0	**cri_dfx_prbsseed_39_32** Default Value: 00h cfg_cri_dfx_prbsseed_39_32_defaultreset Access: R/W PRBS seed.

DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD8

DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD8

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	168BFCh-168BFFh
Name:	DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD8
ShortName:	DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD8
Reset:	global

DFXPRBSSEED3

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:8	**cri_dfx_prbsseed_23_16** Default Value: FFh cri_dfx_prbsseed_23_16_defaultreset Access: R/W reserved
	7:0	**cri_dfx_prbsseed_31_24** Default Value: FFh cri_dfx_prbsseed_31_24_defaultreset Access: R/W PRBS seed. Can be used for HVM determinism. The use of all zeroes can result in lockup.

DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD9

DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD9

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	168C00h-168C03h
Name:	DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD9
ShortName:	DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD9
Reset:	global

DFXPRBSSEED1

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:8	**cri_dfx_prbsseed_7_0** Default Value: FFh cri_dfx_prbsseed_7_0_defaultreset Access: R/W reserved
	7:0	**cri_dfx_prbsseed_15_8** Default Value: FFh cri_dfx_prbsseed_15_8_defaultreset Access: R/W reserved

DKLP_PCS_PCS_DWORD3

DKLP_PCS_PCS_DWORD3

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	168BE8h-168BEBh
Name:	DKLP_PCS_PCS_DWORD3
ShortName:	DKLP_PCS_PCS_DWORD3
Reset:	global

PRBS

DWord	Bit	Description
0	31:24	**cri_kalign_timer_value_gen1** Default Value: 00h cri_kalign_timer_value_gen1_defaultreset Access: R/W When K-Align timer mode is enabled, this value represents the timer value for Gen1 rates.[3:0] = PS1 timer value. Multiply this value by 32ns[7:4] = Non-PS1 timer value. Multiply this value by 64ns
	23:6	**Reserved** Default Value: 00h Access: RO
	5	**Reserved**
	4	**o_cfg_pcs_dp_calbypass_disable** Access: R/W To disable the DP calbypass. Value: 0b — enable **[Default]** Value: 1b — disble
	3	**o_cri_tx_raw_mode_enable** Access: R/W To enable the PRBS raw mode for TX data. Value: 0b — disable **[Default]** Value: 1b — enable
	2:0	**Reserved**

DKLP_PCS_PCS_DWORD23

DKLP_PCS_PCS_DWORD23		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
SET50		
DWord	**Bit**	**Description**
0	31:29	**cfg_reserved583** Default Value: 000b cfg_reserved583_defaultreset Access: RO Reserved
	28:24	**cfg_reserved584** Default Value: 00h cfg_reserved584_defaultreset Access: RO Hysteresis ip74pppxdkltypecfamilyew_timer value for corewell gating. Power gating criteria(expect PMC) must me met for these many susclks before power gating is initiated. Crireg resides in suswell and retained in aonwell.
	23:21	**cfg_reserved581** Default Value: 000b cfg_reserved581_defaultreset Access: RO Reserved
	20:16	**cfg_reserved582** Default Value: 00h cfg_reserved582_defaultreset Access: RO Hysteresis ip74pppxdkltypecfamilyew_timer value for suswell gating. Power gating criteria(expect PMC) must me met for these many susclks before power gating is initiated. Crireg resides in aonwell.
	15:9	**cfg_reserved579** Default Value: 00h cfg_reserved579_defaultreset Access: RO Reserved
	8	**cfg_reserved580** Default Value: 0b cfg_reserved580_defaultreset Access: RO pianclkbufen assertion to rxidle deassertion delay ip74pppxdkltypecfamilyew_timer override

DKLP_PCS_PCS_DWORD23

	7:3	**cfg_reserved578** Default Value: 00h cfg_reserved578_defaultreset Access: RO Reserved
	2	**cfg_cri_disable_powerdown_based_gating** Default Value: 1b cfg_cri_disable_powerdown_based_gating_defaultreset Access: R/W This is for just incase something unexpected happen and internally we powergate in be;ow PS0.
	1	**cfg_reg_rxlpsen_enable_raw_all_powerdown** Default Value: 0b cfg_reg_rxlpsen_enable_raw_all_powerdown_defaultreset Access: R/W When set to 1 raw mode is enabled in all powerdown state. lfpsen from PIPE is passed to RxuP asynchronously.
	0	**cfg_reg_rxlfpsen_use_raw** Default Value: 0b cfg_reg_rxlfpsen_use_raw_defaultreset Access: R/W When set to 1, in PS3 (and above) susclk is not requested. As a result, lfps is not reported on rxelecidle. Additionally lfpsen from PIPE is passed to RxuP asynchronously. Implemented with AON retention.

DKLP_PCS_PCS_DWORD25

DKLP_PCS_PCS_DWORD25

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	168BECh-168BEFh
Name:	DKLP_PCS_PCS_DWORD25
ShortName:	DKLP_PCS_PCS_DWORD25
Reset:	global

Used for INDEX_REGISTERS programming, to program DKLP_PCS_INDEXED_PCS_ICL_INDEXED_DWORD* registers.
INDEX 0 refers to DWORD0 and INDEX 63 refers to DWORD63.

DWord	Bit	Description
0	31:24	**dfx_regaccess_31_24** Default Value: 00h dfx_regaccess_31_24_defaultreset Access: R/W Upper 3 bits are RNW(MSB) Bit31 - 0 - Read 1 - Write Bit30 - Byte1_sel Bit29 - Byte0_sel
	23:16	**dfx_regaccess_23_16** Default Value: 00h dfx_regaccess_23_16_defaultreset Access: R/W Lower 6bits i.e., [21:16] are address select
	15:8	**dfx_regaccess_15_8** Default Value: 00h dfx_regaccess_15_8_defaultreset Access: R/W Write or Read Data[15:8]
	7:0	**dfx_regaccess_7_0** Default Value: 00h dfx_regaccess_7_0_defaultreset Access: R/W Write or Read Data[7:0]

DKLP_PLL0_BIAS

DKLP_PLL0_BIAS		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
BIAS Register		
DWord	**Bit**	**Description**
0	31	**cfg_i_tdc_fine_res** Default Value: 1b cfg_i_tdc_fine_res_defaultreset Access: R/W RET DFUSE STRAP TDC fine resolution select 0: Coarse resolution / 8 1: Coarse resolution / 4
	30	**cfg_i_fracnen_h** Default Value: 1b cfg_i_fracnen_h_defaultreset Access: R/W RET DFUSE STRAP Enables fractional modulator. For SSC, this bit needs to be set to '1', even though it starts with integer division ratio. This is not part of direct pin IF. This is only integer to integer with BW optimization across all supported interger divisions or fractional to fractional with small PPM changes. Integer to fractional or vice versa is not supported.
	29:24	**cfg_i_fbdiv_frac_21_16** Default Value: 0Dh cfg_i_fbdiv_frac_21_16_defaultreset Access: R/W RET DFUSE STRAP Fractional Modulator settings
	23:16	**cfg_i_fbdiv_frac_15_8** Default Value: 55h cfg_i_fbdiv_frac_15_8_defaultreset Access: R/W RET DFUSE STRAP Fractional Modulator settings
	15:8	**cfg_i_fbdiv_frac_7_0** Default Value: 55h cfg_i_fbdiv_frac_7_0_defaultreset Access: R/W RET DFUSE STRAP Fractional Modulator settings
	7:0	**cfg_i_sscinj_stepsize_7_0** Default Value: 00h cfg_i_sscinj_stepsize_7_0_defaultreset Access: R/W RET DFUSE SSCInjection Step Size [7:0]

DKLP_PLL0_DIV0

DKLP_PLL0_DIV0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
DIV0 Register		
DWord	**Bit**	**Description**
0	31:30	**cfg_i_truelock_criteria_1_0** Default Value: 01b cfg_i_truelock_criteria_1_0_defaultreset Access: R/W RET DFUSE STRAP True lock indicator criteria. After early lock generation, external PLL lock indicator asserted high when phase error is less than threshold value for 00: 16 consecutive cycles 01: 32 consecutive cycles 10: 48 consecutive cycles 11: 64 consecutive cycles
	29:28	**cfg_i_earlylock_criteria_1_0** Default Value: 11b cfg_i_earlylock_criteria_1_0_defaultreset Access: R/W RET DFUSE STRAP Early lock indicator criteria. Early PLL lock indicator asserted high when phase error is less than threshold value for 00: 16 consecutive cycles 01: 32 consecutive cycles 10: 48 consecutive cycles 11: 64 consecutive cycles
	27:25	**cfg_i_afc_startup_2_0** Default Value: 000b cfg_i_afc_startup_2_0_defaultreset Access: R/W RET DFUSE STRAP This is for AFC start point. NOTE: i_afc_startup[2] - DW4, byte17, bit[0] 000: fine = 511 001: fine = 639 (+128) 010: fine = 767 (+256) 011: fine = 895 (+384) 100: NA 101: fine = 127 (-384) 110: fine = 255 (-256) 111: fine = 383 (-128)
	24	**cfg_i_divretimeren** Default Value: 0b cfg_i_divretimeren_defaultreset Access: R/W RET DFUSE STRAP Retiming of feedback clock
	23:21	**cfg_i_gainctrl_2_0** Default Value: 001b cfg_i_gainctrl_2_0_defaultreset Access: R/W RET DFUSE STRAP Adjustable gain for loop filter. Both coefficients shifted right by gainctrl before lock

DKLP_PLL0_DIV0

	20:16	**cfg_i_int_coeff_4_0** Default Value: 07h cfg_i_int_coeff_4_0_defaultreset Access: R/W RET DFUSE STRAP integral coeff. = 2^(-int_coeff), tageting up to 2^-11
	15:12	**cfg_i_prop_coeff_3_0** Default Value: 3h cfg_i_prop_coeff_3_0_defaultreset Access: R/W RET DFUSE STRAP proportional coeff. = 2^(-prop_coeff+1)
	11:8	**cfg_i_fbprediv_3_0** Default Value: 2h cfg_i_fbprediv_3_0_defaultreset Access: R/W RET DFUSE STRAP predivider ratio 0000,0001 : reserved 0010: /2 0100: /4 0011: reserved Rest: reserved For external IF pin use case with
	7:0	**cfg_i_fbdiv_intgr_7_0** Default Value: 82h cfg_i_fbdiv_intgr_7_0_defaultreset Access: R/W RET DFUSE STRAP Feedback divider post division (M2)

DKLP_PLL0_DIV1

DKLP_PLL0_DIV1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
DIV1 Register		
DWord	**Bit**	**Description**
0	31	**cfg_i_bw_ampmeas_window** Default Value: 0b cfg_i_bw_ampmeas_window_defaultreset Access: R/W RET DFUSE STRAP 0: 10 modulation cycles of averaging for mplitude measurement 1: 20 modulation cycles of averaging for mplitude measurement
	30:29	**cfg_i_bias_calib_stepsize_1_0** Default Value: 00b cfg_i_bias_calib_stepsize_1_0_defaultreset Access: R/W RET DFUSE STRAP Bias Calibration Step Size during linear search 00: 1 01: 2 10: 3 11: 4
	28:24	**cfg_i_ctrim_4_0** Default Value: 0Ch cfg_i_ctrim_4_0_defaultreset Access: R/W RET DFUSE STRAP ip74pppxdkltypecfamilyew_Cap trimming for irefout. This also has refclock dependency. Current default is for 24MHz.
	23	**cfg_i_fastlock_internal_reset** Default Value: 1b cfg_i_fastlock_internal_reset_defaultreset Access: R/W RETCLR DFUSE STRAP clears internal fastlock memory so that next cold start will do both TDC and AFC calibration instead of fast lock. NOTE: this does not clear the i_fastlock_en_h register bit, clears functional register and self-clears Formerly, i_bbthresh[3] (no longer strap)
	22:21	**cfg_i_bias_r_programability_1_0** Default Value: 10b cfg_i_bias_r_programability_1_0_defaultreset Access: R/W RET DFUSE STRAP [1:0] : Bias Filter R programmability - note mapped to bias_bonus[1:0]
	20:16	**cfg_i_ireftrim_4_0** Default Value: 1Ch cfg_i_ireftrim_4_0_defaultreset Access: R/W RET DFUSE STRAP Output current trim . Mirror ratio is changed based on constant current

DKLP_PLL0_DIV1

		requirement. SInce it is refclock dependent, needs to be reconfigurable. i_ireftrim[4:0] - 38.4/19.2 MHz - 5'h1C i_ireftrim[4:0] - 25/100 MHz = 5'h18 Step size: 20 uA
	15	**cfg_i_biasfilter_en_delay** Default Value: 1b cfg_i_biasfilter_en_delay_defaultreset Access: R/W RET DFUSE 0: Filter enabled even before pll enabled 1: Filter enable is delay until lock acquisition This bit is sensitive only when i_bias_filter_en is set (bit3)
	14	**cfg_i_bias_filter_en** Default Value: 1b cfg_i_bias_filter_en_defaultreset Access: R/W RET DFUSE 0: Disables bias filter 1: Enables bias filter
	13	**cfg_i_biascal_en_h** Default Value: 0b cfg_i_biascal_en_h_defaultreset Access: R/W RET DFUSE STRAP Bias Calibration Signal. Bias cal should be disable when override DCO coarse code.
	12	**cfg_i_dcodither_config** Default Value: 0b cfg_i_dcodither_config_defaultreset Access: R/W RET DFUSE Whenever we have binary weighted MFC ip74pppxdkltypecfamilyew_cap, this should be set to 1'b0. Ex: i_dcofine_resolution = 1'b0. For this case, this should be set to 1'b0. 0: No floating dither 1: Floating dither (511+Nobinary - Floating dither)
	11:8	**cfg_i_lockthresh_3_0** Default Value: 5h cfg_i_lockthresh_3_0_defaultreset Access: R/W RET DFUSE STRAP Digital lock detect threshold, the PLL will generate plllockout when the phase error detected by TDC is within lockthresh number of TDC counts for 64 consecutive cycles 5*16 (TDC Code) = 80 (16 is internally hard coded scalar value) - This setting isin middle of coarse range
	7:0	**cfg_i_tdctargetcnt_7_0** Default Value: 11h cfg_i_tdctargetcnt_7_0_defaultreset Access: R/W RET DFUSE STRAP TDC tristate buffer calibration ip74pppxdkltypecfamilyew_counter value. Delay line loop oscillation is counted over two refclk cycles. This is used for TDC coarse code calibration

DKLP_PLL0_FRAC_LOCK

DKLP_PLL0_FRAC_LOCK		
Register Space: MMIO: 0/2/0 Size (in bits): 32		
FRAC_LOCK Register		
DWord	**Bit**	**Description**
0	31:30	**cfg_i_cml2cmosbonus_1_0** Default Value: 00b cfg_i_cml2cmosbonus_1_0_defaultreset Access: R/W RET DFUSE STRAP NOTE: These bits are ported to anatop bit [0] mapped to cml2cmosbonus[1] port - o_plllc_ck_dcocmosclkp_ana disable '1' = o_plllc_ck_dcocmosclkp_ana is disabled (ie for MG B0) '0' = both phases of the cmos clock ip74pppxdkltypecfamilyew_toggle at the interface bit[1] mapped to cml2cmosbonus[2] port - available
	29:27	**cfg_i_bb_gain2_2_0** Default Value: 000b cfg_i_bb_gain2_2_0_defaultreset Access: R/W RET DFUSE STRAP bb_gain2[2:0] : BB gain for second BB range; expect bb_gain2 >= bb_gain1
	26:24	**cfg_i_bb_gain1_2_0** Default Value: 000b cfg_i_bb_gain1_2_0_defaultreset Access: R/W RET DFUSE STRAP BB gain for first BB range
	23	**cfg_i_fastlock_en_h** Default Value: 0b cfg_i_fastlock_en_h_defaultreset Access: R/W RET DFUSE STRAP Enable FLL based AFC; this replaces binary search based AFC. FLL based AFC faster than binary search
	22:19	**cfg_i_fllafc_gain_3_0** Default Value: 8h cfg_i_fllafc_gain_3_0_defaultreset Access: R/W RET DFUSE STRAP Initial FLL gain that decrements down to 1 every refclk cycle in the beginning of FLL based AFC

DKLP_PLL0_FRAC_LOCK

	18:16	**cfg_i_fllafc_lockcnt_2_0** Default Value: 100b cfg_i_fllafc_lockcnt_2_0_defaultreset Access: R/W RET DFUSE STRAP Number of refclk cycles for FLL lock after gain is reduced to 1
	15:8	**cfg_i_max_cselafc_7_0** Default Value: 3Fh cfg_i_max_cselafc_7_0_defaultreset Access: R/W RET DFUSE STRAP Max. AFC code for a given DCO
	7:0	**cfg_i_init_cselafc_7_0** Default Value: 6Ah cfg_i_init_cselafc_7_0_defaultreset Access: R/W RET DFUSE STRAP Initial AFC code for FLL AFC; apply approximate AFC, and starting at closer frequency helps fast/accurate calibration

DKLP_PLL0_LF

DKLP_PLL0_LF		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
LF Register		
DWord	**Bit**	**Description**
0	31	**cfg_i_afc_divratio** Default Value: 0b cfg_i_afc_divratio_defaultreset Access: R/W RET DFUSE STRAP 0: DCO/4 (prediv: 2, Mdithdiv: 2) 1: DCO/8 (prediv: 4 Mdithdiv: 2)
	30:29	**cfg_i_plllock_sel_1_0** Default Value: 00b cfg_i_plllock_sel_1_0_defaultreset Access: R/W RET DFUSE select between lockdetect-based plllock or ip74pppxdkltypecfamilyew_counter based plllock 11: Sticky lock 10: ip74pppxdkltypecfamilyew_Counter-based 01: Lock Detection + ip74pppxdkltypecfamilyew_Counter 00: Lock Detection
	28:24	**cfg_i_bwphase_4_0** Default Value: 00h cfg_i_bwphase_4_0_defaultreset Access: R/W RET DFUSE STRAP Phase amplitude for bandwidth measurement
	23:21	**cfg_i_ft_mode_sel_2_0** Default Value: 010b cfg_i_ft_mode_sel_2_0_defaultreset Access: R/W RET DFUSE STRAP ftmodesel[2:0] : 9b vs 10b finetune selection, lsb0 tuning 3'b000 : 10b Nom -> Dither = LSB0, LSB0 = LSB0 3'b001 : 10b+ -> Dither = LSB+, LSB0 = LSB0+ 3'b010 : 10b- -> Dither = LSB-, LSB0 = LSB0- 3'b011 : 10b DNL -> Dither = LSB01, LSB0 = LSB0+ 3'h1xx: 9b -> Dither = LSB1, LSB0 = N/A
	20:19	**cfg_i_bw_mode_1_0** Default Value: 00b cfg_i_bw_mode_1_0_defaultreset Access: R/W RET DFUSE STRAP 00: No measurement 01: BW measurement 10: BW calibration up to +1 direction 11: BW calibration up to +2 direction
	18:16	**cfg_i_bw_upperbound_2_0** Default Value: 000b cfg_i_bw_upperbound_2_0_defaultreset Access: R/W RET DFUSE STRAP Upper bound setting for BW 000: 0.1MHz 001: 1 MHz 010: 2 MHz 011: 5 MHz 100: 6 MHz 101: 8 MHz 110: 16 MHz 111: 25MHz

DKLP_PLL0_LF

15:13 cfg_i_bw_lowerbound_2_0

Default Value:	000b cfg_i_bw_lowerbound_2_0_defaultreset
Access:	R/W

RET DFUSE STRAP Lower bound setting for BW 000: 0.1MHz 001: 1 MHz 010: 2 MHz 011: 5 MHz 100: 6 MHz 101: 8 MHz 110: 16 MHz 111: 25MHz

12:9 cfg_i_dcoamp_3_0

Default Value:	0h cfg_i_dcoamp_3_0_defaultreset
Access:	R/W

RET DFUSE Amplitude override value for DCO, 0000 is min amplitude, 1111 is max amplitude, applied when i_dcoampovrden_h is high. 4bit is left shifted to generate 6bit code (x4)

8 cfg_i_dcoampovrden_h

Default Value:	0b cfg_i_dcoampovrden_h_defaultreset
Access:	R/W

RET DFUSE DCO amplitude override enable: 0 DCO amplitude set internally (default) 1 DCO amplitude is set by i_dcoamp[3:0]

7:5 cfg_i_bbthresh2_2_0

Default Value:	000b cfg_i_bbthresh2_2_0_defaultreset
Access:	R/W

RET DFUSE STRAP threshold for second (outer) BB range; expect bbthresh2 >= bbthresh1

4:2 cfg_i_bbthresh1_2_0

Default Value:	000b cfg_i_bbthresh1_2_0_defaultreset
Access:	R/W

RET DFUSE STRAP threshold for first (inner) bang-bang (BB) range

1:0 cfg_i_tdc_offset_lock_1_0

Default Value:	00b cfg_i_tdc_offset_lock_1_0_defaultreset
Access:	R/W

RET DFUSE TDC Offset during lock for integer mode

DKLP_PLL0_TDC_COLDST_BIAS

DKLP_PLL0_TDC_COLDST_BIAS

Register Space: MMIO: 0/2/0

Size (in bits): 32

TDC_COLDST_BIAS Register

DWord	Bit	Description
0	31:29	**cfg_i_cloadctrllext_4_2** Default Value: 000b cfg_i_cloadctrllext_4_2_defaultreset Access: R/W RET VFUSE cload control override for H Vernier line. This is applied when i_tdccalexten_h = 1
	28:24	**cfg_i_tribufctrlext_4_0** Default Value: 00h cfg_i_tribufctrlext_4_0_defaultreset Access: R/W RET VFUSE Tribufctrl control override. This is applied when i_tdccalexten_h = 1
	23:16	**cfg_i_dcocoarse_7_0** Default Value: 00h cfg_i_dcocoarse_7_0_defaultreset Access: R/W RET VFUSE STRAP DCO coarse tune frequency value, when dcocoarse_ovrd_h is '1', this input is used to override the value calculated from the Automatic Frequency Calibration (AFC) block
	15:8	**cfg_i_sscstepsize_7_0** Default Value: 13h cfg_i_sscstepsize_7_0_defaultreset Access: R/W RET DFUSE STRAP Fractional value for one SSC frequency step.
	7:0	**cfg_i_feedfwrdgain_7_0** Default Value: 1Ch cfg_i_feedfwrdgain_7_0_defaultreset Access: R/W RET DFUSE STRAP Feedforwad gain for fractional mode/SSC mode PLL This setting is needed whenever PLL is configured in fractional mode or configured for SSC clock generation.

DKLP_PLL1_BIAS

DKLP_PLL1_BIAS		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
BIAS Register		
DWord	**Bit**	**Description**
0	31	**cfg_i_tdc_fine_res** Default Value: 1b cfg_i_tdc_fine_res_defaultreset Access: R/W RET DFUSE STRAP TDC fine resolution select 0: Coarse resolution / 8 1: Coarse resolution / 4
	30	**cfg_i_fracnen_h** Default Value: 1b cfg_i_fracnen_h_defaultreset Access: R/W RET DFUSE STRAP Enables fractional modulator. For SSC, this bit needs to be set to '1', even though it starts with integer division ratio. This is not part of direct pin IF. This is only integer to integer with BW optimization across all supported interger divisions or fractional to fractional with small PPM changes. Integer to fractional or vice versa is not supported.
	29:24	**cfg_i_fbdiv_frac_21_16** Default Value: 1Eh cfg_i_fbdiv_frac_21_16_defaultreset Access: R/W RET DFUSE STRAP Fractional Modulator settings
	23:16	**cfg_i_fbdiv_frac_15_8** Default Value: 00h cfg_i_fbdiv_frac_15_8_defaultreset Access: R/W RET DFUSE STRAP Fractional Modulator settings
	15:8	**cfg_i_fbdiv_frac_7_0** Default Value: 00h cfg_i_fbdiv_frac_7_0_defaultreset Access: R/W RET DFUSE STRAP Fractional Modulator settings
	7:0	**cfg_i_sscinj_stepsize_7_0** Default Value: 00h cfg_i_sscinj_stepsize_7_0_defaultreset Access: R/W RET DFUSE SSCInjection Step Size [7:0]

DKLP_PLL1_DIV0

DKLP_PLL1_DIV0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
DIV0 Register		
DWord	**Bit**	**Description**
0	31:30	**cfg_i_truelock_criteria_1_0** Default Value: 01b cfg_i_truelock_criteria_1_0_defaultreset Access: R/W RET DFUSE STRAP True lock indicator criteria. After early lock generation, external PLL lock indicator asserted high when phase error is less than threshold value for 00: 16 consecutive cycles 01: 32 consecutive cycles 10: 48 consecutive cycles 11: 64 consecutive cycles
	29:28	**cfg_i_earlylock_criteria_1_0** Default Value: 01b cfg_i_earlylock_criteria_1_0_defaultreset Access: R/W RET DFUSE STRAP Early lock indicator criteria. Early PLL lock indicator asserted high when phase error is less than threshold value for 00: 16 consecutive cycles 01: 32 consecutive cycles 10: 48 consecutive cycles 11: 64 consecutive cycles
	27:25	**cfg_i_afc_startup_2_0** Default Value: 000b cfg_i_afc_startup_2_0_defaultreset Access: R/W RET DFUSE STRAP This is for AFC start point. NOTE: i_afc_startup[2] - DW4, byte17, bit[0] 000: fine = 511 001: fine = 639 (+128) 010: fine = 767 (+256) 011: fine = 895 (+384) 100: NA 101: fine = 127 (-384) 110: fine = 255 (-256) 111: fine = 383 (-128)
	24	**cfg_i_divretimeren** Default Value: 0b cfg_i_divretimeren_defaultreset Access: R/W RET DFUSE STRAP Retiming of feedback clock
	23:21	**cfg_i_gainctrl_2_0** Default Value: 001b cfg_i_gainctrl_2_0_defaultreset Access: R/W RET DFUSE STRAP Adjustable gain for loop filter. Both coefficients shifted right by gainctrl before lock

DKLP_PLL1_DIV0

	20:16	**cfg_i_int_coeff_4_0** Default Value: 07h cfg_i_int_coeff_4_0_defaultreset Access: R/W RET DFUSE STRAP integral coeff. = 2^(-int_coeff), tageting up to 2^-11
	15:12	**cfg_i_prop_coeff_3_0** Default Value: 3h cfg_i_prop_coeff_3_0_defaultreset Access: R/W RET DFUSE STRAP proportional coeff. = 2^(-prop_coeff+1)
	11:8	**cfg_i_fbprediv_3_0** Default Value: 2h cfg_i_fbprediv_3_0_defaultreset Access: R/W RET DFUSE STRAP predivider ratio 0000,0001 : reserved 0010: /2 0100: /4 0011: reserved Rest: reserved For external IF pin use case with
	7:0	**cfg_i_fbdiv_intgr_7_0** Default Value: 69h cfg_i_fbdiv_intgr_7_0_defaultreset Access: R/W RET DFUSE STRAP Feedback divider post division (M2)

DKLP_PLL1_DIV1

DKLP_PLL1_DIV1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
DIV1 Register		
DWord	**Bit**	**Description**
0	31	**cfg_i_bw_ampmeas_window** Default Value: 0b cfg_i_bw_ampmeas_window_defaultreset Access: R/W RET DFUSE STRAP 0: 10 modulation cycles of averaging for mplitude measurement 1: 20 modulation cycles of averaging for mplitude measurement
	30:29	**cfg_i_bias_calib_stepsize_1_0** Default Value: 00b cfg_i_bias_calib_stepsize_1_0_defaultreset Access: R/W RET DFUSE STRAP Bias Calibration Step Size during linear search 00: 1 01: 2 10: 3 11: 4
	28:24	**cfg_i_ctrim_4_0** Default Value: 0Ch cfg_i_ctrim_4_0_defaultreset Access: R/W RET DFUSE STRAP ip74pppxdkltypecfamilyew_Cap trimming for irefout. This also has refclock dependency. Current default is for 24MHz.
	23	**cfg_i_fastlock_internal_reset** Default Value: 1b cfg_i_fastlock_internal_reset_defaultreset Access: R/W RETCLR DFUSE STRAP clears internal fastlock memory so that next cold start will do both TDC and AFC calibration instead of fast lock. NOTE: this does not clear the i_fastlock_en_h register bit, clears functional register and self-clears Formerly, i_bbthresh[3] (no longer strap)
	22:21	**cfg_i_bias_r_programability_1_0** Default Value: 10b cfg_i_bias_r_programability_1_0_defaultreset Access: R/W RET DFUSE STRAP [1:0] : Bias Filter R programmability - note mapped to bias_bonus[1:0]
	20:16	**cfg_i_ireftrim_4_0** Default Value: 1Ch cfg_i_ireftrim_4_0_defaultreset Access: R/W RET DFUSE STRAP Output current trim . Mirror ratio is changed based on constant current

DKLP_PLL1_DIV1

		requirement. SInce it is refclock dependent, needs to be reconfigurable. i_ireftrim[4:0] - 38.4/19.2 MHz - 5'h1C i_ireftrim[4:0] - 25/100 MHz = 5'h18 Step size: 20 uA
	15	**cfg_i_biasfilter_en_delay** Default Value: 1b cfg_i_biasfilter_en_delay_defaultreset Access: R/W RET DFUSE 0: Filter enabled even before pll enabled 1: Filter enable is delay until lock acquisition This bit is sensitive only when i_bias_filter_en is set (bit3)
	14	**cfg_i_bias_filter_en** Default Value: 1b cfg_i_bias_filter_en_defaultreset Access: R/W RET DFUSE 0: Disables bias filter 1: Enables bias filter
	13	**cfg_i_biascal_en_h** Default Value: 0b cfg_i_biascal_en_h_defaultreset Access: R/W RET DFUSE STRAP Bias Calibration Signal. Bias cal should be disable when override DCO coarse code.
	12	**cfg_i_dcodither_config** Default Value: 0b cfg_i_dcodither_config_defaultreset Access: R/W RET DFUSE Whenever we have binary weighted MFC ip74pppxdkltypecfamilyew_cap, this should be set to 1'b0. Ex: i_dcofine_resolution = 1'b0. For this case, this should be set to 1'b0. 0: No floating dither 1: Floating dither (511+Nobinary - Floating dither)
	11:8	**cfg_i_lockthresh_3_0** Default Value: 5h cfg_i_lockthresh_3_0_defaultreset Access: R/W RET DFUSE STRAP Digital lock detect threshold, the PLL will generate plllockout when the phase error detected by TDC is within lockthresh number of TDC counts for 64 consecutive cycles 5*16 (TDC Code) = 80 (16 is internally hard coded scalar value) - This setting isin middle of coarse range
	7:0	**cfg_i_tdctargetcnt_7_0** Default Value: 22h cfg_i_tdctargetcnt_7_0_defaultreset Access: R/W RET DFUSE STRAP TDC tristate buffer calibration ip74pppxdkltypecfamilyew_counter value. Delay line loop oscillation is counted over two refclk cycles. This is used for TDC coarse code calibration

DKLP_PLL1_FRAC_LOCK

DKLP_PLL1_FRAC_LOCK

Register Space: MMIO: 0/2/0

Size (in bits): 32

FRAC_LOCK Register

DWord	Bit	Description
0	31:30	**cfg_i_cml2cmosbonus_1_0** Default Value: 00b cfg_i_cml2cmosbonus_1_0_defaultreset Access: R/W RET DFUSE STRAP NOTE: These bits are ported to anatop bit [0] mapped to cml2cmosbonus[1] port - o_plllc_ck_dcocmosclkp_ana disable '1' = o_plllc_ck_dcocmosclkp_ana is disabled (ie for MG B0) '0' = both phases of the cmos clock ip74pppxdkltypecfamilyew_toggle at the interface bit[1] mapped to cml2cmosbonus[2] port - available
	29:27	**cfg_i_bb_gain2_2_0** Default Value: 000b cfg_i_bb_gain2_2_0_defaultreset Access: R/W RET DFUSE STRAP bb_gain2[2:0] : BB gain for second BB range; expect bb_gain2 >= bb_gain1
	26:24	**cfg_i_bb_gain1_2_0** Default Value: 000b cfg_i_bb_gain1_2_0_defaultreset Access: R/W RET DFUSE STRAP BB gain for first BB range
	23	**cfg_i_fastlock_en_h** Default Value: 0b cfg_i_fastlock_en_h_defaultreset Access: R/W RET DFUSE STRAP Enable FLL based AFC; this replaces binary search based AFC. FLL based AFC faster than binary search
	22:19	**cfg_i_fllafc_gain_3_0** Default Value: 8h cfg_i_fllafc_gain_3_0_defaultreset Access: R/W RET DFUSE STRAP Initial FLL gain that decrements down to 1 every refclk cycle in the beginning of FLL based AFC
	18:16	**cfg_i_fllafc_lockcnt_2_0** Default Value: 100b cfg_i_fllafc_lockcnt_2_0_defaultreset Access: R/W RET DFUSE STRAP Number of refclk cycles for FLL lock after gain is reduced to 1

DKLP_PLL1_FRAC_LOCK		
	15:8	**cfg_i_max_cselafc_7_0** Default Value: 7Fh cfg_i_max_cselafc_7_0_defaultreset Access: R/W RET DFUSE STRAP Max. AFC code for a given DCO
	7:0	**cfg_i_init_cselafc_7_0** Default Value: 6Ah cfg_i_init_cselafc_7_0_defaultreset Access: R/W RET DFUSE STRAP Initial AFC code for FLL AFC; apply approximate AFC, and starting at closer frequency helps fast/accurate calibration

DKLP_PLL1_LF

DKLP_PLL1_LF		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
LF Register		
DWord	**Bit**	**Description**
0	31	**cfg_i_afc_divratio** Default Value: 0b cfg_i_afc_divratio_defaultreset Access: R/W RET DFUSE STRAP 0: DCO/4 (prediv: 2, Mdithdiv: 2) 1: DCO/8 (prediv: 4 Mdithdiv: 2)
	30:29	**cfg_i_plllock_sel_1_0** Default Value: 00b cfg_i_plllock_sel_1_0_defaultreset Access: R/W RET DFUSE select between lockdetect-based plllock or ip74pppxdkltypecfamilyew_counter based plllock 11: Sticky lock 10: ip74pppxdkltypecfamilyew_Counter-based 01: Lock Detection + ip74pppxdkltypecfamilyew_Counter 00: Lock Detection
	28:24	**cfg_i_bwphase_4_0** Default Value: 00h cfg_i_bwphase_4_0_defaultreset Access: R/W RET DFUSE STRAP Phase amplitude for bandwidth measurement
	23:21	**cfg_i_ft_mode_sel_2_0** Default Value: 010b cfg_i_ft_mode_sel_2_0_defaultreset Access: R/W RET DFUSE STRAP ftmodesel[2:0] : 9b vs 10b finetune selection, lsb0 tuning 3'b000 : 10b Nom -> Dither = LSB0, LSB0 = LSB0 3'b001 : 10b+ -> Dither = LSB+, LSB0 = LSB0+ 3'b010 : 10b- -> Dither = LSB-, LSB0 = LSB0- 3'b011 : 10b DNL -> Dither = LSB01, LSB0 = LSB0+ 3'h1xx: 9b -> Dither = LSB1, LSB0 = N/A
	20:19	**cfg_i_bw_mode_1_0** Default Value: 00b cfg_i_bw_mode_1_0_defaultreset Access: R/W RET DFUSE STRAP 00: No measurement 01: BW measurement 10: BW calibration up to +1 direction 11: BW calibration up to +2 direction
	18:16	**cfg_i_bw_upperbound_2_0** Default Value: 000b cfg_i_bw_upperbound_2_0_defaultreset Access: R/W RET DFUSE STRAP Upper bound setting for BW 000: 0.1MHz 001: 1 MHz 010: 2 MHz 011: 5 MHz 100: 6 MHz 101: 8 MHz 110: 16 MHz 111: 25MHz

DKLP_PLL1_LF

	15:13	**cfg_i_bw_lowerbound_2_0** Default Value: 000b cfg_i_bw_lowerbound_2_0_defaultreset Access: R/W RET DFUSE STRAP Lower bound setting for BW 000: 0.1MHz 001: 1 MHz 010: 2 MHz 011: 5 MHz 100: 6 MHz 101: 8 MHz 110: 16 MHz 111: 25MHz
	12:9	**cfg_i_dcoamp_3_0** Default Value: 0h cfg_i_dcoamp_3_0_defaultreset Access: R/W RET DFUSE Amplitude override value for DCO, 0000 is min amplitude, 1111 is max amplitude, applied when i_dcoampovrden_h is high. 4bit is left shifted to generate 6bit code (x4)
	8	**cfg_i_dcoampovrden_h** Default Value: 0b cfg_i_dcoampovrden_h_defaultreset Access: R/W RET DFUSE DCO amplitude override enable: 0 DCO amplitude set internally (default) 1 DCO amplitude is set by i_dcoamp[3:0]
	7:5	**cfg_i_bbthresh2_2_0** Default Value: 000b cfg_i_bbthresh2_2_0_defaultreset Access: R/W RET DFUSE STRAP threshold for second (outer) BB range; expect bbthresh2 >= bbthresh1
	4:2	**cfg_i_bbthresh1_2_0** Default Value: 000b cfg_i_bbthresh1_2_0_defaultreset Access: R/W RET DFUSE STRAP threshold for first (inner) bang-bang (BB) range
	1:0	**cfg_i_tdc_offset_lock_1_0** Default Value: 00b cfg_i_tdc_offset_lock_1_0_defaultreset Access: R/W RET DFUSE TDC Offset during lock for integer mode

DKLP_PLL1_TDC_COLDST_BIAS

DKLP_PLL1_TDC_COLDST_BIAS		
Register Space: MMIO: 0/2/0 Size (in bits): 32		
TDC_COLDST_BIAS Register		
DWord	**Bit**	**Description**
0	31:29	**cfg_i_cloadctrllext_4_2** Default Value: 000b cfg_i_cloadctrllext_4_2_defaultreset Access: R/W RET VFUSE cload control override for H Vernier line. This is applied when i_tdccalexten_h = 1
	28:24	**cfg_i_tribufctrlext_4_0** Default Value: 00h cfg_i_tribufctrlext_4_0_defaultreset Access: R/W RET VFUSE Tribufctrl control override. This is applied when i_tdccalexten_h = 1
	23:16	**cfg_i_dcocoarse_7_0** Default Value: 00h cfg_i_dcocoarse_7_0_defaultreset Access: R/W RET VFUSE STRAP DCO coarse tune frequency value, when dcocoarse_ovrd_h is '1', this input is used to override the value calculated from the Automatic Frequency Calibration (AFC) block
	15:8	**cfg_i_sscstepsize_7_0** Default Value: 0Fh cfg_i_sscstepsize_7_0_defaultreset Access: R/W RET DFUSE STRAP Fractional value for one SSC frequency step.
	7:0	**cfg_i_feedfwrdgain_7_0** Default Value: 23h cfg_i_feedfwrdgain_7_0_defaultreset Access: R/W RET DFUSE STRAP Feedforwad gain for fractional mode/SSC mode PLL This setting is needed whenever PLL is configured in fractional mode or configured for SSC clock generation.

DKLP_PMD_LANE_ANA_CSR_DIG_LFPS_CAL

DKLP_PMD_LANE_ANA_CSR_DIG_LFPS_CAL

Register Space: MMIO: 0/2/0

Size (in bits): 32

PMD_LANE_ANA_CSR::DIG_LFPS_CAL

DWord	Bit	Description
0	31:23	**cfg_reserved1** Default Value: 000h cfg_reserved1_defaultreset Access: R/W PMD_LANE_ANA_CSR::DIG_LFPS_CAL::reserved1
	22	**cfg_filter1** Default Value: 0b cfg_filter1_defaultreset Access: R/W PMD_LANE_ANA_CSR::DIG_LFPS_CAL::filter1
	21	**cfg_osc_dfx_en** Default Value: 0b cfg_osc_dfx_en_defaultreset Access: R/W PMD_LANE_ANA_CSR::DIG_LFPS_CAL::osc_dfx_en
	20:19	**cfg_ck2logic_sel** Default Value: 00b cfg_ck2logic_sel_defaultreset Access: R/W PMD_LANE_ANA_CSR::DIG_LFPS_CAL::ck2logic_sel
	18	**cfg_ck_div_en** Default Value: 1b cfg_ck_div_en_defaultreset Access: R/W PMD_LANE_ANA_CSR::DIG_LFPS_CAL::ck_div_en
	17:12	**cfg_osc_trim** Default Value: 3Fh cfg_osc_trim_defaultreset Access: R/W Reserved

DKLP_PMD_LANE_ANA_CSR_DIG_LFPS_CAL

	11	**cfg_reserved** Default Value: 0b cfg_reserved_defaultreset Access: R/W added by Yoni to provide local fix.
	10:1	**cfg_amon_sel** Default Value: 000h cfg_amon_sel_defaultreset Access: R/W added by Yoni to provide local fix.
	0	**cfg_amon_puldn_en** Default Value: 1b cfg_amon_puldn_en_defaultreset Access: R/W modified by Yoni to provide local fix. originally width 24 RW, reset value - 800FDA

DKLP_PMD_LANE_CDR_CDR_SAVE_RESTORE1

DKLP_PMD_LANE_CDR_CDR_SAVE_RESTORE1		
Register Space: MMIO: 0/2/0 Size (in bits): 32		
PMD_LANE_CDR::CDR_SAVE_RESTORE1		
DWord	**Bit**	**Description**
0	31:28	**cfg_reserved1** Default Value: 0h cfg_reserved1_defaultreset Access: R/W PMD_LANE_CDR::CDR_SAVE_RESTORE1::reserved1
	27:19	**cfg_dco_coarse_backup_sr** Default Value: 000h cfg_dco_coarse_backup_sr_defaultreset Access: R/W this is a save restore register, to save last dco calibration code, for reuse in warm boot.
	18:10	**cfg_dco_coarse_init** Default Value: 098h cfg_dco_coarse_init_defaultreset Access: R/W PMD_LANE_CDR::CDR_SAVE_RESTORE1::dco_coarse_init
	9:5	**cfg_kp_datalock_acq** Default Value: 16h cfg_kp_datalock_acq_defaultreset Access: R/W PMD_LANE_CDR::CDR_SAVE_RESTORE1::kp_datalock_acq
	4:0	**cfg_ki_datalock_acq** Default Value: 18h cfg_ki_datalock_acq_defaultreset Access: R/W PMD_LANE_CDR::CDR_SAVE_RESTORE1::ki_datalock_acq

DKLP_PMD_LANE_SUSWELL_LFPS_SPARE

DKLP_PMD_LANE_SUSWELL_LFPS_SPARE		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
PMD_LANE_SUSWELL::LFPS_SPARE		
DWord	**Bit**	**Description**
0	31:17	**cfg_reserved1** Default Value: 0000h cfg_reserved1_defaultreset Access: R/W Reserved
	16	**cfg_enable_squelch_periodic_cal** Default Value: 0b cfg_enable_squelch_periodic_cal_defaultreset Access: R/W PMD_LANE_SUSWELL::LFPS_SPARE::enable_squelch_periodic_cal
	15:11	**cfg_rx_lfps_delay_filter_depth** Default Value: 04h cfg_rx_lfps_delay_filter_depth_defaultreset Access: R/W rx_lfps_delay_filter_depth
	10:9	**cfg_counter_init** Default Value: 00b cfg_counter_init_defaultreset Access: R/W PMD_LANE_SUSWELL::LFPS_SPARE::counter_init
	8	**cfg_1ms_timer_en** Default Value: 0b cfg_1ms_timer_en_defaultreset Access: R/W PMD_LANE_SUSWELL::LFPS_SPARE::1ms_timer_en
	7	**cfg_rx_lfps_delay_timer_en** Default Value: 0b cfg_rx_lfps_delay_timer_en_defaultreset Access: R/W PMD_LANE_SUSWELL::LFPS_SPARE::rx_lfps_delay_timer_en
	6	**cfg_mipi_dif_pn_hs_filt_dis** Default Value: 0b cfg_mipi_dif_pn_hs_filt_dis_defaultreset Access: R/W PMD_LANE_SUSWELL::LFPS_SPARE::mipi_dif_pn_hs_filt_dis
	5	**cfg_ecsr_force_val_pmd_txdorxdetect_tx1** Default Value: 0b cfg_ecsr_force_val_pmd_txdorxdetect_tx1_defaultreset Access: R/W PMD_LANE_SUSWELL::LFPS_SPARE::ecsr_force_val_pmd_txdorxdetect_tx1

DKLP_PMD_LANE_SUSWELL_LFPS_SPARE		
	4	**cfg_ecsr_force_en_pmd_txdorxdetect_tx1** Default Value: 0b cfg_ecsr_force_en_pmd_txdorxdetect_tx1_defaultreset Access: R/W PMD_LANE_SUSWELL::LFPS_SPARE::ecsr_force_en_pmd_txdorxdetect_tx1
	3:0	**cfg_idig_lfps_spare** Default Value: 0h cfg_idig_lfps_spare_defaultreset Access: R/W PMD_LANE_SUSWELL::LFPS_SPARE::idig_lfps_spare

DKLP_PMD_LANE_SUSWELL_TX1_RCV_DETECT_CTRL

DKLP_PMD_LANE_SUSWELL_TX1_RCV_DETECT_CTRL		
Register Space: MMIO: 0/2/0 Size (in bits): 32		
PMD_LANE_SUS_LN0/1::TX1_RCV_DETECT_CTRL		
DWord	**Bit**	**Description**
0	31	**reserved1** Access: RO
	30	**ecsr_tx_rcv_detect_shunt_low_tx1** HASH(0x18841C0)
	29	**ecsr_tx_rcv_detect_shunt_high_tx1** HASH(0x18CD070)
	28:22	**ecsr_tx_rcv_detect_cell_hz_tx1** Default Value: 0x7F HASH(0x188F060)
	21	**Reserved**
	20:19	**ecsr_tx_rcv_detect_fltr_sel_tx1** Default Value: 0x2 wait state during sampling select beween 1,2,3 wait states
	18	**ecsr_tx_rcv_detect_status_force_val_tx1** force detection value
	17	**ecsr_tx_rcv_detect_status_force_en_tx1** force detection value
	16:12	**ecsr_tx_rcv_detect_fsm_force_val_tx1** force rcv detect fsm state
	11	**ecsr_tx_rcv_detect_fsm_force_en_tx1** force rcv detect fsm
	10:4	**ecsr_tx_rcv_detect_cnt_limit_tx1** Default Value: 0x5 count limit of valid to sample tx detect signal
	3	**ecsr_tx_rcv_detect_p_on_high_en_tx1** take detect on high level
	2	**ecsr_tx_rcv_detect_n_on_high_en_tx1** HASH(0x1AC4C40)
	1	**ecsr_tx_rcv_detect_inv_p_ana_sig_tx1** invert analog input
	0	**ecsr_tx_rcv_detect_inv_n_ana_sig_tx1** HASH(0x197E1B0)

DKLP_TX_GLUE_TX_DWORD14

DKLP_TX_GLUE_TX_DWORD14		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
TX related, imported from PMD_LANE_MISC		
DWord	**Bit**	**Description**
0	31:25	**cfg_reserved_1** Default Value: 00h cfg_reserved_1_defaultreset Access: R/W Tx_Glue::tx_dword14::reserved_1
	24:14	**cfg_ecsr_tx_clk_dccdacctrl_offset** Default Value: 000h cfg_ecsr_tx_clk_dccdacctrl_offset_defaultreset Access: R/W reserved
	13	**cfg_ecsr_tx_dccdacenrbp** Default Value: 0b cfg_ecsr_tx_dccdacenrbp_defaultreset Access: R/W if set, chooses coarse step (x2) of dccdacctrl
	12:0	**cfg_ecsr_tx_clk_dccdacctrl** Default Value: 148Ch cfg_ecsr_tx_clk_dccdacctrl_defaultreset Access: R/W ecsr_tx_clk_dccdacctrl

DKLP_TX2_PMD_LANE_MISC_TX1_SPARE

DKLP_TX2_PMD_LANE_MISC_TX1_SPARE		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
TX2_PMD_LANE_MISC::TX1_SPARE		
DWord	**Bit**	**Description**
0	31:8	**cfg_reserved1** Default Value: 000000h cfg_reserved1_defaultreset Access: R/W Reserved
	7:0	**cfg_dig_tx_spare_tx1** Default Value: 0Fh cfg_dig_tx_spare_tx1_defaultreset Access: R/W Spare Config Bits for Tx

DMA Address Register 0 High

DMA_ADDR_0_HIGH - DMA Address Register 0 High

Register Space: MMIO: 0/2/0
Access: R/W
Size (in bits): 32

The lower 32 bits of the first DMA address register.

Programming Notes

This register is saved in the power context

DWord	Bit	Description
0	31:25	**Reserved** Access: RO Format: MBZ
	24	**Access Region** 0h: Main Memory **[Default]** 1h: SRAM
	23:21	**Reserved** Access: RO Format: MBZ
	20:16	**Address Space** 0h: Normal Memory Access **[Default]** – Use the Per-process GTT if enabled, otherwise uses the Global GTT 7h: WOPCM Access 8h: Global GTT Memory Access **Programming Notes**: This field is ignored if the selected Access Region is SRAM.
	15:0	**Address Upper DWord** Format: GraphicsAddress[47:32]

DMA Address Register 0 Low

DMA_ADDR_0_LOW - DMA Address Register 0 Low		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
The lower 32 bits of the first DMA address register.		
Programming Notes		
This register is saved in the power context		

DWord	Bit	Description
0	31:0	**Address** Format: GraphicsAddress[31:0] This field contains an offset into the associated space or a Graphics Address.

DMA Address Register 1 High

DMA_ADDR_1_HIGH - DMA Address Register 1 High

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

The lower 32 bits of the second DMA address register.

Programming Notes

This register is saved in the power context

DWord	Bit	Description
0	31:25	**Reserved** Access: RO Format: MBZ
	24	**Access Region** 0h: Main Memory **[Default]** 1h: SRAM
	23:21	**Reserved** Access: RO Format: MBZ
	20:16	**Address Space** 0h: Normal Memory Access **[Default]** 7h: WOPCM Access 8h: Global GTT Memory Access **Programming Notes** This field is ignored if the selected Access Region is SRAM.
	15:0	**Address Upper DWord** Format: GraphicsAddress[47:32]

DMA Address Register 1 Low

DMA_ADDR_1_LOW - DMA Address Register 1 Low	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

The lower 32 bits of the second DMA address register.

Programming Notes
This register is saved in the power context

DWord	Bit	Description
0	31:0	**Address** Format: GraphicsAddress[31:0] This field contains an offset into the associated space or a Graphics Address.

DMA Configuration

DMA_CFG - DMA Configuration

Register Space: MMIO: 0/2/0
Access: R/W
Size (in bits): 32

Programming Notes
This register is saved in the power context. SW shall set bits[1:0] = '00'
Programming of Bits[1:0] are only applies to explicit DMA operation's triggered by FW Or HOST.

DWord	Bit	Description
0	31:3	**Reserved** Access: RO Format: MBZ
	2	**C6 Restore Control** 0h – Active C6 Restore **[Default]** – SRAM contents will be restored as part of the C6 restore. 1h – Lazy C6 Restore – SRAM contents will only be restored on the first active write to the GuC. **Programming Notes**: Hardware restores SRAM contents coming out of RC6 only after a valid kernel has been loaded. If host SW uses DMA to load multiple entities (e.g micro-apps and then kernel), it shall inhibit RC6 entry till the kernel is loaded.
	1	**Main Memory To SRAM Performance Optimization Mode** Mode selection control when bit 0 is set. 0h – **[Default]** – If bit 0 is set, stream writes will be done only uOS/uApp being DMA-ed. 1h – If bit 0 is set, stream writes will be done for all DMA to SRAM. Note that if MinIA is active, this setting can cause writes to be dropped.
	0	**Main Memory To SRAM Performance Optimization** 0h – Reorder Writes **[Default]** – Reorder DMA writes to be in order when DMA'ing from GAM to SRAM. 1h – Stream – Stream DMA writes as the read data comes in. Use the 'tag' field in the read return as the address to SRAM.

DMA Control

DMA_CTRL - DMA Control

Register Space: MMIO: 0/2/0
Access: R/W
Size (in bits): 32

Programming Notes

This register is saved in the power context
The DMA hardware performs several checks of the parameters provided in the DMA registers before allowing DMA to proceed. These checks are documented in the GuC DMA description PRM section.
If these checks fail then DMA will not start (i.e: it will appear that writes to this register were dropped)

DWord	Bit	Description
0	31:16	**Write Enable Mask** Format: Mask[15:0] A write enable mask for bits 15:0
	15	**Catastrophic Error Encountered during DMA** This bit is set if the DMA transfer encountered a catastrophic fault during its operation. Bit is reset when the next DMA transfer is initiated.
	14:11	**Reserved** Access: RO Format: MBZ
	10	**Last DFX DMA** Allow for SHA to preserve signature between DMA transfers - This feature should be enabled only when DFX PRIMARY Control bit C068[7] is 1 . Dfx now needs to indicate to GUC which is the last DMA in its set of DMA transfers, whose SHA is being accumulated. Dfx will program all these DMAs as uApp at different destination addresses. When Dfx sets this bit along with C314[0], it indicates the last DMA in Dfx's set of DMA transfers.
	9	**Reserved**
	8	**BOOTROM in WOPCM** Allows host to DMA BootROM code to WOPCM region.BootROM code shall be located at offset 0 in GuC's WOPCM regionMode available with fuse_auth_disable == '1'
	7:6	**uApp Index** Specific AppID for one of the 2 uApps, for storing into WOPCM area as destination. Will be only valid when WOPCM is the Destination. Value: [0,1] Name:
	5	**uApp Move** Whether the uApp (index specified the "uApp Index" field) is being moved to WOPCM.

DMA_CTRL - DMA Control

		Programming Notes Must be set only when the DMA is moving uApp Code, and not uOS code.
	4	**uOS Move** Whether the uOS is being moved to WOPCM. **Programming Notes** Must be set only when the DMA is moving uOS Code, and not uApp code.
	3	**DMA Completion Interrupt Enable** Whether an interrupt will be generated when the DMA copy is complete.
	2	**DMA Completion Interrupt Routing** Where the DMA completion interrupt will be routed. **Value** / **Name** / **Description** 0h / Host **[Default]** / Interrupt will be routed to the host CPU. 1h / Reserved / Reserved
	1	**DMA Source** Which DMA address register contains the source of the DMA transfer. **Value** / **Name** / **Description** 0h / Register 0 **[Default]** / Address register 0 is the source of the DMA transfer, address register 1 is the destination. 1h / Register 1 / Address register 1 is the source of the DMA transfer, address register 0 is the destination.
	0	**Start DMA Transfer** This bit should be programmed last since it will trigger the DMA transfer. Hardware will clear this bit once DMA completes.

DMA Copy Size

DMA_COPY_SIZE - DMA Copy Size

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

Size of the DMA copy.

Programming Notes

The following rules hold good regarding the uOS and uApp size:

- uOS and uApps sizes are always divisible by 64-bytes.
- uOS and uApps hash value take into account any padding needed for uOS/uApps divisible by 64-byte requirement.

This register is saved in the power context

DWord	Bit	Description
0	31:0	**DMA Transfer Size** **Description** Size of the DMA transfer, in bytes. For smaller transfers SW should use the MinuteIA core directly (DMA programming overhead is higher than copying the data using the MinuteIA core. Approximate inflection point would be ~ 256 bits.)

DMA Global MicroController WOPCM Offset

DMA_GUC_WOPCM_OFFSET - DMA Global MicroController WOPCM Offset

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

Programming Notes

The WOPCM space referenced by this base address/offset is not IA visible space. uKernel/uApps are stored here.

This register is required to be programmed before the DMA engine can be invoked to load uKernel (access WOPCM memory). The register is locked after the write - (Write Once).

This register is saved in the power context

DWord	Bit	Description
0	31:14	**GuC WOPCM Offset** Format: WOPCMBaseOffset[31:14] This bit field defines the "GuC WOPCM Start Address" as an 16KB alligned offset from "WOPCM Base".
	13:2	**Reserved** Access: RO Format: MBZ
	1	**Reserved**
	0	**Offset Valid** Reports whether the "GuC WOPCM Offset" is valid and loaded. HW sets this bit when the register is programmed with the GuC WOPCM Offset value.

DMA Microkernel Base

DMA_UOS_BASE - DMA Microkernel Base

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32

Programming Notes

This register is saved in the power context

DWord	Bit	Description
0	31:6	**Microkernel Base Offset** Format: WOPCMBaseOffset[31:6] This is the value that was programmed by the host. The value is relative to the SRAM base == 0x0 offset in WOPCM.
	5:4	**State of Microkernel** This field is valid only if the "Valid" field (bit 0) is set Value / Name / Description: 0h / In WOPCM **[Default]** / The microkernel is in WOPCM. 1h / In WOPCM and SRAM / The microkernel is in WOPCM and SRAM. 2h / In WOPCM and SRAM and partially SRAM / The microkernel is in WOPCM and partially in SRAM.
	3:1	**Reserved** Access: RO Format: MBZ
	0	**Valid** Format: Boolean Whether a microkernel has been loaded into WOPCM and the hash check has passed.

DMA Microkernel Top

DMA_UOS_TOP - DMA Microkernel Top

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32

Programming Notes
This register is saved in the power context

DWord	Bit	Description
0	31:6	**Microkernel Top Address** Format: WOPCMBaseOffset[31:6] This value is derived from the initial uOS DMA programming to WOPCM by the host. The value is relative to the SRAM base == 0x0 offset in WOPCM. This offset will point to the beginning of the last cacheline DMA'd.
	5:0	**Reserved** Access: RO Format: MBZ

DMA Protected Range

DPR_0_0_0_PCI - DMA Protected Range

Register Space:	PCI: 0/0/0
Size (in bits):	32
Address:	0005Ch

DMA protected range register.

DWord	Bit	Description
0	31:20	**Top of DPR** Default Value: 000000000000b Access: RO _Custom_GTIReset: BUS RESERVED for DEVICE 2 shadow copy. Device 0 maintains these register bits. Top address + 1 of DPR. This is the base of TSEG. Bits 19:0 of the BASE reported here are 0x0_0000.
	19:12	**Reserved** Access: RO Format: MBZ
	11:4	**DPRSIZE** Default Value: 00000000b Access: R/W Lock _Custom_GTIReset: BUS This is the size of memory, in MB, that will be protected from DMA accesses. A value of 0x00 in this field means no additional memory is protected. The maximum amount of memory that will be protected is 255MB. The amount of memory reported in this field will be protected from all DMA accesses, including translated CPU accesses and graphics. The top of the protected range is the BASE of TSEG -1. Note: If TSEG is not enabled, then the top of this range becomes the base of stolen graphics, or ME stolen space or TOLUD, whichever would have been the location of TSEG, assuming it had been enabled. The DPR range works independently of any other range, including the NoDMA.TABLE protection or the PMRC checks in VTd, and is done post any VTd translation or LT NoDMA lookup. Therefore incoming cycles are checked against this range after the VTd translation and faulted if they hit this protected range, even if they passed the VTd translation or were clean in the NoDMA lookup. All the memory checks are OR'ed with respect to NOT being allowed to go to memory. So if either PMRC, DPR, NoDMA table lookup, NoDMA.TABLE.PROTECT OR a VTd translation disallows the cycle, then the cycle is not allowed to go to memory. Or in other words, all of the above checks must pass before a cycle is allowed to DRAM.
	3	**Reserved** Access: RO Format: MBZ

DPR_0_0_0_PCI - DMA Protected Range

	2	**EPM** Default Value: 0b Access: R/W Lock _Custom_GTIReset: BUS This field controls DMA accesses to the DMA Protected Range (DPR) region. 0: DPR is disabled. 1: DPR is enabled. All DMA requests accessing DPR region are blocked. Dev 0 HW reports the status of DPR enable/disabled through the PRS field in this register.
	1	**Protected Range Status** Default Value: 0b Access: RO _Custom_GTIReset: BUS RESERVED for DEVICE 2 shadow copy. Dev0 maintains this bit. This field indicated the status of DPR. 0: DPR protection disabled. 1: DPR protection enabled.
	0	**Lock** Default Value: 0b Access: R/W Key Lock _Custom_GTIReset: BUS All bits which may be updated by SW in this register are locked down when this bit is set.

Doorbell Control

DOORBELL_CTRL - Doorbell Control

Register Space: MMIO: 0/2/0
Access: R/W
Size (in bits): 32

Programming Notes
This register is saved in the power context

DWord	Bit	Description
0	31:25	**Reserved** Access: RO Format: MBZ
	24:16	**Write Enable Mask** Format: Mask[8:0] A write enable mask for bits 8:0
	15:9	**Reserved** Access: RO Format: MBZ
	8	**Interrupt Routing** Where interrupts will be routed. 0h – Host **[Default]** – Interrupts will be routed to the host CPU. 1h – Reserved – Reserved
	7	**Doorbell Rung in Group 7** Format: Boolean Whether a doorbell has rung in group 7 (doorbells 224 through 255). This field is written by GTI, or the host CPU. It is cleared by the GuC interrupt hardware, and is read by the host CPU. 0h – None **[Default]** – No doorbell in the group was rung 1h – Doorbell Rung – A doorbell in the group was rung.
	6	**Doorbell Rung in Group 6** Format: Boolean Whether a doorbell has rung in group 6 (doorbells 192 through 223). This field is written by GTI, or the host CPU. It is cleared by the GuC interrupt hardware, and is read by the host CPU. 0h – None **[Default]** – No doorbell in the group was rung 1h – Doorbell Rung – A doorbell in the group was rung.

DOORBELL_CTRL - Doorbell Control

5 — Doorbell Rung in Group 5

Format:	Boolean

Whether a doorbell has rung in group 5 (doorbells 160 through 191). This field is written by GTI, or the host CPU. It is cleared by the GuC interrupt hardware, and is read by the host CPU..

Value	Name	Description
0h	None **[Default]**	No doorbell in the group was rung
1h	Doorbell Rung	A doorbell in the group was rung.

4 — Doorbell Rung in Group 4

Format:	Boolean

Whether a doorbell has rung in group 4 (doorbells 128 through 159). This field is written by GTI, or the host CPU. It is cleared by the GuC interrupt hardware, and is read by the host CPU.

Value	Name	Description
0h	None **[Default]**	No doorbell in the group was rung
1h	Doorbell Rung	A doorbell in the group was rung.

3 — Doorbell Rung in Group 3

Format:	Boolean

Whether a doorbell has rung in group 3 (doorbells 96 through 127). This field is written by GTI, or the host CPU. It is cleared by the GuC interrupt hardware, and is read by the host CPU.

Value	Name	Description
0h	None **[Default]**	No doorbell in the group was rung
1h	Doorbell Rung	A doorbell in the group was rung.

2 — Doorbell Rung in Group 2

Format:	Boolean

Whether a doorbell has rung in group 2 (doorbells 64 through 95). This field is written by GTI, or the host CPU. It is cleared by the GuC interrupt hardware, and is read by the host CPU.

Value	Name	Description
0h	None **[Default]**	No doorbell in the group was rung
1h	Doorbell Rung	A doorbell in the group was rung.

1 — Doorbell Rung in Group 1

Format:	Boolean

Whether a doorbell has rung in group 1 (doorbells 32 through 63). This field is written by GTI, or the host CPU. It is cleared by the GuC interrupt hardware, and is read by the host CPU.

Value	Name	Description
0h	None **[Default]**	No doorbell in the group was rung
1h	Doorbell Rung	A doorbell in the group was rung.

DOORBELL_CTRL - Doorbell Control

	0	**Doorbell Rung in Group 0** Format: Boolean Whether a doorbell has rung in group 0 (doorbells 0 through 31). This field is written by GTI, or the host CPU. It is cleared by the GuC interrupt hardware, and is read by the host CPU. **Value** / **Name** / **Description** 0h / None **[Default]** / No doorbell in the group was rung 1h / Doorbell Rung / A doorbell in the group was rung.

Doorbell Cookie Register 0..255

DRB0..255COOK - Doorbell Cookie Register 0..255		
Register Space: MMIO: 0/2/0 Size (in bits): 32		
Address: 01A00h - 01DFCh		
DWord	**Bit**	**Description**
0	31:0	**Doorbell #0 Cookie Data** Access: R/W _Custom_GTIReset: BUS Value of the cookie data for doorbell.
..	..	..
255	31:0	**Doorbell #255 Cookie Data** Access: R/W _Custom_GTIReset: BUS Value of the cookie data for doorbell.

Doorbell IDI Register

DRBIDI1 - Doorbell IDI Register		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	01988h	
DWord	**Bit**	**Description**
0	31:10	**Reserved** Access: RO Format: MBZ
	9:8	**Doorbell Class of Service** Access: R/W _Custom_GTIReset: BUS Identifies the Class of Service (CLOS) to send with all C2UREQ initiated by doorbell unit. CLOS determines one of 4 possible uses in number of ways of LLC. Since the doorbells are used with only all 16 ways available, these registers should be programmed to match the CLOS which allocates all 16 ways. It is also expected that these values are changed once before any doorbells are armed/active and will stay static for the duration of any test. 00=> MDRB will echo CLOS of 0 with each C2UREQ - DEFAULT 01=> MDRB will echo CLOS of 1 with each C2UREQ 10=> MDRB will echo CLOS of 2 with each C2UREQ 11=> MDRB will echo CLOS of 3 with each C2UREQ
	7:0	**Reserved** Access: RO Format: MBZ

Doorbell Lower Address Register 0..255

DRB0..255REGL - Doorbell Lower Address Register 0..255

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	01000h - 017F8h

DWord	Bit	Description
0	31:6	**Address to be monitored** Access: R/W _Custom_GTIReset: BUS Address to be Monitored: Value of the cacheline address that needs to be monitored. This address has to be HPA in case of 2nd level translations are enabled, else it is GPA.
	5:3	**Reserved** Access: RO Format: MBZ
	2	**RFO request** Access: RO _Custom_GTIReset: BUS RFO request pending: This is a hardware-maintained flag that an RFO (Request for ownership) needs to be sent to BGF. This is set to 1 when Ownership pending is activated and reset to 0 when the RFO is sent to uncore via BGF. This bit is reflected to MMIO else managed within HW.
	1	**Ownership Flow Pending** Access: RO _Custom_GTIReset: BUS Ownership Flow Pending: This is a hardware-maintained flag which would indicate that corresponding flag has lost ownership and pending for RFO. This is set to 1 when RFO is needed through snoop hit or activation of valid and reset to 0 when the RFO is completed (data and response received). It is reflected to MMIO else managed within HW.
	0	**Valid** Access: R/W _Custom_GTIReset: BUS Valid: Determines whether the doorbell is active or not. As part of the register update flow, SW first need to write the upper portion of this register (i.e. 63:32) and update the lower portion along with the VALID flag which activates the monitor system. SW has to clear the contents of this register before allocating a new doorbell.
..	..	..

DRB0..255REGL - Doorbell Lower Address Register 0..255

255	31:6	**Address to be monitored** Access: R/W _Custom_GTIReset: BUS Address to be Monitored: Value of the cacheline address that needs to be monitored. This address has to be HPA in case of 2nd level translations are enabled, else it is GPA.
	5:3	**Reserved** Access: RO Format: MBZ
	2	**RFO request** Access: RO _Custom_GTIReset: BUS RFO request pending: This is a hardware-maintained flag that an RFO (Request for ownership) needs to be sent to BGF. This is set to 1 when Ownership pending is activated and reset to 0 when the RFO is sent to uncore via BGF. This bit is reflected to MMIO else managed within HW.
	1	**Ownership Flow Pending** Access: RO _Custom_GTIReset: BUS Ownership Flow Pending: This is a hardware-maintained flag which would indicate that corresponding flag has lost ownership and pending for RFO. This is set to 1 when RFO is needed through snoop hit or activation of valid and reset to 0 when the RFO is completed (data and response received). It is reflected to MMIO else managed within HW.
	0	**Valid** Access: R/W _Custom_GTIReset: BUS Valid: Determines whether the doorbell is active or not. As part of the register update flow, SW first need to write the upper portion of this register (i.e. 63:32) and update the lower portion along with the VALID flag which activates the monitor system. SW has to clear the contents of this register before allocating a new doorbell.

Doorbell Upper Address Register 0..255

DRB0..255REGU - Doorbell Upper Address Register 0..255

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 01004h - 017FCh

DWord	Bit	Description
0	31:7	**Reserved** Access: RO Format: MBZ
	6:0	**Address to be monitored** Access: R/W _Custom_GTIReset: BUS Upper Address to be Monitored: Value of the cacheline address that needs to be monitored. This address corresponds to address bits [38:32],and has to be HPA in case of 2nd level translations are enabled, else it is GPA.
..	..	..
255	31:7	**Reserved** Access: RO Format: MBZ
	6:0	**Address to be monitored** Access: R/W _Custom_GTIReset: BUS Upper Address to be Monitored: Value of the cacheline address that needs to be monitored. This address corresponds to address bits [38:32],and has to be HPA in case of 2nd level translations are enabled, else it is GPA.

DOUBLE_BUFFER_CTL

DOUBLE_BUFFER_CTL

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	44500h-44503h
Name:	Double Buffer Control
ShortName:	DOUBLE_BUFFER_CTL
Reset:	soft

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7:4	**Reserved** Access: RO Format: MBZ
	3:1	**Reserved** Access: RO Format: MBZ
	0	**Global Double Buffer Update Disable** Description: This field is used to disable double buffer updates for all pipes and transcoders. 0b – Not Disabled 1b – Disabled

DP_TP_CTL

DP_TP_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60540h-60543h
Name:	DisplayPort Transport Control
ShortName:	DP_TP_CTL_A
Reset:	soft
Address:	61540h-61543h
Name:	DisplayPort Transport Control
ShortName:	DP_TP_CTL_B
Reset:	soft
Address:	62540h-62543h
Name:	DisplayPort Transport Control
ShortName:	DP_TP_CTL_C
Reset:	soft
Address:	63540h-63543h
Name:	DisplayPort Transport Control
ShortName:	DP_TP_CTL_D
Reset:	soft

DWord	Bit	Description
0	31	**Transport Enable** This bit enables the DisplayPort transport function. Value: 0b – Disable; 1b – Enable
	30	**FEC Enable** Forward Error Correction (FEC) coding for Display Ports (DP). The data M and data N must account for the FEC overhead when FEC is enabled. FEC can only be enabled after DP_TP_CTL isenabled. FEC can only be disabled after DP_TP_CTL is disabled. Value: 0b – disable; 1b – enable

DP_TP_CTL

29:28 **Reserved**

Access:	RO
Format:	MBZ

27 **Transport Mode Select**

This bit selects between DisplayPort SST and MST modes of operation.

This bit must be programmed for all version of displayport protocol.

Value	Name	Description
0b	SST mode	DisplayPort SST mode
1b	MST mode	DisplayPort MST mode

Restriction
The DisplayPort mode (SST or MST) selected here must match the mode selected in the Transcoder DDI Function Control registers for the transcoders attached to this transport. This field must not be changed while the DDI function is enabled.

26 **Reserved**

Access:	RO
Format:	MBZ

25 **Force ACT**

This bit forces DisplayPort MST ACT to be sent one time at the next link frame boundary. After ACT is sent, as indicated in the ACT sent status bit, this bit can be cleared and set again to send ACT again.

Value	Name	Description
0b	Do not force	Do not force ACT to be sent
1b	Force	Force ACT to be sent one time

24:21 **Reserved**

Access:	RO
Format:	MBZ

20:19 **Training Pattern 4 Select**

Value	Name	Description
00b	Training Pattern 4a **[Default]**	**CP2520 Pattern 3**:SR-BS-BS-SR-248 00hs (after data symbol scrambling and ANSI8B/10B coding)
01b	Training Pattern 4b	**CP2520 Pattern 2**:SR-BF-BF-SR-248 00hs (after data symbol scrambling and ANSI8B/10B coding)
10b	Training Pattern 4c	**CP2520 Pattern 1**:SR-CP-CP-SR-248 of 00hs (after data symbol scrambling and ANSI8B/10B coding)
11b	Reserved	

DP_TP_CTL

18 Enhanced Framing Enable

This bit selects enhanced framing for DisplayPort SST.

Hardware internally enables enhanced framing for DisplayPort MST.

Value	Name
0b	Disabled
1b	Enabled

Restriction
In DisplayPort MST mode this bit must be set to Disabled. This field must not be changed while the DDI function is enabled.

17:11 Reserved

Access:	RO
Format:	MBZ

10:8 DP Link Training Enable

These bits are used for DisplayPort link initialization as defined in the DisplayPort specification. During training patterns, hardware will internally manage enabling and disabling of scrambling. Scrambling disable bit will be ignored at that time.
DP_TP_STATUS has an indication that the required number of idle patterns has been sent.

Value	Name	Description
000b	Pattern 1	Training Pattern 1 enabled.
001b	Pattern 2	Training Pattern 2 enabled.
010b	Idle	Idle Pattern enabled.
011b	Normal	Link not in training: Send normal pixels
100b	Pattern 3	Training Pattern 3 enabled.
101b	Pattern 4	Training Pattern 4 enabled.
Others	Reserved	Reserved

Restriction
When enabling or re-enabling the port, it must be turned on with pattern 1 enabled.

7 Reserved

6 Alternate SR Enable

This bit enables the DisplayPort Alternate Scrambler Reset, intended for use only with embedded DisplayPort receivers.

Value	Name
0b	Disable
1b	Enable

DP_TP_CTL		
		Restriction This field must not be changed while the DDI function is enabled.
	5:0	**Reserved** Access: RO Format: MBZ

DP_TP_STATUS

DP_TP_STATUS	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60544h-60547h
Name:	DisplayPort Transport Status
ShortName:	DP_TP_STATUS_A
Reset:	soft
Address:	61544h-61547h
Name:	DisplayPort Transport Status
ShortName:	DP_TP_STATUS_B
Reset:	soft
Address:	62544h-62547h
Name:	DisplayPort Transport Status
ShortName:	DP_TP_STATUS_C
Reset:	soft
Address:	63544h-63547h
Name:	DisplayPort Transport Status
ShortName:	DP_TP_STATUS_D
Reset:	soft

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28	**FEC enable live status** Access: RO This bit provides live status of FEC enable/disable in hardware.
	27	**Idle Link Frame Status** Access: R/WC This bit indicates if a link frame boundary has been sent in idle pattern. This is a sticky bit, cleared by writing 1b to it. Value: 0b — Name: Idle link frame not sent Value: 1b — Name: Idle link frame sent

DP_TP_STATUS

26 **Active Link Frame Status**

Access:	R/WC

This bit indicates if a link frame boundary has been sent in active (at least one VC enabled). This is a sticky bit, cleared by writing 1b to it.

Value	Name
0b	Active link frame not sent
1b	Active link frame sent

25 **Min Idles Sent**

Access:	RO

This bit indicates that the minimum required number of idle patterns has been sent when DP_TP_CTL is set to send idle patterns. This bit will clear itself when DP_TP_CTL is not longer set to send idle patterns.

Value	Name
0b	Min idles not sent
1b	Min idles sent

24 **ACT Sent Status**

Access:	R/WC

This bit indicates if DisplayPort MST ACT has been sent. This is a sticky bit, cleared by writing 1b to it.

Value	Name
0b	ACT not sent
1b	ACT sent

23 **Mode Status**

Access:	RO

This bit indicates what mode the transport is currently in.

Value	Name	Description
0b	SST	Single-stream mode
1b	MST	Multi-stream mode

22 **DP Stream Status**

Access:	RO

Value	Name
0b	Init
1b	Active

21:19 **DP Init Status**

Access:	RO

Value	Name	Description

DP_TP_STATUS

000b	Pattern1	Training Pattern 1
001b	Pattern2	Training Pattern 2
010b	Pattern3	Training Pattern 3
100b	Idle SST	Sending SST Idle Pattern
101b	Idle MST	Sending MST Idle Pattern
110b	Active SST	In Active SST Mode
111b	Active MST	In Active MST Mode
011b	Reserved	Reserved

18:16 **Streams Enabled**

Access:	RO

This field indicates the number of streams (transcoders) enabled on this port during multistream operation. This field should be ignored in single stream mode.

Value	Name
000b	0
001b	1
010b	2
011b	3
100b	4

15:14 **Reserved**

Access:	RO
Format:	MBZ

13:12 **Payload Mapping VC3**

Access:	RO

This field indicates which transcoder is mapped to Virtual Channel 3 during multistream operation. This field should be ignored if the number of streams enabled is less than four. This field should be ignored in single stream mode.

Value	Name	Description
00b	A	Transcoder A mapped to this VC
01b	B	Transcoder B mapped to this VC
10b	C	Transcoder C mapped to this VC
11b	D	Transcoder D mapped to this VC

11:10 **Reserved**

Access:	RO
Format:	MBZ

DP_TP_STATUS

9:8 Payload Mapping VC2

Access:	RO

This field indicates which transcoder is mapped to Virtual Channel 2 during multistream operation. This field should be ignored if the number of streams enabled is less than three. This field should be ignored in single stream mode.

Value	Name	Description
00b	A	Transcoder A mapped to this VC
01b	B	Transcoder B mapped to this VC
10b	C	Transcoder C mapped to this VC
11b	D	Transcoder D mapped to this VC

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Payload Mapping VC1

Access:	RO

This field indicates which transcoder is mapped to Virtual Channel 1 during multistream operation. This field should be ignored if the number of streams enabled is less than two. This field should be ignored in single stream mode.

Value	Name	Description
00b	A	Transcoder A mapped to this VC
01b	B	Transcoder B mapped to this VC
10b	C	Transcoder C mapped to this VC
11b	D	Transcoder D mapped to this VC

3:2 Reserved

Access:	RO
Format:	MBZ

1:0 Payload Mapping VC0

Access:	RO

This field indicates which transcoder is mapped to Virtual Channel 0 during multistream operation. This field should be ignored if the number of streams enabled is less than one. This field should be ignored in single stream mode.

Value	Name	Description
00b	A	Transcoder A mapped to this VC
01b	B	Transcoder B mapped to this VC
10b	C	Transcoder C mapped to this VC
11b	D	Transcoder D mapped to this VC

DPCLKA_CFGCR0

DPCLKA_CFGCR0	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	164280h-164283h
Name:	DPCLKA_CFGCR0
ShortName:	DPCLKA_CFGCR0
Reset:	global

This register is not reset by the device 2 FLR.

DWord	Bit	Description
0	31	**Reserved**
	30	**Reserved**
	29	**Reserved**
	28:27	**Reserved** Access: RO Format: MBZ
	26	**Reserved**
	25	**Reserved**
	24	**DDIC Clock Off** This field gates off the clock going to the display engine. 0b: On 1b: Off **[Default]**
	23	**TC6 Clock Off** This field gates off the clock going to the display engine. 0b: On 1b: Off **[Default]**
	22	**TC5 Clock Off** This field gates off the clock going to the display engine. 0b: On 1b: Off **[Default]**
	21	**TC4 Clock Off** This field gates off the clock going to the display engine.

DPCLKA_CFGCR0

	Value / **Name** 0b – On 1b – Off **[Default]**
20	**Reserved**
19	**Reserved**
18	**Reserved**
17	**Reserved**
16	**Reserved**
15	**Reserved**
14	**TC3 Clock Off** This field gates off the clock going to the display engine. **Value** / **Name** 0b – On 1b – Off **[Default]**
13	**TC2 Clock Off** This field gates off the clock going to the display engine. **Value** / **Name** 0b – On 1b – Off **[Default]**
12	**TC1 Clock Off** This field gates off the clock going to the display engine. **Value** / **Name** 0b – On 1b – Off **[Default]**
11	**DDIB Clock Off** This field gates off the DDIB clock going to the display engine. DSI1 clock gating is independent and controlled by the MIPI DSI mode programming. **Value** / **Name** 0b – On 1b – Off **[Default]**
10	**DDIA Clock Off** This field gates off the DDIA clock going to the display engine. DSI0 clock gating is independent and controlled by the MIPI DSI mode programming. **Value** / **Name** 0b – On 1b – Off **[Default]**
9:8	**Reserved**

DPCLKA_CFGCR0

	7:6	**Reserved**
	5:4	**DDIC Clock Select** This field selects which DPLL will drive the port clock for DDIC.

Value	Name
00b	DPLL0
01b	DPLL1
10b	DPLL4

	3:2	**DDIB Clock Select** This field selects which DPLL will drive the port clock for DDIB and DSI1.

Value	Name
00b	DPLL0
01b	DPLL1
10b	DPLL4

	1:0	**DDIA Clock Select** This field selects which DPLL will drive the port clock for DDIA and DSI0.

Value	Name
00b	DPLL0
01b	DPLL1
10b	DPLL4

DPFC_CONTROL_SA

DPFC_CTL_SA - DPFC_CONTROL_SA

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	100100h

This register contains control bits related to Display Frame Buffer Compression Host Invalidation in System Agent.

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29	**CPUFNCEN** Default Value: 0b Access: R/W _Custom_GTIReset: BUS 0: Display Buffer is not in a CPU fence. No modifications are allowed from CPU to the Display Buffer. 1: Display Buffer exists in a CPU fence.
	28:5	**Reserved** Access: RO Format: MBZ
	4:0	**CPUFNCNUM** Default Value: 00h Access: R/W _Custom_GTIReset: BUS This field specifies the CPU visible FENCE number corresponding to the placement of the uncompressed frame buffer.

DPFC_CPU_FENCE_OFFSET

DPFC_CFO - DPFC_CPU_FENCE_OFFSET

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	100104h

This register contains control bits related to Display Frame Buffer Compression Host Invalidation in System Agent.

DWord	Bit	Description
0	31:22	**Reserved** Access: RO Format: MBZ
	21:0	**YFNCDISP** Default Value: 000000h Access: R/W _Custom_GTIReset: BUS Y offset from the CPU fence to the Display Buffer base

DPHY_CLK_TIMING_PARAM

DPHY_CLK_TIMING_PARAM

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

Address:	162180h-162183h
Name:	DPHY 0 Clock Lane Timing Parameter
ShortName:	DPHY_CLK_TIMING_PARAM_0
Reset:	global

Address:	6C180h-6C183h
Name:	DPHY 1 Clock Lane Timing Parameter
ShortName:	DPHY_CLK_TIMING_PARAM_1
Reset:	global

This register specifies the D-PHY timing parameters for the Clock Lane, if SW is overriding the HW defaults.
This register is located within the combo-PHY and is used to apply the overrides to the Clock Lane. The DSI Controller within the Display Core has an identical register (DSI_CLK_TIMING_PARAM) that is used to calculate the transition latencies of the Clock Lane. Both registers should be programmed by Software if an override is to be applied to the Clock Lane.
All fields are defined in number of Escape clocks.

Restriction

Overall restriction is that the timing parameters must be non-zero if SW is overriding the HW timing parameters.
The programming of this register must be equal to or less than the programming of its sister register that lives within the Display Core (DSI_CLK_TIMING_PARAM).

DWord	Bit	Description
0	31	**CLK_PREPARE Override** Access: R/W This field controls the override of the CLK-PREPARE timing parameter. Value 0b: HW maintains **[Default]** Value 1b: SW overrides
	30:28	**CLK_PREPARE** Access: R/W This parameter defines the time that the Host drives the Clock Lane with the LP-00 Lane state (the Bridge state) immediately before the HS-0 Line state. This field represents a hexadecimal value with a precision of 1.2 i.e. the most significant bit is the integer and the least significant 2 bits are fraction bits. So, the field can represent a range of 0.25 to 1.75 (12.5ns to 87.5ns assuming an Escape clock with a 20MHz frequency) HW maintains this timing parameter at1 Escape clock (minimum50ns)

DPHY_CLK_TIMING_PARAM

Value	Name
001b	0.25 Escape clocks
010b	0.50 Escape clocks
011b	0.75 Escape clocks
100b	1.00 Escape clocks
101b	1.25 Escape clocks
110b	1.50 Escape clocks
111b	1.75 Escape clocks
Others	Reserved

Programming Notes
Caution: The MIPI D-PHY specification has a maximum of 95ns for this parameter.

27 **CLK_ZERO Override**

Access:	R/W

This field controls the override of the CLK-ZERO timing parameter

Value	Name
0	HW Maintains
1	SW overrides

26:24 **Reserved**

Access:	RO
Format:	MBZ

23:20 **CLK_ZERO**

Access:	R/W

This parameter defines the time that the Host drives the HS-0 Lane state on the Clock Lane.
HW maintains this parameter at 5 Escape clocks (minimum250ns)

19 **CLK_PRE Override**

Access:	R/W

This field controls the override of the CLK-PRE timing parameter.

Value	Name
0	HW Maintains
1	SW overrides

18 **Reserved**

Access:	RO
Format:	MBZ

DPHY_CLK_TIMING_PARAM

	17:16	**CLK_PRE** Access: R/W This parameter defines the time that the HS clock shall be driven by the Host prior to any Data Lane beginning its transition from the LP state to the HS state. HW maintains this parameter at 8 UI (1 Byte clock). This field will override the parameter with a value measured in Escape clocks which will be much greater than 8 UI.
	15	**CLK_POST Override** Access: R/W This field controls the override of the CLK-POST timing parameter Value: 0 — Name: HW Maintains Value: 1 — Name: SW overrides
	14:11	**Reserved** Access: RO Format: MBZ
	10:8	**CLK_POST** Access: R/W This parameter defines the time the Host continues to transmit the HS clock after the last Data Lane has transitioned to the LP state. HW maintains this parameter at 1.25 Escape clocks plus 7 Byte clocks (minimum62.5ns + 56 UI)
	7	**CLK_TRAIL Override** Access: R/W This field controls the override of the CLK-TRAIL timing parameter Value: 0 — Name: HW Maintains Value: 1 — Name: SW overrides
	6:3	**Reserved** Access: RO Format: MBZ
	2:0	**CLK_TRAIL** Access: R/W This parameter defines the time that the Host drives the HS-0 Lane state on the Clock Lane after the CLK-POST time has been achieved. HW maintains this parameter at 1.25 Escape clocks (minimum 62.5ns)

DPHY_DATA_TIMING_PARAM

DPHY_DATA_TIMING_PARAM

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

Address:	162184h-162187h
Name:	DPHY 0 Data Lane Timing Parameter
ShortName:	DPHY_DATA_TIMING_PARAM_0
Reset:	global

Address:	6C184h-6C187h
Name:	DPHY 1 Data Lane Timing Parameter
ShortName:	DPHY_DATA_TIMING_PARAM_1
Reset:	global

This register specifies the D-PHY timing parameters for the Data Lane, if SW is overriding the HW defaults.
This register is located within the combo-PHY and is used to apply the overrides to the Data Lanes. The DSI Controller within the Display Core has an identical register (DSI_DATA_TIMING_PARAM) that is used to calculate the transition latencies of the Data Lanes. Both registers should be programmed by Software if an override is to be applied to the Clock Lane.
All fields are defined in number of Escape clocks.

Restriction

Overall restriction is that the timing parameters must be non-zero if SW is overriding the HW timing parameters.
The programming of this register must be equal to or less than the programming of it's sister register that lives within the Display Core (DSI_DATA_TIMING_PARAM).

DWord	Bit	Description
0	31	**HS_PREPARE Override** Access: R/W This field controls the override of the HS-PREPARE timing parameter. Value 0: HW maintains Value 1: SW overrides
	30:27	**Reserved** Access: RO Format: MBZ
	26:24	**HS_PREPARE** Access: R/W This parameter defines the time that the Host drives a Data Lane with the LP-00 Lane state (the Bridge state) immediately before driving the HS-0 Line state.

DPHY_DATA_TIMING_PARAM

This field represents a hexadecimal value with a precision of 1.2 i.e. the most significant bit is the integer and the least significant 2 bits are fraction bits. So, the field can represent a range of 0.25 to 1.75 (12.5ns to 87.5ns assuming an Escape clock with a 20MHz frequency)
HW maintains this parameter at 1 Escape clock (minimum 50ns)

Value	Name
001b	0.25 Escape clocks
010b	0.50 Escape clocks
011b	0.75 Escape clocks
100b	1.0 Escape clocks
101b	1.25 Escape clocks
110b	1.50 Escape clocks
111b	1.75 Escape clocks
Others	Reserved

Programming Notes
Caution: The MIPI D-PHY specification has a maximum of 85ns + 6UI for this parameter.

23 **HS_ZERO Override**

Access:	R/W

This field controls the override of the HS-ZERO timing parameter

Value	Name
0	HW maintains
1	SW overrides

22:20 **Reserved**

Access:	RO
Format:	MBZ

19:16 **HS_ZERO**

Access:	R/W

This parameter defines the time that the Host drives the HS-0 Lane state on a Data Lane.
HW maintains this parameter at 2 Escape clocks plus 1 Byte clock (minimum 100ns + 8UI)

15 **HS_TRAIL Override**

Access:	R/W

This field controls the override of the HS-TRAIL timing parameter

Value	Name
0	HW maintains
1	SW overrides

DPHY_DATA_TIMING_PARAM

	14:11	**Reserved** Access: RO Format: MBZ
	10:8	**HS_TRAIL** Access: R/W This parameter defines the time that the Host drives the flipped differential state of the last payload data bit of a HS transmission on a Data Lane. HW maintains this parameter at 1.5 Escape clocks (minimum 75ns)
	7	**HS_EXIT Override** Access: R/W This field controls the override of the HS-EXIT timing parameter Value: 0 — Name: HW maintains Value: 1 — Name: SW overrides
	6:3	**Reserved** Access: RO Format: MBZ
	2:0	**HS_EXIT** Access: R/W This parameter defines the time that the Host drives the LP-11 Lane state (i.e. the Stop state) following a HS burst. HW maintains this parameter at 2 Escape clocks (minimum 100ns)

DPHY_ESC_CLK_DIV

DPHY_ESC_CLK_DIV

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	162190h-162193h
Name:	DPHY 0 Escape Clock Divider
ShortName:	DPHY_ESC_CLK_DIV_0
Reset:	global
Address:	6C190h-6C193h
Name:	DPHY 1 Escape Clock Divider
ShortName:	DPHY_ESC_CLK_DIV_1
Reset:	global

This register defines the clock divider variable M needed to generate an Escape clock from the 8X clock.
This register is located within the combo-PHY. There is an identical register (DSI_ESC_CLK_DIV) located within the Display Core. Both of these registers should be programmed by Software.

Restriction :
The programming of this register must be identical to the programming of its sister register that lives within the Display Core (DSI_ESC_CLK_DIV).
If operating in Dual Link mode, then both Combo-PHY registers (DPHY_ESC_CLK_DIV_0 and DPHY_ESC_CLK_DIV_1) have to be programmed to the same value as the sister register that lives within the Display Core's primary port (DSI_ESC_CLK_DIV_0)

DWord	Bit	Description
0	31:21	**Reserved** Access: RO Format: MBZ
	20:16	**Byte Clocks per Escape Clock** Access: RO This field reports the number of Byte clocks present within a given Escape clock. The DSI transcoder calculates this variable based off of the Escape clock divider M. N = Ceiling(M/8) The DSI complex (transcoder and D-PHY) use this information to emulate an Escape clock using the Byte clock.
	15:9	**Reserved** Access: RO Format: MBZ

DPHY_ESC_CLK_DIV

	8:0	**Escape Clock Divider M** Access: R/W **Description** This field specifies the divider variable (M) needed to derive the Escape clock from the Link clock (i.e. the 8X frequency) Escape frequency= 8X frequency / M The DSI transcoder does not use a physical Escape clock, so there is no physical divider, but the transcoder needs to know the value of M to emulate the Escape clock in Byte clocks. **Restriction** The Escape clock frequency must be as close to, but not greater than 20MHz. Therefore, the programming of M should be: M = Ceiling(8X Frequency (in MHz) / 20 MHz)

DPHY_TA_TIMING_PARAM

DPHY_TA_TIMING_PARAM

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

Address:	162188h-16218Bh
Name:	DPHY 0 Turnaround Timing Parameter
ShortName:	DPHY_TA_TIMING_PARAM_0
Reset:	global

Address:	6C188h-6C18Bh
Name:	DPHY 1 Turnaround Timing Parameter
ShortName:	DPHY_TA_TIMING_PARAM_1
Reset:	global

This register specifies the D-PHY timing parameters used for the Bus Turn-Around flow, if SW is overriding the HW defaults.
All fields are defined in number of Escape clocks.

Restriction
Overall restriction is that the timing parameters must be non-zero if SW is overriding the HW timing parameters. If operating in Dual Link mode, then SW should program both Combo-PHY registers (DPHY_TA_TIMING_PARAM_0 and DPHY_TA_TIMING_PARAM_1), if necessary.

DWord	Bit	Description
0	31	**TA_SURE Override** Access: R/W This field controls the override of the TA-SURE timing parameter Value: 0 — Name: HW maintains Value: 1 — Name: SW overrides
	30:21	**Reserved** Access: RO Format: MBZ
	20:16	**TA_SURE** Access: R/W This parameter defines the time that the new transmitter waits after the LP-10 state before transmitting the Bridge state (LP-00) during a Link Turnaround. This field represents a hexadecimal value with a precision of 3.2 i.e. the most significant 3 bits are the integer and the least significant 2 bits are fraction bits. So, the field can represent a range of 0.25 to 7.75 (12.5ns to 387.5ns assuming an Escape clock with a 20MHz frequency)

DPHY_TA_TIMING_PARAM

HW maintains this parameter at 1 Escape clock (minimum 50ns).

Programming Notes
1. Caution: The MIPI D-PHY specification has a maximum of 2 Escape clocks for this parameter 2. If operating at or below an 800MHz Link frequency, this parameter should be overridden and programmed to a value of 0

15 TA_GO Override

Access:	R/W

This field controls the override of the TA-GO timing parameter

Value	Name
0	HW maintains
1	SW overrides

14:12 Reserved

Access:	RO
Format:	MBZ

11:8 TA_GO

Access:	R/W

This parameter defines the time that the transmitter drives the Bridge state (LP-00) before releasing control during a Link Turnaround.
HW maintains this parameter at 4 Escape clocks (minimum 200ns)

Programming Notes
Caution: The MIPI D-PHY specification has a fixed requirement of 4 Escape clocks for this parameter

7 TA_GET Override

Access:	R/W

This field controls the override of the TA-GET timing parameter

Value	Name
0	HW maintains
1	SW overrides

6:4 Reserved

Access:	RO
Format:	MBZ

3:0 TA_GET

Access:	R/W

This parameter defines the time that the new transmitter drives the Bridge state (LP-00) after accepting control during a Link Turnaround.
HW maintains this parameter at 5 Escape clocks (minimum 250ns)

DPHY_TA_TIMING_PARAM		
		Programming Notes
		Caution: The MIPI D-PHY specification has a fixed requirement of 5 Escape clocks for this parameter

DPHY_TRIG_EXT

DPHY_TRIG_EXT	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	16218Ch-16218Fh
Name:	DPHY 0 Trigger Extension
ShortName:	DPHY_TRIG_EXT_0
Reset:	global
Address:	6C18Ch-6C18Fh
Name:	DPHY 1 Trigger Extension
ShortName:	DPHY_TRIG_EXT_1
Reset:	global

This register specifies the amount of time to extend a Trigger message to the Peripheral.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Trigger Extension** Access: R/W This field specifies the number of Escape clocks to extend a trigger message by. This effectively extends the duration that the trigger is asserted at the Peripherals Protocol Layer. This field is only used if a Trigger Message is initiated from the DSI_LP_MSG register.

DPLC_CTL

DPLC_CTL

Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled

Address:	49400h-49403h
Name:	Pipe A LDPST Control
ShortName:	DPLC_CTL_A
Reset:	soft

Restriction : LDPST is supported for horizontal and vertical pipe sizes up to 4096 pixels.

Restriction

In cases where software wants to use the Image enhancement functionality with pre-determined IE values when the pipe starts, the following programming sequence should be followed.

1. Enable LDPST with the Orientation and the Tile Size bits programmed correctly.
2. Program IE coefficients
3. Enable pipe.

Orientation and Tile Size must not be changed when the LDPST function is enabled.

DWord	Bit	Description
0	31	**Function Enable** This field enables the LDPST function. Histogram is enabled directly by this field. Image enhancement is enabled if both this field and IE Enable are set. **Value** / **Name** 0b / Disable 1b / Enable
	30	**IE Enable** This field enables LDPST image enhancement. This field is ignored if the function is disabled. **Value** / **Name** / **Description** 0b / Disable / Input pixels are routed to output with no modification 1b / Enable / Input pixels will go through image enhancement before output
	29	**Load IE** Access: R/W Set This field is set by software to request the new image enhancement table to be loaded into the working RAMs on the next start of vertical blank or Pipe not enabled. Hardware clears this field after the load is complete.

DPLC_CTL

Value	Name
0b	Ready/Done
1b	Loading

28 **Orientation**

This field sets the orientation. In case of preload when there is a change in Orientation for next frame, this bit needs to be set before the IE correction points are written.

Value	Name	Description
0b	Landscape	16x9 tile arrangement
1b	Portrait	9x16 tile arrangement

27 **Frame Histogram Done**

Access:	R/WC

This bit is set by H/W when done creating histograms for all valid tiles (based on hsize and vsize). S/W should start reading from histogram buffers only when this bit is set. S/W must clear the bit by a write of '1' once the histogram read is complete.

Value	Name	Description
0b	Not Done	Histogram creation not done
1b	Done	Histogram creation done

26 **Histogram Buffer ID**

Access:	RO

This bit is set or cleared by H/W to indicate which double buffered BANK H/W is working on for creating histogram for current frame. This bit toggles one clk after Vblank if Hist_buffer_delay bit is not set.

Value	Name	Description
0b	Bank0	Creating Histogram in Bank0
1b	Bank1	Creating Histogram in Bank1

25 **IE Buffer ID**

Access:	RO

This bit is set or cleared by H/W to indicate which double buffered BANK H/W is using for reading correction factors for current frame. This bit will toggle at Vblank when load IE bit is set.

Value	Name	Description
0b	Bank0	Reading correction factors from Bank 0
1b	Bank1	Reading correction factors from Bank 0

24 **Allow Double Buffer Update Disable**

Access:	R/W

This field controls whether double buffer updates are allowed to be disabled for the double buffered LACE (DPLC) registers. The DOUBLE_BUFFER_CTL register can be configured to globally disable double buffer updates for those resources that allow them to be disabled.

DPLC_CTL

		Value / **Name** 0b – Not Allowed 1b – Allowed **[Default]**
	23	**Fast Access Mode Enable** This field enables the fast access mode where hardware transfers the histogram and IET to and from memory. **Value** / **Name** 0b – Disable 1b – Enable
	22	**Reserved**
	21	**Reserved**
	20	**Reserved**
	19	**Reserved**
	18:14	**Reserved** Access: RO Format: MBZ
	13:12	**Enhancement mode** These bits are the control bits to select between Look up table mode and Multiplier mode. **Value** / **Name** / **Description** 00b – Direct **[Default]** – Direct look up Mode 01b – Multiplicative – Multiplicative Mode 10b – Reserved – Reserved 11b – Reserved – Reserved
	11	**Reserved**
	10	**Reserved**
	9	**Reserved**
	8	**Reserved**
	7	**Reserved**
	6	**Reserved**
	5	**Reserved**
	4	**Reserved**
	3	**Reserved**
	2	**Reserved**

DPLC_CTL		
	1	**Hist Buffer Delay** Access: R/W This field controls when the histogram readback buffer is loaded in H/W. S/W can set this bit while reading the histogram bin registers to ensure that the H/W does not overwrite the histogram registers with the data for the new frame. This bit must be cleared by the S/W as soon as the histogram bin read is complete.
	0	**Reserved**

DPLC_FA_IIR

DPLC_FA_IIR

Register Space:	MMIO: 0/2/0
Access:	R/WC
Size (in bits):	32
Address:	49468h-4946Bh
Name:	Pipe A LDPST Fast Access Interrupt Identity
ShortName:	DPLC_FA_IIR_A
Reset:	soft

This register holds the persistent values of the interrupt bits which are unmasked by IMR. Bits set in this register will propagate to the LACE interrupt in the Pipe Interrupts.

DWord	Bit	Description
0	31:23	**Reserved** Access: RO Format: MBZ
	22	**Part B Histogram Copy Done** This field is set by hardware when DMC indicates it has finished copying this part to memory. Clear by writing with a 1. 0b – Not Done 1b – Done
	21	**Part B Load IE** This field is set by hardware on the rising edge of the load of the image enhancement table for this part. Clear by writing with a 1. 0b – Ready/Done 1b – Loading
	20	**Part B Histogram Ready** This field is set by hardware on the rising edge of when the histogram for this part is ready inside LACE. Clear by writing with a 1. 0b – Not ready 1b – Ready
	19	**Part B IET Overlap** This field is set by LACE hardware when the IET for this part is not loaded (Part Load IE bit set) before the other part is loaded. Clear by writing with a 1. 0b – Not Late

DPLC_FA_IIR

	1b	Late

18 Part B IET Late

This field is set by LACE hardware when the IET for this part is not loaded (Part Load IE bit set) before this part starts in the next frame. Clear by writing with a 1.

Value	Name
0b	Not Late
1b	Late

17 Part B Histogram Overlap

This field is set by LACE hardware when the histogram for this part is not read out (Part Histogram Done bit cleared) before the other part histogram becomes ready. Clear by writing with a 1.

Value	Name
0b	Not Overlap
1b	Overlap

16 Part B Histogram Late

This field is set by LACE hardware when the histogram for this part is not read out (Part Histogram Done bit cleared) before this part starts in the next frame. Clear by writing with a 1.

Value	Name
0b	Not Late
1b	Late

15:7 Reserved

Access:	RO
Format:	MBZ

6 Part A Histogram Copy Done

This field is set by hardware when DMC indicates it has finished copying this part to memory. Clear by writing with a 1.

Value	Name
0b	Not Done
1b	Done

5 Part A Load IE

This field is set by hardware on the rising edge of the load of the image enhancement table for this part. Clear by writing with a 1.

Value	Name
0b	Ready/Done
1b	Loading

4 Part A Histogram Ready

This field is set by hardware on the rising edge of when the histogram for this part is ready inside LACE. Clear by writing with a 1.

DPLC_FA_IIR

Value	Name
0b	Not ready
1b	Ready

3	**Part A IET Overlap** This field is set by hardware when the IET for this part is not loaded (Part Load IE bit set) before the other part is loaded. Clear by writing with a 1.

Value	Name
0b	Not Late
1b	Late

2	**Part A IET Late** This field is set by hardware when the IET for this part is not loaded (Part Load IE bit set) before this part starts in the next frame. Clear by writing with a 1.

Value	Name
0b	Not Late
1b	Late

1	**Part A Histogram Overlap** This field is set by hardware when the histogram for this part is not read out (Part Histogram Done bit cleared) before the other part histogram becomes ready. Clear by writing with a 1.

Value	Name
0b	Not Overlap
1b	Overlap

0	**Part A Histogram Late** This field is set by hardware when the histogram for this part is not read out (Part Histogram Done bit cleared) before this part starts in the next frame. Clear by writing with a 1.

Value	Name
0b	Not Late
1b	Late

DPLC_FA_IMR

DPLC_FA_IMR

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	49464h-49467h
Name:	Pipe A LDPST Fast Access Interrupt Mask
ShortName:	DPLC_FA_IMR_A
Reset:	soft

This register contains a bit mask which selects the events that will be reported in the IIR.

DWord	Bit	Description
0	31:23	**Reserved** Access: RO Format: MBZ
	22	**Part B Histogram Copy Done Mask** Access: R/W 0b: Unmask 1b: Mask **[Default]**
	21	**Part B Load IE Mask** Access: R/W 0b: Unmask 1b: Mask **[Default]**
	20	**Part B Histogram Ready Mask** Access: R/W 0b: Unmask 1b: Mask **[Default]**
	19	**Part B IET Overlap Mask** Access: R/W 0b: Unmask

DPLC_FA_IMR

	1b	Mask **[Default]**
18	**Part B IET Late Mask**	
	Access:	R/W
	Value	**Name**
	0b	Unmask
	1b	Mask **[Default]**
17	**Part B Histogram Overlap Mask**	
	Access:	R/W
	Value	**Name**
	0b	Unmask
	1b	Mask **[Default]**
16	**Part B Histogram Late Mask**	
	Access:	R/W
	Value	**Name**
	0b	Unmask
	1b	Mask **[Default]**
15:7	**Reserved**	
	Access:	RO
	Format:	MBZ
6	**Part A Histogram Copy Done Mask**	
	Access:	R/W
	Value	**Name**
	0b	Unmask
	1b	Mask **[Default]**
5	**Part A Load IE Mask**	
	Access:	R/W
	Value	**Name**
	0b	Unmask
	1b	Mask **[Default]**
4	**Part A Histogram Ready Mask**	
	Access:	R/W

DPLC_FA_IMR

Value	Name
0b	Unmask
1b	Mask **[Default]**

3 **Part A IET Overlap Mask**

Access:	R/W

Value	Name
0b	Unmask
1b	Mask **[Default]**

2 **Part A IET Late Mask**

Access:	R/W

Value	Name
0b	Unmask
1b	Mask **[Default]**

1 **Part A Histogram Overlap Mask**

Access:	R/W

Value	Name
0b	Unmask
1b	Mask **[Default]**

0 **Part A Histogram Late Mask**

Access:	R/W

Value	Name
0b	Unmask
1b	Mask **[Default]**

DPLC_FA_STATUS

DPLC_FA_STATUS

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	49460h-49463h
Name:	Pipe A LDPST Fast Access Status
ShortName:	DPLC_FA_STATUS_A
Reset:	soft

DWord	Bit	Description
0	31:19	**Reserved** Access: RO Format: MBZ
	18	**Part B Histogram Copy Done** Access: R/W Set This field is set by DMC when it finishes copying this part to memory. LACE hardware clears this field immediately. The pulse is used to set the histogram copy done interrupt. 0b – Not Done 1b – Done
	17	**Part B Load IE** Access: R/W Set This field is set by DSB to request LACE hardware to load the new image enhancement table for this part. LACE hardware clears this field at the end of this part. 0b – Ready/Done 1b – Loading
	16	**Part B Histogram Ready** Access: R/WC This field is set by LACE hardware when the histogram for this part is ready inside LACE. DMC writes 1 to clear this field after reading out the histogram. 0b – Not ready 1b – Ready
	15:3	**Reserved** Access: RO Format: MBZ

DPLC_FA_STATUS

	2	**Part A Histogram Copy Done** Access: R/W Set This field is set by DMC when it finishes copying this part to memory. LACE hardware clears this field immediately. The pulse is used to set the histogram copy done interrupt. **Value** / **Name** 0b – Not Done 1b – Done
	1	**Part A Load IE** Access: R/W Set This field is set by DSB to request LACE hardware to load the new image enhancement table for this part. LACE hardware clears this field at the end of this part. **Value** / **Name** 0b – Ready/Done 1b – Loading
	0	**Part A Histogram Ready** Access: R/WC This field is set by LACE hardware when the histogram for this part is ready inside LACE. DMC writes 1 to clear this field after reading out the histogram. **Value** / **Name** 0b – Not ready 1b – Ready

DPLC_FA_SURF

DPLC_FA_SURF	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled
Address:	49470h-49473h
Name:	Pipe A LDPST Fast Access Surface Part A
ShortName:	DPLC_FA_SURF_PARTA_A
Reset:	soft
Address:	49474h-49477h
Name:	Pipe A LDPST Fast Access Surface Part B
ShortName:	DPLC_FA_SURF_PARTB_A
Reset:	soft

DWord	Bit	Description
0	31:12	**Surface Base Address** This field specifies the surface base address bits 31:12 that the LACE histogram is copied to in fast access mode. It is mapped to physical pages through the global GTT. **Restriction** The surface must be linear. This address must be at least 4KB aligned.
	11:0	**Reserved** Access: RO Format: MBZ

DPLC_FA_SURF_LIVE

DPLC_FA_SURF_LIVE			
Register Space:	MMIO: 0/2/0		
Access:	RO		
Size (in bits):	32		
Address:	49478h-4947Bh		
Name:	Pipe A LDPST Fast Access Surface Live Part A		
ShortName:	DPLC_FA_SURF_LIVE_PARTA_A		
Reset:	soft		
Address:	4947Ch-4947Fh		
Name:	Pipe A LDPST Fast Access Surface Live Part B		
ShortName:	DPLC_FA_SURF_LIVE_PARTB_A		
Reset:	soft		
DWord	**Bit**	**Description**	
0 The live register updates at the start of the associated part.	31:12	**Surface Base Address** This field specifies the live (double-buffered value in use currently) surface base address.	
	11:0	**Reserved**	
		Access:	RO
		Format:	MBZ

DPLC_HIST_DATA

DPLC_HIST_DATA	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	49408h-4940Bh
Name:	Pipe A LDPST Histogram Data
ShortName:	DPLC_HIST_DATA_A
Reset:	soft

This register contains the histogram values for the array of LDPST Histogram table entries.
The data format is arranged as 32 dwords for each tile, containing 32 bins. The index register controls which bin and dword is accessed.

DWord	Bit	Description
0	31:17	**Reserved** Access: RO Format: MBZ
	16:0	**Bin** Histogram Data for Bin

DPLC_HIST_INDEX

DPLC_HIST_INDEX

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	49404h-49407h
Name:	Pipe A LDPST Histogram Index
ShortName:	DPLC_HIST_INDEX_A
Reset:	soft

Description

The LDPST Histogram table tile entries are accessed through index and data registers.
Each tile is composed of 32 histogram bins in 32 data Dwords.
The index fields address the individual tiles and Dwords.
Hardware will automatically walk the indexes through each Dword and raster scan through the X and Y, starting from the upper left corner of the display. The automatic walk is based on the programmed pipe source size.
Software can manually program the index to access specific tiles and Dwords.

There are 260 tiles arranged in a 20x13 (horizontal x vertical) or 13x20 array, depending on the DPLC_CTL Orientation setting.

DWord	Bit	Description
0	31:21	**Reserved** Access: RO Format: MBZ
	20:16	**Y Index** Access: Write/Read Status This index points to the current vertical tile row in the array. This index auto increments by one after each horizontal tile row is completed. It will automatically rollover when the final (20th or 13th) vertical tile row is completed. Value: [0,19] — Name:
	15:13	**Reserved** Access: RO Format: MBZ
	12:8	**X Index** Access: Write/Read Status This index points to current horizontal tile in the array. This index auto increments by one after the final (32nd) Dword is accessed. It will automatically rollover when the final (20th or 13th) horizontal tile is completed.

DPLC_HIST_INDEX

		Value / **Name** [0,19] /
	7:5	**Reserved** Access: RO Format: MBZ
	4:0	**DW Index** Access: Write/Read Status This index points to the next Dword within the tile. This index auto increments by one after each read to the data register is completed. It will automatically rollover when the final (32nd) Dword is accessed. **Value** / **Name** [0,31] /

DPLC_IE_DATA

DPLC_IE_DATA	
Register Space:	MMIO: 0/2/0
Access:	Write/Read Status
Size (in bits):	32
Address:	49410h-49413h
Name:	Pipe A LDPST Image Enhancement Data
ShortName:	DPLC_IE_DATA_A
Reset:	soft

This register contains the image enhanced pixel values for the array of LDPST Image Enhancement table entries. The data format is arranged as 17 dwords for each tile, containing 33 image enhancement points. The points are packed two per dword, aligned on 16 bit boundaries. The index register controls which bin and dword is accessed.
Point 34 is invalid and not used.
Direct Lookup mode:
If the calculated Image enhancement value is using less than 12 bits, it must be shifted left to 12 bits before programming.
Multiplicative mode:
Mulitplicative image enhancement values must be programmed directly in the 3.9 fixed point format.

DWord	Bit	Description
0	31:28	**Reserved** Access: RO Format: MBZ
	27:16	**Odd Point** Enhancement Data for Point 2*N+1
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Even Point** Enhancement Data for Point 2*N

DPLC_IE_INDEX

DPLC_IE_INDEX

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	4940Ch-4940Fh
Name:	Pipe A LDPST Image Enhancement Index
ShortName:	DPLC_IE_INDEX_A
Reset:	soft

Description

The LDPST Histogram table tile entries are accessed through index and data registers.
Each tile is composed of 33 image enhancement points in 17 data Dwords. Any IE data writes must start from DW index 0 and update all 17 Dwords
The index fields address the individual tiles and Dwords.
Hardware will automatically walk the indexes through each Dword and raster scan through the X and Y, starting from the upper left corner of the display. The automatic walk is based on the programmed pipe source size.
Software can manually program the index to access specific tiles and Dwords.

There are 260 tiles arranged in a 20x13 (horizontal x vertical) or 13x20 array, depending on the DPLC_CTL Orientation setting.

DWord	Bit	Description
0	31:21	**Reserved** Access: RO Format: MBZ
	20:16	**Y Index** Access: Write/Read Status This index points to the current vertical tile row in the array. This index auto increments by one after each horizontal tile row is completed. It will automatically rollover when the final (20th or 13th) vertical tile row is completed. Value: [0,19] — Name:
	15:13	**Reserved** Access: RO Format: MBZ
	12:8	**X Index** Access: Write/Read Status This index points to current horizontal tile in the array. This index auto increments by one after the final (17th) Dword is accessed. It will automatically rollover when the final (20th or 13th) horizontal tile is completed.

DPLC_IE_INDEX

		Value / **Name** [0,19] /
	7:5	**Reserved** Access: RO Format: MBZ
	4:0	**DW Index** Access: Write/Read Status This index points to the next Dword within the tile. This index auto increments by one after each read or write to the data register is completed. It will automatically rollover when the final (17th) Dword is accessed. **Value** / **Name** [0,16] /

DPLC_PART_CTL

DPLC_PART_CTL		
Register Space:	MMIO: 0/2/0	
Access:	Double Buffered	
Size (in bits):	32	
_Custom_Display_DoubleBufferUpdatePoint: Start of vertical blank or pipe not enabled		
Address:	49430h-49433h	
Name:	Pipe A LDPST Partition Control	
ShortName:	DPLC_PART_CTL_A	
Reset:	soft	

DWord	Bit	Description
0	31:26	**Reserved** Access: RO Format: MBZ
	25:21	**Part B End Tile Row** This field specifies the tile row that ends part B.
	20:16	**Part B Start Tile Row** This field specifies the tile row that starts part B.
	15:10	**Reserved** Access: RO Format: MBZ
	9:5	**Part A End Tile Row** This field specifies the tile row that ends part A.
	4:0	**Part A Start Tile Row** This field specifies the tile row that starts part A.

DPLC_PTRCFG

DPLC_PTRCFG		
Register Space:	MMIO: 0/2/0	
Access:	Double Buffered	
Size (in bits):	32	
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled	
Address:	49434h-49437h	
Name:	Pipe A LDPST Pointer Config Part A	
ShortName:	DPLC_PTRCFG_PARTA_A	
Reset:	soft	
Address:	49438h-4943Bh	
Name:	Pipe A LDPST Pointer Config Part B	
ShortName:	DPLC_PTRCFG_PARTB_A	
Reset:	soft	

DWord	Bit	Description
0	31:0	**Histogram Index** This field specifies the histogram index for fast LACE access.

DPLC_PTRCFG_LIVE

DPLC_PTRCFG_LIVE	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	49444h-49447h
Name:	Pipe A LDPST Pointer Config Live Part A
ShortName:	DPLC_PTRCFG_LIVE_PARTA_A
Reset:	soft
Address:	49448h-4944Bh
Name:	Pipe A LDPST Pointer Config Live Part B
ShortName:	DPLC_PTRCFG_LIVE_PARTB_A
Reset:	soft

DWord	Bit	Description
0 The live register updates at the start of the associated part.	31:0	**Histogram Index** This field specifies the live (double-buffered value in use currently) histogram index for fast LACE access.

DPLC_RDLENGTH

DPLC_RDLENGTH	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled
Address:	4943Ch-4943Fh
Name:	Pipe A LDPST Read Length Part A
ShortName:	DPLC_RDLENGTH_PARTA_A
Reset:	soft
Address:	49440h-49443h
Name:	Pipe A LDPST Read Length Part B
ShortName:	DPLC_RDLENGTH_PARTB_A
Reset:	soft

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Read Length** This field specifies the number of cachelines of histogram to read for fast LACE access. Program CEILING(number of histogram DWords/16).

DPLC_RDLENGTH_LIVE

DPLC_RDLENGTH_LIVE	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	4944Ch-4944Fh
Name:	Pipe A LDPST Read Length Live Part A
ShortName:	DPLC_RDLENGTH_LIVE_PARTA_A
Reset:	soft
Address:	49450h-49453h
Name:	Pipe A LDPST Read Length Live Part B
ShortName:	DPLC_RDLENGTH_LIVE_PARTB_A
Reset:	soft

DWord	Bit	Description
0 The live register updates at the start of the associated part.	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Read Length** This field specifies the live (double-buffered value in use currently) number of cachelines of histogram to read for fast LACE access.

DPLL_CFGCR0

DPLL_CFGCR0	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	164284h-164287h
Name:	DPLL0_CFGCR0
ShortName:	DPLL0_CFGCR0
Reset:	global
Address:	16428Ch-16428Fh
Name:	DPLL1_CFGCR0
ShortName:	DPLL1_CFGCR0
Reset:	global
Address:	16429Ch-16429Fh
Name:	TBTPLL_CFGCR0
ShortName:	TBTPLL_CFGCR0
Reset:	global
Address:	164294h-164297h
Name:	DPLL4_CFGCR0
ShortName:	DPLL4_CFGCR0
Reset:	global

This register is used to configure the DPLL mode, frequency, and SSC.
This register is not reset by the device 2 FLR.

DWord	Bit	Description
0	31:25	**Reserved** Access: RO Format: MBZ
	24:10	**DCO Fraction** Default Value: 0x4000 (DCO Frequency/Reference Frequency - INT(DCO Frequency/Reference Frequency)) * 2^15
	9:0	**DCO Integer** Default Value: 0x151 INT (DCO Frequency/Reference Frequency)

DPLL_CFGCR1

DPLL_CFGCR1	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	164288h-16428Bh
Name:	DPLL0_CFGCR1
ShortName:	DPLL0_CFGCR1
Reset:	global
Address:	164290h-164293h
Name:	DPLL1_CFGCR1
ShortName:	DPLL1_CFGCR1
Reset:	global
Address:	1642A0h-1642A3h
Name:	TBTPLL_CFGCR1
ShortName:	TBTPLL_CFGCR1
Reset:	global
Address:	164298h-16429Bh
Name:	DPLL4_CFGCR1
ShortName:	DPLL4_CFGCR1
Reset:	global

This register, together with DPLL_CFGCR0, is used to configure the DPLL frequency.

This register is not reset by the device 2 FLR.

Programming Notes

The post divider is P*Q*K

DWord	Bit	Description
0	31:18	**Reserved** Access: RO Format: MBZ
	17:10	**Qdiv Ratio** This field specifies the Q divider ratio. This field is only used when Qdiv Mode is set to Enable to get a divider value other than 1.
	9	**Qdiv Mode** This field enables the Q divider when the ratio is not 1. Value: 0b — Name: Disable — Description: Q divider = 1

DPLL_CFGCR1

1b	Enable	Q divider = Qdiv Ratio

Restriction
If K is not 2, Q MUST be 1 to ensure 50% duty cycle.

8:6 **Kdiv**

This field specifies the K divider ratio.

Value	Name
001b	1 **[Default]**
010b	2
100b	3

5:2 **Pdiv**

This field specifies the P divider ratio.

Value	Name
0001b	2
0010b	3 **[Default]**
0100b	5
1000b	7

1:0 **cfselovrd**

Display controller programs this field to set desired reference clock source in the case of genlocked systems.

Value	Name	Description
00b	Normal XTAL	Normal XTAL cannot be picked as genlock clock source if the transcoder is programmed as genlock remote secondary.
01b	Unfiltered genlock ref	
11b	Filtered genlock ref	
10b	Reserved	

DPLL_DIV0

DPLL_DIV0	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	164B00h-164B03h
Name:	DPLL0_DIV0
ShortName:	DPLL0_DIV0
Reset:	global
Address:	164C00h-164C03h
Name:	DPLL1_DIV0
ShortName:	DPLL1_DIV0
Reset:	global
Address:	164E00h-164E03h
Name:	DPLL4_DIV0
ShortName:	DPLL4_DIV0
Reset:	global

DPLL tuning.

DWord	Bit	Description
0	31:30	**i_truelock_criteria_1_0** Default Value: 01b True lock indicator criteria. After early lock generation, external PLL lock indicator asserted high when phase error is less than threshold value. 00: 16 consecutive cycles 01: 32 consecutive cycles 10: 48 consecutive cycles 11: 64 consecutive cycles
	29:28	**i_earlylock_criteria_1_0** Default Value: 11b Early lock indicator criteria. Early PLL lock indicator asserted high when phase error is less than threshold value. 00: 16 consecutive cycles 01: 32 consecutive cycles 10: 48 consecutive cycles 11: 64 consecutive cycles
	27:25	**i_afc_startup_2_0** Default Value: 000b This is for AFC start point. 000: fine = 511

DPLL_DIV0		
		001: fine = 639 (+128) 010: fine = 767 (+256) 011: fine = 895 (+384) 100: NA 101: fine = 127 (-384) 110: fine = 255 (-256) 111: fine = 383 (-128).
	24	**i_divretimeren** Retiming of feedback clock.
	23:21	**i_gainctrl_2_0** Default Value: 001b Adjustable gain for loop filter. Both coefficients shifted right by gainctrl before lock.
	20:16	**i_int_coeff_4_0** Default Value: 7h integral coeff. = 2^(-int_coeff), tageting up to 2^-11.
	15:12	**i_prop_coeff_3_0** Default Value: 0010b proportional coeff. = 2^(-prop_coeff+1).
	11:8	**i_fbprediv_3_0** Default Value: 0010b Predivider ratio 0000,0001 : reserved 0010: /2 0100: /4 0011: reserved Rest: reserved
	7:0	**i_fbdiv_intgr** Default Value: 69h Feedback divider post division (M2 integer).

DPLL_ENABLE

DPLL_ENABLE	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	46010h-46013h
Name:	DPLL 0 Enable
ShortName:	DPLL0_ENABLE
Reset:	soft
Address:	46014h-46017h
Name:	DPLL 1 Enable
ShortName:	DPLL1_ENABLE
Reset:	soft
Address:	46018h-4601Bh
Name:	DPLL 4 Enable
ShortName:	DPLL4_ENABLE
Reset:	soft
Address:	46020h-46023h
Name:	Thunderbolt PLL Enable
ShortName:	TBT_PLL_ENABLE
Reset:	soft
Address:	46030h-46033h
Name:	MG PLL 1 Enable
ShortName:	MGPLL1_ENABLE
Reset:	soft
Address:	46034h-46037h
Name:	MG PLL 2 Enable
ShortName:	MGPLL2_ENABLE
Reset:	soft
Address:	46038h-4603Bh
Name:	MG PLL 3 Enable
ShortName:	MGPLL3_ENABLE
Reset:	soft
Address:	4603Ch-4603Fh
Name:	MG PLL 4 Enable

DPLL_ENABLE

ShortName:	MGPLL4_ENABLE
Reset:	soft

Address:	46040h-46043h
Name:	MG PLL 5 Enable
ShortName:	MGPLL5_ENABLE
Reset:	soft

Address:	46044h-46047h
Name:	MG PLL 6 Enable
ShortName:	MGPLL6_ENABLE
Reset:	soft

These registers are used to enable the PLLs for driving the ports.

Note that MG register instances are being used for TypeC in general, so they include the Dekel PLLs as well.

DWord	Bit	Description
0	31	**PLL Enable** This field enables or disables the PLL. Value: 0b = Disable; 1b = Enable
	30	**PLL Lock** Access: RO This fields indicates the status of the PLL Lock. Value: 0b = Not locked or not enabled; 1b = Locked
	29	**Reserved**
	28	**Reserved** Access: RO Format: MBZ
	27	**Power Enable** This field enables or disables the PLL power. Value: 0b = Disable; 1b = Enable
	26	**Power State** Access: RO This fields indicates the status of the PLL power.

DPLL_ENABLE

		Value / **Name** 0b / Disabled 1b / Enabled
	25:12	**Reserved** Access: RO Format: MBZ
	11	**Reserved** Access: RO Format: MBZ
	10:0	**Reserved** Access: RO Format: MBZ

DPLL_SSC

DPLL_SSC	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	164B10h-164B13h
Name:	DPLL0_SSC
ShortName:	DPLL0_SSC
Reset:	global
Address:	164C10h-164C13h
Name:	DPLL1_SSC
ShortName:	DPLL1_SSC
Reset:	global
Address:	164E10h-164E13h
Name:	DPLL4_SSC
ShortName:	DPLL4_SSC
Reset:	global

DPLL SSC tuning

DWord	Bit	Description
0	31:29	**iref_ndivratio** Default Value: 0x4 Refclock divider control
	28:26	**sscstepnum_offset** SSC step number offset
	25	**sscinj_adapt_en_h** SSC injection Adaptive Gain Change Enable Value: 0b — Name: Disable Value: 1b — Name: Enable
	24	**sscinj_en_h** SSC injection enable Value: 0b — Name: Disable Value: 1b — Name: Enable
	23:16	**sscsteplength** Num of refclk cycles of one SSC step

<table>
<tr><th colspan="3">DPLL_SSC</th></tr>
<tr><td></td><td>15:14</td><td>sscfll_update_sel
Select frequency update rate for FLL SSC</td></tr>
<tr><td></td><td>13:11</td><td>sscstepnum
<table><tr><td>Default Value:</td><td>0x4</td></tr></table>SSC step number</td></tr>
<tr><td></td><td>10</td><td>ssc_openloop_en
Openloop SSC enable</td></tr>
<tr><td></td><td>9</td><td>sscen
Enables SSC modulator<table><tr><th>Value</th><th>Name</th></tr><tr><td>0b</td><td>Disable</td></tr><tr><td>1b</td><td>Enable</td></tr></table></td></tr>
<tr><td></td><td>8</td><td>sscfllen
Enable frequency adjustment for FLL SSC</td></tr>
<tr><td></td><td>7:6</td><td>bias_gb_sel
<table><tr><td>Default Value:</td><td>0x3</td></tr></table>Select guard band after bias calibration</td></tr>
<tr><td></td><td>5:0</td><td>init_dcoamp
<table><tr><td>Default Value:</td><td>0x3F</td></tr></table>Initial DCOAMP value</td></tr>
</table>

DPST_BIN

DPST_BIN	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank

Address:	490C4h-490C7h
Name:	Pipe DPST Bin Data
ShortName:	DPST_BIN_A
Reset:	soft

Address:	491C4h-491C7h
Name:	Pipe DPST Bin Data
ShortName:	DPST_BIN_B
Reset:	soft

Address:	492C4h-492C7h
Name:	Pipe DPST Bin Data
ShortName:	DPST_BIN_C
Reset:	soft

Address:	493C4h-493C7h
Name:	Pipe DPST Bin Data
ShortName:	DPST_BIN_D
Reset:	soft

Access to this address are steered to the correct register by programming the Bin Register Function Select and the Bin Register Index.Updates take place at the start of vertical blank.

DWord	Bit	Description

0	31	**Busy Bit**
		If (DPST_CTL:Bin Register Function Select = Threshold Count) {
		This is a read only bit. If set, the engine is busy and the rest of the register is undefined. If clear, the register contains valid data.
		}
		Else (Image Enhancement) {
		This bit is reserved.
		}

	30:24	**Reserved** Access: RO Format: MBZ

DPST_BIN		

	23:0	**Data**
		If (DPST_CTL : Bin Register Function Select = Threshold Count) {
		Bits 23:0 are read only bits when the Restore DPST bit (DPST_CTL) is cleared and read/write when the Restore DPST bit is set.
		They indicate the total number of pixels in this bin. The bin value is updated when guardband interrupt delay is met, and is not valid until after a histogram event has occurred. The bin value will stop incrementing once the maximum has been reached.
		}
		Else (Image Enhancement) {
		Bits 23:10 are reserved and should be written as zeroes. Bits 9:0 are R/W double-buffered and program the correction value for this bin. Writes to this register are double buffered on the next vblank. The value written here is the 10bit corrected channel value for the lowest point of the bin.
		}

DPST_CTL

DPST_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	490C0h-490C3h
Name:	Pipe DPST Control
ShortName:	DPST_CTL_A
Reset:	soft
Address:	491C0h-491C3h
Name:	Pipe DPST Control
ShortName:	DPST_CTL_B
Reset:	soft
Address:	492C0h-492C3h
Name:	Pipe DPST Control
ShortName:	DPST_CTL_C
Reset:	soft
Address:	493C0h-493C3h
Name:	Pipe DPST Control
ShortName:	DPST_CTL_D
Reset:	soft

DWord	Bit	Description
0	31	**IE Histogram Enable** This bit enables the Image Enhancement histogram logic to collect data. The collected data will be valid after a histogram event has occurred. **Value** / **Name** 0b – Disable 1b – Enable **Programming Notes** If histogram is enabled while no planes are enabled on the pipe, it may get an incorrect pixel count for a frame.
	30:29	**Reserved** Access: RO Format: MBZ

DPST_CTL

28 Reserved

Access:	RO
Format:	MBZ

27 IE Modification Table Enable

This bit enables the Image Enhancement modification table. When enabled, modifications begin after the next vertical blank.

Value	Name
0b	Disable
1b	Enable

26:25 Reserved

Access:	RO
Format:	MBZ

24 Histogram Mode Select

Value	Name	Description
0b	YUV	YUV Luma Mode
1b	HSV	HSV Intensity Mode

23:16 Reserved

Access:	RO
Format:	MBZ

15 IE Table Value Format

This field indicates what format is used for the image enhancement table values in multiplicative mode. The other modes use a 0.10 (0 integer and 10 fractional bits) format.

Value	Name	Description
0b	1.9	1 integer and 9 fractional bits
1b	2.8	2 integer and 8 fractional bits

14:13 Enhancement mode

Value	Name	Description
00b	Direct	Direct look up mode
01b	Additive	Additive mode
10b	Multiplicative	Multiplicative mode
11b	Reserved	Reserved

12 Reserved

Access:	RO
Format:	MBZ

DPST_CTL

	11	**Bin Register Function Select** This field indicates what data is being written to or read from the bin data register. **Value** / **Name** / **Description** 0b / TC / Threshold Count. A read from the bin data register returns that bin's threshold value from the most recent vblank load event (guardband threshold trip). Valid range for the Bin Index is 0 to 31. 1b / IE / Image Enhancement Value. Valid range for the Bin Index is 0 to 32
	10:7	**Reserved** Access: RO Format: MBZ
	6:0	**Bin Register Index** This field indicates the bin number whose data can be accessed through the bin data register. This value is automatically incremented by a read or a write to the bin data register if the busy bit is not set.

DPST_GUARD

DPST_GUARD	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank
Address:	490C8h-490CBh
Name:	Pipe DPST Threshold Guardband
ShortName:	DPST_GUARD_A
Reset:	soft
Address:	491C8h-491CBh
Name:	Pipe DPST Threshold Guardband
ShortName:	DPST_GUARD_B
Reset:	soft
Address:	492C8h-492CBh
Name:	Pipe DPST Threshold Guardband
ShortName:	DPST_GUARD_C
Reset:	soft
Address:	493C8h-493CBh
Name:	Pipe DPST Threshold Guardband
ShortName:	DPST_GUARD_D
Reset:	soft

Updates take place at the start of vertical blank.

DWord	Bit	Description
0	31	**Histogram Interrupt enable** Value / Name / Description 0b / Disable / Disabled 1b / Enable / This generates a histogram interrupt once a Histogram event occurs.
	30	**Histogram Event status** Access: R/WC When a Histogram event has occurred, this will get set by the hardware. For any more Histogram events to occur, clear this bit by writing a '1'. Value / Name / Description 0b / Not Occurred / Histogram event has not occurred 1b / Occurred / Histogram event has occurred

DPST_GUARD		
	29:22	**Guardband Interrupt Delay** An interrupt is always generated after this many consecutive frames of the guardband threshold being surpassed .This value is double buffered on start of vblank. **Restriction** A value of 0 is invalid.
	21:0	**Threshold Guardband** This value is used to determine the guardband for the threshold interrupt generation. This single value is used for all the segments. This value is double buffered on start of vblank. This value is shifted left 2 bits (multiplied by 4) for use with the 24 bit bin values.

Driver Media Force Wake Ack

DRIVER_MEDIA_FWAKE_ACK - Driver Media Force Wake Ack	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00D88h

Message registers have bit-wise masking applied for writes. The register consists of 16 bits of data in [15:0], and 16 bits of corresponding masks in [31:16].
To set bit0, for example, the data would be 0x0001_0001.
To clear bit0, for example, the data would be 0x0001_0000.
Note that mask bit is the data bit offset + 16. Message registers are protected from non-GT writes via the Message Channel.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO
	15:0	**GPM Driver Media ForceWake Ack** Access: R/W _Custom_GTIReset: BUS 1'b0 : GT Media Can be powered down (default) 1'b1 : GT Media cannot be powered down

DRIVER MEDIA FORCE WAKE REQ

DRIVER_MEDIA_FWAKE - DRIVER MEDIA FORCE WAKE REQ

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	0A270h

The newly loaded Gfx driver must first initialize this register by clearing all bits (0xFF00).

DWord	Bit	Description
0	31:16	**Multiple Force Wake Mask** Access: R/W _Custom_GTIReset: DEV Mask bits applied to [15:0] of same register. If mask is set to 1, corresponding bit in [15:0 is written. If mask is set to 0, corresponding bit in [15:0] is unaffected.
	15	**Force Wake Request for Thread 15** Access: R/W _Custom_GTIReset: DEV Thread 15 - When set with corresponding mask bit 31, Media cannot be powered down.
	14	**Force Wake Request for Thread 14** Access: R/W _Custom_GTIReset: DEV Thread 14 - When set with corresponding mask bit 30, Media cannot be powered down.
	13	**Force Wake Request for Thread 13** Access: R/W _Custom_GTIReset: DEV Thread 13 - When set with corresponding mask bit 29, Media cannot be powered down.
	12	**Force Wake Request for Thread 12** Access: R/W _Custom_GTIReset: DEV Thread 12 - When set with corresponding mask bit 28, Media cannot be powered down.
	11	**Force Wake Request for Thread 11** Access: R/W _Custom_GTIReset: DEV Thread 11 - When set with corresponding mask bit 27, Media cannot be powered down.

DRIVER_MEDIA_FWAKE - DRIVER MEDIA FORCE WAKE REQ

	10	**Force Wake Request for Thread 10** Access: R/W _Custom_GTIReset: DEV Thread 10 - When set with corresponding mask bit 26, Media cannot be powered down.
	9	**Force Wake Request for Thread 9** Access: R/W _Custom_GTIReset: DEV Thread 9 - When set with corresponding mask bit 25, Media cannot be powered down.
	8	**Force Wake Request for Thread 8** Access: R/W _Custom_GTIReset: DEV Thread 8 - When set with corresponding mask bit 24, Media cannot be powered down.
	7	**Force Wake Request for Thread 7** Access: R/W _Custom_GTIReset: DEV Thread 7 - When set with corresponding mask bit 23, Media cannot be powered down.
	6	**Force Wake Request for Thread 6** Access: R/W _Custom_GTIReset: DEV Thread 6 - When set with corresponding mask bit 22, Media cannot be powered down.
	5	**Force Wake Request for Thread 5** Access: R/W _Custom_GTIReset: DEV Thread 5 - When set with corresponding mask bit 21, Media cannot be powered down.
	4	**Force Wake Request for Thread 4** Access: R/W _Custom_GTIReset: DEV Thread 4 - When set with corresponding mask bit 20, Media cannot be powered down.
	3	**Force Wake Request for Thread 3** Access: R/W _Custom_GTIReset: DEV Thread 3 - When set with corresponding mask bit 19, Media cannot be powered down.

DRIVER_MEDIA_FWAKE - DRIVER MEDIA FORCE WAKE REQ

	2	**Force Wake Request for Thread 2** Access: R/W _Custom_GTIReset: DEV Thread 2 - When set with corresponding mask bit 18, Media cannot be powered down.
	1	**Force Wake Request for Thread 1** Access: R/W _Custom_GTIReset: DEV Thread 1 - When set with corresponding mask bit 17, Media cannot be powered down.
	0	**Force Wake Request for Thread 0** Access: R/W _Custom_GTIReset: DEV Thread 0 - When set with corresponding mask bit 16, Media cannot be powered down.

Driver Render Force Wake Ack

DRIVER_RENDER_FWAKE_ACK - Driver Render Force Wake Ack	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00D84h

Message registers have bit-wise masking applied for writes. The register consists of 16 bits of data in [15:0], and 16 bits of corresponding masks in [31:16].
To set bit0, for example, the data would be 0x0001_0001.
To clear bit0, for example, the data would be 0x0001_0000.
Note that mask bit is the data bit offset + 16. Message registers are protected from non-GT writes via the Message Channel.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**GPM Driver ForceWake Ack** Access: R/W _Custom_GTIReset: BUS 1'b0 : GT Render Can be powered down (default) 1'b1 : GT Render cannot be powered down

Driver VDBox0 Force Wake Ack

DRIVER_VDBOX0_FWAKE_ACK - Driver VDBox0 Force Wake Ack

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00D50h
Name:	Driver VDBox0 Force Wake Ack
ShortName:	DRIVER_VDBOX0_FWAKE_ACK

This register is used for GPM to handshake the driver's VDBox0 forcewake request

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**GPM Driver Vdbox0 ForceWake Ack** Access: R/W _Custom_GTIReset: BUS 1'b0 : GT Media Slice 0 can be powered down (default) 1'b1 : GT Media Slice 0 cannot be powered down

Driver VDBox1 Force Wake Ack

DRIVER_VDBOX1_FWAKE_ACK - Driver VDBox1 Force Wake Ack

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00D54h
Name:	Driver VDBox1 Force Wake Ack
ShortName:	DRIVER_VDBOX1_FWAKE_ACK

This register is used for GPM to handshake the driver's VDBox1 forcewake request

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**GPM Driver Vdbox1 ForceWake Ack** Access: R/W _Custom_GTIReset: BUS 1'b0 : GT Media Slice 0 can be powered down (default) 1'b1 : GT Media Slice 0 cannot be powered down

Driver VDBox2 Force Wake Ack

DRIVER_VDBOX2_FWAKE_ACK - Driver VDBox2 Force Wake Ack

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00D58h
Name:	Driver VDBox2 Force Wake Ack
ShortName:	DRIVER_VDBOX2_FWAKE_ACK

This register is used for GPM to handshake the driver's VDBox2 forcewake request

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**GPM Driver Vdbox2 ForceWake Ack** Access: R/W _Custom_GTIReset: BUS 1'b0 : GT Media Slice 1 can be powered down (default) 1'b1 : GT Media Slice 1 cannot be powered down

Driver VDBox3 Force Wake Ack

DRIVER_VDBOX3_FWAKE_ACK - Driver VDBox3 Force Wake Ack

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00D5Ch
Name:	Driver VDBox3 Force Wake Ack
ShortName:	DRIVER_VDBOX3_FWAKE_ACK

This register is used for GPM to handshake the driver's VDBox3 forcewake request

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**GPM Driver Vdbox3 ForceWake Ack** Access: R/W _Custom_GTIReset: BUS 1'b0 : GT Media Slice 1 can be powered down (default) 1'b1 : GT Media Slice 1 cannot be powered down

Driver VDBox4 Force Wake Ack

DRIVER_VDBOX4_FWAKE_ACK - Driver VDBox4 Force Wake Ack

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00D60h
Name:	Driver VDBox4 Force Wake Ack
ShortName:	DRIVER_VDBOX4_FWAKE_ACK

This register is used for GPM to handshake the driver's VDBox4 forcewake request

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**GPM Driver Vdbox4 ForceWake Ack** Access: R/W _Custom_GTIReset: BUS 1'b0 : GT Media Slice 2 can be powered down (default) 1'b1 : GT Media Slice 2 cannot be powered down

Driver VDBox5 Force Wake Ack

DRIVER_VDBOX5_FWAKE_ACK - Driver VDBox5 Force Wake Ack

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00D64h
Name:	Driver VDBox5 Force Wake Ack
ShortName:	DRIVER_VDBOX5_FWAKE_ACK

This register is used for GPM to handshake the driver's VDBox5 forcewake request

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**GPM Driver Vdbox5 ForceWake Ack** Access: R/W _Custom_GTIReset: BUS 1'b0 : GT Media Slice 2 can be powered down (default) 1'b1 : GT Media Slice 2 cannot be powered down

Driver VDBox6 Force Wake Ack

DRIVER_VDBOX6_FWAKE_ACK - Driver VDBox6 Force Wake Ack

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00D68h
Name:	Driver VDBox6 Force Wake Ack
ShortName:	DRIVER_VDBOX6_FWAKE_ACK

This register is used for GPM to handshake the driver's VDBox6 forcewake request

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**GPM Driver Vdbox6 ForceWake Ack** Access: R/W _Custom_GTIReset: BUS 1'b0 : GT Media Slice 3 can be powered down (default) 1'b1 : GT Media Slice 3 cannot be powered down

Driver VDBox7 Force Wake Ack

DRIVER_VDBOX7_FWAKE_ACK - Driver VDBox7 Force Wake Ack

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00D6Ch
Name:	Driver VDBox7 Force Wake Ack
ShortName:	DRIVER_VDBOX7_FWAKE_ACK

This register is used for GPM to handshake the driver's VDBox7 forcewake request

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**GPM Driver Vdbox7 ForceWake Ack** Access: R/W _Custom_GTIReset: BUS 1'b0 : GT Media Slice 3 can be powered down (default) 1'b1 : GT Media Slice 3 cannot be powered down

Driver VEBox0 Force Wake Ack

DRIVER_VEBOX0_FWAKE_ACK - Driver VEBox0 Force Wake Ack

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00D70h
Name:	Driver VEBox0 Force Wake Ack
ShortName:	DRIVER_VEBOX0_FWAKE_ACK

This register is used for GPM to handshake the driver's VEBox0 forcewake request

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**GPM Driver Vebox0 ForceWake Ack** Access: R/W _Custom_GTIReset: BUS 1'b0 : GT Media Slice 0 can be powered down (default) 1'b1 : GT Media Slice 0 cannot be powered down

Driver VEBox1 Force Wake Ack

DRIVER_VEBOX1_FWAKE_ACK - Driver VEBox1 Force Wake Ack	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00D74h
Name:	Driver VEBox1 Force Wake Ack
ShortName:	DRIVER_VEBOX1_FWAKE_ACK

This register is used for GPM to handshake the driver's VEBox1 forcewake request

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**GPM Driver Vebox1 ForceWake Ack** Access: R/W _Custom_GTIReset: BUS 1'b0 : GT Media Slice 1 can be powered down (default) 1'b1 : GT Media Slice 1 cannot be powered down

Driver Vebox2 Force Wake Ack

DRIVER_VEBOX2_FWAKE_ACK - Driver Vebox2 Force Wake Ack

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00D78h
Name:	Driver Vebox2 Force Wake Ack
ShortName:	DRIVER_VEBOX2_FWAKE_ACK

This register stores the ACK, from GPMunit, once the driver forcewake has been serviced
Message registers have bit-wise masking applied for writes. The register consists of 16 bits of data in [15:0], and 16 bits of corresponding masks in [31:16].
To set bit0, for example, the data would be 0x0001_0001.
To clear bit0, for example, the data would be 0x0001_0000.
Note that mask bit is the data bit offset + 16. Message registers are protected from non-GT writes via the Message Channel.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**GPM Driver Vebox2 ForceWake Ack** Access: R/W _Custom_GTIReset: BUS 1'b0 : GT Media Can be powered down (default) 1'b1 : GT Media cannot be powered down

Driver Vebox3 Force Wake Ack

DRIVER_VEBOX3_FWAKE_ACK - Driver Vebox3 Force Wake Ack

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00D7Ch
Name:	Driver Vebox3 Force Wake Ack
ShortName:	DRIVER_VEBOX3_FWAKE_ACK

This register stores the Force wake ACK, from GPM, for driver forcewake to VEBOX3
Message registers have bit-wise masking applied for writes. The register consists of 16 bits of data in [15:0], and 16 bits of corresponding masks in [31:16].
To set bit0, for example, the data would be 0x0001_0001.
To clear bit0, for example, the data would be 0x0001_0000.
Note that mask bit is the data bit offset + 16. Message registers are protected from non-GT writes via the Message Channel.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**GPM Driver Vebox3 ForceWake Ack** Access: R/W _Custom_GTIReset: BUS 1'b0 : GT Media Can be powered down (default) 1'b1 : GT Media cannot be powered down

DSB_BUFRPT_CNT

DSB_BUFRPT_CNT	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	70B48h-70B4Bh
Name:	DSB BUFFER REPEAT COUNTER
ShortName:	DSB_BUFRPT_CNT_0_A
Reset:	soft
Address:	70C48h-70C4Bh
Name:	DSB BUFFER REPEAT COUNTER
ShortName:	DSB_BUFRPT_CNT_1_A
Reset:	soft
Address:	70D48h-70D4Bh
Name:	DSB BUFFER REPEAT COUNTER
ShortName:	DSB_BUFRPT_CNT_2_A
Reset:	soft
Address:	71B48h-71B4Bh
Name:	DSB BUFFER REPEAT COUNTER
ShortName:	DSB_BUFRPT_CNT_0_B
Reset:	soft
Address:	71C48h-71C4Bh
Name:	DSB BUFFER REPEAT COUNTER
ShortName:	DSB_BUFRPT_CNT_1_B
Reset:	soft
Address:	71D48h-71D4Bh
Name:	DSB BUFFER REPEAT COUNTER
ShortName:	DSB_BUFRPT_CNT_2_B
Reset:	soft
Address:	72B48h-72B4Bh
Name:	DSB BUFFER REPEAT COUNTER
ShortName:	DSB_BUFRPT_CNT_0_C
Reset:	soft
Address:	72C48h-72C4Bh
Name:	DSB BUFFER REPEAT COUNTER
ShortName:	DSB_BUFRPT_CNT_1_C

DSB_BUFRPT_CNT

Reset:	soft

Address:	72D48h-72D4Bh
Name:	DSB BUFFER REPEAT COUNTER
ShortName:	DSB_BUFRPT_CNT_2_C
Reset:	soft

Address:	73B48h-73B4Bh
Name:	DSB BUFFER REPEAT COUNTER
ShortName:	DSB_BUFRPT_CNT_0_D
Reset:	soft

Address:	73C48h-73C4Bh
Name:	DSB BUFFER REPEAT COUNTER
ShortName:	DSB_BUFRPT_CNT_1_D
Reset:	soft

Address:	73D48h-73D4Bh
Name:	DSB BUFFER REPEAT COUNTER
ShortName:	DSB_BUFRPT_CNT_2_D
Reset:	soft

This register can be updated only before the DSB_CTRL register is programmed to enable the DSB engine.

DWord	Bit	Description
0	31:0	**REPEAT_COUNT** Default Value: 00000008h repeat_value Access: R/W Number of times that the buffer play will be repeated when the Buffer Re-iterate is set in DSB_CTRL register. Do not program this value to 0 if buffer re-iterate is set to 1 in DSB_CTRL register.

DSB_CTRL

DSB_CTRL	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	70B08h-70B0Bh
Name:	DSB Control
ShortName:	DSB_CTRL_0_A
Reset:	soft
Address:	70C08h-70C0Bh
Name:	DSB Control
ShortName:	DSB_CTRL_1_A
Reset:	soft
Address:	70D08h-70D0Bh
Name:	DSB Control
ShortName:	DSB_CTRL_2_A
Reset:	soft
Address:	71B08h-71B0Bh
Name:	DSB Control
ShortName:	DSB_CTRL_0_B
Reset:	soft
Address:	71C08h-71C0Bh
Name:	DSB Control
ShortName:	DSB_CTRL_1_B
Reset:	soft
Address:	71D08h-71D0Bh
Name:	DSB Control
ShortName:	DSB_CTRL_2_B
Reset:	soft
Address:	72B08h-72B0Bh
Name:	DSB Control
ShortName:	DSB_CTRL_0_C
Reset:	soft
Address:	72C08h-72C0Bh
Name:	DSB Control
ShortName:	DSB_CTRL_1_C

DSB_CTRL

Reset:	soft
Address:	72D08h-72D0Bh
Name:	DSB Control
ShortName:	DSB_CTRL_2_C
Reset:	soft
Address:	73B08h-73B0Bh
Name:	DSB Control
ShortName:	DSB_CTRL_0_D
Reset:	soft
Address:	73C08h-73C0Bh
Name:	DSB Control
ShortName:	DSB_CTRL_1_D
Reset:	soft
Address:	73D08h-73D0Bh
Name:	DSB Control
ShortName:	DSB_CTRL_2_D
Reset:	soft

This register controls the Display State Buffer (DSB) engines.

DWord	Bit	Description
0	31	**DSB Enable** Access: R/W This bit enables the DSB engine. Notes: 1. The act of enabling the DSB engine will not result in the engine to start fetching a state buffer from memory. Only a write to the Tail Pointer will start a fetch 2. A write to the Tail Pointer of the DSB engine will have no affect if the DSB engine is not enabled 3. SW should not update the DSB control registers while DSB engine is not in IDLE state. Bit 0 indicates the status of DSB engine. 4. If SW desires to reset the DSB engine then following sequence can be followed. This sequence is also described in the DSB programming details 1. To reset the DSB engine, SW can set this bit to 0. Bit31 transition from 1 to 0 initiates a reset sequence in the DSB HW. 2. Poll for the DSB status (bit 0) to be in IDLE. 3. If DSB status is 0 then the DSB engine reset sequence is complete and DSB is in reset state.

DSB_CTRL

Value	Name
0b	Disabled
1b	Enabled

30 **Reserved**

Access:	RO
Format:	MBZ

29 **Buffer Reiterate**

Access:	R/W

This bit controls what the DSB engine does when finished with a state buffer program.
If this bit is cleared, then the engine will stop and wait for Software to update the Head/Tail pointers before it starts processing the next state buffer.
If this bit is set, then the engine will start to re-fetch the state buffer defined by the current Head/Tail pointers when it finishes with the current program.
Note that the DSB engine will not take any action on this bit if it is not currently processing a state buffer. I.e. When Software sets this bit on an idle DSB engine, it must then write to the Tail Pointer to start the reiteration process.

Value	Name
0b	Disabled
1b	Enabled

Programming Notes
Software should ensure there is some form of flow control when this bit is set (i.e. the state buffer has a Wait opcode and/or the Wait for VBLANK bit of this register is set)

28 **Wait for VBLANK**

Access:	R/W

This bit controls when the DSB engine starts to fetch a state buffer from memory.
When cleared, then the DSB engine will start fetching the state buffer immediately after Software writes the Tail Pointer
When set, then the DSB engine will start fetching the state buffer only after it sees a rising edge of VBLANK following the write to the Tail Pointer.
SW cannot have both bits 27 and 28 set at the same time.

27 **Wait for Line in Range**

Access:	R/W

This bit controls when the DSB engine starts to fetch a state buffer from memory.
When cleared, then the DSB engine will start fetching the state buffer immediately after Software writes the Tail Pointer
When set, then the DSB engine will start fetching the state buffer only when the scanline falls in the range defined in DSB_LINERANGE_PF register following the write to the Tail Pointer.
SW cannot have both bits 27 and 28 set at the same time.

DSB_CTRL

	26:17	**Reserved** Access: RO Format: MBZ
	16	**DSB Halt** Access: R/W When this bit is set DSB engine will halt. DMA engine will stop generating new memory requests and all the pending instructions in the buffer will be executed and DSB will be halted. The pointers will stop moving. SW can read the DSB_CURRENT_HEAD_PTR register to check the DMA engine pointer status.
	15:9	**Reserved** Access: RO Format: MBZ
	8	**Non Posted Enable** Access: R/W When this bit is set all the MMIO writes including the indexed writes will be non posted to the clients. If SW sets this bit within the DSB program then SW must ensure to include at least 4 NO Ops instructions following this register programming. This is required to allow the register in the DSB to be set correctly.
	7:1	**Reserved** Access: RO Format: MBZ
	0	**DSB Status** Access: RO This field reflects the status of the DSB engine. It is set when the DMA starts to read the state buffer and is cleared when the state buffer is finished and last instruction is completed. **Value** / **Name** 0b – Idle 1b – Busy

DSB_CURRENT_HEAD_PTR

DSB_CURRENT_HEAD_PTR	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	70B2Ch-70B2Fh
Name:	DSB CURRENT HEAD PTR
ShortName:	DSB_CURRENT_HEAD_PTR_0_A
Reset:	soft
Address:	70C2Ch-70C2Fh
Name:	DSB CURRENT HEAD PTR
ShortName:	DSB_CURRENT_HEAD_PTR_1_A
Reset:	soft
Address:	70D2Ch-70D2Fh
Name:	DSB CURRENT HEAD PTR
ShortName:	DSB_CURRENT_HEAD_PTR_2_A
Reset:	soft
Address:	71B2Ch-71B2Fh
Name:	DSB CURRENT HEAD PTR
ShortName:	DSB_CURRENT_HEAD_PTR_0_B
Reset:	soft
Address:	71C2Ch-71C2Fh
Name:	DSB CURRENT HEAD PTR
ShortName:	DSB_CURRENT_HEAD_PTR_1_B
Reset:	soft
Address:	71D2Ch-71D2Fh
Name:	DSB CURRENT HEAD PTR
ShortName:	DSB_CURRENT_HEAD_PTR_2_B
Reset:	soft
Address:	72B2Ch-72B2Fh
Name:	DSB CURRENT HEAD PTR
ShortName:	DSB_CURRENT_HEAD_PTR_0_C
Reset:	soft
Address:	72C2Ch-72C2Fh
Name:	DSB CURRENT HEAD PTR

DSB_CURRENT_HEAD_PTR

ShortName:	DSB_CURRENT_HEAD_PTR_1_C
Reset:	soft

Address:	72D2Ch-72D2Fh
Name:	DSB CURRENT HEAD PTR
ShortName:	DSB_CURRENT_HEAD_PTR_2_C
Reset:	soft

Address:	73B2Ch-73B2Fh
Name:	DSB CURRENT HEAD PTR
ShortName:	DSB_CURRENT_HEAD_PTR_0_D
Reset:	soft

Address:	73C2Ch-73C2Fh
Name:	DSB CURRENT HEAD PTR
ShortName:	DSB_CURRENT_HEAD_PTR_1_D
Reset:	soft

Address:	73D2Ch-73D2Fh
Name:	DSB CURRENT HEAD PTR
ShortName:	DSB_CURRENT_HEAD_PTR_2_D
Reset:	soft

DWord	Bit	Description
0	31:6	**Head Pointer** Access: RO This register will have the current head pointer of the DSB HW. Driver can read to check the status of the DSB pointer.
	5:0	**Reserved** Access: RO Format: MBZ

DSB_HEAD_PTR

DSB_HEAD_PTR	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	70B00h-70B03h
Name:	DSB Head Pointer
ShortName:	DSB_HEAD_PTR_0_A
Reset:	soft
Address:	70C00h-70C03h
Name:	DSB Head Pointer
ShortName:	DSB_HEAD_PTR_1_A
Reset:	soft
Address:	70D00h-70D03h
Name:	DSB Head Pointer
ShortName:	DSB_HEAD_PTR_2_A
Reset:	soft
Address:	71B00h-71B03h
Name:	DSB Head Pointer
ShortName:	DSB_HEAD_PTR_0_B
Reset:	soft
Address:	71C00h-71C03h
Name:	DSB Head Pointer
ShortName:	DSB_HEAD_PTR_1_B
Reset:	soft
Address:	71D00h-71D03h
Name:	DSB Head Pointer
ShortName:	DSB_HEAD_PTR_2_B
Reset:	soft
Address:	72B00h-72B03h
Name:	DSB Head Pointer
ShortName:	DSB_HEAD_PTR_0_C
Reset:	soft
Address:	72C00h-72C03h
Name:	DSB Head Pointer
ShortName:	DSB_HEAD_PTR_1_C

DSB_HEAD_PTR

Reset:	soft
Address:	72D00h-72D03h
Name:	DSB Head Pointer
ShortName:	DSB_HEAD_PTR_2_C
Reset:	soft
Address:	73B00h-73B03h
Name:	DSB Head Pointer
ShortName:	DSB_HEAD_PTR_0_D
Reset:	soft
Address:	73C00h-73C03h
Name:	DSB Head Pointer
ShortName:	DSB_HEAD_PTR_1_D
Reset:	soft
Address:	73D00h-73D03h
Name:	DSB Head Pointer
ShortName:	DSB_HEAD_PTR_2_D
Reset:	soft

DWord	Bit	Description
0	31:6	**Head Pointer** Access: R/W This register defines the base address (i.e. head pointer) of the Display State Buffering memory. The address within this register is a graphics address. Restriction : 1. The address is cacheline aligned within this DWord (HW enforced by reserving register bits 5:0) 2. The address should be pointing to the first cacheline of the state buffer 3. Software should not modify this register while the DSB engine is busy
	5:0	**Reserved** Access: RO Format: MBZ

DSB_INTERRUPT

DSB_INTERRUPT	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	70B28h-70B2Bh
Name:	DSB INTERRUPT
ShortName:	DSB_INTERRUPT_0_A
Reset:	soft
Address:	70C28h-70C2Bh
Name:	DSB INTERRUPT
ShortName:	DSB_INTERRUPT_1_A
Reset:	soft
Address:	70D28h-70D2Bh
Name:	DSB INTERRUPT
ShortName:	DSB_INTERRUPT_2_A
Reset:	soft
Address:	71B28h-71B2Bh
Name:	DSB INTERRUPT
ShortName:	DSB_INTERRUPT_0_B
Reset:	soft
Address:	71C28h-71C2Bh
Name:	DSB INTERRUPT
ShortName:	DSB_INTERRUPT_1_B
Reset:	soft
Address:	71D28h-71D2Bh
Name:	DSB INTERRUPT
ShortName:	DSB_INTERRUPT_2_B
Reset:	soft
Address:	72B28h-72B2Bh
Name:	DSB INTERRUPT
ShortName:	DSB_INTERRUPT_0_C
Reset:	soft
Address:	72C28h-72C2Bh
Name:	DSB INTERRUPT
ShortName:	DSB_INTERRUPT_1_C

DSB_INTERRUPT

Reset:	soft

Address:	72D28h-72D2Bh
Name:	DSB INTERRUPT
ShortName:	DSB_INTERRUPT_2_C
Reset:	soft

Address:	73B28h-73B2Bh
Name:	DSB INTERRUPT
ShortName:	DSB_INTERRUPT_0_D
Reset:	soft

Address:	73C28h-73C2Bh
Name:	DSB INTERRUPT
ShortName:	DSB_INTERRUPT_1_D
Reset:	soft

Address:	73D28h-73D2Bh
Name:	DSB INTERRUPT
ShortName:	DSB_INTERRUPT_2_D
Reset:	soft

DWord	Bit	Description
0	31	**SPARE_31** Access: R/W
	30	**SPARE_30** Access: R/W
	29	**SPARE_29** Access: R/W
	28	**SPARE_28** Access: R/W
	27	**SPARE_27** Access: R/W
	26	**SPARE_26** Access: R/W
	25	**SPARE_25** Access: R/W
	24	**SPARE_24** Access: R/W
	23	**SPARE_23** Access: R/W

DSB_INTERRUPT		
	22	**SPARE_22** Access: R/W
	21	**SPARE_21** Access: R/W
	20	**SPARE_20** Access: R/W
	19	**DSB_gtt_fault_interrupt_enable** Access: R/W When set the interrupt for gtt faults is enabled. If faults occurs DSB_gtt_fault_interrupt is set.
	18	**DSB_rsptimeout_interrupt_enable** Access: R/W When set the interrupt for response timeout error is enabled. If error occurs DSB_rstimeout_interrupt is set.
	17	**DSB_poll_error_interrupt_enable** Access: R/W When set the interrupt for read poll error is enabled. If error occurs DSB_poll_error_interrupt is set.
	16	**DSB_program_interrupt_enable** Access: R/W When set the interrupt for interrupt instruction is enabled. If program interrupt instruction occurs DSB_program_interrupt is set.
	15	**SPARE_15** Access: R/W
	14	**SPARE_14** Access: R/W
	13	**SPARE_13** Access: R/W
	12	**SPARE_12** Access: R/W
	11	**SPARE_11** Access: R/W
	10	**SPARE_10** Access: R/W
	9	**SPARE_9** Access: R/W

DSB_INTERRUPT

	8	**SPARE_8** Access: R/W
	7	**SPARE_7** Access: R/W
	6	**SPARE_6** Access: R/W
	5	**SPARE_5** Access: R/W
	4	**SPARE_4** Access: R/W
	3	**DSB_gtt_fault_interrupt** Access: R/WC When gtt fault occurs this bit is set. SW must write a 1 to clear this status bit.
	2	**DSB_rsptimeout_interrupt** Access: R/WC When response timeout error occurs this bit is set. SW must write a 1 to clear this status bit.
	1	**DSB_poll_error_interrupt** Access: R/WC When read poll error occurs this bit is set. SW must write a 1 to clear this status bit.
	0	**DSB_program_interrupt** Access: R/WC When program interrupt occurs this bit is set. SW must write a 1 to clear this status bit.

DSB_MMIOCTRL

DSB_MMIOCTRL	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	70B0Ch-70B0Fh
Name:	DSB MMIO Control
ShortName:	DSB_MMIOCTRL_0_A
Reset:	soft
Address:	70C0Ch-70C0Fh
Name:	DSB MMIO Control
ShortName:	DSB_MMIOCTRL_1_A
Reset:	soft
Address:	70D0Ch-70D0Fh
Name:	DSB MMIO Control
ShortName:	DSB_MMIOCTRL_2_A
Reset:	soft
Address:	71B0Ch-71B0Fh
Name:	DSB MMIO Control
ShortName:	DSB_MMIOCTRL_0_B
Reset:	soft
Address:	71C0Ch-71C0Fh
Name:	DSB MMIO Control
ShortName:	DSB_MMIOCTRL_1_B
Reset:	soft
Address:	71D0Ch-71D0Fh
Name:	DSB MMIO Control
ShortName:	DSB_MMIOCTRL_2_B
Reset:	soft
Address:	72B0Ch-72B0Fh
Name:	DSB MMIO Control
ShortName:	DSB_MMIOCTRL_0_C
Reset:	soft
Address:	72C0Ch-72C0Fh
Name:	DSB MMIO Control
ShortName:	DSB_MMIOCTRL_1_C

DSB_MMIOCTRL

Reset:	soft

Address:	72D0Ch-72D0Fh
Name:	DSB MMIO Control
ShortName:	DSB_MMIOCTRL_2_C
Reset:	soft

Address:	73B0Ch-73B0Fh
Name:	DSB MMIO Control
ShortName:	DSB_MMIOCTRL_0_D
Reset:	soft

Address:	73C0Ch-73C0Fh
Name:	DSB MMIO Control
ShortName:	DSB_MMIOCTRL_1_D
Reset:	soft

Address:	73D0Ch-73D0Fh
Name:	DSB MMIO Control
ShortName:	DSB_MMIOCTRL_2_D
Reset:	soft

This register can be updated only before the DSB_CTRL register is programmed to enable the DSB engine.

DWord	Bit	Description
0	31	**DSB HDR meta data delay en** Access: R/W SW must set this bit to 1 and program the bits 15:8 to correct values based on the cd and link clock ratios when DSB is used for HDR meta data transfer. Value: 0b — Name: Delay disabled **[Default]** Value: 1b — Name: Delay enabled
	30:16	**Reserved** Access: RO Format: MBZ
	15:8	**DSB MMIO Dead Clocks Count** Default Value: 00001000b Eight Access: R/W When this counter is enabled by programming bit 31 of this register, DSB will insert programmed number of dead clocks after every 8 back to back MMIO cycles. This is needed to take care of the clock crossing timing issues. Driver must ensure the cdclk/dotclock ratio to adjust the programming. Below is the formula to calculate the number of dead clock programming is as follows. If (cdclk/dotclk) <= 1 then DSB MMIO dead clock count = default value

<table>
<tr><th colspan="3">DSB_MMIOCTRL</th></tr>
<tr><td></td><td></td><td>else if (cdclk/dotclk) > 1 then DSB MMIO dead clock count = 11 *ROUNDUP(cdclk/dotclk)
elseDSB MMIO dead clock count = default value</td></tr>
<tr><td></td><td>7:0</td><td>DSB MMIO Cycles
<table><tr><td>Default Value:</td><td>00001000b Eight</td></tr><tr><td>Access:</td><td>R/W</td></tr></table>These bits can be programmed to change the number of clocks that the DSB engine has control over the MMIO bus. DSB will send programmed number of MMIO cycles before giving away access of RMBUS.</td></tr>
</table>

DSB_PF_LN_LOWER

DSB_PF_LN_LOWER	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	70B40h-70B43h
Name:	DSB PREFETCH SCANLINE LOWER LIMIT
ShortName:	DSB_PF_LN_LOWER_0_A
Reset:	soft
Address:	70C40h-70C43h
Name:	DSB PREFETCH SCANLINE LOWER LIMIT
ShortName:	DSB_PF_LN_LOWER_1_A
Reset:	soft
Address:	70D40h-70D43h
Name:	DSB PREFETCH SCANLINE LOWER LIMIT
ShortName:	DSB_PF_LN_LOWER_2_A
Reset:	soft
Address:	71B40h-71B43h
Name:	DSB PREFETCH SCANLINE LOWER LIMIT
ShortName:	DSB_PF_LN_LOWER_0_B
Reset:	soft
Address:	71C40h-71C43h
Name:	DSB PREFETCH SCANLINE LOWER LIMIT
ShortName:	DSB_PF_LN_LOWER_1_B
Reset:	soft
Address:	71D40h-71D43h
Name:	DSB PREFETCH SCANLINE LOWER LIMIT
ShortName:	DSB_PF_LN_LOWER_2_B
Reset:	soft
Address:	72B40h-72B43h
Name:	DSB PREFETCH SCANLINE LOWER LIMIT
ShortName:	DSB_PF_LN_LOWER_0_C
Reset:	soft
Address:	72C40h-72C43h
Name:	DSB PREFETCH SCANLINE LOWER LIMIT
ShortName:	DSB_PF_LN_LOWER_1_C

DSB_PF_LN_LOWER

Reset:	soft
Address:	72D40h-72D43h
Name:	DSB PREFETCH SCANLINE LOWER LIMIT
ShortName:	DSB_PF_LN_LOWER_2_C
Reset:	soft
Address:	73B40h-73B43h
Name:	DSB PREFETCH SCANLINE LOWER LIMIT
ShortName:	DSB_PF_LN_LOWER_0_D
Reset:	soft
Address:	73C40h-73C43h
Name:	DSB PREFETCH SCANLINE LOWER LIMIT
ShortName:	DSB_PF_LN_LOWER_1_D
Reset:	soft
Address:	73D40h-73D43h
Name:	DSB PREFETCH SCANLINE LOWER LIMIT
ShortName:	DSB_PF_LN_LOWER_2_D
Reset:	soft

DWord	Bit	Description
0	31:0	**Scanline_lower_limit** Access: R/W SW must program this register before Tail pointer is updated if the bit 27 in the DSB_CTRL register is set. This is the lower limit of the scanline number to be compared with the HW line number to start the DSB data prefetch.

DSB_PF_LN_UPPER

DSB_PF_LN_UPPER	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	70B44h-70B47h
Name:	DSB PREFETCH SCANLINE UPPER LIMIT
ShortName:	DSB_PF_LN_UPPER_0_A
Reset:	soft
Address:	70C44h-70C47h
Name:	DSB PREFETCH SCANLINE UPPER LIMIT
ShortName:	DSB_PF_LN_UPPER_1_A
Reset:	soft
Address:	70D44h-70D47h
Name:	DSB PREFETCH SCANLINE UPPER LIMIT
ShortName:	DSB_PF_LN_UPPER_2_A
Reset:	soft
Address:	71B44h-71B47h
Name:	DSB PREFETCH SCANLINE UPPER LIMIT
ShortName:	DSB_PF_LN_UPPER_0_B
Reset:	soft
Address:	71C44h-71C47h
Name:	DSB PREFETCH SCANLINE UPPER LIMIT
ShortName:	DSB_PF_LN_UPPER_1_B
Reset:	soft
Address:	71D44h-71D47h
Name:	DSB PREFETCH SCANLINE UPPER LIMIT
ShortName:	DSB_PF_LN_UPPER_2_B
Reset:	soft
Address:	72B44h-72B47h
Name:	DSB PREFETCH SCANLINE UPPER LIMIT
ShortName:	DSB_PF_LN_UPPER_0_C
Reset:	soft
Address:	72C44h-72C47h
Name:	DSB PREFETCH SCANLINE UPPER LIMIT
ShortName:	DSB_PF_LN_UPPER_1_C

DSB_PF_LN_UPPER

Reset:	soft
Address:	72D44h-72D47h
Name:	DSB PREFETCH SCANLINE UPPER LIMIT
ShortName:	DSB_PF_LN_UPPER_2_C
Reset:	soft
Address:	73B44h-73B47h
Name:	DSB PREFETCH SCANLINE UPPER LIMIT
ShortName:	DSB_PF_LN_UPPER_0_D
Reset:	soft
Address:	73C44h-73C47h
Name:	DSB PREFETCH SCANLINE UPPER LIMIT
ShortName:	DSB_PF_LN_UPPER_1_D
Reset:	soft
Address:	73D44h-73D47h
Name:	DSB PREFETCH SCANLINE UPPER LIMIT
ShortName:	DSB_PF_LN_UPPER_2_D
Reset:	soft

DWord	Bit	Description
0	31:0	**Scanline_upper_limit** Access: R/W SW must program this register before Tail pointer is updated if the bit 27 in the DSB_CTRL register is set. This is the upper limit of the scanline number to be compared with the HW line number to start the DSB data prefetch.

DSB_PMCTRL_2

DSB_PMCTRL_2	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	70B3Ch-70B3Fh
Name:	DSB POWER MANAGEMENT CONTROL 2
ShortName:	DSB_PMCTRL_2_0_A
Reset:	soft
Address:	70C3Ch-70C3Fh
Name:	DSB POWER MANAGEMENT CONTROL 2
ShortName:	DSB_PMCTRL_2_1_A
Reset:	soft
Address:	70D3Ch-70D3Fh
Name:	DSB POWER MANAGEMENT CONTROL 2
ShortName:	DSB_PMCTRL_2_2_A
Reset:	soft
Address:	71B3Ch-71B3Fh
Name:	DSB POWER MANAGEMENT CONTROL 2
ShortName:	DSB_PMCTRL_2_0_B
Reset:	soft
Address:	71C3Ch-71C3Fh
Name:	DSB POWER MANAGEMENT CONTROL 2
ShortName:	DSB_PMCTRL_2_1_B
Reset:	soft
Address:	71D3Ch-71D3Fh
Name:	DSB POWER MANAGEMENT CONTROL 2
ShortName:	DSB_PMCTRL_2_2_B
Reset:	soft
Address:	72B3Ch-72B3Fh
Name:	DSB POWER MANAGEMENT CONTROL 2
ShortName:	DSB_PMCTRL_2_0_C
Reset:	soft
Address:	72C3Ch-72C3Fh
Name:	DSB POWER MANAGEMENT CONTROL 2
ShortName:	DSB_PMCTRL_2_1_C

DSB_PMCTRL_2

Reset:	soft

Address:	72D3Ch-72D3Fh
Name:	DSB POWER MANAGEMENT CONTROL 2
ShortName:	DSB_PMCTRL_2_2_C
Reset:	soft

Address:	73B3Ch-73B3Fh
Name:	DSB POWER MANAGEMENT CONTROL 2
ShortName:	DSB_PMCTRL_2_0_D
Reset:	soft

Address:	73C3Ch-73C3Fh
Name:	DSB POWER MANAGEMENT CONTROL 2
ShortName:	DSB_PMCTRL_2_1_D
Reset:	soft

Address:	73D3Ch-73D3Fh
Name:	DSB POWER MANAGEMENT CONTROL 2
ShortName:	DSB_PMCTRL_2_2_D
Reset:	soft

This register can be updated only before the DSB_CTRL register or it can also be updated within DSB program with appropriate byte enable settings.

DWord	Bit	Description
0	31	**MMIOGEN_DEWAKE_dis** Access: R/W Setting this bit will disable the dewake logic from MMIO generation logic.
	30:24	**Reserved** Access: RO Format: MBZ
	23	**FORCE_DEWAKE** Access: R/W Setting this bit will force a DEWAKE from the DSB engine.
	22:16	**Reserved** Access: RO Format: MBZ
	15	**BLOCK_DEWAKE_EXTENSION** Access: R/W

DSB_PMCTRL_2

Value	Name	Description
0	Extend Dewake	When this bit is set to 0 then the dewake is extended from the time it is set when the Program scanline number in PMCTRL is reached to the point when DSB is done processing all the instructions.
1	Disable Extend Dewake **[Default]**	When this bit is set to 1 then the dewake is not extended and will be de-asserted on a vblank. Dewake then depends on the other logic in DSB.

14:8 **Reserved**

Access:	RO
Format:	MBZ

7 **OVERRIDE_DC5_DC6_OK**

Access:	R/W

Setting this bit will force a DC5 or DC6 OK from the DSB engine.

6:0 **Reserved**

Access:	RO
Format:	MBZ

DSB_PMCTRL

DSB_PMCTRL	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	70B38h-70B3Bh
Name:	DSB POWER MANAGEMENT CONTROL
ShortName:	DSB_PMCTRL_0_A
Reset:	soft
Address:	70C38h-70C3Bh
Name:	DSB POWER MANAGEMENT CONTROL
ShortName:	DSB_PMCTRL_1_A
Reset:	soft
Address:	70D38h-70D3Bh
Name:	DSB POWER MANAGEMENT CONTROL
ShortName:	DSB_PMCTRL_2_A
Reset:	soft
Address:	71B38h-71B3Bh
Name:	DSB POWER MANAGEMENT CONTROL
ShortName:	DSB_PMCTRL_0_B
Reset:	soft
Address:	71C38h-71C3Bh
Name:	DSB POWER MANAGEMENT CONTROL
ShortName:	DSB_PMCTRL_1_B
Reset:	soft
Address:	71D38h-71D3Bh
Name:	DSB POWER MANAGEMENT CONTROL
ShortName:	DSB_PMCTRL_2_B
Reset:	soft
Address:	72B38h-72B3Bh
Name:	DSB POWER MANAGEMENT CONTROL
ShortName:	DSB_PMCTRL_0_C
Reset:	soft
Address:	72C38h-72C3Bh
Name:	DSB POWER MANAGEMENT CONTROL
ShortName:	DSB_PMCTRL_1_C

DSB_PMCTRL

Reset:	soft

Address:	72D38h-72D3Bh
Name:	DSB POWER MANAGEMENT CONTROL
ShortName:	DSB_PMCTRL_2_C
Reset:	soft

Address:	73B38h-73B3Bh
Name:	DSB POWER MANAGEMENT CONTROL
ShortName:	DSB_PMCTRL_0_D
Reset:	soft

Address:	73C38h-73C3Bh
Name:	DSB POWER MANAGEMENT CONTROL
ShortName:	DSB_PMCTRL_1_D
Reset:	soft

Address:	73D38h-73D3Bh
Name:	DSB POWER MANAGEMENT CONTROL
ShortName:	DSB_PMCTRL_2_D
Reset:	soft

This register can be updated only before the DSB_CTRL register is programmed to enable the DSB engine.

DWord	Bit	Description
0	31	**Enable DE Wake Generation** Access: R/W Setting this bit to 1 enables the DE wake generation logic based on the scan line number programmed to trigger DE wake.
	30:0	**Scanline number for DE Wake Generation** Access: R/W Scanline number which when reached will trigger a DE wake from DSB if memory is down. 31 bits to cover VRR range

DSB_POLLFUNC

DSB_POLLFUNC	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	70B10h-70B13h
Name:	DSB POLL FUNCTION
ShortName:	DSB_POLLFUNC_0_A
Reset:	soft
Address:	70C10h-70C13h
Name:	DSB POLL FUNCTION
ShortName:	DSB_POLLFUNC_1_A
Reset:	soft
Address:	70D10h-70D13h
Name:	DSB POLL FUNCTION
ShortName:	DSB_POLLFUNC_2_A
Reset:	soft
Address:	71B10h-71B13h
Name:	DSB POLL FUNCTION
ShortName:	DSB_POLLFUNC_0_B
Reset:	soft
Address:	71C10h-71C13h
Name:	DSB POLL FUNCTION
ShortName:	DSB_POLLFUNC_1_B
Reset:	soft
Address:	71D10h-71D13h
Name:	DSB POLL FUNCTION
ShortName:	DSB_POLLFUNC_2_B
Reset:	soft
Address:	72B10h-72B13h
Name:	DSB POLL FUNCTION
ShortName:	DSB_POLLFUNC_0_C
Reset:	soft
Address:	72C10h-72C13h
Name:	DSB POLL FUNCTION
ShortName:	DSB_POLLFUNC_1_C

DSB_POLLFUNC

Reset:	soft
Address:	72D10h-72D13h
Name:	DSB POLL FUNCTION
ShortName:	DSB_POLLFUNC_2_C
Reset:	soft
Address:	73B10h-73B13h
Name:	DSB POLL FUNCTION
ShortName:	DSB_POLLFUNC_0_D
Reset:	soft
Address:	73C10h-73C13h
Name:	DSB POLL FUNCTION
ShortName:	DSB_POLLFUNC_1_D
Reset:	soft
Address:	73D10h-73D13h
Name:	DSB POLL FUNCTION
ShortName:	DSB_POLLFUNC_2_D
Reset:	soft

If poll function is needed, this register should be programmed before the Poll function is used in the DSB program. Can be part of the DSB program itself.

DWord	Bit	Description
0	31	**Poll Enable** This field enables the Poll function in DSB. Value: 0b – Disable **[Default]**; 1b – Enable
	30:23	**Wait in Microseconds** Default Value: 00000010b 2us Access: R/W This field defines the number of micro seconds that DSB will wait before retrying the poll.
	22:15	**Number of times** Default Value: 00110010b 50 times Access: R/W This field defines the number of times that DSB will retry the poll if not satisfied before time out.

DSB_POLLFUNC		
	14:0	**Reserved** Access: RO Format: MBZ

DSB_POLLMASK

DSB_POLLMASK	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	70B1Ch-70B1Fh
Name:	DSB POLL MASK
ShortName:	DSB_POLLMASK_0_A
Reset:	soft
Address:	70C1Ch-70C1Fh
Name:	DSB POLL MASK
ShortName:	DSB_POLLMASK_1_A
Reset:	soft
Address:	70D1Ch-70D1Fh
Name:	DSB POLL MASK
ShortName:	DSB_POLLMASK_2_A
Reset:	soft
Address:	71B1Ch-71B1Fh
Name:	DSB POLL MASK
ShortName:	DSB_POLLMASK_0_B
Reset:	soft
Address:	71C1Ch-71C1Fh
Name:	DSB POLL MASK
ShortName:	DSB_POLLMASK_1_B
Reset:	soft
Address:	71D1Ch-71D1Fh
Name:	DSB POLL MASK
ShortName:	DSB_POLLMASK_2_B
Reset:	soft
Address:	72B1Ch-72B1Fh
Name:	DSB POLL MASK
ShortName:	DSB_POLLMASK_0_C
Reset:	soft
Address:	72C1Ch-72C1Fh
Name:	DSB POLL MASK
ShortName:	DSB_POLLMASK_1_C

DSB_POLLMASK

Reset:	soft
Address:	72D1Ch-72D1Fh
Name:	DSB POLL MASK
ShortName:	DSB_POLLMASK_2_C
Reset:	soft
Address:	73B1Ch-73B1Fh
Name:	DSB POLL MASK
ShortName:	DSB_POLLMASK_0_D
Reset:	soft
Address:	73C1Ch-73C1Fh
Name:	DSB POLL MASK
ShortName:	DSB_POLLMASK_1_D
Reset:	soft
Address:	73D1Ch-73D1Fh
Name:	DSB POLL MASK
ShortName:	DSB_POLLMASK_2_D
Reset:	soft

If poll function is needed, this register should be programmed before the Poll function is used in the DSB program. Can be part of the DSB program itself.

DWord	Bit	Description
0	31:0	**DSB_POLL_MASK** Access: R/W SW must program these masks bits before the POLL RD instruction is called in the DSB program. This register programming can be part of the DSB program itself and the MMIO write instruction to this register can be placed before the poll read instruction. HW will use these bits mask the bits that are not needed to be compared with the poll read data.

DSB_RM_TIMEOUT

DSB_RM_TIMEOUT	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	70B30h-70B33h
Name:	DSB RMTIMEOUT
ShortName:	DSB_RM_TIMEOUT_0_A
Reset:	soft
Address:	70C30h-70C33h
Name:	DSB RMTIMEOUT
ShortName:	DSB_RM_TIMEOUT_1_A
Reset:	soft
Address:	70D30h-70D33h
Name:	DSB RMTIMEOUT
ShortName:	DSB_RM_TIMEOUT_2_A
Reset:	soft
Address:	71B30h-71B33h
Name:	DSB RMTIMEOUT
ShortName:	DSB_RM_TIMEOUT_0_B
Reset:	soft
Address:	71C30h-71C33h
Name:	DSB RMTIMEOUT
ShortName:	DSB_RM_TIMEOUT_1_B
Reset:	soft
Address:	71D30h-71D33h
Name:	DSB RMTIMEOUT
ShortName:	DSB_RM_TIMEOUT_2_B
Reset:	soft
Address:	72B30h-72B33h
Name:	DSB RMTIMEOUT
ShortName:	DSB_RM_TIMEOUT_0_C
Reset:	soft
Address:	72C30h-72C33h
Name:	DSB RMTIMEOUT
ShortName:	DSB_RM_TIMEOUT_1_C

DSB_RM_TIMEOUT

Reset:	soft

Address:	72D30h-72D33h
Name:	DSB RMTIMEOUT
ShortName:	DSB_RM_TIMEOUT_2_C
Reset:	soft

Address:	73B30h-73B33h
Name:	DSB RMTIMEOUT
ShortName:	DSB_RM_TIMEOUT_0_D
Reset:	soft

Address:	73C30h-73C33h
Name:	DSB RMTIMEOUT
ShortName:	DSB_RM_TIMEOUT_1_D
Reset:	soft

Address:	73D30h-73D33h
Name:	DSB RMTIMEOUT
ShortName:	DSB_RM_TIMEOUT_2_D
Reset:	soft

This register can be updated only before the DSB_CTRL register is programmed to enable the DSB engine.

DWord	Bit	Description
0	31	**RM Claim Timeout** Access: R/WC Sticky bit set to 1 when RM hits ready timeout when non posted cycles are enabled in DSB. Clear by writing with a 1.
	30	**RM Ready Timeout** Access: R/WC Sticky bit set to 1 when RM hits claim counter when non posted cycles are enabled in DSB. Clear by writing with a 1.
	29:24	**Reserved** Access: RO Format: MBZ
	23:16	**RM Claim Timeout Count** Default Value: 20h 32 Clocks Access: R/W Number of clocks that DSB will wait for a claim response from clients before generating a claim for error condition.

DSB_RM_TIMEOUT		
	15:0	**RM Ready Timeout Value** Access: R/W This field selects the RM timeout amount in microseconds. **Value** / **Name** FFFFh: 65,535 microseconds **[Default]** 0000h: Timeout disabled

DSB_RMTIMEOUTREG_CAPTURE

DSB_RMTIMEOUTREG_CAPTURE	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	70B34h-70B37h
Name:	DSB RMTIMEOUT REGISTER ADDRESS CAPTURE
ShortName:	DSB_RMTIMEOUTREG_CAPTURE_0_A
Reset:	soft
Address:	70C34h-70C37h
Name:	DSB RMTIMEOUT REGISTER ADDRESS CAPTURE
ShortName:	DSB_RMTIMEOUTREG_CAPTURE_1_A
Reset:	soft
Address:	70D34h-70D37h
Name:	DSB RMTIMEOUT REGISTER ADDRESS CAPTURE
ShortName:	DSB_RMTIMEOUTREG_CAPTURE_2_A
Reset:	soft
Address:	71B34h-71B37h
Name:	DSB RMTIMEOUT REGISTER ADDRESS CAPTURE
ShortName:	DSB_RMTIMEOUTREG_CAPTURE_0_B
Reset:	soft
Address:	71C34h-71C37h
Name:	DSB RMTIMEOUT REGISTER ADDRESS CAPTURE
ShortName:	DSB_RMTIMEOUTREG_CAPTURE_1_B
Reset:	soft
Address:	71D34h-71D37h
Name:	DSB RMTIMEOUT REGISTER ADDRESS CAPTURE
ShortName:	DSB_RMTIMEOUTREG_CAPTURE_2_B
Reset:	soft
Address:	72B34h-72B37h
Name:	DSB RMTIMEOUT REGISTER ADDRESS CAPTURE
ShortName:	DSB_RMTIMEOUTREG_CAPTURE_0_C
Reset:	soft
Address:	72C34h-72C37h
Name:	DSB RMTIMEOUT REGISTER ADDRESS CAPTURE
ShortName:	DSB_RMTIMEOUTREG_CAPTURE_1_C

DSB_RMTIMEOUTREG_CAPTURE

Reset:	soft

Address:	72D34h-72D37h
Name:	DSB RMTIMEOUT REGISTER ADDRESS CAPTURE
ShortName:	DSB_RMTIMEOUTREG_CAPTURE_2_C
Reset:	soft

Address:	73B34h-73B37h
Name:	DSB RMTIMEOUT REGISTER ADDRESS CAPTURE
ShortName:	DSB_RMTIMEOUTREG_CAPTURE_0_D
Reset:	soft

Address:	73C34h-73C37h
Name:	DSB RMTIMEOUT REGISTER ADDRESS CAPTURE
ShortName:	DSB_RMTIMEOUTREG_CAPTURE_1_D
Reset:	soft

Address:	73D34h-73D37h
Name:	DSB RMTIMEOUT REGISTER ADDRESS CAPTURE
ShortName:	DSB_RMTIMEOUTREG_CAPTURE_2_D
Reset:	soft

This register captures the register on which the RMTIMEOUT error happened.

DWord	Bit	Description
0	31:0	**DSB TIMEOUT REGISTER CAPTURE** Access: RO Offset of the Register that caused RM TIMEOUT.

DSB_STATUS

DSB_STATUS	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	70B24h-70B27h
Name:	DSB STATUS
ShortName:	DSB_STATUS_0_A
Reset:	soft
Address:	70C24h-70C27h
Name:	DSB STATUS
ShortName:	DSB_STATUS_1_A
Reset:	soft
Address:	70D24h-70D27h
Name:	DSB STATUS
ShortName:	DSB_STATUS_2_A
Reset:	soft
Address:	71B24h-71B27h
Name:	DSB STATUS
ShortName:	DSB_STATUS_0_B
Reset:	soft
Address:	71C24h-71C27h
Name:	DSB STATUS
ShortName:	DSB_STATUS_1_B
Reset:	soft
Address:	71D24h-71D27h
Name:	DSB STATUS
ShortName:	DSB_STATUS_2_B
Reset:	soft
Address:	72B24h-72B27h
Name:	DSB STATUS
ShortName:	DSB_STATUS_0_C
Reset:	soft
Address:	72C24h-72C27h
Name:	DSB STATUS
ShortName:	DSB_STATUS_1_C

DSB_STATUS

Reset:	soft

Address:	72D24h-72D27h
Name:	DSB STATUS
ShortName:	DSB_STATUS_2_C
Reset:	soft

Address:	73B24h-73B27h
Name:	DSB STATUS
ShortName:	DSB_STATUS_0_D
Reset:	soft

Address:	73C24h-73C27h
Name:	DSB STATUS
ShortName:	DSB_STATUS_1_D
Reset:	soft

Address:	73D24h-73D27h
Name:	DSB STATUS
ShortName:	DSB_STATUS_2_D
Reset:	soft

DWord	Bit	Description
0	31	**SPARE_31** Access: RO This bit indicates the status for hp idle state.
	30	**SPARE_30** Access: RO This bit indicates the status of the de wake state.
	29:27	**DSB_REQARB_SM_state** Access: RO Shows the DSB HP requests machine states. IDLE 3'b000 DSB0 3'b001 DSB1 3'b010 DSB2 3'b011
	26	**SPARE_26** Access: RO

DSB_STATUS

		Description This bit indicates the live status of the safe window signal.
	25:23	**DSB_VTDFAULT_ARB_SM_state** Access: RO Shows the DSB VTD Fault arbitration machine states. IDLE_VTD_ERR 3'b000 DSB0_VTD_ERR 3'b001 DSB1_VTD_ERR 3'b010 DSB2_VTD_ERR 3'b011 PIPEDMC_VTD_ERR 3'b100
	22	**SPARE_22** Access: RO Shows bit 2 of DSB_TLBTRANS_SM
	21:20	**DSB_TLBTRANS_SM_state** Access: RO Shows the DSB TLB translation machine states. IDLE 2'b00 INPUT_FIFO_RD 2'b01 WAIT_FOT_VTD 2'b10 OUTPUT_FIFO_WRITE 2'b11
	19	**SPARE_19** Access: R/WC **Description** This bit is set when DSB receives a safe window signal as 1 from dptunit. SW must write a 1 to clear this register bit.
	18:17	**DSB_POINTERS_SM_state** Access: RO Shows the DSB Pointers increment state machine states. IDLE 2'b00 LOAD_HTP 2'b01 INCREMENT_ADDR 2'b10 HALT 2'b11
	16	**SPARE_16** Access: R/WC

DSB_STATUS

		Description This bit is set when DSB is still in non-IDLE state after delayed vblank. SW must write a 1 to clear this register bit.
	15:13	**DSB_MMIO_ARB_SM_state** Access: RO Shows the DSB MMIO arbitor state machine states. MAIN_DMC 3'b000 PIPE_DMC 3'b001 DSB0 3'b010 DSB1 3'b011 DSB2 3'b100
	12	**SPARE_12** Access: RO
	11:7	**DSB_MMIO_INST_SM_state** Access: RO Shows the DSB MMIO Instructions state machine states. IDLE 5'b00000 MMIO_WR 5'b00001 WT_VBLANKS 5'b00010 WT_USEC 5'b00011 WT_LINES 5'b00100 WT_LINES_IN_RANGE 5'b00101 WT_LINES_OUT_OF_RANGE 5'b00110 GENERATE_INT 5'b00111 GENERATE_FRAME_START 5'b01000 WT_FOR_RMBUS_ACCESS 5'b01001 MMIO_INDEXED_WR 5'b01010 INDEXED_EVEN 5'b01011 WT_FOR_VALID_DATA 5'b01100 MMIO_POLL_RD 5'b01101 MMIO_POLL_RD_RECEIVED 5'b01110 WT_POLL_US 5'b01111 POLL_ERROR 5'b10000
	6	**SPARE_6** Access: RO
	5:4	**DSB_reset_SM_state** Access: RO Shows the DSB Reset state machine states. IDLE_RST 2'b00 STOP_REQ_WAIT_FOR_DATA 2'b01 START_SYNC_RESET 2'b10 DEASSERT_SYNC_RESET 2'b11

DSB_STATUS		
	3	**SPARE_3** Access: RO
	2:0	**DSB_run_SM_state** Access: RO Shows the DMA Run state machine states. IDLE 3'b000 DSB_CTRL_UPDATED 3'b001 WT_FOR_VBLANK_SCANLINE_IN_RANGE 3'b010 NOT_WT_FOR_VBLANK_SCANLINE_IN_RANGE 3'b011 VBLANK_LINE_IN_RANGE_HAPPENED 3'b100 DSB_RUN 3'b101

DSB_TAIL_PTR

DSB_TAIL_PTR	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	70B04h-70B07h
Name:	DSB Tail Pointer
ShortName:	DSB_TAIL_PTR_0_A
Reset:	soft
Address:	70C04h-70C07h
Name:	DSB Tail Pointer
ShortName:	DSB_TAIL_PTR_1_A
Reset:	soft
Address:	70D04h-70D07h
Name:	DSB Tail Pointer
ShortName:	DSB_TAIL_PTR_2_A
Reset:	soft
Address:	71B04h-71B07h
Name:	DSB Tail Pointer
ShortName:	DSB_TAIL_PTR_0_B
Reset:	soft
Address:	71C04h-71C07h
Name:	DSB Tail Pointer
ShortName:	DSB_TAIL_PTR_1_B
Reset:	soft
Address:	71D04h-71D07h
Name:	DSB Tail Pointer
ShortName:	DSB_TAIL_PTR_2_B
Reset:	soft
Address:	72B04h-72B07h
Name:	DSB Tail Pointer
ShortName:	DSB_TAIL_PTR_0_C
Reset:	soft
Address:	72C04h-72C07h
Name:	DSB Tail Pointer
ShortName:	DSB_TAIL_PTR_1_C

DSB_TAIL_PTR

Reset:	soft
Address:	72D04h-72D07h
Name:	DSB Tail Pointer
ShortName:	DSB_TAIL_PTR_2_C
Reset:	soft
Address:	73B04h-73B07h
Name:	DSB Tail Pointer
ShortName:	DSB_TAIL_PTR_0_D
Reset:	soft
Address:	73C04h-73C07h
Name:	DSB Tail Pointer
ShortName:	DSB_TAIL_PTR_1_D
Reset:	soft
Address:	73D04h-73D07h
Name:	DSB Tail Pointer
ShortName:	DSB_TAIL_PTR_2_D
Reset:	soft

DWord	Bit	Description
0	31:6	**Tail Pointer** Access: R/W This register defines the end address (i.e. tail pointer) of the Display Sate Buffer in memory. The address within this register is a graphics address. The act of writing to this register is the event that will activate the DSB engine (i.e. the DSB engine will start fetching the state buffer from memory) as long as the DSB is enabled Restriction : 1. The address is cacheline aligned (HW enforced) 2. The address should be pointing to the cacheline **after** the last cacheline of the state buffer (i.e. Tail Pointer = Number of State Buffer cachelines + 1) 3. This address should be greater than the Head Pointer address 4. Software should not modify this register while the DSB engine is busy
	5:0	**Reserved** Access: RO Format: MBZ

DSC_CRC_CTL

DSC_CRC_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	7802Ch-7802Fh
Name:	DSC_CRC_CTL
ShortName:	DSC_CRC_CTL_DSC0_PA
Reset:	soft
Address:	7812Ch-7812Fh
Name:	DSC_CRC_CTL
ShortName:	DSC_CRC_CTL_DSC1_PA
Reset:	soft
Address:	7822Ch-7822Fh
Name:	DSC_CRC_CTL
ShortName:	DSC_CRC_CTL_DSC0_PB
Reset:	soft
Address:	7832Ch-7832Fh
Name:	DSC_CRC_CTL
ShortName:	DSC_CRC_CTL_DSC1_PB
Reset:	soft
Address:	7842Ch-7842Fh
Name:	DSC_CRC_CTL
ShortName:	DSC_CRC_CTL_DSC0_PC
Reset:	soft
Address:	7852Ch-7852Fh
Name:	DSC_CRC_CTL
ShortName:	DSC_CRC_CTL_DSC1_PC
Reset:	soft
Address:	7862Ch-7862Fh
Name:	DSC_CRC_CTL
ShortName:	DSC_CRC_CTL_DSC0_PD
Reset:	soft
Address:	7872Ch-7872Fh
Name:	DSC_CRC_CTL

DSC_CRC_CTL

ShortName: DSC_CRC_CTL_DSC1_PD

Reset: soft

DWord	Bit	Description
0	31	**Enable CRC** Access: R/W Enables the CRC calculations. The CRC will give a done indication and a new result at the end of each frame. Value: 0b — Name: Disable Value: 1b — Name: Enable
	30	**CRC Done** Access: R/WC This bit is set on the rising edge of the CRC done indication. This is a sticky bit, cleared by writing 1b to it. Value: 0b — Name: Not Done Value: 1b — Name: Done
	29	**CRC Change** Access: R/WC This bit is set if the CRC result value changes from the previous value. This is a sticky bit, cleared by writing 1b to it. Value: 0b — Name: No Change Value: 1b — Name: Change
	28:0	**Reserved** Access: RO Format: MBZ

DSC_CRC_RES

DSC_CRC_RES	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	78030h-78033h
Name:	DSC_CRC_RES
ShortName:	DSC_CRC_RES_DSC0_PA
Reset:	soft
Address:	78130h-78133h
Name:	DSC_CRC_RES
ShortName:	DSC_CRC_RES_DSC1_PA
Reset:	soft
Address:	78230h-78233h
Name:	DSC_CRC_RES
ShortName:	DSC_CRC_RES_DSC0_PB
Reset:	soft
Address:	78330h-78333h
Name:	DSC_CRC_RES
ShortName:	DSC_CRC_RES_DSC1_PB
Reset:	soft
Address:	78430h-78433h
Name:	DSC_CRC_RES
ShortName:	DSC_CRC_RES_DSC0_PC
Reset:	soft
Address:	78530h-78533h
Name:	DSC_CRC_RES
ShortName:	DSC_CRC_RES_DSC1_PC
Reset:	soft
Address:	78630h-78633h
Name:	DSC_CRC_RES
ShortName:	DSC_CRC_RES_DSC0_PD
Reset:	soft
Address:	78730h-78733h
Name:	DSC_CRC_RES

DSC_CRC_RES

ShortName: DSC_CRC_RES_DSC1_PD

Reset: soft

DWord	Bit	Description
0	31:0	**CRC Result Value** This field contains the CRC result at the end of a CRC frame. The CRC done bit indicates when the result is valid.

DSC_PICTURE_PARAMETER_SET_0

DSC_PICTURE_PARAMETER_SET_0	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	78070h-78073h
Name:	DSC_PICTURE_PARAMETER_SET_0
ShortName:	DSC_PICTURE_PARAMETER_SET_0_DSC0_PA
Reset:	soft
Address:	78170h-78173h
Name:	DSC_PICTURE_PARAMETER_SET_0
ShortName:	DSC_PICTURE_PARAMETER_SET_0_DSC1_PA
Reset:	soft
Address:	78270h-78273h
Name:	DSC_PICTURE_PARAMETER_SET_0
ShortName:	DSC_PICTURE_PARAMETER_SET_0_DSC0_PB
Reset:	soft
Address:	78370h-78373h
Name:	DSC_PICTURE_PARAMETER_SET_0
ShortName:	DSC_PICTURE_PARAMETER_SET_0_DSC1_PB
Reset:	soft
Address:	78470h-78473h
Name:	DSC_PICTURE_PARAMETER_SET_0
ShortName:	DSC_PICTURE_PARAMETER_SET_0_DSC0_PC
Reset:	soft
Address:	78570h-78573h
Name:	DSC_PICTURE_PARAMETER_SET_0
ShortName:	DSC_PICTURE_PARAMETER_SET_0_DSC1_PC
Reset:	soft
Address:	78670h-78673h
Name:	DSC_PICTURE_PARAMETER_SET_0
ShortName:	DSC_PICTURE_PARAMETER_SET_0_DSC0_PD
Reset:	soft
Address:	78770h-78773h
Name:	DSC_PICTURE_PARAMETER_SET_0
ShortName:	DSC_PICTURE_PARAMETER_SET_0_DSC1_PD

DSC_PICTURE_PARAMETER_SET_0

Reset: soft

DWord	Bit	Description
0	31	**Allow Double Buffer Update Disable** Access: R/W This field controls whether double buffer updates are allowed to be disabled for this instance of DSC. The DOUBLE_BUFFER_CTL register can be configured to globally disable double buffer updates for resources that allow them to be disabled. NOTE: This bit itself is not double buffered Value: 0b – Not Allowed Value: 1b – Allowed
	30:26	**Reserved** Access: RO Format: MBZ
	25:20	**Reserved** Access: RO Format: MBZ
	19	**vbr_enable** Access: R/W Restriction : DSC variable bit rate mode is not supported. Value: 0b – Name: disable **[Default]** – Description: 0 padding bits are stuffed at the end of a slice to ensure that the total number of bits within the slice is equal to the slice bit budget. Value: 1b – Name: Enable – Description: Bit stuffing is bypassed
	18	**enable_422** Access: R/W This bit value shall be 0 if native_422 or native_420 is set to 1. DSC standard calls this mode as simple_422. This mode is currently not supported. Value: 0b – Name: 444 **[Default]** – Description: Input uses 4:4:4 sampling. Decoder does not drop samples to reconstruct a 4:2:2 picture Value: 1b – Name: 422 – Description: Input uses 4:2:2 sampling. Decoder drops samples to reconstruct a 4:2:2 picture.
	17	**convert_rgb** Access: R/W Indicates whether DSC color space conversion is active. Value: 0b – Name: YCbCr – Description: Color space is YCbCr

DSC_PICTURE_PARAMETER_SET_0

1b	convert_rgb	Encoder converts RGB to YCoCg-R, and decoder converts YCoCg-R to RGB.

16 block_pred_enable

Access:	R/W

Value	Name	Description
0b	disable	BP is not used to code any groups within the picture
1b	enable	Decoder must select between BP and MMAP

15:12 linebuf_depth

Access:	R/W

Contains the line buffer bit depth used to generate the bitstream. If a components bit depth (after color space conversion) is greater than this value, the line storage rounds the reconstructed values to this number of bits.
0x8 = 8 bits
0x9 = 9 bits
0xA = 10 bits
0xB = 11 bits
0xC = 12 bits
0xD = 13 bits
All other encodings are RESERVED

11:8 bits_per_component

Access:	R/W

Indicates the number of bits per component for the original pixels of the encoded picture.
0x8 = 8bpc
0xA = 10bpc
0xC = 12bpc
All other encodings are RESERVED

7:4 dsc_version_minor

Access:	R/W

Description
Contains the major version of DSC.

3:0 dsc_version_major

Access:	R/W

Contains the major version of DSC.
0x1 = Encoder implements DSC

DSC_PICTURE_PARAMETER_SET_1

DSC_PICTURE_PARAMETER_SET_1	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	78074h-78077h
Name:	DSC_PICTURE_PARAMETER_SET_1
ShortName:	DSC_PICTURE_PARAMETER_SET_1_DSC0_PA
Reset:	soft
Address:	78174h-78177h
Name:	DSC_PICTURE_PARAMETER_SET_1
ShortName:	DSC_PICTURE_PARAMETER_SET_1_DSC1_PA
Reset:	soft
Address:	78274h-78277h
Name:	DSC_PICTURE_PARAMETER_SET_1
ShortName:	DSC_PICTURE_PARAMETER_SET_1_DSC0_PB
Reset:	soft
Address:	78374h-78377h
Name:	DSC_PICTURE_PARAMETER_SET_1
ShortName:	DSC_PICTURE_PARAMETER_SET_1_DSC1_PB
Reset:	soft
Address:	78474h-78477h
Name:	DSC_PICTURE_PARAMETER_SET_1
ShortName:	DSC_PICTURE_PARAMETER_SET_1_DSC0_PC
Reset:	soft
Address:	78574h-78577h
Name:	DSC_PICTURE_PARAMETER_SET_1
ShortName:	DSC_PICTURE_PARAMETER_SET_1_DSC1_PC
Reset:	soft
Address:	78674h-78677h
Name:	DSC_PICTURE_PARAMETER_SET_1
ShortName:	DSC_PICTURE_PARAMETER_SET_1_DSC0_PD
Reset:	soft
Address:	78774h-78777h
Name:	DSC_PICTURE_PARAMETER_SET_1

DSC_PICTURE_PARAMETER_SET_1

ShortName: DSC_PICTURE_PARAMETER_SET_1_DSC1_PD

Reset: soft

DWord	Bit	Description
0	31:20	**Reserved** Access: RO Format: MBZ
	19:10	**Reserved** Access: RO Format: MBZ
	9:0	**bits_per_pixel** Access: R/W Specifies the target bits/pixel (bpp) rate that is used by the encoder, in steps of 1 bit per pixel. Only values greater than or equal to 6.0 are allowed. If vbr_enable is cleared to 0, this value must be less than or equal to the sustained rate that would apply if MPP is always selected with QP = 0, which is a function of bits_per_component, convert_rgb, and rc_range_parameters[0]. If native_422 or native_420 is set to 1, this value shall be programmed to double the target bits per pixel rate.

DSC_PICTURE_PARAMETER_SET_2

DSC_PICTURE_PARAMETER_SET_2	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	78078h-7807Bh
Name:	DSC_PICTURE_PARAMETER_SET_2
ShortName:	DSC_PICTURE_PARAMETER_SET_2_DSC0_PA
Reset:	soft
Address:	78178h-7817Bh
Name:	DSC_PICTURE_PARAMETER_SET_2
ShortName:	DSC_PICTURE_PARAMETER_SET_2_DSC1_PA
Reset:	soft
Address:	78278h-7827Bh
Name:	DSC_PICTURE_PARAMETER_SET_2
ShortName:	DSC_PICTURE_PARAMETER_SET_2_DSC0_PB
Reset:	soft
Address:	78378h-7837Bh
Name:	DSC_PICTURE_PARAMETER_SET_2
ShortName:	DSC_PICTURE_PARAMETER_SET_2_DSC1_PB
Reset:	soft
Address:	78478h-7847Bh
Name:	DSC_PICTURE_PARAMETER_SET_2
ShortName:	DSC_PICTURE_PARAMETER_SET_2_DSC0_PC
Reset:	soft
Address:	78578h-7857Bh
Name:	DSC_PICTURE_PARAMETER_SET_2
ShortName:	DSC_PICTURE_PARAMETER_SET_2_DSC1_PC
Reset:	soft
Address:	78678h-7867Bh
Name:	DSC_PICTURE_PARAMETER_SET_2
ShortName:	DSC_PICTURE_PARAMETER_SET_2_DSC0_PD
Reset:	soft
Address:	78778h-7877Bh
Name:	DSC_PICTURE_PARAMETER_SET_2

DSC_PICTURE_PARAMETER_SET_2

ShortName: DSC_PICTURE_PARAMETER_SET_2_DSC1_PD

Reset: soft

DWord	Bit	Description
0	31:16	**pic_width** Access: Double Buffered **Description** Double buffer update will happen at the standard pipe double buffer point. On a single pipe if we are using 1 VDSC instance, picture_width of that VDSC instance = input frame picture_width. On a single pipe if we are using 2 VDSC instances, picture_width of each instance =input frame picture_width divided by 2. In the case a full frame is processed by2*N VDSC instances from N pipes, picture_width of each instance = input framepicture_width divided by 2*N.
	15:0	**pic_height** Access: Double Buffered **Description** Double buffer update will happen at the standard pipe double buffer point. This field is always programmed to input frame picture height.

DSC_PICTURE_PARAMETER_SET_3

DSC_PICTURE_PARAMETER_SET_3	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	7807Ch-7807Fh
Name:	DSC_PICTURE_PARAMETER_SET_3
ShortName:	DSC_PICTURE_PARAMETER_SET_3_DSC0_PA
Reset:	soft
Address:	7817Ch-7817Fh
Name:	DSC_PICTURE_PARAMETER_SET_3
ShortName:	DSC_PICTURE_PARAMETER_SET_3_DSC1_PA
Reset:	soft
Address:	7827Ch-7827Fh
Name:	DSC_PICTURE_PARAMETER_SET_3
ShortName:	DSC_PICTURE_PARAMETER_SET_3_DSC0_PB
Reset:	soft
Address:	7837Ch-7837Fh
Name:	DSC_PICTURE_PARAMETER_SET_3
ShortName:	DSC_PICTURE_PARAMETER_SET_3_DSC1_PB
Reset:	soft
Address:	7847Ch-7847Fh
Name:	DSC_PICTURE_PARAMETER_SET_3
ShortName:	DSC_PICTURE_PARAMETER_SET_3_DSC0_PC
Reset:	soft
Address:	7857Ch-7857Fh
Name:	DSC_PICTURE_PARAMETER_SET_3
ShortName:	DSC_PICTURE_PARAMETER_SET_3_DSC1_PC
Reset:	soft
Address:	7867Ch-7867Fh
Name:	DSC_PICTURE_PARAMETER_SET_3
ShortName:	DSC_PICTURE_PARAMETER_SET_3_DSC0_PD
Reset:	soft
Address:	7877Ch-7877Fh
Name:	DSC_PICTURE_PARAMETER_SET_3

DSC_PICTURE_PARAMETER_SET_3

ShortName: DSC_PICTURE_PARAMETER_SET_3_DSC1_PD

Reset: soft

DWord	Bit	Description
0	31:16	**slice_width** Access: R/W This defines the width of the slice in number of pixels.
	15:0	**slice_height** Access: R/W This defines the height of the slice in number of pixels.

DSC_PICTURE_PARAMETER_SET_4

DSC_PICTURE_PARAMETER_SET_4	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	78080h-78083h
Name:	DSC_PICTURE_PARAMETER_SET_4
ShortName:	DSC_PICTURE_PARAMETER_SET_4_DSC0_PA
Reset:	soft
Address:	78180h-78183h
Name:	DSC_PICTURE_PARAMETER_SET_4
ShortName:	DSC_PICTURE_PARAMETER_SET_4_DSC1_PA
Reset:	soft
Address:	78280h-78283h
Name:	DSC_PICTURE_PARAMETER_SET_4
ShortName:	DSC_PICTURE_PARAMETER_SET_4_DSC0_PB
Reset:	soft
Address:	78380h-78383h
Name:	DSC_PICTURE_PARAMETER_SET_4
ShortName:	DSC_PICTURE_PARAMETER_SET_4_DSC1_PB
Reset:	soft
Address:	78480h-78483h
Name:	DSC_PICTURE_PARAMETER_SET_4
ShortName:	DSC_PICTURE_PARAMETER_SET_4_DSC0_PC
Reset:	soft
Address:	78580h-78583h
Name:	DSC_PICTURE_PARAMETER_SET_4
ShortName:	DSC_PICTURE_PARAMETER_SET_4_DSC1_PC
Reset:	soft
Address:	78680h-78683h
Name:	DSC_PICTURE_PARAMETER_SET_4
ShortName:	DSC_PICTURE_PARAMETER_SET_4_DSC0_PD
Reset:	soft
Address:	78780h-78783h
Name:	DSC_PICTURE_PARAMETER_SET_4
ShortName:	DSC_PICTURE_PARAMETER_SET_4_DSC1_PD

DSC_PICTURE_PARAMETER_SET_4		
Reset: soft		
DWord	**Bit**	**Description**
0	31:16	**initial_dec_delay** Access: R/W
	15:10	**Reserved** Access: RO Format: MBZ
	9:0	**initial_xmit_delay** Access: R/W

DSC_PICTURE_PARAMETER_SET_5

DSC_PICTURE_PARAMETER_SET_5	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	78084h-78087h
Name:	DSC_PICTURE_PARAMETER_SET_5
ShortName:	DSC_PICTURE_PARAMETER_SET_5_DSC0_PA
Reset:	soft
Address:	78184h-78187h
Name:	DSC_PICTURE_PARAMETER_SET_5
ShortName:	DSC_PICTURE_PARAMETER_SET_5_DSC1_PA
Reset:	soft
Address:	78284h-78287h
Name:	DSC_PICTURE_PARAMETER_SET_5
ShortName:	DSC_PICTURE_PARAMETER_SET_5_DSC0_PB
Reset:	soft
Address:	78384h-78387h
Name:	DSC_PICTURE_PARAMETER_SET_5
ShortName:	DSC_PICTURE_PARAMETER_SET_5_DSC1_PB
Reset:	soft
Address:	78484h-78487h
Name:	DSC_PICTURE_PARAMETER_SET_5
ShortName:	DSC_PICTURE_PARAMETER_SET_5_DSC0_PC
Reset:	soft
Address:	78584h-78587h
Name:	DSC_PICTURE_PARAMETER_SET_5
ShortName:	DSC_PICTURE_PARAMETER_SET_5_DSC1_PC
Reset:	soft
Address:	78684h-78687h
Name:	DSC_PICTURE_PARAMETER_SET_5
ShortName:	DSC_PICTURE_PARAMETER_SET_5_DSC0_PD
Reset:	soft
Address:	78784h-78787h
Name:	DSC_PICTURE_PARAMETER_SET_5
ShortName:	DSC_PICTURE_PARAMETER_SET_5_DSC1_PD

DSC_PICTURE_PARAMETER_SET_5

Reset: soft

DWord	Bit	Description
0	31:28	**Reserved** Access: RO Format: MBZ
	27:16	**scale_decrement_interval** Access: R/W
	15:0	**scale_increment_interval** Access: R/W

DSC_PICTURE_PARAMETER_SET_6

DSC_PICTURE_PARAMETER_SET_6	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	78088h-7808Bh
Name:	DSC_PICTURE_PARAMETER_SET_6
ShortName:	DSC_PICTURE_PARAMETER_SET_6_DSC0_PA
Reset:	soft
Address:	78188h-7818Bh
Name:	DSC_PICTURE_PARAMETER_SET_6
ShortName:	DSC_PICTURE_PARAMETER_SET_6_DSC1_PA
Reset:	soft
Address:	78288h-7828Bh
Name:	DSC_PICTURE_PARAMETER_SET_6
ShortName:	DSC_PICTURE_PARAMETER_SET_6_DSC0_PB
Reset:	soft
Address:	78388h-7838Bh
Name:	DSC_PICTURE_PARAMETER_SET_6
ShortName:	DSC_PICTURE_PARAMETER_SET_6_DSC1_PB
Reset:	soft
Address:	78488h-7848Bh
Name:	DSC_PICTURE_PARAMETER_SET_6
ShortName:	DSC_PICTURE_PARAMETER_SET_6_DSC0_PC
Reset:	soft
Address:	78588h-7858Bh
Name:	DSC_PICTURE_PARAMETER_SET_6
ShortName:	DSC_PICTURE_PARAMETER_SET_6_DSC1_PC
Reset:	soft
Address:	78688h-7868Bh
Name:	DSC_PICTURE_PARAMETER_SET_6
ShortName:	DSC_PICTURE_PARAMETER_SET_6_DSC0_PD
Reset:	soft
Address:	78788h-7878Bh
Name:	DSC_PICTURE_PARAMETER_SET_6
ShortName:	DSC_PICTURE_PARAMETER_SET_6_DSC1_PD

DSC_PICTURE_PARAMETER_SET_6

Reset: soft

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:24	**flatness_max_qp** Access: R/W
	23:21	**Reserved** Access: RO Format: MBZ
	20:16	**flatness_min_qp** Access: R/W
	15:13	**Reserved** Access: RO Format: MBZ
	12:8	**first_line_bpg_offset** Access: R/W
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**initial_scale_value** Access: R/W Specifies the initial rcXformScale factor value used at the beginning of a slice. This is an unsigned field with three fractional bits.

DSC_PICTURE_PARAMETER_SET_7

DSC_PICTURE_PARAMETER_SET_7	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	7808Ch-7808Fh
Name:	DSC_PICTURE_PARAMETER_SET_7
ShortName:	DSC_PICTURE_PARAMETER_SET_7_DSC0_PA
Reset:	soft
Address:	7818Ch-7818Fh
Name:	DSC_PICTURE_PARAMETER_SET_7
ShortName:	DSC_PICTURE_PARAMETER_SET_7_DSC1_PA
Reset:	soft
Address:	7828Ch-7828Fh
Name:	DSC_PICTURE_PARAMETER_SET_7
ShortName:	DSC_PICTURE_PARAMETER_SET_7_DSC0_PB
Reset:	soft
Address:	7838Ch-7838Fh
Name:	DSC_PICTURE_PARAMETER_SET_7
ShortName:	DSC_PICTURE_PARAMETER_SET_7_DSC1_PB
Reset:	soft
Address:	7848Ch-7848Fh
Name:	DSC_PICTURE_PARAMETER_SET_7
ShortName:	DSC_PICTURE_PARAMETER_SET_7_DSC0_PC
Reset:	soft
Address:	7858Ch-7858Fh
Name:	DSC_PICTURE_PARAMETER_SET_7
ShortName:	DSC_PICTURE_PARAMETER_SET_7_DSC1_PC
Reset:	soft
Address:	7868Ch-7868Fh
Name:	DSC_PICTURE_PARAMETER_SET_7
ShortName:	DSC_PICTURE_PARAMETER_SET_7_DSC0_PD
Reset:	soft
Address:	7878Ch-7878Fh
Name:	DSC_PICTURE_PARAMETER_SET_7
ShortName:	DSC_PICTURE_PARAMETER_SET_7_DSC1_PD

DSC_PICTURE_PARAMETER_SET_7		
Reset:	soft	
DWord	**Bit**	**Description**
0	31:16	**nfl_bpg_offset** Access: R/W Specifies the number of bits (including fractional bits) that are deallocated for each group, for groups after the first line of a slice. This is an unsigned value with 11 fractional bits.
	15:0	**slice_bpg_offset** Access: R/W Specifies the number of bits (including fractional bits) that are deallocated for each group to enforce the slice constraint (i.e., the final buffer model fullness cannot exceed the initial transmission delay times bits per group), while allowing a programmable initial_offset. This is an unsigned value with 11 fractional bits.

DSC_PICTURE_PARAMETER_SET_8

DSC_PICTURE_PARAMETER_SET_8	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	78090h-78093h
Name:	DSC_PICTURE_PARAMETER_SET_8
ShortName:	DSC_PICTURE_PARAMETER_SET_8_DSC0_PA
Reset:	soft
Address:	78190h-78193h
Name:	DSC_PICTURE_PARAMETER_SET_8
ShortName:	DSC_PICTURE_PARAMETER_SET_8_DSC1_PA
Reset:	soft
Address:	78290h-78293h
Name:	DSC_PICTURE_PARAMETER_SET_8
ShortName:	DSC_PICTURE_PARAMETER_SET_8_DSC0_PB
Reset:	soft
Address:	78390h-78393h
Name:	DSC_PICTURE_PARAMETER_SET_8
ShortName:	DSC_PICTURE_PARAMETER_SET_8_DSC1_PB
Reset:	soft
Address:	78490h-78493h
Name:	DSC_PICTURE_PARAMETER_SET_8
ShortName:	DSC_PICTURE_PARAMETER_SET_8_DSC0_PC
Reset:	soft
Address:	78590h-78593h
Name:	DSC_PICTURE_PARAMETER_SET_8
ShortName:	DSC_PICTURE_PARAMETER_SET_8_DSC1_PC
Reset:	soft
Address:	78690h-78693h
Name:	DSC_PICTURE_PARAMETER_SET_8
ShortName:	DSC_PICTURE_PARAMETER_SET_8_DSC0_PD
Reset:	soft
Address:	78790h-78793h
Name:	DSC_PICTURE_PARAMETER_SET_8
ShortName:	DSC_PICTURE_PARAMETER_SET_8_DSC1_PD

DSC_PICTURE_PARAMETER_SET_8		
Reset: soft		
DWord	**Bit**	**Description**
0	31:16	**initial_offset** Access: R/W
	15:0	**final_offset** Access: R/W

DSC_PICTURE_PARAMETER_SET_9

DSC_PICTURE_PARAMETER_SET_9	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	78094h-78097h
Name:	DSC_PICTURE_PARAMETER_SET_9
ShortName:	DSC_PICTURE_PARAMETER_SET_9_DSC0_PA
Reset:	soft
Address:	78194h-78197h
Name:	DSC_PICTURE_PARAMETER_SET_9
ShortName:	DSC_PICTURE_PARAMETER_SET_9_DSC1_PA
Reset:	soft
Address:	78294h-78297h
Name:	DSC_PICTURE_PARAMETER_SET_9
ShortName:	DSC_PICTURE_PARAMETER_SET_9_DSC0_PB
Reset:	soft
Address:	78394h-78397h
Name:	DSC_PICTURE_PARAMETER_SET_9
ShortName:	DSC_PICTURE_PARAMETER_SET_9_DSC1_PB
Reset:	soft
Address:	78494h-78497h
Name:	DSC_PICTURE_PARAMETER_SET_9
ShortName:	DSC_PICTURE_PARAMETER_SET_9_DSC0_PC
Reset:	soft
Address:	78594h-78597h
Name:	DSC_PICTURE_PARAMETER_SET_9
ShortName:	DSC_PICTURE_PARAMETER_SET_9_DSC1_PC
Reset:	soft
Address:	78694h-78697h
Name:	DSC_PICTURE_PARAMETER_SET_9
ShortName:	DSC_PICTURE_PARAMETER_SET_9_DSC0_PD
Reset:	soft
Address:	78794h-78797h
Name:	DSC_PICTURE_PARAMETER_SET_9
ShortName:	DSC_PICTURE_PARAMETER_SET_9_DSC1_PD

DSC_PICTURE_PARAMETER_SET_9

Reset: soft

DWord	Bit	Description
0	31:20	**Reserved** Access: RO Format: MBZ
	19:16	**rc_edge_factor** Access: R/W
	15:0	**rc_model_Size** Access: R/W

DSC_PICTURE_PARAMETER_SET_10

DSC_PICTURE_PARAMETER_SET_10	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	78098h-7809Bh
Name:	DSC_PICTURE_PARAMETER_SET_10
ShortName:	DSC_PICTURE_PARAMETER_SET_10_DSC0_PA
Reset:	soft
Address:	78198h-7819Bh
Name:	DSC_PICTURE_PARAMETER_SET_10
ShortName:	DSC_PICTURE_PARAMETER_SET_10_DSC1_PA
Reset:	soft
Address:	78298h-7829Bh
Name:	DSC_PICTURE_PARAMETER_SET_10
ShortName:	DSC_PICTURE_PARAMETER_SET_10_DSC0_PB
Reset:	soft
Address:	78398h-7839Bh
Name:	DSC_PICTURE_PARAMETER_SET_10
ShortName:	DSC_PICTURE_PARAMETER_SET_10_DSC1_PB
Reset:	soft
Address:	78498h-7849Bh
Name:	DSC_PICTURE_PARAMETER_SET_10
ShortName:	DSC_PICTURE_PARAMETER_SET_10_DSC0_PC
Reset:	soft
Address:	78598h-7859Bh
Name:	DSC_PICTURE_PARAMETER_SET_10
ShortName:	DSC_PICTURE_PARAMETER_SET_10_DSC1_PC
Reset:	soft
Address:	78698h-7869Bh
Name:	DSC_PICTURE_PARAMETER_SET_10
ShortName:	DSC_PICTURE_PARAMETER_SET_10_DSC0_PD
Reset:	soft
Address:	78798h-7879Bh
Name:	DSC_PICTURE_PARAMETER_SET_10
ShortName:	DSC_PICTURE_PARAMETER_SET_10_DSC1_PD

DSC_PICTURE_PARAMETER_SET_10

Reset: soft

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23:20	**rc_tgt_offset_lo** Access: R/W
	19:16	**rc_tgt_offset_hi** Access: R/W
	15:13	**Reserved** Access: RO Format: MBZ
	12:8	**rc_quant_incr_limit1** Access: R/W
	7:5	**Reserved** Access: RO Format: MBZ
	4:0	**rc_quant_incr_limit0** Access: R/W

DSC_PICTURE_PARAMETER_SET_11

DSC_PICTURE_PARAMETER_SET_11	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	7809Ch-7809Fh
Name:	DSC_PICTURE_PARAMETER_SET_11
ShortName:	DSC_PICTURE_PARAMETER_SET_11_DSC0_PA
Reset:	soft
Address:	7819Ch-7819Fh
Name:	DSC_PICTURE_PARAMETER_SET_11
ShortName:	DSC_PICTURE_PARAMETER_SET_11_DSC1_PA
Reset:	soft
Address:	7829Ch-7829Fh
Name:	DSC_PICTURE_PARAMETER_SET_11
ShortName:	DSC_PICTURE_PARAMETER_SET_11_DSC0_PB
Reset:	soft
Address:	7839Ch-7839Fh
Name:	DSC_PICTURE_PARAMETER_SET_11
ShortName:	DSC_PICTURE_PARAMETER_SET_11_DSC1_PB
Reset:	soft
Address:	7849Ch-7849Fh
Name:	DSC_PICTURE_PARAMETER_SET_11
ShortName:	DSC_PICTURE_PARAMETER_SET_11_DSC0_PC
Reset:	soft
Address:	7859Ch-7859Fh
Name:	DSC_PICTURE_PARAMETER_SET_11
ShortName:	DSC_PICTURE_PARAMETER_SET_11_DSC1_PC
Reset:	soft
Address:	7869Ch-7869Fh
Name:	DSC_PICTURE_PARAMETER_SET_11
ShortName:	DSC_PICTURE_PARAMETER_SET_11_DSC0_PD
Reset:	soft
Address:	7879Ch-7879Fh
Name:	DSC_PICTURE_PARAMETER_SET_11
ShortName:	DSC_PICTURE_PARAMETER_SET_11_DSC1_PD

DSC_PICTURE_PARAMETER_SET_11		
Reset: soft		
DWord	**Bit**	**Description**
0	31:0	**Reserved** Access: RO Format: MBZ

DSC_PICTURE_PARAMETER_SET_12

DSC_PICTURE_PARAMETER_SET_12	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	780A0h-780A3h
Name:	DSC_PICTURE_PARAMETER_SET_12
ShortName:	DSC_PICTURE_PARAMETER_SET_12_DSC0_PA
Reset:	soft
Address:	781A0h-781A3h
Name:	DSC_PICTURE_PARAMETER_SET_12
ShortName:	DSC_PICTURE_PARAMETER_SET_12_DSC1_PA
Reset:	soft
Address:	782A0h-782A3h
Name:	DSC_PICTURE_PARAMETER_SET_12
ShortName:	DSC_PICTURE_PARAMETER_SET_12_DSC0_PB
Reset:	soft
Address:	783A0h-783A3h
Name:	DSC_PICTURE_PARAMETER_SET_12
ShortName:	DSC_PICTURE_PARAMETER_SET_12_DSC1_PB
Reset:	soft
Address:	784A0h-784A3h
Name:	DSC_PICTURE_PARAMETER_SET_12
ShortName:	DSC_PICTURE_PARAMETER_SET_12_DSC0_PC
Reset:	soft
Address:	785A0h-785A3h
Name:	DSC_PICTURE_PARAMETER_SET_12
ShortName:	DSC_PICTURE_PARAMETER_SET_12_DSC1_PC
Reset:	soft
Address:	786A0h-786A3h
Name:	DSC_PICTURE_PARAMETER_SET_12
ShortName:	DSC_PICTURE_PARAMETER_SET_12_DSC0_PD
Reset:	soft
Address:	787A0h-787A3h
Name:	DSC_PICTURE_PARAMETER_SET_12
ShortName:	DSC_PICTURE_PARAMETER_SET_12_DSC1_PD

DSC_PICTURE_PARAMETER_SET_12		
Reset: soft		
DWord	**Bit**	**Description**
0	31:0	**Reserved** Access: RO Format: MBZ

DSC_PICTURE_PARAMETER_SET_13

DSC_PICTURE_PARAMETER_SET_13	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	780A4h-780A7h
Name:	DSC_PICTURE_PARAMETER_SET_13
ShortName:	DSC_PICTURE_PARAMETER_SET_13_DSC0_PA
Reset:	soft
Address:	781A4h-781A7h
Name:	DSC_PICTURE_PARAMETER_SET_13
ShortName:	DSC_PICTURE_PARAMETER_SET_13_DSC1_PA
Reset:	soft
Address:	782A4h-782A7h
Name:	DSC_PICTURE_PARAMETER_SET_13
ShortName:	DSC_PICTURE_PARAMETER_SET_13_DSC0_PB
Reset:	soft
Address:	783A4h-783A7h
Name:	DSC_PICTURE_PARAMETER_SET_13
ShortName:	DSC_PICTURE_PARAMETER_SET_13_DSC1_PB
Reset:	soft
Address:	784A4h-784A7h
Name:	DSC_PICTURE_PARAMETER_SET_13
ShortName:	DSC_PICTURE_PARAMETER_SET_13_DSC0_PC
Reset:	soft
Address:	785A4h-785A7h
Name:	DSC_PICTURE_PARAMETER_SET_13
ShortName:	DSC_PICTURE_PARAMETER_SET_13_DSC1_PC
Reset:	soft
Address:	786A4h-786A7h
Name:	DSC_PICTURE_PARAMETER_SET_13
ShortName:	DSC_PICTURE_PARAMETER_SET_13_DSC0_PD
Reset:	soft
Address:	787A4h-787A7h
Name:	DSC_PICTURE_PARAMETER_SET_13
ShortName:	DSC_PICTURE_PARAMETER_SET_13_DSC1_PD

DSC_PICTURE_PARAMETER_SET_13		
Reset: soft		
DWord	**Bit**	**Description**
0	31:0	**Reserved** Access: RO Format: MBZ

DSC_PICTURE_PARAMETER_SET_14

DSC_PICTURE_PARAMETER_SET_14	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	780A8h-780ABh
Name:	DSC_PICTURE_PARAMETER_SET_14
ShortName:	DSC_PICTURE_PARAMETER_SET_14_DSC0_PA
Reset:	soft
Address:	781A8h-781ABh
Name:	DSC_PICTURE_PARAMETER_SET_14
ShortName:	DSC_PICTURE_PARAMETER_SET_14_DSC1_PA
Reset:	soft
Address:	782A8h-782ABh
Name:	DSC_PICTURE_PARAMETER_SET_14
ShortName:	DSC_PICTURE_PARAMETER_SET_14_DSC0_PB
Reset:	soft
Address:	783A8h-783ABh
Name:	DSC_PICTURE_PARAMETER_SET_14
ShortName:	DSC_PICTURE_PARAMETER_SET_14_DSC1_PB
Reset:	soft
Address:	784A8h-784ABh
Name:	DSC_PICTURE_PARAMETER_SET_14
ShortName:	DSC_PICTURE_PARAMETER_SET_14_DSC0_PC
Reset:	soft
Address:	785A8h-785ABh
Name:	DSC_PICTURE_PARAMETER_SET_14
ShortName:	DSC_PICTURE_PARAMETER_SET_14_DSC1_PC
Reset:	soft
Address:	786A8h-786ABh
Name:	DSC_PICTURE_PARAMETER_SET_14
ShortName:	DSC_PICTURE_PARAMETER_SET_14_DSC0_PD
Reset:	soft
Address:	787A8h-787ABh
Name:	DSC_PICTURE_PARAMETER_SET_14
ShortName:	DSC_PICTURE_PARAMETER_SET_14_DSC1_PD

DSC_PICTURE_PARAMETER_SET_14		
Reset: soft		
DWord	**Bit**	**Description**
0	31:0	**Reserved** Access: RO Format: MBZ

DSC_PICTURE_PARAMETER_SET_15

DSC_PICTURE_PARAMETER_SET_15	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	780ACh-780AFh
Name:	DSC_PICTURE_PARAMETER_SET_15
ShortName:	DSC_PICTURE_PARAMETER_SET_15_DSC0_PA
Reset:	soft
Address:	781ACh-781AFh
Name:	DSC_PICTURE_PARAMETER_SET_15
ShortName:	DSC_PICTURE_PARAMETER_SET_15_DSC1_PA
Reset:	soft
Address:	782ACh-782AFh
Name:	DSC_PICTURE_PARAMETER_SET_15
ShortName:	DSC_PICTURE_PARAMETER_SET_15_DSC0_PB
Reset:	soft
Address:	783ACh-783AFh
Name:	DSC_PICTURE_PARAMETER_SET_15
ShortName:	DSC_PICTURE_PARAMETER_SET_15_DSC1_PB
Reset:	soft
Address:	784ACh-784AFh
Name:	DSC_PICTURE_PARAMETER_SET_15
ShortName:	DSC_PICTURE_PARAMETER_SET_15_DSC0_PC
Reset:	soft
Address:	785ACh-785AFh
Name:	DSC_PICTURE_PARAMETER_SET_15
ShortName:	DSC_PICTURE_PARAMETER_SET_15_DSC1_PC
Reset:	soft
Address:	786ACh-786AFh
Name:	DSC_PICTURE_PARAMETER_SET_15
ShortName:	DSC_PICTURE_PARAMETER_SET_15_DSC0_PD
Reset:	soft
Address:	787ACh-787AFh
Name:	DSC_PICTURE_PARAMETER_SET_15
ShortName:	DSC_PICTURE_PARAMETER_SET_15_DSC1_PD

DSC_PICTURE_PARAMETER_SET_15		
Reset: soft		
DWord	**Bit**	**Description**
0	31:0	**Reserved** Access: RO Format: MBZ

DSC_PICTURE_PARAMETER_SET_16

DSC_PICTURE_PARAMETER_SET_16	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	780B0h-780B3h
Name:	DSC_PICTURE_PARAMETER_SET_16
ShortName:	DSC_PICTURE_PARAMETER_SET_16_DSC0_PA
Reset:	soft
Address:	781B0h-781B3h
Name:	DSC_PICTURE_PARAMETER_SET_16
ShortName:	DSC_PICTURE_PARAMETER_SET_16_DSC1_PA
Reset:	soft
Address:	782B0h-782B3h
Name:	DSC_PICTURE_PARAMETER_SET_16
ShortName:	DSC_PICTURE_PARAMETER_SET_16_DSC0_PB
Reset:	soft
Address:	783B0h-783B3h
Name:	DSC_PICTURE_PARAMETER_SET_16
ShortName:	DSC_PICTURE_PARAMETER_SET_16_DSC1_PB
Reset:	soft
Address:	784B0h-784B3h
Name:	DSC_PICTURE_PARAMETER_SET_16
ShortName:	DSC_PICTURE_PARAMETER_SET_16_DSC0_PC
Reset:	soft
Address:	785B0h-785B3h
Name:	DSC_PICTURE_PARAMETER_SET_16
ShortName:	DSC_PICTURE_PARAMETER_SET_16_DSC1_PC
Reset:	soft
Address:	786B0h-786B3h
Name:	DSC_PICTURE_PARAMETER_SET_16
ShortName:	DSC_PICTURE_PARAMETER_SET_16_DSC0_PD
Reset:	soft
Address:	787B0h-787B3h
Name:	DSC_PICTURE_PARAMETER_SET_16
ShortName:	DSC_PICTURE_PARAMETER_SET_16_DSC1_PD

DSC_PICTURE_PARAMETER_SET_16

Reset: soft

DWord	Bit	Description
0	31:20	**slice_row_per_frame** Access: Double Buffered **Description** Double buffer update will happen at the standard pipe double buffer point. This is the field driver will program to indicate slice_row_per_frame for full frame. There is another field in DSC_PICTURE_PARAMETER_SET_1 to indicate slice_row_per_frame based onPSR2 SU region size. This field indicates number of slices stacked in the vertical direction. Example: Input to DSS unit: 3840x2160 to be compressed as 4 slices Input to each VDSC instance: 1920x2160 slice_per_line: 1 slice_row_per_frame: 2
	19	**Reserved** Access: RO Format: MBZ
	18:16	**slice_per_line** Access: R/W Refer to the description under slice_row_per_frame.
	15:0	**slice_chunk_size** Access: R/W

DSC_RC_BUF_THRESH_0

DSC_RC_BUF_THRESH_0	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
Address:	78054h-7805Bh
Name:	DSC_RC_BUF_THRESH_0
ShortName:	DSC_RC_BUF_THRESH_0_DSC0_PA
Reset:	soft
Address:	78154h-7815Bh
Name:	DSC_RC_BUF_THRESH_0
ShortName:	DSC_RC_BUF_THRESH_0_DSC1_PA
Reset:	soft
Address:	78254h-7825Bh
Name:	DSC_RC_BUF_THRESH_0
ShortName:	DSC_RC_BUF_THRESH_0_DSC0_PB
Reset:	soft
Address:	78354h-7835Bh
Name:	DSC_RC_BUF_THRESH_0
ShortName:	DSC_RC_BUF_THRESH_0_DSC1_PB
Reset:	soft
Address:	78454h-7845Bh
Name:	DSC_RC_BUF_THRESH_0
ShortName:	DSC_RC_BUF_THRESH_0_DSC0_PC
Reset:	soft
Address:	78554h-7855Bh
Name:	DSC_RC_BUF_THRESH_0
ShortName:	DSC_RC_BUF_THRESH_0_DSC1_PC
Reset:	soft
Address:	78654h-7865Bh
Name:	DSC_RC_BUF_THRESH_0
ShortName:	DSC_RC_BUF_THRESH_0_DSC0_PD
Reset:	soft
Address:	78754h-7875Bh
Name:	DSC_RC_BUF_THRESH_0
ShortName:	DSC_RC_BUF_THRESH_0_DSC1_PD

DSC_RC_BUF_THRESH_0		
Reset: soft		
DWord	**Bit**	**Description**
0	31:24	**rc_buf_thresh_3** Access: R/W
	23:16	**rc_buf_thresh_2** Access: R/W
	15:8	**rc_buf_thresh_1** Access: R/W
	7:0	**rc_buf_thresh_0** Access: R/W
1	31:24	**rc_buf_thresh_7** Access: R/W
	23:16	**rc_buf_thresh_6** Access: R/W
	15:8	**rc_buf_thresh_5** Access: R/W
	7:0	**rc_buf_thresh_4** Access: R/W

DSC_RC_BUF_THRESH_1

DSC_RC_BUF_THRESH_1	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
Address:	7805Ch-78063h
Name:	DSC_RC_BUF_THRESH_1
ShortName:	DSC_RC_BUF_THRESH_1_DSC0_PA
Reset:	soft
Address:	7815Ch-78163h
Name:	DSC_RC_BUF_THRESH_1
ShortName:	DSC_RC_BUF_THRESH_1_DSC1_PA
Reset:	soft
Address:	7825Ch-78263h
Name:	DSC_RC_BUF_THRESH_1
ShortName:	DSC_RC_BUF_THRESH_1_DSC0_PB
Reset:	soft
Address:	7835Ch-78363h
Name:	DSC_RC_BUF_THRESH_1
ShortName:	DSC_RC_BUF_THRESH_1_DSC1_PB
Reset:	soft
Address:	7845Ch-78463h
Name:	DSC_RC_BUF_THRESH_1
ShortName:	DSC_RC_BUF_THRESH_1_DSC0_PC
Reset:	soft
Address:	7855Ch-78563h
Name:	DSC_RC_BUF_THRESH_1
ShortName:	DSC_RC_BUF_THRESH_1_DSC1_PC
Reset:	soft
Address:	7865Ch-78663h
Name:	DSC_RC_BUF_THRESH_1
ShortName:	DSC_RC_BUF_THRESH_1_DSC0_PD
Reset:	soft
Address:	7875Ch-78763h
Name:	DSC_RC_BUF_THRESH_1
ShortName:	DSC_RC_BUF_THRESH_1_DSC1_PD

DSC_RC_BUF_THRESH_1		
Reset: soft		
DWord	**Bit**	**Description**
0	31:24	**rc_buf_thresh_11** Access: R/W
	23:16	**rc_buf_thresh_10** Access: R/W
	15:8	**rc_buf_thresh_9** Access: R/W
	7:0	**rc_buf_thresh_8** Access: R/W
1	31:16	**Reserved** Access: RO Format: MBZ
	15:8	**rc_buf_thresh_13** Access: R/W
	7:0	**rc_buf_thresh_12** Access: R/W

DSC_RC_RANGE_PARAMETERS_0

DSC_RC_RANGE_PARAMETERS_0	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
Address:	78008h-7800Fh
Name:	DSC_RC_RANGE_PARAMETERS_0
ShortName:	DSC_RC_RANGE_PARAMETERS_0_DSC0_PA
Reset:	soft
Address:	78108h-7810Fh
Name:	DSC_RC_RANGE_PARAMETERS_0
ShortName:	DSC_RC_RANGE_PARAMETERS_0_DSC1_PA
Reset:	soft
Address:	78208h-7820Fh
Name:	DSC_RC_RANGE_PARAMETERS_0
ShortName:	DSC_RC_RANGE_PARAMETERS_0_DSC0_PB
Reset:	soft
Address:	78308h-7830Fh
Name:	DSC_RC_RANGE_PARAMETERS_0
ShortName:	DSC_RC_RANGE_PARAMETERS_0_DSC1_PB
Reset:	soft
Address:	78408h-7840Fh
Name:	DSC_RC_RANGE_PARAMETERS_0
ShortName:	DSC_RC_RANGE_PARAMETERS_0_DSC0_PC
Reset:	soft
Address:	78508h-7850Fh
Name:	DSC_RC_RANGE_PARAMETERS_0
ShortName:	DSC_RC_RANGE_PARAMETERS_0_DSC1_PC
Reset:	soft
Address:	78608h-7860Fh
Name:	DSC_RC_RANGE_PARAMETERS_0
ShortName:	DSC_RC_RANGE_PARAMETERS_0_DSC0_PD
Reset:	soft
Address:	78708h-7870Fh
Name:	DSC_RC_RANGE_PARAMETERS_0
ShortName:	DSC_RC_RANGE_PARAMETERS_0_DSC1_PD

DSC_RC_RANGE_PARAMETERS_0		
Reset: soft		
DWord	**Bit**	**Description**
0	31:26	**rc_bpg_offset_1** Access: R/W
	25:21	**rc_max_qp_1**
	20:16	**rc_min_qp_1**
	15:10	**rc_bpg_offset_0** Access: R/W
	9:5	**rc_max_qp_0**
	4:0	**rc_min_qp_0**
1	31:26	**rc_bpg_offset_3** Access: R/W
	25:21	**rc_max_qp_3**
	20:16	**rc_min_qp_3**
	15:10	**rc_bpg_offset_2** Access: R/W
	9:5	**rc_max_qp_2**
	4:0	**rc_min_qp_2**

DSC_RC_RANGE_PARAMETERS_1

DSC_RC_RANGE_PARAMETERS_1	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
Address:	78010h-78017h
Name:	DSC_RC_RANGE_PARAMETERS_1
ShortName:	DSC_RC_RANGE_PARAMETERS_1_DSC0_PA
Reset:	soft
Address:	78110h-78117h
Name:	DSC_RC_RANGE_PARAMETERS_1
ShortName:	DSC_RC_RANGE_PARAMETERS_1_DSC1_PA
Reset:	soft
Address:	78210h-78217h
Name:	DSC_RC_RANGE_PARAMETERS_1
ShortName:	DSC_RC_RANGE_PARAMETERS_1_DSC0_PB
Reset:	soft
Address:	78310h-78317h
Name:	DSC_RC_RANGE_PARAMETERS_1
ShortName:	DSC_RC_RANGE_PARAMETERS_1_DSC1_PB
Reset:	soft
Address:	78410h-78417h
Name:	DSC_RC_RANGE_PARAMETERS_1
ShortName:	DSC_RC_RANGE_PARAMETERS_1_DSC0_PC
Reset:	soft
Address:	78510h-78517h
Name:	DSC_RC_RANGE_PARAMETERS_1
ShortName:	DSC_RC_RANGE_PARAMETERS_1_DSC1_PC
Reset:	soft
Address:	78610h-78617h
Name:	DSC_RC_RANGE_PARAMETERS_1
ShortName:	DSC_RC_RANGE_PARAMETERS_1_DSC0_PD
Reset:	soft
Address:	78710h-78717h
Name:	DSC_RC_RANGE_PARAMETERS_1
ShortName:	DSC_RC_RANGE_PARAMETERS_1_DSC1_PD

DSC_RC_RANGE_PARAMETERS_1		
Reset: soft		
DWord	**Bit**	**Description**
0	31:26	**rc_bpg_offset_5** Access: R/W
	25:21	**rc_max_qp_5**
	20:16	**rc_min_qp_5**
	15:10	**rc_bpg_offset_4** Access: R/W
	9:5	**rc_max_qp_4**
	4:0	**rc_min_qp_4**
1	31:26	**rc_bpg_offset_7** Access: R/W
	25:21	**rc_max_qp_7**
	20:16	**rc_min_qp_7**
	15:10	**rc_bpg_offset_6** Access: R/W
	9:5	**rc_max_qp_6**
	4:0	**rc_min_qp_6**

DSC_RC_RANGE_PARAMETERS_2

DSC_RC_RANGE_PARAMETERS_2	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
Address:	78018h-7801Fh
Name:	DSC_RC_RANGE_PARAMETERS_2
ShortName:	DSC_RC_RANGE_PARAMETERS_2_DSC0_PA
Reset:	soft
Address:	78118h-7811Fh
Name:	DSC_RC_RANGE_PARAMETERS_2
ShortName:	DSC_RC_RANGE_PARAMETERS_2_DSC1_PA
Reset:	soft
Address:	78218h-7821Fh
Name:	DSC_RC_RANGE_PARAMETERS_2
ShortName:	DSC_RC_RANGE_PARAMETERS_2_DSC0_PB
Reset:	soft
Address:	78318h-7831Fh
Name:	DSC_RC_RANGE_PARAMETERS_2
ShortName:	DSC_RC_RANGE_PARAMETERS_2_DSC1_PB
Reset:	soft
Address:	78418h-7841Fh
Name:	DSC_RC_RANGE_PARAMETERS_2
ShortName:	DSC_RC_RANGE_PARAMETERS_2_DSC0_PC
Reset:	soft
Address:	78518h-7851Fh
Name:	DSC_RC_RANGE_PARAMETERS_2
ShortName:	DSC_RC_RANGE_PARAMETERS_2_DSC1_PC
Reset:	soft
Address:	78618h-7861Fh
Name:	DSC_RC_RANGE_PARAMETERS_2
ShortName:	DSC_RC_RANGE_PARAMETERS_2_DSC0_PD
Reset:	soft
Address:	78718h-7871Fh
Name:	DSC_RC_RANGE_PARAMETERS_2
ShortName:	DSC_RC_RANGE_PARAMETERS_2_DSC1_PD

DSC_RC_RANGE_PARAMETERS_2		
Reset: soft		
DWord	**Bit**	**Description**
0	31:26	**rc_bpg_offset_9** Access: R/W
	25:21	**rc_max_qp_9**
	20:16	**rc_min_qp_9**
	15:10	**rc_bpg_offset_8** Access: R/W
	9:5	**rc_max_qp_8**
	4:0	**rc_min_qp_8**
1	31:26	**rc_bpg_offset_11** Access: R/W
	25:21	**rc_max_qp_11**
	20:16	**rc_min_qp_11**
	15:10	**rc_bpg_offset_10** Access: R/W
	9:5	**rc_max_qp_10**
	4:0	**rc_min_qp_10**

DSC_RC_RANGE_PARAMETERS_3

DSC_RC_RANGE_PARAMETERS_3	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
Address:	78020h-78027h
Name:	DSC_RC_RANGE_PARAMETERS_3
ShortName:	DSC_RC_RANGE_PARAMETERS_3_DSC0_PA
Reset:	soft
Address:	78120h-78127h
Name:	DSC_RC_RANGE_PARAMETERS_3
ShortName:	DSC_RC_RANGE_PARAMETERS_3_DSC1_PA
Reset:	soft
Address:	78220h-78227h
Name:	DSC_RC_RANGE_PARAMETERS_3
ShortName:	DSC_RC_RANGE_PARAMETERS_3_DSC0_PB
Reset:	soft
Address:	78320h-78327h
Name:	DSC_RC_RANGE_PARAMETERS_3
ShortName:	DSC_RC_RANGE_PARAMETERS_3_DSC1_PB
Reset:	soft
Address:	78420h-78427h
Name:	DSC_RC_RANGE_PARAMETERS_3
ShortName:	DSC_RC_RANGE_PARAMETERS_3_DSC0_PC
Reset:	soft
Address:	78520h-78527h
Name:	DSC_RC_RANGE_PARAMETERS_3
ShortName:	DSC_RC_RANGE_PARAMETERS_3_DSC1_PC
Reset:	soft
Address:	78620h-78627h
Name:	DSC_RC_RANGE_PARAMETERS_3
ShortName:	DSC_RC_RANGE_PARAMETERS_3_DSC0_PD
Reset:	soft
Address:	78720h-78727h
Name:	DSC_RC_RANGE_PARAMETERS_3
ShortName:	DSC_RC_RANGE_PARAMETERS_3_DSC1_PD

DSC_RC_RANGE_PARAMETERS_3		
Reset: soft		
DWord	**Bit**	**Description**
0	31:26	**rc_bpg_offset_13** Access: R/W
	25:21	**rc_max_qp_13**
	20:16	**rc_min_qp_13**
	15:10	**rc_bpg_offset_12** Access: R/W
	9:5	**rc_max_qp_12**
	4:0	**rc_min_qp_12**
1	31:16	**Reserved** Access: RO Format: MBZ
	15:10	**rc_bpg_offset_14** Access: R/W
	9:5	**rc_max_qp_14**
	4:0	**rc_min_qp_14**

DSI_CALIB_TO

DSI_CALIB_TO	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	6B050h-6B053h
Name:	DSI 0 Calibration Timeout
ShortName:	DSI_CALIB_TO_0
Reset:	soft
Address:	6B850h-6B853h
Name:	DSI 1 Calibration Timeout
ShortName:	DSI_CALIB_TO_1
Reset:	soft

This register specifies the amount of time that the Host will drive the Link with the HS calibration sequence. The values are specified in Byte clocks and the values specified should be zero based (i.e. value of 1 = 2 Byte clocks, value of 2 = 3 Byte clocks, etc.)

Restriction: The timeout values should be greater than zero if the respective calibration type is enabled.

DWord	Bit	Description
0	31:20	**Periodic Calibration Timeout** Default Value: 07Fh Access: R/W This field represents the amount of time that the Host will give to transmitting the HS calibration sequence for Periodic Calibrations. The transcoder will ignore this field if Periodic Calibration is not enabled within the TRANS_DSI_FUNC_CONF register. The default value will be set to 128 Byte clocks Restriction: When periodic calibration is enabled in Video Mode, the Horizontal size between synchronous packets must be great enough to support the HS calibration burst without the loss of synchronous packets. Software will need to ensure that the following equation is met to prevent synchronization packet loss: Periodic Calib Duration < (H. Size * Bits per Pixel) / (Link Width * 8) 16*N Notes: 1. The "H. Size" term is dependent on the mode of operation (i.e. Sync Pulse or Sync Event). In Sync Pulse, H. Size = H. Sync Start + (H. Total H. Sync End). In Sync Event, H. Size = H. Total. 2. HS bursts and calibrations cannot be concatenated together on Data Lanes which means the Data Lanes have to enter the LP state on either end of the calibration. The 16*N term within the equation is used to account for the HS to HS latency of transitioning the Data Lanes on either end of the calibration.The variable N is the number of Byte clocks within

DSI_CALIB_TO		
		an Escape clock (N = ceiling(((8X Frequency (in MHz) / 20 MHz) / 8))
	19:16	**Reserved** Access: RO Format: MBZ
	15:0	**Initial Calibration Timeout** Default Value: 0FFFh Access: R/W This field represents the amount of time the Host will give to transmitting the HS calibration sequence for the Initial Calibration, if enabled. The transcoder will ignore this field if Calibration is not enabled within the TRANS_DSI_FUNC_CONF register. The default value will be set to 4096 Byte clocks

DSI_CLK_TIMING_PARAM

DSI_CLK_TIMING_PARAM

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B080h-6B083h
Name:	DSI 0 Clock Lane Timing Parameter
ShortName:	DSI_CLK_TIMING_PARAM_0
Reset:	soft
Address:	6B880h-6B883h
Name:	DSI 1 Clock Lane Timing Parameter
ShortName:	DSI_CLK_TIMING_PARAM_1
Reset:	soft

This register specifies the D-PHY timing parameters for the Clock Lane, if SW is overriding the HW defaults.
This register is located within the Core Display and is used by the DSI Controller to calculate Link transition latencies of the Clock Lane. There is an identical register (DPHY_CLK_TIMING_PARAM) located within the combo-PHY that actually applies the overrides to the D-PHY Clock Lane. Both registers should be programmed by Software if an override needs to be applied to the Clock Lane within the D-PHY.
Since this register is being used to calculate the Link transition latencies of the Clock Lane, but does not actually affect the transition times within the D-PHY, this register can be used to add guardbands to the DSI Controller's transition latency calculations.
The lower 12 bits of the offset address for this register should correspond to the lower offset address of its sister D-PHY register within the combo-PHY.
All fields are defined in number of Escape clocks.

Restriction

Overall restriction is that the timing parameters must be non-zero if SW is overriding the HW timing parameters.
The programming of this register must be equal to or greater than the programming of its sister register that lives within the combo-PHY (DPHY_CLK_TIMING_PARAM).

DWord	Bit	Description
0	31	**CLK_PREPARE Override** Access: R/W This field controls the override of the CLK-PREPARE timing parameter. 0b: HW maintains **[Default]** 1b: SW overrides
	30:28	**CLK_PREPARE** Access: R/W This parameter defines the time that the Host drives the Clock Lane with the LP-00 Lane state (the Bridge state) immediately before the HS-0 Line state.

DSI_CLK_TIMING_PARAM

This field represents a hexadecimal value with a precision of 1.2 i.e. the most significant bit is the integer and the least significant 2 bits are fraction bits. So, the field can represent a range of 0.25 to 1.75 (12.5ns to 87.5ns assuming an Escape clock with a 20MHz frequency)
HW maintains this timing parameter at1 Escape clock (minimum50ns)

Value	Name
001b	0.25 Escape clocks
010b	0.50 Escape clocks
011b	0.75 Escape clocks
100b	1.00 Escape clocks
101b	1.25 Escape clocks
110b	1.50 Escape clocks
111b	1.75 Escape clocks
Others	Reserved

Programming Notes
Caution: The MIPI D-PHY specification has a maximum of 95ns for this parameter.

27 **CLK_ZERO Override**

Access:	R/W

This field controls the override of the CLK-ZERO timing parameter

Value	Name
0	HW Maintains
1	SW overrides

26:24 **Reserved**

Access:	RO
Format:	MBZ

23:20 **CLK_ZERO**

Access:	R/W

This parameter defines the time that the Host drives the HS-0 Lane state on the Clock Lane.
HW maintains this parameter at 5 Escape clocks (minimum250ns)

19 **CLK_PRE Override**

Access:	R/W

This field controls the override of the CLK-PRE timing parameter.

Value	Name
0	HW Maintains
1	SW overrides

18 **Reserved**

Access:	RO
Format:	MBZ

DSI_CLK_TIMING_PARAM

	17:16	**CLK_PRE** Access: R/W This parameter defines the time that the HS clock shall be driven by the Host prior to any Data Lane beginning its transition from the LP state to the HS state. HW maintains this parameter at 8 UI (1 Byte clock). This field will override the parameter with a value measured in Escape clocks which will be much greater than 8 UI.
	15	**CLK_POST Override** Access: R/W This field controls the override of the CLK-POST timing parameter Value: 0 — Name: HW Maintains Value: 1 — Name: SW overrides
	14:11	**Reserved** Access: RO Format: MBZ
	10:8	**CLK_POST** Access: R/W This parameter defines the time the Host continues to transmit the HS clock after the last Data Lane has transitioned to the LP state. HW maintains this parameter at 1.25 Escape clocks plus 7 Byte clocks (minimum62.5ns + 56 UI)
	7	**CLK_TRAIL Override** Access: R/W This field controls the override of the CLK-TRAIL timing parameter Value: 0 — Name: HW Maintains Value: 1 — Name: SW overrides
	6:3	**Reserved** Access: RO Format: MBZ
	2:0	**CLK_TRAIL** Access: R/W This parameter defines the time that the Host drives the HS-0 Lane state on the Clock Lane after the CLK-POST time has been achieved. HW maintains this parameter at 1.25 Escape clocks (minimum 62.5ns)

DSI_CMD_FRMCTL

DSI_CMD_FRMCTL

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B034h-6B037h
Name:	DSI Transcoder 0 Command Mode Frame Control
ShortName:	DSI_CMD_FRMCTL_0
Reset:	soft
Address:	6B834h-6B837h
Name:	DSI Transcoder 1 Command Mode Frame Control
ShortName:	DSI_CMD_FRMCTL_1
Reset:	soft

This register is used to control initiating frame updates to the Peripheral in Command Mode
The fields within this register are only observed by the DSI transcoder when it is in the Command Mode of operation

DWord	Bit	Description
0	31	**Frame Update Request** Access: R/W Set This bit controls when the transcoder will start the next frame when it is in Command Mode. The transcoder will act on this bit only when it is in the Command Mode and the Transcoder is enabled (TRANS_CONF). Software can write to this bit, but Hardware will be responsible for clearing it. 0b – No frame request present 1b – Frame request present
	30	**Reserved** Access: RO Format: MBZ
	29	**Periodic Frame Update Enable** Access: R/W When the Intel graphics driver is not present (e.g. the driver is uninstalled, the OS basic driver is present, or pre-boot time) there will be no frame update requests to the DSI transcoder through the Frame Update Request of this register. Therefore, this field will enable a mechanism that will initiate periodic frame update requests when TE events are received from the Panel. When enabling this feature, the expectation will be that SW has configured the Panel to send TE events to the Host (i.e. set_tear_on, etc.). HW will do the following when this bit is set: 1. It will override the TE Source bit of the TRANS_DSI_FUNC_CONF register to the GPIO setting. The DSI transcoder HW does not have a mechanism to automatically send Bus

Turn-Arounds to the Panel to receive in-band TE events

2. It will override the Mode of Operation of the TRANS_DSI_FUNC_CONF register to a non-gating Command Mode of operation

Value	Name
0b	Periodic Frame Update disabled
1b	Periodic Frame Update Enabled

Programming Notes
Note that when the Takeover MIPI DBI bit is set within the MSG_MEDE_KVMR_SPR_CTL register then HW will do the following if the DSI transcoder is operating in a Command Mode (i.e. DBI): 1. It will enable the Periodic Frame Update mode of operation 2. It will override the TE Source bit of the TRANS_DSI_FUNC_CONF register to the GPIO setting 3. It will override the Mode of Operation of the TRANS_DSI_FUNC_CONF register to a non-gating Command Mode of operation If the DSI transcoder is not operating in a DBI mode, then the transcoder will ignore the Takeover MIPI DBI bit.

28 **Null Packet Enable**

Access:	R/W

This bit controls whether Null Packets will be transmitted between Pixel Packets in Command Mode operation.
If a Pixel Packet ends and the next Pixel Packet is within the transcoders pipeline but not visible to the HS arbiter, then the transcoder will begin transmitting Null Packet bursts to keep the Link in the HS state.
If the current Pixel Packet ends and the next Pixel Packet is not within the transcoders pipeline, then the transcoder will allow the Link to enter the LP state.
This field is ignored when the transcoder is operating in Video Mode.

Value	Name
0b	Null packet injection disabled
1b	Null packet injection enabled

27 **Single Panel Update**

Access:	R/W

This is an attribute applied to the Frame Update Request bit.
When this transcoder is synchronized to another transcoder in Dual Link Dual Pipe Sync mode (i.e. both Port Sync Mode Enable and Dual Pipe Sync Enable are set), then this bit controls whether this transcoder waits for the other port to receive its Frame Update Request and TE event (i.e. both Panels are going to receive a frame), or whether this transcoder just waits for the other transcoders TE event (i.e. just this Panel is going to receive a frame)

Value	Name
0b	Dual Panel update

1b	Single Panel update

Programming Notes

1. HW only acts on this bit if the Port Sync Mode Enable is set for this transcoder. If the transcoders Port Sync Enable is not set, then a Frame Update Request is implicitly a single Panel update. The difference is that a non-synchronized transcoder will not wait for TE events from another port.
2. If this bit is set and the Port Sync Enable bit is set, then it is the responsibility of SW to ensure that the Panel for the other transcoder is generating TE events
3. Setting this bit should be mutually exclusive with the other Frame Update Request attributes Accumulate Frame Update Request and Forward Frame Update Request

26 **Accumulate Frame Update Requests**

Access:	R/W Set

This is an attribute applied to the Frame Update Request bit.
This bit will control whether two synchronized transcoders in a Dual Link - Dual Pipe mode of operation will synchronize the Frame Update Requests to each transcoder (i.e. the Frame Start to the two Pipes will be sent out from each transcoder at the same time)
HW will clear this bit when the V. Blank for this transcoder is observed

Value	Name
0b	No Accumulation
1b	Accumulate

Programming Notes

1. HW only acts on this attribute if:
   a. The Port Sync Enable is set for both transcoders
   b. The transcoders are in a Dual Link Dual Pipe sync mode
2. For Dual Link Single Pipe sync mode (i.e. Port Sync Enable is set and both transcoders are bound to the same Pipe), only DSI 0 sends a Frame Start
3. If SW is going to use this attribute, it must use it one of two ways:
   a. The attribute must be set to the same value for both Frame Update Requests
   b. The attribute must be set to a 1 for the first Frame Update Request and a 0 for the second Frame Update Request
4. HW will not start looking for TE events (if TE gating is enabled) until both Frame Update Requests have been received.
5. Setting this bit should be mutually exclusive with the other Frame Update Request attributes Single Panel Update and Forward Frame Update Request

25 **Forward Frame Update Request**

Access:	R/W

This is an attribute applied to the Frame Update Request bit.

DSI_CMD_FRMCTL

This bit will control whether the transcoder forwards the Frame Update Request received by this transcoder to a synchronized transcoder

Value	Name
0b	Request not forwarded
1b	Request forwarded

Programming Notes

1. HW ignores this bit if the Port Sync Enable bit for the transcoders are not set
2. HW will automatically override the setting of this bit for DSI0 when in Dual Link Single Pipe (DLSP) mode. In DLSP, HW ignores any Frame Update Requests made to DSI1 and automatically forwards DSI0 Frame Update Requests to DSI1.
3. Setting this bit should be mutually exclusive with the other Frame Update Request attributes Accumulate Frame Update Request and Single Panel Update

24:2 **Reserved**

Access:	RO
Format:	MBZ

1 **Reserved**

Access:	RO
Format:	MBZ

0 **Frame in Progress**

Access:	RO

This bit reflects whether the DSI transcoder is currently processing/sending a frame to the Peripheral.

DSI_CMD_RXCTL

DSI_CMD_RXCTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B0D4h-6B0D7h
Name:	DSI Transcoder 0 Command Receive Control
ShortName:	DSI_CMD_RXCTL_0
Reset:	soft
Address:	6B8D4h-6B8D7h
Name:	DSI Transcoder 1 Command Receive Control
ShortName:	DSI_CMD_RXCTL_1
Reset:	soft

This register controls how received DSI packets from the Peripheral are handled.

DWord	Bit	Description
0	31:17	**Reserved** Access: RO Format: MBZ
	16	**Read Unloads DW** Access: R/W This bit controls whether the DSI_RXDATA register read unloads the DW from the transcoders receive queue. If DSI_RXDATA reads do not unload the transcoders receive queue, then the transcoder will continue to return the DW of data at the head of the queue. Otherwise, every read to the DSI_RXDATA will return a new DW of data. Value 0b: DSI_RXDATA reads do not unload DW Value 1b: DSI_RXDATA reads unload DW
	15	**Received Unassigned Trigger** Access: R/WC The unassigned trigger (10100000 [lsb on the left to the msb on the right]) has been received. Value 0b: Trigger message not received Value 1b: Trigger message received
	14	**Received Acknowledge Trigger** Access: R/WC The Acknowledge trigger message (00100001 [lsb to msb]) has been received.

DSI_CMD_RXCTL

Value	Name
0b	Trigger message not received
1b	Trigger message received

13 **Received Tear Effect Trigger**

Access:	R/WC

The Tear Effect (TE) trigger message (01011101 [lsb to msb]) has been received.

Value	Name
0b	Trigger message not received
1b	Trigger message received

12 **Received Reset Trigger**

Access:	R/WC

The Reset trigger message (01100010 [lsb to msb]) has been received.
The Peripheral is not expected to send this trigger message to the Host, but even if it does, the hardware does not take any action on this message

Value	Name
0b	Trigger message not received
1b	Trigger message received

11 **Received Payload was Lost**

Access:	R/WC

This bit indicates if the DSI transcoder had to drop one or more of the payload bytes from a response packet being received from the Peripheral.
Software should check that the "Maximum Return Packet Size" programming is set correctly within the Peripheral

Value	Name
0b	No payload bytes dropped
1b	Payload bytes dropped

10 **Received CRC was Lost**

Access:	R/WC

When set, the DSI transcoder had to drop one or more of the CRC bytes from a response packet being received from the Peripheral.
If the "Received Payload was Lost" bit is set, then this bit will most likely also be set.
It is not imperative that the CRC is captured within the queue, so this is not an error but only a heads up that it may not be present.
If Software wishes the CRC to always be present within the Payload Receive queue, then it should adjust the "Maximum Return Packet Size" programming within the Peripheral to account for the CRC

Value	Name
0b	No CRC bytes dropped
1b	CRC bytes dropped

DSI_CMD_RXCTL		
	9:8	**Reserved** Access: RO Format: MBZ
	7:0	**Number Rx Payload DW** Access: RO This field represents the number of DWs currently within the transcoders Payload Receive queue. Note that the queue will be flushed at the beginning of a Bus Turn-Around (BTA) HW currently maintains an 8 DW queue.

DSI_CMD_RXHDR

DSI_CMD_RXHDR	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	6B0E0h-6B0E3h
Name:	DSI 0 Command Receive Header
ShortName:	DSI_CMD_RXHDR_0
Reset:	soft
Address:	6B8E0h-6B8E3h
Name:	DSI 1 Command Receive Header
ShortName:	DSI_CMD_RXHDR_1
Reset:	soft

This register reads the READ Response packet header received from the Periphery. This register is RO.

DWord	Bit	Description
0	31:0	**Received Header** Access: RO This field contains the READ Response packet header received from the Periphery. Software can read this as many times as it wishes (i.e. the Read Unloads DW bit of the DSI_CMD_RXCTL has no effect on this data).

DSI_CMD_RXPYLD

DSI_CMD_RXPYLD	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	6B0E4h-6B0E7h
Name:	DSI 0 Command Receive Payload
ShortName:	DSI_CMD_RXPYLD_0
Reset:	soft
Address:	6B8E4h-6B8E7h
Name:	DSI 1 Command Receive Payload
ShortName:	DSI_CMD_RXPYLD_1
Reset:	soft

This register reads from the Receive Payload queue within the transcoder which contains the payload data of READ Response packets received from the Periphery. A read to this register will pull a DW of data from a receive queue within the DSI transcoder unless the DSI_CMD_RXCTL is programmed to prevent this. This register is RO.

DWord	Bit	Description
0	31:0	**Received Payload** Access: RO This field contains the READ Response payload data and CRC received from the Periphery. This data is taken from the head of the DSI transcoders receive queue. If the Read Unloads DW bit of the DSI_CMD_RXCTL is set, then multiple reads to this register will return the same data (i.e. the data at the head of the receive queue). If there is a read to this register when the transcoders receive queue is empty (i.e. the Number Rx DW within DSI_CMD_RXDATA is zero), then the data returned will be all zeros. Note that the contents of the Receive Payload queue within the transcoder is flushed for every entry into a Bus Turn-Around state and for every exit from a content protection session.

DSI_CMD_TXCTL

DSI_CMD_TXCTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B0D0h-6B0D3h
Name:	DSI Transcoder 0 Command Transmit Control
ShortName:	DSI_CMD_TXCTL_0
Reset:	soft
Address:	6B8D0h-6B8D3h
Name:	DSI Transcoder 1 Command Transmit Control
ShortName:	DSI_CMD_TXCTL_1
Reset:	soft

This register controls how DSI command packets are built and transmitted over the DSI Link.

DWord	Bit	Description
0	31:25	**Reserved** Access: RO Format: MBZ
	24	**Keep Link in HS** Access: R/W This field will keep the Link in the HS state between SW initiated command packets to the Periphery. The transitions between the LP and HS states can result in significant latencies and can have a negative impact on SW performance. If between HS commands there is no other HS traffic to transmit, then the DSI transcoder will begin transmitting Null packets until the next HS packet arrives. **Restriction** Restrictions: 1. HW will only act on this bit when the DSI transcoder is in the Command Mode of operation. 2. SW must clear this bit if it is going to initiate a LP transaction (e.g. a LPDT, BTA, Trigger, etc.). 3. SW should clear this bit when it is finished sending its burst of commands to the Periphery.
	23:13	**Reserved** Access: RO Format: MBZ

DSI_CMD_TXCTL		
	12:8	**Free Header Credits** Access: RO This field represents the number of Header resources that are currently available to SW. SW will need a Header Credit to write to either the Command Tx Header (DSI_CMD_TXHDR) or the LP Message (DSI_LP_MSG) registers. A write to either of these registers will consume a Header Credit. The transcoder will release a Header Credit after it has pulled the command from its internal command header queue. HW currently maintains 16 Header Credits.
	7:0	**Free Payload Credits** Access: RO This field represents the number of Payload resources that are currently available to SW. SW will need a Payload Credit to write to the Command Tx Payload register (DSI_CMD_TXPYLD). A write to the Tx Payload register will consume 1 Payload Credit (i.e., 1 Payload Credit is equal to 1 to 4 bytes of payload). The transcoder will release a Payload Credit after it has pulled a DW of data from its internal command payload queue. HW currently maintains 64 Payload Credits (i.e. HW can accept payloads of up to 256 bytes).

DSI_CMD_TXHDR

DSI_CMD_TXHDR	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B100h-6B103h
Name:	DSI Transcoder 0 Transmit Packet Header
ShortName:	DSI_CMD_TXHDR_0
Reset:	soft
Address:	6B900h-6B903h
Name:	DSI Transcoder 1 Transmit Packet Header
ShortName:	DSI_CMD_TXHDR_1
Reset:	soft

This register is used to write a packet header to the DSI transcoder.

DWord	Bit	Description
0	31	**Payload** Access: R/W When set, the DSI packet carries a payload of data. 0b: Short packet format (no payload) 1b: Long packet format (payload) **Restriction** Restrictions: 1. SW must ensure that the Data Type encoding matches the packet format specified by this attribute. 2. SW must ensure that the payload data is written into the Command Payload queue before writing to this register, if this bit is set.
	30	**LPDT** Access: R/W Low Power Data Transfer. This field will direct the DSI transcoder on what mode (HS or LP) to transmit the packet in. 0b: Cmd transmitted in HS state 1b: Cmd transmitted in LP Escape mode **Restriction** SW must expect synchronization events to be missed, if it sets this attribute when the DSI

DSI_CMD_TXHDR

controller is in the Video Mode of operation and the Timing Generator is enabled.
SW is highly discouraged from setting this bit when in Video Mode.

29 Vertical Blank Fence

Access:	R/W

This field will direct the DSI transcoder to wait until the start of the next V. Blank (in Video Mode) or the end of the frame (in Command Mode) before it executes the Command.
This can be used for Frame Synchronized Commands that have to be fenced after the transmission of an Execute Queue.

Value	Name
0b	Cmd will not be fenced
1b	Cmd will be fenced

28 Reserved

Access:	RO
Format:	MBZ

27:24 Reserved

Access:	RO
Format:	MBZ

23:8 Word Count - Parameters

Access:	R/W

This field specifies either the DSI Word Count (i.e. the length of the payload in bytes), or the two bytes of Parameters.
The interpretation of this field by the DSI transcoder will be based off of the Payload attribute above.
When the Payload bit is '0', then these bytes represent Parameters
When the Payload bit is '1', then these bytes represent the Word Count of the Payload
The interpretation of this field by the Periphery will be based off of the Data Type field below

Restriction
When the command carries a payload, SW must ensure that the value within this field matches the amount of payload data loaded into the Command Payload queue associated with this packet For Short packets, it will be the responsibility of SW to adhere to the DSI protocols if the packet carries only one, or no parameter

7:6 Virtual Channel

Access:	R/W

This field specifies the virtual channel to transmit the command over the DSI Link.

5:0 Data Type

Access:	R/W

This field specifies the DSI command Data Type.

DSI_CMD_TXPYLD

DSI_CMD_TXPYLD	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B104h-6B107h
Name:	DSI Transcoder 0 Transmit Packet Payload
ShortName:	DSI_CMD_TXPYLD_0
Reset:	soft
Address:	6B904h-6B907h
Name:	DSI Transcoder 1 Transmit Packet Payload
ShortName:	DSI_CMD_TXPYLD_1
Reset:	soft

This register is used to write a DW of packet payload to the DSI transcoder.

DWord	Bit	Description
0	31:0	**Payload Data** Access: R/W This register is a proxy into the payload of the command packet to be injected onto the DSI Link. Writes to this register will load the DW of data into a Command Payload queue that will be unloaded by HW after the packet has been validated via a write to the DSI_CMD_TXHDR register. When reading from this register, only the payload data from the last write will be available. **Restriction** Software must have an available Payload credit to write to this register. SW must write the payload in ascending order (i.e. DW 0 is written first, the final DW is written last). SW must load the entire payload within the Command Payload queue before writing to the Command Header register (DSI_CMD_TXHDR). The write to the Command Header register validates the payload written to this queue. SW must ensure that the WC of the Packet Header matches the amount of data written to the Command Payload queue.

DSI_DATA_TIMING_PARAM

DSI_DATA_TIMING_PARAM

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B084h-6B087h
Name:	DSI 0 Data Lane Timing Parameter
ShortName:	DSI_DATA_TIMING_PARAM_0
Reset:	soft
Address:	6B884h-6B887h
Name:	DSI 1 Data Lane Timing Parameter
ShortName:	DSI_DATA_TIMING_PARAM_1
Reset:	soft

This register specifies the D-PHY timing parameters for the Data Lane, if SW is overriding the HW defaults.
This register is located within the Core Display and is used by the DSI Controller to calculate Link transition latencies of the Data Lanes. There is an identical register (DPHY_DATA_TIMING_PARAM) located within the combo-PHY that actually applies the overrides to the D-PHY Data Lanes. Both registers should be programmed by Software if an override needs to be applied to the Data Lanes within the D-PHY.
Since this register is being used to calculate the Link transition latencies of the Data Lanes, but does not actually affect the transition times within the D-PHY, this register can be used to add guardbands to the DSI Controller's transition latency calculations.
The lower 12 bits of the offset address for this register should correspond to the lower offset address of its sister D-PHY register within the combo-PHY.
All fields are defined in number of Escape clocks.

Restriction

Overall restriction is that the timing parameters must be non-zero if SW is overriding the HW timing parameters.
The programming of this register must be equal to or greater than the programming of it's sister register that lives within the combo-PHY (DPHY_DATA_TIMING_PARAM).

DWord	Bit	Description
0	31	**HS_PREPARE Override** Access: R/W This field controls the override of the HS-PREPARE timing parameter. Value 0: HW maintains Value 1: SW overrides
	30:27	**Reserved** Access: RO Format: MBZ

DSI_DATA_TIMING_PARAM

26:24 **HS_PREPARE**

Access:	R/W

This parameter defines the time that the Host drives a Data Lane with the LP-00 Lane state (the Bridge state) immediately before driving the HS-0 Line state.
This field represents a hexadecimal value with a precision of 1.2 i.e. the most significant bit is the integer and the least significant 2 bits are fraction bits. So, the field can represent a range of 0.25 to 1.75 (12.5ns to 87.5ns assuming an Escape clock with a 20MHz frequency)
HW maintains this parameter at 1 Escape clock (minimum 50ns)

Value	Name
001b	0.25 Escape clocks
010b	0.50 Escape clocks
011b	0.75 Escape clocks
100b	1.0 Escape clocks
101b	1.25 Escape clocks
110b	1.50 Escape clocks
111b	1.75 Escape clocks
Others	Reserved

Programming Notes
Caution: The MIPI D-PHY specification has a maximum of 85ns + 6UI for this parameter.

23 **HS_ZERO Override**

Access:	R/W

This field controls the override of the HS-ZERO timing parameter

Value	Name
0	HW maintains
1	SW overrides

22:20 **Reserved**

Access:	RO
Format:	MBZ

19:16 **HS_ZERO**

Access:	R/W

This parameter defines the time that the Host drives the HS-0 Lane state on a Data Lane.
HW maintains this parameter at 2 Escape clocks plus 1 Byte clock (minimum 100ns + 8UI)

15 **HS_TRAIL Override**

Access:	R/W

This field controls the override of the HS-TRAIL timing parameter

Value	Name
0	HW maintains
1	SW overrides

DSI_DATA_TIMING_PARAM

	14:11	**Reserved** Access: RO Format: MBZ
	10:8	**HS_TRAIL** Access: R/W This parameter defines the time that the Host drives the flipped differential state of the last payload data bit of a HS transmission on a Data Lane. HW maintains this parameter at 1.5 Escape clocks (minimum 75ns)
	7	**HS_EXIT Override** Access: R/W This field controls the override of the HS-EXIT timing parameter Value 0: HW maintains Value 1: SW overrides
	6:3	**Reserved** Access: RO Format: MBZ
	2:0	**HS_EXIT** Access: R/W This parameter defines the time that the Host drives the LP-11 Lane state (i.e. the Stop state) following a HS burst. HW maintains this parameter at 2 Escape clocks (minimum 100ns)

DSI_ESC_CLK_DIV

DSI_ESC_CLK_DIV

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	6B090h-6B093h
Name:	DSI 0 Escape Clock Divider
ShortName:	DSI_ESC_CLK_DIV_0
Reset:	soft
Address:	6B890h-6B893h
Name:	DSI 1 Escape Clock Divider
ShortName:	DSI_ESC_CLK_DIV_1
Reset:	soft

This register defines the clock divider variable M needed to generate an Escape clock from the 8X clock
This register is located within the Core Display. There is an identical register (DPHY_ESC_CLK_DIV) located within the combo-PHY. Both of these registers should be programmed by Software.
The lower 12 bits of the offset address for this register should correspond to the lower offset address of its sister D-PHY register within the combo-PHY.

Restriction : The programming of this register must be identical to the programming of its sister register that lives within the combo-PHY (DPHY_ESC_CLK_DIV)

DWord	Bit	Description
0	31:21	**Reserved** Access: RO Format: MBZ
	20:16	**Byte Clocks per Escape Clock** Access: RO This field reports the number of Byte clocks present within a given Escape clock. The DSI transcoder calculates this variable based off of the Escape clock divider M. N = Ceiling(M/8) The DSI complex (transcoder and D-PHY) use this information to emulate an Escape clock using the Byte clock. Note that the value reported here is zero-based (i.e. Ceiling(M/8) - 1)
	15:9	**Reserved** Access: RO Format: MBZ
	8:0	**Escape Clock Divider M** Access: R/W **Description** This field specifies the divider variable (M) needed to derive the Escape clock from the Link

DSI_ESC_CLK_DIV		
		clock (i.e. the 8X frequency) Escape frequency= 8X frequency / M The DSI transcoder does not use a physical Escape clock, so there is no physical divider, but the transcoder needs to know the value of M to emulate the Escape clock in Byte clocks. **Restriction** The Escape clock frequency must be as close to, but not greater than 20MHz. Therefore, the programming of M should be: M = Ceiling(8X Frequency (in MHz) / 20 MHz)

DSI_HTX_TO

DSI_HTX_TO	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B044h-6B047h
Name:	DSI 0 HS Transmit Timeout
ShortName:	DSI_HTX_TO_0
Reset:	soft
Address:	6B844h-6B847h
Name:	DSI 1 HS Transmit Timeout
ShortName:	DSI_HTX_TO_1
Reset:	soft

This register specifies the HS Tx timeout.

DWord	Bit	Description
0	31:16	**HS TX Timeout** Default Value: FFFFh Access: R/W This field represents the upper 16 bits of the HS Transmit Timeout (i.e. the timeout has a granularity of 64K). The time is specified in Byte clocks. This field will default to the maximum possible value. **Programming Notes** This timer is zero-based (i.e. a value of 0 will have a timeout of 64K) If the DSI transcoder is in the Video Mode and is not able to place the Link in the LP state during the H. Blank regions of the V. Active region, then SW must program this to be greater that the amount of time it takes to transmit the full frame (V. Total * H. Total). The value should be set to a value greater than the Peripherals HS RX Timeout.
	15:1	**Reserved** Access: RO Format: MBZ
	0	**HTX_TO** Access: R/WC The HS TX Timer has timed out. HW will set this bit, SW will clear it with a write of 1b.

DSI_INTER_IDENT_REG

DSI_INTER_IDENT_REG	
Register Space:	MMIO: 0/2/0
Access:	R/WC
Size (in bits):	32
Address:	6B074h-6B077h
Name:	DSI Transcoder 0 Interrupt Identity Register
ShortName:	DSI_INTER_IDENT_REG_0
Reset:	soft
Address:	6B874h-6B877h
Name:	DSI Transcoder 1 Interrupt Identity Register
ShortName:	DSI_INTER_IDENT_REG_1
Reset:	soft

The DSI Interrupt Identity Register (IIR)logs non-masked DSI interrupts received from the Periphery and Host. The DSI_INTER_MSK_REG (IMR) controls which interrupts will be logged within the IIR.

DWord	Bit	Description
0	31	**TE Event** Access: R/WC A Tear Effect (TE) event was received
	30	**Rx Data / BTA Terminated** Access: R/WC READ response data has been received, or the BTA has been terminated
	29	**Tx Data** Access: R/WC A transmit credit has been freed
	28	**ULPS Entry Done** Access: R/WC A Ultra Low Power State Entry flow has completed
	27	**Non-TE Trigger Received** Access: R/WC A non-TE trigger has been received from the Periphery. The DSI_CMD_RXCTL will indicate the trigger received.
	26	**Host Checksum Error** Access: R/WC Host reported a checksum error
	25	**Host Multi ECC Error** Access: R/WC Host reported a multi-bit ECC error

DSI_INTER_IDENT_REG

	24	**Host Single ECC Error** Access: R/WC Host reported a single-bit ECC
	23	**Host Contention Detected** Access: R/WC Host reported a LP contention
	22	**Host False Control Error** Access: R/WC Host reported a False Control error
	21	**Host Timeout Error** Access: R/WC Host reported a Timeout error
	20	**Host Low Power Transmit Sync Error** Access: R/WC Host reported a LP Transmission byte alignment problem
	19	**Host Escape Mode Entry Command Error** Access: R/WC Host reported an Escape Mode entry command error
	18:17	**Spare 18_17** Access: R/WC Spare R/WC bits for future use
	16	**Frame Update Done** Access: R/WC A frame update is done. This interrupt is only valid when the transcoder is in Command Mode
	15	**Protocol Violation** Access: R/WC Peripheral reported a protocol violation
	14	**Spare 14** Access: R/WC Spare R/WC bit for future use
	13	**Invalid Tx Length** Access: R/WC Peripheral reported an invalid transmission length
	12	**Invalid VC** Access: R/WC Peripheral reported an invalid DSI VC ID
	11	**Invalid Data Type** Access: R/WC

DSI_INTER_IDENT_REG		
		Peripheral reported a non-recognizable DSI Data Type
	10	**Peripheral Checksum Error** Access: R/WC Peripheral reported a checksum error
	9	**Peripheral Multi ECC Error** Access: R/WC Peripheral reported a multi-bit ECC error
	8	**Peripheral Single ECC Error** Access: R/WC Peripheral reported a single-bit ECC error
	7	**Peripheral Contention Detected** Access: R/WC Peripheral reported a LP contention
	6	**Peripheral False Control Error** Access: R/WC Peripheral reported a False Control error
	5	**Peripheral Timeout Error** Access: R/WC Peripheral reported a timeout error
	4	**Peripheral Low Power Transmit Sync Error** Access: R/WC Peripheral reported a LP Transmission byte alignment problem
	3	**Peripheral Escape Mode Entry Command Error** Access: R/WC Peripheral reported an Escape Mode entry command error
	2	**EoT Sync Error** Access: R/WC Peripheral reported an End of Transmission byte alignment problem
	1	**SoT Sync Error** Access: R/WC Peripheral reported a Start of Transmission leader sequence corruption error
	0	**SoT Error** Access: R/WC Peripheral reported a Start of Transmission Error

DSI_INTER_MSK_REG

DSI_INTER_MSK_REG

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B070h-6B073h
Name:	DSI Transcoder 0 Interrupt Mask Register
ShortName:	DSI_INTER_MSK_REG_0
Reset:	soft
Address:	6B870h-6B873h
Name:	DSI Transcoder 1 Interrupt Mask Register
ShortName:	DSI_INTER_MSK_REG_1
Reset:	soft

The DSI Interrupt Mask Register (IMR) provides a filter to the events that can cause interrupts (i.e. the register will determine which events will be logged within the DSI Interrupt Identity Register (IIR))

DWord	Bit	Description
0	31	**TE Event** Access: R/W Tear Effect (TE) interrupt mask Value 0: Event Unmasked Value 1: Event Masked **[Default]**
	30	**Rx Data/BTA Terminated** Access: R/W READ response data received, or the BTA has been terminated interrupt mask Value 0: Event Unmasked Value 1: Event Masked **[Default]**
	29	**Tx Data** Access: R/W Freed transmit credit interrupt mask Value 0: Event Unmasked Value 1: Event Masked **[Default]**
	28	**ULPS Entry Done** Access: R/W Ultra Low Power State Entry flow interrupt mask

DSI_INTER_MSK_REG

Value	Name
0	Event Unmasked
1	Event Masked **[Default]**

27 **Non-TE Trigger Received**

Access:	R/W

Non-TE trigger received interrupt mask

Value	Name
0	Event Unmasked
1	Event Masked **[Default]**

26 **Host Checksum Error**

Access:	R/W

Host reported a checksum error interrupt mask

Value	Name
0	Event Unmasked
1	Event Masked **[Default]**

Programming Notes
Masking this event effectively disables the CRC checking for received payloads

25 **Host Multi ECC Error**

Access:	R/W

Host reported a multi-bit ECC error interrupt mask

Value	Name
0	Event Unmasked
1	Event Masked **[Default]**

Programming Notes
Masking this event along with the Host Single ECC Error mask bit will disable ECC checking for received packet headers

24 **Host Single ECC Error**

Access:	R/W

Host reported a single-bit ECC error interrupt mask

Value	Name
0	Event Unmasked
1	Event Masked **[Default]**

Programming Notes
Masking this event along with the Host Multi ECC Error mask bit will disable ECC checking for received packet headers

DSI_INTER_MSK_REG

23	**Host Contention Detected** Access: R/W Host reported a LP contention interrupt mask 0: Event Unmasked 1: Event Masked **[Default]**
22	**Host False Control Error** Access: R/W Host reported a False Control error interrupt mask 0: Event Unmasked 1: Event Masked **[Default]**
21	**Host Timeout Error** Access: R/W Host reported a Timeout error interrupt mask 0: Event Unmasked 1: Event Masked **[Default]**
20	**Host Low Power Transmit Sync Error** Access: R/W Host reported a LP Transmission byte alignment problem interrupt mask 0: Event Unmasked 1: Event Masked **[Default]**
19	**Host Escape Mode Entry Command Error** Access: R/W Host reported an Escape Mode entry command error interrupt mask 0: Event Unmasked 1: Event Masked **[Default]**
18:17	**Spare 18_17** Default Value: 11b Access: R/W Spare R/W bits for future use
16	**Frame Update Done** Frame update finished mask

DSI_INTER_MSK_REG

Value	Name
0b	Event Unmasked
1b	Event Masked **[Default]**

15 — Protocol Violation

Access:	R/W

Peripheral reported a protocol violation interrupt mask

Value	Name
0	Event Unmasked
1	Event Masked **[Default]**

14 — Spare 14

Default Value:	1b
Access:	R/W

Spare R/W bit for future use

13 — Invalid Tx Length

Access:	R/W

Peripheral reported an invalid transmission length interrupt mask

Value	Name
0	Event Unmasked
1	Event Masked **[Default]**

12 — Invalid VC

Access:	R/W

Peripheral reported an invalid DSI VC ID interrupt mask

Value	Name
0	Event Unmasked
1	Event Masked **[Default]**

11 — Invalid Data Type

Access:	R/W

Peripheral reported a non-recognizable DSI Data Type interrupt mask

Value	Name
0	Event Unmasked
1	Event Masked **[Default]**

10 — Peripheral Checksum Error

Access:	R/W

Peripheral reported a checksum error interrupt mask

Value	Name
0	Event Unmasked
1	Event Masked **[Default]**

DSI_INTER_MSK_REG

9 — Peripheral Multi ECC Error

Access:	R/W

Peripheral reported a multi-bit ECC error interrupt mask

Value	Name
0	Event Unmasked
1	Event Masked **[Default]**

8 — Peripheral Single ECC Error

Access:	R/W

Peripheral reported a single-bit ECC error interrupt mask

Value	Name
0	Event Unmasked
1	Event Masked **[Default]**

7 — Peripheral Contention Detected

Access:	R/W

Peripheral reported a LP contention interrupt mask

Value	Name
0	Event Unmasked
1	Event Masked **[Default]**

6 — Peripheral False Control Error

Access:	R/W

Peripheral reported a False Control error interrupt mask

Value	Name
0	Event Unmasked
1	Event Masked **[Default]**

5 — Peripheral Timeout Error

Access:	R/W

Peripheral reported a timeout error interrupt mask

Value	Name
0	Event Unmasked
1	Event Masked **[Default]**

4 — Peripheral Low Power Transmit Sync Error

Access:	R/W

Peripheral reported a LP Transmission byte alignment problem interrupt mask

Value	Name
0	Event Unmasked
1	Event Masked **[Default]**

DSI_INTER_MSK_REG

	3	**Peripheral Escape Mode Entry Command Error** Access: R/W Peripheral reported an Escape Mode entry command error interrupt mask **Value** / **Name** 0 – Event Unmasked 1 – Event Masked **[Default]**
	2	**EoT Sync Error** Access: R/W Peripheral reported an End of Transmission byte alignment problem interrupt mask **Value** / **Name** 0 – Event Unmasked 1 – Event Masked **[Default]**
	1	**SoT Sync Error** Access: R/W Peripheral reported a Start of Transmission leader sequence corruption error interrupt mask **Value** / **Name** 0 – Event Unmasked 1 – Event Masked **[Default]**
	0	**SoT Error** Access: R/W Peripheral reported a Start of Transmission Error interrupt mask **Value** / **Name** 0 – Event Unmasked 1 – Event Masked **[Default]**

DSI_IO_MODECTL

DSI_IO_MODECTL

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	6B094h-6B097h
Name:	DSI Transcoder 0 IO Mode Control
ShortName:	DSI_IO_MODECTL_0
Reset:	soft
Address:	6B894h-6B897h
Name:	DSI Transcoder 1 IO Mode Control
ShortName:	DSI_IO_MODECTL_1
Reset:	soft

This register is used to control the mode of operation within the Combo-PHY which is shared between the MIPI DSI transcoder and the eDP/DP DDI.
Each DSI transcoder is attached to the following Combo-PHY:
DSI0: Combo-PHY A
DSI1: Combo-PHY B
Note that the Combo-PHY's are also referred to as DDI A / DDI B

Restriction: If the Combo-PHY is going to be used by the DSI transcoder, then this register must be programmed **before** the power request is sent to the Combo-PHY

DWord	Bit	Description
0	31:18	**Reserved** Access: RO Format: MBZ
	17:16	**Reserved** Access: RO Format: MBZ
	15:2	**Reserved** Access: RO Format: MBZ
	1	**Reserved** Access: RO Format: MBZ
	0	**Combo PHY Mode** Access: R/W For products using a Combo-PHY for the DSI port, this bit selects the mode of operation within the Combo-PHY to which the DSI transcoder is attached. For products using a dedicated DSI PHY, this bit controls the clock request to the PHY.

DSI_IO_MODECTL			
		Value	Name
		0b	DDI Mode / No Clock Request
		1b	MIPI DSI Mode / Clock Request

DSI_LP_MSG

DSI_LP_MSG

Register Space:	MMIO: 0/2/0
Access:	R/W Set
Size (in bits):	32
Address:	6B0D8h-6B0DBh
Name:	DSI 0 Low Power Message
ShortName:	DSI_LP_MSG_0
Reset:	soft
Address:	6B8D8h-6B8DBh
Name:	DSI 1 Low Power Message
ShortName:	DSI_LP_MSG_1
Reset:	soft

This register contains the LP messages that can be sent to the Periphery.

Restriction

Software must have a Header credit to write to this register.

DWord	Bit	Description
0	31:19	**Reserved** Access: RO Format: MBZ
	18	**Link Direction** Access: RO This field advertises the current state of the Link direction 0b – Link is in the Forward direction 1b – Link is in the Reverse direction
	17	**LP Tx in Progress** Access: RO This status bit indicates whether the DSI transcoder is currently servicing a LP transaction. 0b – Transcoder is not transmitting in the LP Esc mode 1b – Transcoder is transmitting in the LP Esc mode
	16	**In ULPS** Access: RO This status bit indicates whether the DSI Link is in the Ultra-Low Power State (ULPS), or not. This bit should accurately reflect the ultra-low power state of the Link even when the DSI transcoder function is disabled.

DSI_LP_MSG

Value	Name
0b	The DSI Link is not in ULPS
1b	The DSI Link is in ULPS

15:11 **Reserved**

Access:	RO
Format:	MBZ

10:9 **Trigger Type**

Access:	R/W

This field specifies the type of Trigger Message to send the Peripheral. It is only sampled when a Trigger Message is being sent to the Peripheral.

Value	Name	Description
00b	Reset Trigger	Entry Command [lsb:msb]: 01100010
01b	Unknown 3	Entry Command [lsb:msb]: 01011101
10b	Unknown 4	Entry Command [lsb:msb]: 00100001
11b	Unknown 5	Entry Command [lsb:msb]: 10100000

Programming Notes
Note that the Unassigned triggers are not specifically assigned to a given action/function within the DSI spec. Therefore, these can be used as general-purpose trigger messages that the Periphery defines

8 **ULPS Type**

Access:	R/W

This bit specifies the LP state that the Lanes (both Data and Clock) will be left in after entering ULPS

Value	Name	Description
0b	LP-00	Lanes will be left in the LP-00 state
1b	LP-11	Lanes will be left in the LP-11 state

7:3 **Reserved**

Access:	RO
Format:	MBZ

2 **Trigger Message**

Access:	R/W Set

This bit will direct the DSI Transcoder to issue a Trigger Message to the Peripheral. The type of trigger is specified with the Trigger Type field within this register.
Trigger signaling is a mechanism to send a flag to the Protocol Layer of the Periphery using the Escape Mode of transmission.

Programming Notes
The trigger flag, as seen by the Peripheral, can be extended using the Trigger Extension in the DPHY_TRIG_EXT.

DSI_LP_MSG

		The DSI Transcoder will clear this bit when it is finished with the transmission of the message (including the Trigger Extension)
	1	**Bus Turnaround** Access: R/W Set This bit will direct the DSI Transcoder to issue a BTA request to the Periphery. **Programming Notes** When in Video Mode, the DSI transcoder will dispatch a BTA request (by itself) within the next Vertical blank line that it sees. SW must program the Turnaround Timeout (DSI_TA_TO) and the LP Rx (Host) Timeout (DSI_LRX_H_TO) properly to avoid having synchronization events being missed. In other words, the total amount of time it takes to send a BTA and receive a response from the Panel needs to be less than the Vertical blank line time.
	0	**ULPS Entry** Access: R/W Set Ultra-Low Power State Entry. When set, the DSI Transcoder will transmit the ULPS Entry Command on all Data Lanes and it will bring the Clock Lane to the ULPS state. The state of the Lanes at the end of the ULPS flow is dictated by the ULPS Type within this register. When HW has finished the UPLS sequence it will clear this bit. **Restriction** Before setting this bit, Software must disable the DSI's Timing Generator

DSI_LRX_H_TO

DSI_LRX_H_TO

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B048h-6B04Bh
Name:	DSI 0 LP Rx (Host) Timeout
ShortName:	DSI_LRX_H_TO_0
Reset:	soft
Address:	6B848h-6B84Bh
Name:	DSI 1 LP Rx (Host) Timeout
ShortName:	DSI_LRX_H_TO_1
Reset:	soft

This register specifies the LP Rx (Host)timeout.

DWord	Bit	Description
0	31:17	**Reserved** Access: RO Format: MBZ
	16	**LRX_H_TO** Default Value: 0b Access: R/WC The LP RX Timer has timed out. HW will set this bit, SW must clear it with a write of 1b.
	15:0	**LP RX_H Timeout** Access: R/W This field represents the maximum amount of time the DSI transcoder will give to the Peripheral to transmit its response back to the Host. If the timer times out, then the DSI transcoder will set the LRX_H_TO bit in this register and the Host Timeout Error bit within the DSI_INTER_IDENT_REG register, if this interrupt event is unmasked(DSI_INTER_MSK_REG). The time is specified in Escape clocks. **Programming Notes** The LP RX Timer will be disabled if this value is zero

DSI_PWAIT_TO

DSI_PWAIT_TO	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B040h-6B043h
Name:	DSI 0 Peripheral Wait Timeout
ShortName:	DSI_PWAIT_TO_0
Reset:	soft
Address:	6B840h-6B843h
Name:	DSI 1 Peripheral Wait Timeout
ShortName:	DSI_PWAIT_TO_1
Reset:	soft

This register represents a Peripheral wait time used in conjunction with either a BTA or a Trigger LP message.
The times specified within this register are in terms of Escape clocks.
The timers are zero-based (i.e. a value of 0 equals 1 Escape clock)

DWord	Bit	Description
0	31:16	**Peripheral Reset Timeout** Access: R/W This field represents the time to wait after a Reset Trigger has been transmitted to the Peripheral (PR_TO). The timer will start after the Trigger message has been sent and the DSI transcoder will block all other traffic until the timer reaches the timeout value.
	15:0	**Peripheral Response Timeout** Access: R/W This field represents the time to wait before asking the Peripheral for a response (PRESP_TO). The timer will start when the DSI transcoder receives the BTA request from SW and the Link enters the Stop state. When the timer reaches the timeout value, the DSI transcoder will begin the BTA request to the Peripheral.

DSI_T_INIT_PRIMARY

DSI_T_INIT_PRIMARY	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B088h-6B08Bh
Name:	DSI 0 Initialization Primary Time
ShortName:	DSI_T_INIT_PRIMARY_0
Reset:	soft
Address:	6B888h-6B88Bh
Name:	DSI 1 Initialization Primary Time
ShortName:	DSI_T_INIT_PRIMARY_1
Reset:	soft

This register specifies the amount of time (in Escape clocks) to drive the Link in the initialization state.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Primary Initialization Time** Default Value: 07D0h Access: R/W This field specifies the INIT_PRIMARY timing parameter used by the Host to drive the Link initialization. This field is specified in Escape clocks where the Escape clock operates at a maximum frequency of 20MHz. **Programming Notes** The default value of this register will produce an initialization duration of 100us which is the minimum requirement for the INIT timing parameter. The Periphery may require a longer initialization period (INIT + INTERNAL_DELAY), so the value programmed should be greater than the Periphery's initialization requirements.

DSI_T_WAKEUP

DSI_T_WAKEUP	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B08Ch-6B08Fh
Name:	DSI 0 Wakeup Timing Parameter
ShortName:	DSI_T_WAKEUP_0
Reset:	soft
Address:	6B88Ch-6B88Fh
Name:	DSI 1 Wakeup Timing Parameter
ShortName:	DSI_T_WAKEUP_1
Reset:	soft

This is the timing parameter T-WAKEUP used to drive the Mark-1 state on the Link when exiting ULPS.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Wakeup Time** Default Value: 4E20h Access: R/W This field represents the T_{WAKEUP} timing parameter used when ULPS is being exited. The time is specified in number of Escape clocks. The default of this field will be set to 1ms (MIPI DSI specification minimum). **Restriction** A value of zero is invalid

DSI_TA_TO

DSI_TA_TO	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B04Ch-6B04Fh
Name:	DSI 0 Turnaround Timeout
ShortName:	DSI_TA_TO_0
Reset:	soft
Address:	6B84Ch-6B84Fh
Name:	DSI 1 Turnaround Timeout
ShortName:	DSI_TA_TO_1
Reset:	soft

This register specifies the Turnaround timeout.

DWord	Bit	Description
0	31:17	**Reserved** Access: RO Format: MBZ
	16	**TA_TO** Access: R/WC The Turnaround Timer has timed out. HW will set this bit, SW must clear it with a write of 1b.
	15:0	**Turnaround Timeout** Access: R/W This field represents the maximum amount of time the Host will give to the Peripheral to acknowledge the Bus Turnaround request. If the timer times out, then the DSI transcoder will set the TA_TO bit in this register and the Host Timeout Error bit within the DSI_INTER_IDENT_REG register, if this interrupt event is unmasked (DSI_INTER_MSK_REG). The time is specified in Escape clocks. **Programming Notes** The TA TO Timer will be disabled if this value is zero

DSI_TRIG_TX_TIME

DSI_TRIG_TX_TIME

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B09Ch-6B09Fh
Name:	DSI 0 Trigger Transmission Time
ShortName:	DSI_TRIG_TX_TIME_0
Reset:	soft
Address:	6B89Ch-6B89Fh
Name:	DSI 1 Trigger Transmission Time
ShortName:	DSI_TRIG_TX_TIME_1
Reset:	soft

This register specifies the amount of time to present a LP Trigger request to the physical layer in Escape clocks When programming this register, SW must consider the following:

1. Escape mode entry and command transmission time: 20 Escape clocks
2. Trigger extension time: Panel specific

Software should not program the value within this register to be less than the time it takes to transmit the Escape mode entry and command. Doing so could pre-empt the transmission of the trigger causing an error at the Panel.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Trigger Tx Time** Default Value: 14h This field specifies the number of Escape clocks that the DSI controller should present a trigger message to the physical layer This field is only used if a Trigger Message is initiated from the DSI_LP_MSG register.

DS Invocation Counter

DS_INVOCATION_COUNT - DS Invocation Counter	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02308h
Name:	DS Invocation Counter
ShortName:	DS_INVOCATION_COUNT

This register stores the number of domain points shaded by the DS threads. Domain points which hit in the DS cache will not cause this register to increment. Note that the spawning of a DS thread which shades two domain points will cause this counter to increment by two. This register is part of the context save and restore.

DWord	Bit	Description
0..1	63:32	**DS Invocation Count UDW** Number of domain points shaded by the DS threads. Updated only when DS Function Enable and Statistics Enable are set in 3DSTATE_DS
	31:0	**DS Invocation Count LDW** Number of domain points shaded by the DS threads. Updated only when DS Function Enable and Statistics Enable are set in 3DSTATE_DS

DSMBASE

DSMBASE - DSMBASE

Register Space:	MMIO: 0/2/0
Size (in bits):	64
SOC_Consumer:	BIOS
Address:	1080C0h

This register contains the base address of graphics data stolen DRAM memory. BIOS determines the base of graphics data stolen memory. BIOS is now able to allocate Gfx Stolen Memory above the 4GB.

DWord	Bit	Description
0..1	63:20	**BDSM** Default Value: 000h Access: R/W _Custom_GTIReset: BUS This BitField contains bits 63 to 20 of the base address of stolen DRAM memory.
	19:1	**Reserved** Access: RO Format: MBZ
	0	**SPARE** Default Value: 0b Access: R/W _Custom_GTIReset: BUS This was a lock bit prior.

DSSM

DSSM

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	51004h-51007h
Name:	Display Strap State
ShortName:	DSSM
Reset:	global

This register contains fuse and strap settings for display. This register is not reset by FLR.

DWord	Bit	Description
0	31:29	**Reference Frequency** Access: RO This field indicates the reference clock frequency. Software should use this value when programming the display clocks. 000b: 24 MHz 001b: 19.2 MHz 010b: 38.4 MHz
	28	**Spare 28**
	27	**Spare 27**
	26	**Spare 26**
	25	**Spare 25**
	24	**Spare 24**
	23	**Spare 23**
	22	**Spare 22**
	21	**Spare 21**
	20	**Spare 20**
	19	**Spare 19**
	18	**Spare 18**
	17	**Spare 17**
	16	**Spare 16**
	15	**Spare 15**
	14	**Spare 14**
	13	**Spare 13**
	12	**Spare 12**
	11	**Spare 11**

DSSM

	10	**Spare 10**
	9	**Spare 9**
	8	**Spare 8**
	7	**Spare 7**
	6	**DE 8k Dis** DE_8K_DIS 8k capability fuse. This bit indicates whether hardware supports screens with widths greater than 5120 pixels. For tiled or joined displays, this is the total width after combining the widths of both pipes. Software must not enable these resolutions when the fuse is configured to disable 8k.

Value	Name	Description
0b	Enable	8k Capability Enabled
1b	Disable	8k Capability Disabled

	5	**Audio IO Flop Bypass** This field specifies whether the audio IO flop should be bypassed for dies with a long path to IO.

Value	Name
0b	Don't Bypass
1b	Bypass

	4	**Audio IO Select** This field specifies which audio IO location to use. It has to match where the PCH audio is connecting to the die.

Value	Name
0b	South
1b	North

	3	**WD Video Fault Continue** This field specifies whether WD video should continue data writes after a fault or stop the writes.

Value	Name
0b	Stop Writes
1b	Continue Writes

	2	**Reserved**
	1	**Part Is SOC**

Description
This field specifies whether this part is a SoC or not.

Value	Name
0b	Not SoC
1b	SoC

	0	**DisplayPort A Present** This bit specifies whether the port was present during initialization. This strap state can also be read in the DDI_BUF_CTL_A 0x64000 register bit 0.

DSSM				
		Value	Name	Description
		0b	Not Present	Port not present
		1b	Present	Port present

Dummy Context Save Register

UNUTILIZED_DUMMY_CTXSAVE - Dummy Context Save Register		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	080FCh	
DWord	**Bit**	**Description**
0	31:0	**Context Save Register** Access: R/W _Custom_GTIReset: DEV This register is purely used as a dummy register when do context save of non-unslice clients. This is the last register saved as part of all other context saves and used as indication that restore is complete

EDRAMCAP

EDRAMCAP - EDRAMCAP		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
SOC_Consumer:	BIOS	
Address:	120010h	
Describes the presence and capabilities of the eDRAM cache.		
DWord	**Bit**	**Description**
0	31:0	**EDRAMCAP_VALUE** Default Value: 00000000h Access: R/W _Custom_GTIReset: BUS **Bits[31:22] - Reserved** **Bits[21:16] - CURRENT_ETAG_WAYS,** Default Value: 11b The total number of ways in each ETAG slice. This field is updated by PCODE after every ETAG shrink or ETAG expand. 0 if 4 ways 1 if 8 ways 2 if 12 ways 3 if 16 ways. **Bits[15:10] - Reserved** **Bits[9:8] - SETS_CONFIGURATION,** Default Value: 10b 1K or 2K sets, to support 64MB and 128MB EDRAM size configurations accordingly. 00 = Reserved. 01 if 1K sets 10 if 2K sets 11 reserved. **Bits[7:5] - WAYS_CONFIGURATION,** Default Value: 011b This field defines the total number of ETAG ways. 000 if 4 ways 001 if 8 ways 010 if 12 ways 011 if 16 ways 1xx reserved. **Bits[4:1] - Super Queue Internal Register Count,** Default Value: 0100b Number of EDRAM TAG banks should be always 4, meaning 4 banks. **Bit[0] - FUSE_EDRAM_ENABLE,** Default Value: 0b PCODE will update this field based on FUSE_EDRAM_ENABLE.

EMRR Mask LSB

EMRRMASK_LSB - EMRR Mask LSB		
Register Space: MMIO: 0/2/0 Size (in bits): 32		
Address: 09208h		
EMRR Mask Value		
DWord	**Bit**	**Description**
0	31:12	**EMRR MASK LSB BITS** Access: RO _Custom_GTIReset: BUS EMRR MASK VALUE.
	11	**EMRR ENABLE** Access: RO _Custom_GTIReset: BUS EMRR Enable.
	10	**EMRR LOCK** Access: RO _Custom_GTIReset: BUS EMRR LOCK bit.
	9:0	**Spares** Access: RO _Custom_GTIReset: BUS

EMRR Mask MSB

EMRRMASK_MSB - EMRR Mask MSB		
Register Space: MMIO: 0/2/0 Size (in bits): 32		
Address: 0920Ch		
EMRR Mask Value		
DWord	**Bit**	**Description**
0	31:0	**EMRR MASK MSB BITS** Access: RO _Custom_GTIReset: BUS EMRR MASK VALUE.

Encoded row max QP vector

ENC_ROW_MAX_QP_VEC - Encoded row max QP vector	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	1024
_Custom_GTIReset:	BUS
Address:	1C2D00h-1C2D7Fh
Name:	Max QP for each Tile row
ShortName:	ENC_ROW_MAX_QP_VEC_VDENC_REG0
Address:	1C6D00h-1C6D7Fh
Name:	Max QP for each Tile row
ShortName:	ENC_ROW_MAX_QP_VEC_VDENC_REG1
Address:	1D2D00h-1D2D7Fh
Name:	Max QP for each Tile row
ShortName:	ENC_ROW_MAX_QP_VEC_VDENC_REG2
Address:	1D6D00h-1D6D7Fh
Name:	Max QP for each Tile row
ShortName:	ENC_ROW_MAX_QP_VEC_VDENC_REG3
Address:	1E2D00h-1E2D7Fh
Name:	Max QP for each Tile row
ShortName:	ENC_ROW_MAX_QP_VEC_VDENC_REG4
Address:	1E6D00h-1E6D7Fh
Name:	Max QP for each Tile row
ShortName:	ENC_ROW_MAX_QP_VEC_VDENC_REG5
Address:	1F2D00h-1F2D7Fh
Name:	Max QP for each Tile row
ShortName:	ENC_ROW_MAX_QP_VEC_VDENC_REG6
Address:	1F6D00h-1F6D7Fh
Name:	Max QP for each Tile row
ShortName:	ENC_ROW_MAX_QP_VEC_VDENC_REG7

This register holds the Max QP of the encoded bitstream for four consecutive LCU rows. Each, LCU row uses 8 bits.

DWord	Bit	Description
0	31:24	**MAX_QP_LCU_ROW_3** This Byte holds the maximum qp value of any CU coded in the row 3. The value could be used to determine the quality of encoded bitstream for row 3.

ENC_ROW_MAX_QP_VEC - Encoded row max QP vector		
	23:16	**MAX_QP_LCU_ROW_2** This Byte holds the maximum qp value of any CU coded in the row 2. The value could be used to determine the quality of encoded bitstream for row 2.
	15:8	**MAX_QP_LCU_ROW_1** This Byte holds the maximum qp value of any CU coded in the row 1. The value could be used to determine the quality of encoded bitstream for row 1.
	7:0	**MAX_QP_LCU_ROW_0** This Byte holds the maximum qp value of any CU coded in the row 0. The value could be used to determine the quality of encoded bitstream for row 0.
1	31:24	**MAX_QP_LCU_ROW_7**
	23:16	**MAX_QP_LCU_ROW_6**
	15:8	**MAX_QP_LCU_ROW_5**
	7:0	**MAX_QP_LCU_ROW_4**
2	31:24	**MAX_QP_LCU_ROW_11**
	23:16	**MAX_QP_LCU_ROW_10**
	15:8	**MAX_QP_LCU_ROW_9**
	7:0	**MAX_QP_LCU_ROW_8**
3	31:24	**MAX_QP_LCU_ROW_15**
	23:16	**MAX_QP_LCU_ROW_14**
	15:8	**MAX_QP_LCU_ROW_13**
	7:0	**MAX_QP_LCU_ROW_12**
4	31:24	**MAX_QP_LCU_ROW_19**
	23:16	**MAX_QP_LCU_ROW_18**
	15:8	**MAX_QP_LCU_ROW_17**
	7:0	**MAX_QP_LCU_ROW_16**
5	31:24	**MAX_QP_LCU_ROW_23**
	23:16	**MAX_QP_LCU_ROW_22**
	15:8	**MAX_QP_LCU_ROW_21**
	7:0	**MAX_QP_LCU_ROW_20**
6	31:24	**MAX_QP_LCU_ROW_27**
	23:16	**MAX_QP_LCU_ROW_26**
	15:8	**MAX_QP_LCU_ROW_25**
	7:0	**MAX_QP_LCU_ROW_24**
7	31:24	**MAX_QP_LCU_ROW_31**
	23:16	**MAX_QP_LCU_ROW_30**
	15:8	**MAX_QP_LCU_ROW_29**
	7:0	**MAX_QP_LCU_ROW_28**
8	31:24	**MAX_QP_LCU_ROW_35**

ENC_ROW_MAX_QP_VEC - Encoded row max QP vector		
	23:16	**MAX_QP_LCU_ROW_34**
	15:8	**MAX_QP_LCU_ROW_33**
	7:0	**MAX_QP_LCU_ROW_32**
9	31:24	**MAX_QP_LCU_ROW_39**
	23:16	**MAX_QP_LCU_ROW_38**
	15:8	**MAX_QP_LCU_ROW_37**
	7:0	**MAX_QP_LCU_ROW_36**
10	31:24	**MAX_QP_LCU_ROW_43**
	23:16	**MAX_QP_LCU_ROW_42**
	15:8	**MAX_QP_LCU_ROW_41**
	7:0	**MAX_QP_LCU_ROW_40**
11	31:24	**MAX_QP_LCU_ROW_47**
	23:16	**MAX_QP_LCU_ROW_46**
	15:8	**MAX_QP_LCU_ROW_45**
	7:0	**MAX_QP_LCU_ROW_44**
12	31:24	**MAX_QP_LCU_ROW_51**
	23:16	**MAX_QP_LCU_ROW_50**
	15:8	**MAX_QP_LCU_ROW_49**
	7:0	**MAX_QP_LCU_ROW_48**
13	31:24	**MAX_QP_LCU_ROW_55**
	23:16	**MAX_QP_LCU_ROW_54**
	15:8	**MAX_QP_LCU_ROW_53**
	7:0	**MAX_QP_LCU_ROW_52**
14	31:24	**MAX_QP_LCU_ROW_59**
	23:16	**MAX_QP_LCU_ROW_58**
	15:8	**MAX_QP_LCU_ROW_57**
	7:0	**MAX_QP_LCU_ROW_56**
15	31:24	**MAX_QP_LCU_ROW_63**
	23:16	**MAX_QP_LCU_ROW_62**
	15:8	**MAX_QP_LCU_ROW_61**
	7:0	**MAX_QP_LCU_ROW_60**
16	31:24	**MAX_QP_LCU_ROW_67**
	23:16	**MAX_QP_LCU_ROW_66**
	15:8	**MAX_QP_LCU_ROW_65**
	7:0	**MAX_QP_LCU_ROW_64**
17	31:24	**MAX_QP_LCU_ROW_71**

ENC_ROW_MAX_QP_VEC - Encoded row max QP vector		
	23:16	**MAX_QP_LCU_ROW_70**
	15:8	**MAX_QP_LCU_ROW_69**
	7:0	**MAX_QP_LCU_ROW_68**
18	31:24	**MAX_QP_LCU_ROW_75**
	23:16	**MAX_QP_LCU_ROW_74**
	15:8	**MAX_QP_LCU_ROW_73**
	7:0	**MAX_QP_LCU_ROW_72**
19	31:24	**MAX_QP_LCU_ROW_79**
	23:16	**MAX_QP_LCU_ROW_78**
	15:8	**MAX_QP_LCU_ROW_77**
	7:0	**MAX_QP_LCU_ROW_76**
20	31:24	**MAX_QP_LCU_ROW_83**
	23:16	**MAX_QP_LCU_ROW_82**
	15:8	**MAX_QP_LCU_ROW_81**
	7:0	**MAX_QP_LCU_ROW_80**
21	31:24	**MAX_QP_LCU_ROW_87**
	23:16	**MAX_QP_LCU_ROW_86**
	15:8	**MAX_QP_LCU_ROW_85**
	7:0	**MAX_QP_LCU_ROW_84**
22	31:24	**MAX_QP_LCU_ROW_91**
	23:16	**MAX_QP_LCU_ROW_90**
	15:8	**MAX_QP_LCU_ROW_89**
	7:0	**MAX_QP_LCU_ROW_88**
23	31:24	**MAX_QP_LCU_ROW_95**
	23:16	**MAX_QP_LCU_ROW_94**
	15:8	**MAX_QP_LCU_ROW_93**
	7:0	**MAX_QP_LCU_ROW_92**
24	31:24	**MAX_QP_LCU_ROW_99**
	23:16	**MAX_QP_LCU_ROW_98**
	15:8	**MAX_QP_LCU_ROW_97**
	7:0	**MAX_QP_LCU_ROW_96**
25	31:24	**MAX_QP_LCU_ROW_103**
	23:16	**MAX_QP_LCU_ROW_102**
	15:8	**MAX_QP_LCU_ROW_101**
	7:0	**MAX_QP_LCU_ROW_100**
26	31:24	**MAX_QP_LCU_ROW_107**

ENC_ROW_MAX_QP_VEC - Encoded row max QP vector		
	23:16	**MAX_QP_LCU_ROW_106**
	15:8	**MAX_QP_LCU_ROW_105**
	7:0	**MAX_QP_LCU_ROW_104**
27	31:24	**MAX_QP_LCU_ROW_111**
	23:16	**MAX_QP_LCU_ROW_110**
	15:8	**MAX_QP_LCU_ROW_109**
	7:0	**MAX_QP_LCU_ROW_108**
28	31:24	**MAX_QP_LCU_ROW_115**
	23:16	**MAX_QP_LCU_ROW_114**
	15:8	**MAX_QP_LCU_ROW_113**
	7:0	**MAX_QP_LCU_ROW_112**
29	31:24	**MAX_QP_LCU_ROW_119**
	23:16	**MAX_QP_LCU_ROW_118**
	15:8	**MAX_QP_LCU_ROW_117**
	7:0	**MAX_QP_LCU_ROW_116**
30	31:24	**MAX_QP_LCU_ROW_123**
	23:16	**MAX_QP_LCU_ROW_122**
	15:8	**MAX_QP_LCU_ROW_121**
	7:0	**MAX_QP_LCU_ROW_120**
31	31:24	**MAX_QP_LCU_ROW_127**
	23:16	**MAX_QP_LCU_ROW_126**
	15:8	**MAX_QP_LCU_ROW_125**
	7:0	**MAX_QP_LCU_ROW_124**

Error Identity Register

EIR - Error Identity Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	020B0h-020B3h
Name:	Error Identity Register
ShortName:	EIR_RCSUNIT
Address:	180B0h-180B3h
Name:	Error Identity Register
ShortName:	EIR_POCSUNIT
Address:	220B0h-220B3h
Name:	Error Identity Register
ShortName:	EIR_BCSUNIT
Address:	1C00B0h-1C00B3h
Name:	Error Identity Register
ShortName:	EIR_VCSUNIT0
Address:	1C40B0h-1C40B3h
Name:	Error Identity Register
ShortName:	EIR_VCSUNIT1
Address:	1C80B0h-1C80B3h
Name:	Error Identity Register
ShortName:	EIR_VECSUNIT0
Address:	1D00B0h-1D00B3h
Name:	Error Identity Register
ShortName:	EIR_VCSUNIT2
Address:	1D40B0h-1D40B3h
Name:	Error Identity Register
ShortName:	EIR_VCSUNIT3
Address:	1D80B0h-1D80B3h
Name:	Error Identity Register
ShortName:	EIR_VECSUNIT1
Address:	1E00B0h-1E00B3h
Name:	Error Identity Register

EIR - Error Identity Register

ShortName:	EIR_VCSUNIT4
Address:	1E40B0h-1E40B3h
Name:	Error Identity Register
ShortName:	EIR_VCSUNIT5
Address:	1E80B0h-1E80B3h
Name:	Error Identity Register
ShortName:	EIR_VECSUNIT2
Address:	1F00B0h-1F00B3h
Name:	Error Identity Register
ShortName:	EIR_VCSUNIT6
Address:	1F40B0h-1F40B3h
Name:	Error Identity Register
ShortName:	EIR_VCSUNIT7
Address:	1F80B0h-1F80B3h
Name:	Error Identity Register
ShortName:	EIR_VECSUNIT3
Address:	1A0B0h-1A0B3h
Name:	Error Identity Register
ShortName:	EIR_CCSUNIT0

The EIR register contains the persistent values of Hardware-Detected Error Condition bits. Any bit set in this register will cause the error bit in the ISR to be set. The EIR register is also used by software to clear detected errors (by writing a 1 to the appropriate bit(s)), except for the unrecoverable bits described.)

DWord	Bit	Description
0	31:16	**Mask** Access: WO Format: Mask
	15:0	**Error Identity Bits** This register contains the persistent values of ESR error status bits that are unmasked via the EMR register. (See Table 3-3. Hardware-Detected Error Bits). The logical OR of all (defined) bits in this register is reported in the error bit of the Interrupt Status Register. In order to clear an error condition, software must first clear the error by writing a 1 to the appropriate bit(s) in this field. If required, software should then proceed to clear the error bit of the IIR. Reserved bits are RO. Refer the table titled "Hardware-Detected Error Bits" for independent bit definitions. Value: 1h — Name: Error occurred

EIR - Error Identity Register

		Programming Notes
		Writing a 1 to a set bit will cause that error condition to be cleared. However, neither the Page Table Error bit (Bit 4) nor the Instruction Error bit (Bit 0) can be cleared except by reset (i.e., it is a fatal error).

Error Mask Register

EMR - Error Mask Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	020B4h-020B7h
Name:	Error Mask Register
ShortName:	EMR_RCSUNIT
Address:	180B4h-180B7h
Name:	Error Mask Register
ShortName:	EMR_POCSUNIT
Address:	220B4h-220B7h
Name:	Error Mask Register
ShortName:	EMR_BCSUNIT
Address:	1C00B4h-1C00B7h
Name:	Error Mask Register
ShortName:	EMR_VCSUNIT0
Address:	1C40B4h-1C40B7h
Name:	Error Mask Register
ShortName:	EMR_VCSUNIT1
Address:	1C80B4h-1C80B7h
Name:	Error Mask Register
ShortName:	EMR_VECSUNIT0
Address:	1D00B4h-1D00B7h
Name:	Error Mask Register
ShortName:	EMR_VCSUNIT2
Address:	1D40B4h-1D40B7h
Name:	Error Mask Register
ShortName:	EMR_VCSUNIT3
Address:	1D80B4h-1D80B7h
Name:	Error Mask Register
ShortName:	EMR_VECSUNIT1
Address:	1E00B4h-1E00B7h
Name:	Error Mask Register

EMR - Error Mask Register

ShortName:	EMR_VCSUNIT4
Address:	1E40B4h-1E40B7h
Name:	Error Mask Register
ShortName:	EMR_VCSUNIT5
Address:	1E80B4h-1E80B7h
Name:	Error Mask Register
ShortName:	EMR_VECSUNIT2
Address:	1F00B4h-1F00B7h
Name:	Error Mask Register
ShortName:	EMR_VCSUNIT6
Address:	1F40B4h-1F40B7h
Name:	Error Mask Register
ShortName:	EMR_VCSUNIT7
Address:	1F80B4h-1F80B7h
Name:	Error Mask Register
ShortName:	EMR_VECSUNIT3
Address:	1A0B4h-1A0B7h
Name:	Error Mask Register
ShortName:	EMR_CCSUNIT0

The EMR register is used by software to control which Error Status Register bits are masked or unmasked. Unmasked bits will be reported in the EIR, thus setting the error ISR bit and possibly triggering a CPU interrupt, and will persist in the EIR until cleared by software. Masked bits will not be reported in the EIR and therefore cannot generate error conditions or CPU interrupts. Reserved bits are RO.

DWord	Bit	Description
0	31:8	**Reserved** Default Value: FFFFFFh Format: PBC **Programming Notes** These bits are not implemented in HW and must be set to '1'
	7:0	**Error Mask Bits** This register contains a bit mask that selects which error condition bits (from the ESR) are reported in the EIR. Refer the table titled "Hardware-Detected Error Bits" for independent bit definitions.

EMR - Error Mask Register

Value	Name	Description
FFh	**[Default]**	
0h	Not Masked	Will be reported in the EIR
1h	Masked	Will not be reported in the EIR

Error Status Register

ESR - Error Status Register	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	020B8h-020BBh
Name:	Error Status Register
ShortName:	ESR_RCSUNIT
Address:	180B8h-180BBh
Name:	Error Status Register
ShortName:	ESR_POCSUNIT
Address:	220B8h-220BBh
Name:	Error Status Register
ShortName:	ESR_BCSUNIT
Address:	1C00B8h-1C00BBh
Name:	Error Status Register
ShortName:	ESR_VCSUNIT0
Address:	1C40B8h-1C40BBh
Name:	Error Status Register
ShortName:	ESR_VCSUNIT1
Address:	1C80B8h-1C80BBh
Name:	Error Status Register
ShortName:	ESR_VECSUNIT0
Address:	1D00B8h-1D00BBh
Name:	Error Status Register
ShortName:	ESR_VCSUNIT2
Address:	1D40B8h-1D40BBh
Name:	Error Status Register
ShortName:	ESR_VCSUNIT3
Address:	1D80B8h-1D80BBh
Name:	Error Status Register
ShortName:	ESR_VECSUNIT1
Address:	1E00B8h-1E00BBh
Name:	Error Status Register

ESR - Error Status Register

ShortName:	ESR_VCSUNIT4
Address:	1E40B8h-1E40BBh
Name:	Error Status Register
ShortName:	ESR_VCSUNIT5
Address:	1E80B8h-1E80BBh
Name:	Error Status Register
ShortName:	ESR_VECSUNIT2
Address:	1F00B8h-1F00BBh
Name:	Error Status Register
ShortName:	ESR_VCSUNIT6
Address:	1F40B8h-1F40BBh
Name:	Error Status Register
ShortName:	ESR_VCSUNIT7
Address:	1F80B8h-1F80BBh
Name:	Error Status Register
ShortName:	ESR_VECSUNIT3
Address:	1A0B8h-1A0BBh
Name:	Error Status Register
ShortName:	ESR_CCSUNIT0

The ESR register contains the current values of all Hardware-Detected Error condition bits (these are all by definition persistent). The EMR register selects which of these error conditions are reported in the persistent EIR (i.e., set bits must be cleared by software) and thereby causing an error interrupt condition to be reported in the ISR.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Error Status Bits** This register contains the non-persistent values of all hardware-detected error condition bits. Refer the table titled "Hardware-Detected Error Bits" for independent bit definitions. Value: 1h — Name: Error Condition Detected

EU_GRF_CLEAR

EU_GRF_CLEAR - EU_GRF_CLEAR		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	0E550h	
Name:	EU_GRF_CLEAR	
ShortName:	EU_GRF_CLEAR	
This is a basic register template		
DWord	**Bit**	**Description**
0	31:0	**GRF_CLEAR** Default Value: 0000000000000000b Access: RO

EUP1 BONUS2 Reg

EUP1SPCBONUS2 - EUP1 BONUS2 Reg

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24698h

Clock Gating Messages Register

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**BONUS2 BIT 7** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	6	**BONUS2 BIT 6** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	5	**BONUS2 BIT 5** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	4	**BONUS2 BIT 4** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	3	**BONUS2 BIT 3** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT:

EUP1SPCBONUS2 - EUP1 BONUS2 Reg

		'0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	2	**BONUS2 BIT 2** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	1	**BONUS2 BIT 1** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**BONUS2 BIT 0** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

EUP1 BONUS11 Reg

EUP1SPCBONUS1 - EUP1 BONUS11 Reg		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	24694h	
Clock Gating Messages Register		
DWord	**Bit**	**Description**
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**BONUS1 BIT 7** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	6	**BONUS1 BIT 6** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	5	**BONUS1 BIT 5** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	4	**BONUS1 BIT 4** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	3	**BONUS1 BIT 3** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT:

EUP1SPCBONUS1 - EUP1 BONUS11 Reg

		'0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	2	**BONUS1 BIT 2** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	1	**BONUS1 BIT 1** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**BONUS1 BIT 0** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

EUP2 BONUS1 Reg

EUP2SPCBONUS1 - EUP2 BONUS1 Reg

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24714h

Clock Gating Messages Register

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**BONUS1 BIT 7** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	6	**BONUS1 BIT 6** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	5	**BONUS1 BIT 5** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	4	**BONUS1 BIT 4** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	3	**BONUS1 BIT 3** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT:

EUP2SPCBONUS1 - EUP2 BONUS1 Reg

		'0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	2	**BONUS1 BIT 2** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	1	**BONUS1 BIT 1** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**BONUS1 BIT 0** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

EUP2 BONUS2 Reg

EUP2SPCBONUS2 - EUP2 BONUS2 Reg

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24718h

Clock Gating Messages Register

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**BONUS2 BIT 7** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	6	**BONUS2 BIT 6** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	5	**BONUS2 BIT 5** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	4	**BONUS2 BIT 4** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	3	**BONUS2 BIT 3** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT:

EUP2SPCBONUS2 - EUP2 BONUS2 Reg

		'0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	2	**BONUS2 BIT 2** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	1	**BONUS2 BIT 1** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**BONUS2 BIT 0** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

EUP 2 Power Down FSM control register with lock

EUP2SPCPOWERDNFSMCTL - EUP 2 Power Down FSM control register with lock		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	24710h	
DWord	**Bit**	**Description**
0	31	**power down control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of EU PAIR 0 POWERDNFSMCTL register are R/W 1 = All bits of EU PAIR 0 POWERDNFSMCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:13	**Reserved** Access: RO Format: MBZ
	12	**Leave firewall disabled** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not firewall the gated domain for a power down flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e firewall gated domain to ungated domain crossing during power down flows 1 = Leave firewall disabled, i.e don't firewall the gated domain, but complete logical flow
	11	**Leave reset de-asserted** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not assert reset for power off flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e assert resets during power down flows 1 = Leave reset de-asserted mode, i.e don't assert reset, but complete logical flow
	10	**Leave CLKs ON** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not gate clks for power off flow. But it will pretend to complete the flow with PM

EUP2SPCPOWERDNFSMCTL - EUP 2 Power Down FSM control register with lock

		Encodings: 0 = Default mode, i.e gate clocks during power down flows 1 = Leave CLKS ON mode, i.e don't clock gate, but complete logical flow
	9	**Leave FET On** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not turn off the PFET even though it will complete the flow with PM Encodings: 0 = Default mode, i.e power off fets during power down flows 1 = Leave ON mode, i.e don't power off pfet, but complete logical flow
	8:6	**Power Down state 3** Default Value: 010b Access: R/W Lock _Custom_GTIReset: BUS This will be the 3rd state before power is turned OFF in the well Encodings: 000 = Assert Reset 001 = Firewall ON 010 = Gate clocks 1xx = Rsvd for future Default : Gate Clocks
	5:3	**Power Down state 2** Default Value: 001b Access: R/W Lock _Custom_GTIReset: BUS This will be the 2nd state before power is turned OFF in the well Encodings: 000 = Assert Reset 001 = Firewall ON 010 = Gate clocks 1xx = Rsvd for future Default :Firewall ON
	2:0	**Power Down state 1** Default Value: 000b Access: R/W Lock _Custom_GTIReset: BUS This will be the 1st state before power is turned OFF in the well Encodings: 000 = Assert Reset 001 = Firewall ON 010 = Gate clocks

EUP2SPCPOWERDNFSMCTL - EUP 2 Power Down FSM control register with lock		
		1xx = Rsvd for future Default : Assert Reset

EUP 2 Power on FSM control register with lock

EUP2SPCPOWERUPFSMCTL - EUP 2 Power on FSM control register with lock

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	2470Ch

DWord	Bit	Description
0	31	**power up control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of EU PAIR 0 POWERUPFSMCTL register are R/W 1 = All bits of EU PAIR 0 POWERUPFSMCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:9	**Reserved** Access: RO Format: MBZ
	8:6	**Power UP state 3** Default Value: 010b Access: R/W Lock _Custom_GTIReset: BUS This will be the 3rd state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - De-assert resets 3'b000: 10ns (or 1 bclk)
	5:3	**Power UP state 2** Default Value: 001b Access: R/W Lock _Custom_GTIReset: BUS This will be the 2nd state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - Firewall OFF

EUP2SPCPOWERUPFSMCTL - EUP 2 Power on FSM control register with lock

	2:0	**Power UP state 1** Default Value: 000b Access: R/W Lock _Custom_GTIReset: BUS This will be the 1st state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - Clock Ungate

EUP3 BONUS1 Reg

EUP3SPCBONUS1 - EUP3 BONUS1 Reg

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24794h
Clock Gating Messages Register	

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**BONUS1 BIT 7** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	6	**BONUS1 BIT 6** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	5	**BONUS1 BIT 5** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	4	**BONUS1 BIT 4** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	3	**BONUS1 BIT 3** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe)

EUP3SPCBONUS1 - EUP3 BONUS1 Reg

		'1' : Initiate power up sequence (clk/rst/fwe)
	2	**BONUS1 BIT 2** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	1	**BONUS1 BIT 1** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**BONUS1 BIT 0** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

EUP3 BONUS2 Reg

EUP3SPCBONUS2 - EUP3 BONUS2 Reg

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24798h
Clock Gating Messages Register	

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**BONUS2 BIT 7** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	6	**BONUS2 BIT 6** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	5	**BONUS2 BIT 5** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	4	**BONUS2 BIT 4** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	3	**BONUS2 BIT 3** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe)

EUP3SPCBONUS2 - EUP3 BONUS2 Reg

		'1' : Initiate power up sequence (clk/rst/fwe)
	2	**BONUS2 BIT 2** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	1	**BONUS2 BIT 1** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**BONUS2 BIT 0** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

EU PAIR 1 PFET control register with lock

EUP1SPCPFETCTL - EU PAIR 1 PFET control register with lock

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 24688h

DWord	Bit	Description
0	31	**PFET Control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of EU PAIR 1 PGFETCTL register are R/W 1 = All bits of EU PAIR 1 PGFETCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:24	**Reserved** Access: RO Format: MBZ
	23	**Power Well Status** Access: RO _Custom_GTIReset: BUS 0 = Well is powered Down 1 = Well is powered up Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	22	**Powergood timer error** Access: RO _Custom_GTIReset: BUS 0 = Well is powered Down 1 = Well is powered up Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	21:19	**Delay from enabling secondary PFETs to power good.** Default Value: 100b Access: R/W Lock _Custom_GTIReset: BUS Delay from enabling secondary PFETs to power good 3'b000: 40ns 3'b001: 80ns 3'b010: 160ns 3'b011: 240ns 3'b100: 320ns

EUP1SPCPFETCTL - EU PAIR 1 PFET control register with lock

		3'b101: 480ns 3'b110: 640ns 3'b111: 1280ns
	18:16	**Strobe pulse period** Access: R/W Lock _Custom_GTIReset: BUS Time period b/w two adjacent strobes to the primary FETs 3'b000: 10ns (or 1 bclk) 3'b001: 20ns (or 2 bclk) 3'b010: 30ns (or 3 bclk) 3'b111: 80ns (or 8 bclk) **Value**: 001b — **Name**: **[Default]**
	15:0	**PFET Ladder Step Sequence** Default Value: 1111111111111111b Access: R/W Lock _Custom_GTIReset: BUS PFET Ladder STEP sequence The PFET ladder has 16 steps, each represented by its corresponding bit in pfetlddrstepseq[15:0]. If the bit location is set, it means we must step on that ladder stage before we go to the next stage. The SPC controller will step through all the steps with asserted bits periodically with period equal to strbpulsprd[2:0] Programming Rule: pfetldrstepseq[15] and pfetldrstepseq[0] must be '1; A '0 setting for these bits is illegal. 15'FFFFh: Ladder step (ladder_sel) goes 0, 1, 2, ?.15. 15'FFF1h: Ladder step goes 0, 4, 5, 6,?15; Steps 1, 2, 3 are skipped. 15'D555h: Ladder step goes 0, 2, 4, 6, 8, 10, 12, 14, 15; Steps 1, 3, 5, 7, 9, 11, 13 are skipped. 15'80001h: Ladder step goes 0, 15; Steps 1, 2, 3, 4, ?14 are skipped.

EU PAIR 1 Power Context Save request

EUP1PGCTXREQ - EU PAIR 1 Power Context Save request		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	24684h	
DWord	**Bit**	**Description**
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask bots for lower 16 bits
	15:10	**Reserved** Access: RO Format: MBZ
	9	**Power context save request** Access: R/W Set _Custom_GTIReset: BUS Power Context Save Request 1'b0 : Power context save is not being requested 1'b1 : Power context save is being requested CPUnit self-clears this bit upon sampling.
	8:0	**Power Context Save request crdit count** Access: R/W _Custom_GTIReset: BUS QWord Credits for Power Context Save Request Minimum Credits = 1 : Unit may send 1 QWord pair (enough for first LRI at least) Maximum Credits = 511 : Unit may send 511 QWord pairs A QWord pair is defined as a 32-bit register address and the corresponding 32-bits of register data. Note that the LRI header and END commands are 64-bits each (32-bit command followed by 32-bit NOOP) and will consume one QWord credit. Only valid with PWRCTX_SAVE_REQ (Bit9).

EU PAIR 1 Power Down FSM control register with lock

EUP1SPCPOWERDNFSMCTL - EU PAIR 1 Power Down FSM control register with lock

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 24690h

DWord	Bit	Description
0	31	**power down control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of EU PAIR 0 POWERDNFSMCTL register are R/W 1 = All bits of EU PAIR 0 POWERDNFSMCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:13	**Reserved** Access: RO Format: MBZ
	12	**Leave firewall disabled** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not firewall the gated domain for a power down flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e firewall gated domain to ungated domain crossing during power down flows 1 = Leave firewall disabled, i.e don't firewall the gated domain, but complete logical flow
	11	**Leave reset de-asserted** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not assert reset for power off flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e assert resets during power down flows 1 = Leave reset de-asserted mode, i.e don't assert reset, but complete logical flow
	10	**Leave CLKs ON** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not gate clks for power off flow. But it will pretend to complete the flow with PM Encodings:

EUP1SPCPOWERDNFSMCTL - EU PAIR 1 Power Down FSM control register with lock

	0 = Default mode, i.e gate clocks during power down flows 1 = Leave CLKS ON mode, i.e dont clock gate, but complete logical flow
9	**Leave FET On** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not turn off the PFET even though it will complete the flow with PM Encodings: 0 = Default mode, i.e power off fets during power down flows 1 = Leave ON mode, i.e don't power off pfet, but complete logical flow
8:6	**Power Down state 3** Access: R/W Lock _Custom_GTIReset: BUS This will be the 3rd state before power is turned OFF in the well Default : Gate Clocks

Value	Name
000b	Assert Reset
1xxb	Reserved
001b	Firewall ON
010b	Gate clocks **[Default]**

5:3	**Power Down state 2** Access: R/W Lock _Custom_GTIReset: BUS This will be the 2nd state before power is turned OFF in the well Default: Firewall ON

Value	Name
1xxb	Reserved
010b	Gate clocks
000b	Assert Reset
001b	Firewall ON **[Default]**

2:0	**Power Down state 1** Access: R/W Lock _Custom_GTIReset: BUS This will be the 1st state before power is turned OFF in the well Default: Assert Reset

Value	Name
001b	Firewall ON
010b	Gate clocks

EUP1SPCPOWERDNFSMCTL - EU PAIR 1 Power Down FSM control register with lock

		1xxb	Reserved
		000b	Assert Reset **[Default]**

EU PAIR 1 Power Gate Control Request

EUP1PGCTLREQ - EU PAIR 1 Power Gate Control Request		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	24680h	
Clock Gating Messages Register		
DWord	**Bit**	**Description**
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask In order to write to bits 15:0, the corresponding message mask bits must be written. For example, for bit 14 to be set, bit 30 needs to be 1 : 40004000
	15:2	**Reserved** Access: RO Format: MBZ
	1	**CLK RST FWE Request** Access: R/W _Custom_GTIReset: BUS EU PAIR 0 CLK RST FWE request: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**Power Gate Request** Access: R/W _Custom_GTIReset: BUS EU PAIR 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

EU PAIR 1 Power on FSM control register with lock

EUP1SPCPOWERUPFSMCTL - EU PAIR 1 Power on FSM control register with lock

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 2468Ch

DWord	Bit	Description
0	31	**power up control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of EU PAIR 0 POWERUPFSMCTL register are R/W 1 = All bits of EU PAIR 0 POWERUPFSMCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:9	**Reserved** Access: RO Format: MBZ
	8:6	**Power UP state 3** Default Value: 010b Access: R/W Lock _Custom_GTIReset: BUS This will be the 3rd state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - De-assert resets 3'b000: 10ns (or 1 bclk)
	5:3	**Power UP state 2** Default Value: 001b Access: R/W Lock _Custom_GTIReset: BUS This will be the 2nd state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - Firewall OFF

EUP1SPCPOWERUPFSMCTL - EU PAIR 1 Power on FSM control register with lock

	2:0	**Power UP state 1** Default Value: 000b Access: R/W Lock _Custom_GTIReset: BUS This will be the 1st state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - Clock Ungate

EU PAIR 2 PGFET control register with lock

EUP2SPCPFETCTL - EU PAIR 2 PGFET control register with lock

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24708h

DWord	Bit	Description
0	31	**PFET Control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of EU PAIR 0 PGFETCTL register are R/W 1 = All bits of EU PAIR 0 PGFETCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:24	**Reserved** Access: RO Format: MBZ
	23	**Power Well Status** Access: RO _Custom_GTIReset: BUS 0 = Well is powered Down 1 = Well is powered up Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	22	**Powergood timer error** Access: RO _Custom_GTIReset: BUS 0 = Well is powered Down 1 = Well is powered up Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	21:19	**Delay from enabling secondary PFETs to power good.** Default Value: 100b Access: R/W Lock _Custom_GTIReset: BUS Delay from enabling secondary PFETs to power good 3'b000: 40ns 3'b001: 80ns 3'b010: 160ns 3'b011: 240ns 3'b100: 320ns

EUP2SPCPFETCTL - EU PAIR 2 PGFET control register with lock

		3'b101: 480ns 3'b110: 640ns 3'b111: 1280ns
	18:16	**Stroble pulse period** Access: R/W Lock _Custom_GTIReset: BUS Time period b/w two adjacent strobes to the primary FETs 3'b000: 10ns (or 1 bclk) 3'b001: 20ns (or 2 bclk) 3'b010: 30ns (or 3 bclk) 3'b111: 80ns (or 8 bclk) **Value** / **Name** 001b / **[Default]**
	15:0	**PFET Ladder Step Sequence** Default Value: 1111111111111111b Access: R/W Lock _Custom_GTIReset: BUS PFET Ladder STEP sequence The PFET ladder has 16 steps, each represented by its corresponding bit in pfetlddrstepseq[15:0]. If the bit location is set, it means we must step on that ladder stage before we go to the next stage. The SPC controller will step through all the steps with asserted bits periodically with period equal to strbpulsprd[2:0] Programming Rule: pfetldrstepseq[15] and pfetldrstepseq[0] must be '1; A '0 setting for these bits is illegal. 15'FFFFh: Ladder step (ladder_sel) goes 0, 1, 2, ?.15. 15'FFF1h: Ladder step goes 0, 4, 5, 6,?15; Steps 1, 2, 3 are skipped. 15'D555h: Ladder step goes 0, 2, 4, 6, 8, 10, 12, 14, 15; Steps 1, 3, 5, 7, 9, 11, 13 are skipped. 15'80001h: Ladder step goes 0, 15; Steps 1, 2, 3, 4, ?14 are skipped.

EU PAIR 2 Power Context Save request

EUP2PGCTXREQ - EU PAIR 2 Power Context Save request		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	24704h	
DWord	**Bit**	**Description**
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask bots for lower 16 bits
	15:10	**Reserved** Access: RO Format: MBZ
	9	**Power context save request** Access: R/W Set _Custom_GTIReset: BUS Power Context Save Request 1'b0 : Power context save is not being requested 1'b1 : Power context save is being requested CPUnit self-clears this bit upon sampling.
	8:0	**Power Context Save request crdit count** Access: R/W _Custom_GTIReset: BUS QWord Credits for Power Context Save Request Minimum Credits = 1 : Unit may send 1 QWord pair (enough for first LRI at least) Maximum Credits = 511 : Unit may send 511 QWord pairs A QWord pair is defined as a 32-bit register address and the corresponding 32-bits of register data. Note that the LRI header and END commands are 64-bits each (32-bit command followed by 32-bit NOOP) and will consume one QWord credit. Only valid with PWRCTX_SAVE_REQ (Bit9).

EU PAIR 2 Power Gate Control Request

EUP2PGCTLREQ - EU PAIR 2 Power Gate Control Request		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	24700h	
Clock Gating Messages Register		
DWord	**Bit**	**Description**
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask In order to write to bits 15:0, the corresponding message mask bits must be written. For example, for bit 14 to be set, bit 30 needs to be 1 : 40004000
	15:2	**Reserved** Access: RO Format: MBZ
	1	**CLK RST FWE Request** Access: R/W _Custom_GTIReset: BUS EU PAIR 0 CLK RST FWE request: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**Power Gate Request** Access: R/W _Custom_GTIReset: BUS EU PAIR 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

EU PAIR 3 PGFET control register with lock

EUP3SPCPFETCTL - EU PAIR 3 PGFET control register with lock

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 24788h

DWord	Bit	Description
0	31	**PFET Control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of EU PAIR 0 PGFETCTL register are R/W 1 = All bits of EU PAIR 0 PGFETCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:24	**Reserved** Access: RO Format: MBZ
	23	**Power Well Status** Access: RO _Custom_GTIReset: BUS 0 = Well is powered Down 1 = Well is powered up Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	22	**Powergood timer error** Access: RO _Custom_GTIReset: BUS 0 = Well is powered Down 1 = Well is powered up Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	21:19	**Delay from enabling secondary PFETs to power good.** Default Value: 100b Access: R/W Lock _Custom_GTIReset: BUS Delay from enabling secondary PFETs to power good 3'b000: 40ns 3'b001: 80ns 3'b010: 160ns 3'b011: 240ns 3'b100: 320ns

EUP3SPCPFETCTL - EU PAIR 3 PGFET control register with lock

3'b101: 480ns
3'b110: 640ns
3'b111: 1280ns

18:16 **Time period last primay pfet strobe to secondary pfet strobe**

Access:	R/W Lock
_Custom_GTIReset:	BUS

Time period b/w two adjacent strobes to the primary FETs
3'b000: 10ns (or 1 bclk)
3'b001: 20ns (or 2 bclk)
3'b010: 30ns (or 3 bclk)
3'b111: 80ns (or 8 bclk)

Value	Name
001b	**[Default]**

15:0 **PFET Ladder Step Sequence**

Default Value:	1111111111111111b
Access:	R/W Lock
_Custom_GTIReset:	BUS

PFET Ladder STEP sequence
The PFET ladder has 16 steps, each represented by its corresponding bit in pfetlddrstepseq[15:0]. If the bit location is set, it means we must step on that ladder stage before we go to the next stage.
The SPC controller will step through all the steps with asserted bits periodically with period equal to strbpulsprd[2:0]
Programming Rule: pfetldrstepseq[15] and pfetldrstepseq[0] must be '1; A '0 setting for these bits is illegal.
15'FFFFh: Ladder step (ladder_sel) goes 0, 1, 2, ?.15.
15'FFF1h: Ladder step goes 0, 4, 5, 6,?15; Steps 1, 2, 3 are skipped.
15'D555h: Ladder step goes 0, 2, 4, 6, 8, 10, 12, 14, 15; Steps 1, 3, 5, 7, 9, 11, 13 are skipped.
15'80001h: Ladder step goes 0, 15; Steps 1, 2, 3, 4, ?14 are skipped.

EU PAIR 3 Power Context Save request

EUP3PGCTXREQ - EU PAIR 3 Power Context Save request		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	24784h	
DWord	**Bit**	**Description**
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask bots for lower 16 bits
	15:10	**Reserved** Access: RO Format: MBZ
	9	**Power context save request** Access: R/W Set _Custom_GTIReset: BUS Power Context Save Request 1'b0 : Power context save is not being requested 1'b1 : Power context save is being requested CPUnit self-clears this bit upon sampling.
	8:0	**Power Context Save request crdit count** Access: R/W _Custom_GTIReset: BUS QWord Credits for Power Context Save Request Minimum Credits = 1 : Unit may send 1 QWord pair (enough for first LRI at least) Maximum Credits = 511 : Unit may send 511 QWord pairs A QWord pair is defined as a 32-bit register address and the corresponding 32-bits of register data. Note that the LRI header and END commands are 64-bits each (32-bit command followed by 32-bit NOOP) and will consume one QWord credit. Only valid with PWRCTX_SAVE_REQ (Bit9).

EU PAIR 3 Power Down FSM control register with lock

EUP3SPCPOWERDNFSMCTL - EU PAIR 3 Power Down FSM control register with lock

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 24790h

DWord	Bit	Description
0	31	**power down control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of EU PAIR 0 POWERDNFSMCTL register are R/W 1 = All bits of EU PAIR 0 POWERDNFSMCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:13	**Reserved** Access: RO Format: MBZ
	12	**Leave firewall disabled** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not firewall the gated domain for a power down flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e firewall gated domain to ungated domain crossing during power down flows 1 = Leave firewall disabled, i.e don't firewall the gated domain, but complete logical flow
	11	**Leave reset de-asserted** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not assert reset for power off flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e assert resets during power down flows 1 = Leave reset de-asserted mode, i.e don't assert reset, but complete logical flow
	10	**Leave CLKs ON** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not gate clks for power off flow. But it will pretend to complete the flow with PM Encodings:

EUP3SPCPOWERDNFSMCTL - EU PAIR 3 Power Down FSM control register with lock

		0 = Default mode, i.e gate clocks during power down flows 1 = Leave CLKS ON mode, i.e don't clock gate, but complete logical flow
	9	**Leave FET On** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not turn off the PFET even though it will complete the flow with PM Encodings: 0 = Default mode, i.e power off fets during power down flows 1 = Leave ON mode, i.e don't power off pfet, but complete logical flow
	8:6	**Power Down state 3** Default Value: 010b Access: R/W Lock _Custom_GTIReset: BUS This will be the 3rd state before power is turned OFF in the well Encodings: 000 = Assert Reset 001 = Firewall ON 010 = Gate clocks 1xx = Rsvd for future Default : Gate Clocks
	5:3	**Power Down state 2** Default Value: 001b Access: R/W Lock _Custom_GTIReset: BUS This will be the 2nd state before power is turned OFF in the well Encodings: 000 = Assert Reset 001 = Firewall ON 010 = Gate clocks 1xx = Rsvd for future Default :Firewall ON

EUP3SPCPOWERDNFSMCTL - EU PAIR 3 Power Down FSM control register with lock

	2:0	**Power Down state 1** Default Value: 000b Access: R/W Lock _Custom_GTIReset: BUS This will be the 1st state before power is turned OFF in the well Encodings: 000 = Assert Reset 001 = Firewall ON 010 = Gate clocks 1xx = Rsvd for future Default : Assert Reset

EU PAIR 3 Power Gate Control Request

EUP3PGCTLREQ - EU PAIR 3 Power Gate Control Request

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 24780h

Clock Gating Messages Register

DWord	Bit	Description
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask In order to write to bits 15:0, the corresponding message mask bits must be written. For example, for bit 14 to be set, bit 30 needs to be 1 : 40004000
	15:2	**Reserved** Access: RO Format: MBZ
	1	**CLK RST FWE Request** Access: R/W _Custom_GTIReset: BUS EU PAIR 0 CLK RST FWE request: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**Power Gate Request** Access: R/W _Custom_GTIReset: BUS EU PAIR 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

EU PAIR 3 Power on FSM control register with lock

EUP3SPCPOWERUPFSMCTL - EU PAIR 3 Power on FSM control register with lock

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	2478Ch

DWord	Bit	Description
0	31	**power up control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of EU PAIR 0 POWERUPFSMCTL register are R/W 1 = All bits of EU PAIR 0 POWERUPFSMCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:9	**Reserved** Access: RO Format: MBZ
	8:6	**Power UP state 3** Default Value: 010b Access: R/W Lock _Custom_GTIReset: BUS This will be the 3rd state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - De-assert resets 3'b000: 10ns (or 1 bclk)
	5:3	**Power UP state 2** Default Value: 001b Access: R/W Lock _Custom_GTIReset: BUS This will be the 2nd state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - Firewall OFF

EUP3SPCPOWERUPFSMCTL - EU PAIR 3 Power on FSM control register with lock

	2:0	**Power UP state 1** Default Value: 000b Access: R/W Lock _Custom_GTIReset: BUS This will be the 1st state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - Clock Ungate

Eviction counter status register

L3EVICTCNTR - Eviction counter status register		
Register Space:	MMIO: 0/2/0	
Access:	RO	
Size (in bits):	32	
_Custom_GTIReset:	DEV	
Address:	0B12Ch	
DWord	**Bit**	**Description**
0	31:16	**Eviction Counter** Counter indicating total number of eviction in multiple of 64 events
	15:0	**C/Z Eviction Counter** Counter indicating total number of eviction (in multiple of 64 events) from C/Z (or unified tile cache) ways

Exec-List Context Offset

CXT_EL_OFFSET - Exec-List Context Offset	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	021ACh-021AFh
Name:	Exec-List Context Offset
ShortName:	CXT_EL_OFFSET_RCSUNIT
Address:	181ACh-181AFh
Name:	Exec-List Context Offset
ShortName:	CXT_EL_OFFSET_POCSUNIT
Address:	221ACh-221AFh
Name:	Exec-List Context Offset
ShortName:	CXT_EL_OFFSET_BCSUNIT
Address:	1C01ACh-1C01AFh
Name:	Exec-List Context Offset
ShortName:	CXT_EL_OFFSET_VCSUNIT0
Address:	1C41ACh-1C41AFh
Name:	Exec-List Context Offset
ShortName:	CXT_EL_OFFSET_VCSUNIT1
Address:	1C81ACh-1C81AFh
Name:	Exec-List Context Offset
ShortName:	CXT_EL_OFFSET_VECSUNIT0
Address:	1D01ACh-1D01AFh
Name:	Exec-List Context Offset
ShortName:	CXT_EL_OFFSET_VCSUNIT2
Address:	1D41ACh-1D41AFh
Name:	Exec-List Context Offset
ShortName:	CXT_EL_OFFSET_VCSUNIT3
Address:	1D81ACh-1D81AFh
Name:	Exec-List Context Offset
ShortName:	CXT_EL_OFFSET_VECSUNIT1
Address:	1E01ACh-1E01AFh
Name:	Exec-List Context Offset
ShortName:	CXT_EL_OFFSET_VCSUNIT4

CXT_EL_OFFSET - Exec-List Context Offset

Address:	1E41ACh-1E41AFh
Name:	Exec-List Context Offset
ShortName:	CXT_EL_OFFSET_VCSUNIT5

Address:	1E81ACh-1E81AFh
Name:	Exec-List Context Offset
ShortName:	CXT_EL_OFFSET_VECSUNIT2

Address:	1F01ACh-1F01AFh
Name:	Exec-List Context Offset
ShortName:	CXT_EL_OFFSET_VCSUNIT6

Address:	1F41ACh-1F41AFh
Name:	Exec-List Context Offset
ShortName:	CXT_EL_OFFSET_VCSUNIT7

Address:	1F81ACh-1F81AFh
Name:	Exec-List Context Offset
ShortName:	CXT_EL_OFFSET_VECSUNIT3

Address:	1A1ACh-1A1AFh
Name:	Exec-List Context Offset
ShortName:	CXT_EL_OFFSET_CCSUNIT0

This register provides the layout format of LRCA in Exec-List mode of scheduling. Each field represents its location in 4KB offset from LRCA base address. Register gets initialized to default value coming out of reset. SW must not program this register.

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23:20	**CSFE Engine Context Size** Value: 8h — Name: **[Default]**
	19:16	**Ring Context Offset** Default Value: 1h
	15:13	**Ring Context Size** Value: 5h — Name: **[Default]**
	12:4	**Reserved** Access: RO Format: MBZ

CXT_EL_OFFSET - Exec-List Context Offset			
	3:0	**PerProcess HW Status Page Offset**	
		Default Value:	0h

Execlist Control Register

EXECLIST_CONTROL - Execlist Control Register	
Register Space:	MMIO: 0/2/0
Access:	WO
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02550h-02553h
Name:	EXECLIST CONTROL
ShortName:	EXECLIST_CONTROL_RCSUNIT
Address:	18550h-18553h
Name:	EXECLIST CONTROL
ShortName:	EXECLIST_CONTROL_POCSUNIT
Address:	22550h-22553h
Name:	EXECLIST CONTROL
ShortName:	EXECLIST_CONTROL_BCSUNIT
Address:	1C0550h-1C0553h
Name:	EXECLIST CONTROL
ShortName:	EXECLIST_CONTROL_VCSUNIT0
Address:	1C4550h-1C4553h
Name:	EXECLIST CONTROL
ShortName:	EXECLIST_CONTROL_VCSUNIT1
Address:	1C8550h-1C8553h
Name:	EXECLIST CONTROL
ShortName:	EXECLIST_CONTROL_VECSUNIT0
Address:	1D0550h-1D0553h
Name:	EXECLIST CONTROL
ShortName:	EXECLIST_CONTROL_VCSUNIT2
Address:	1D4550h-1D4553h
Name:	EXECLIST CONTROL
ShortName:	EXECLIST_CONTROL_VCSUNIT3
Address:	1D8550h-1D8553h
Name:	EXECLIST CONTROL
ShortName:	EXECLIST_CONTROL_VECSUNIT1
Address:	1E0550h-1E0553h
Name:	EXECLIST CONTROL

EXECLIST_CONTROL - Execlist Control Register

ShortName:	EXECLIST_CONTROL_VCSUNIT4
Address:	1E4550h-1E4553h
Name:	EXECLIST CONTROL
ShortName:	EXECLIST_CONTROL_VCSUNIT5
Address:	1E8550h-1E8553h
Name:	EXECLIST CONTROL
ShortName:	EXECLIST_CONTROL_VECSUNIT2
Address:	1F0550h-1F0553h
Name:	EXECLIST CONTROL
ShortName:	EXECLIST_CONTROL_VCSUNIT6
Address:	1F4550h-1F4553h
Name:	EXECLIST CONTROL
ShortName:	EXECLIST_CONTROL_VCSUNIT7
Address:	1F8550h-1F8553h
Name:	EXECLIST CONTROL
ShortName:	EXECLIST_CONTROL_VECSUNIT3
Address:	1A550h-1A553h
Name:	EXECLIST CONTROL
ShortName:	EXECLIST_CONTROL_CCSUNIT0

DWord	Bit	Description
0	31:3	**Reserved** Access: RO Format: MBZ
	2	**Use HW Element Pointer** HW element pointer gets saved on a context getting preempted due to Preempt to Idle(indicates the element number of the preempted context in the EQ). On a load following Preempt to Idle SW can set "Use Element Pointer" to indicate HW to resume from the element on which preemption has occurred due to Preempt to Idle, not setting Use Element Pointer will result in HW executing from Element-0. o UseHWElementPointer = 1 : HW saves its position in the N deep execution Q across any C6 events. When HW sees Load + UseHWElementPointer, HW will restart execution at the element pointed to by the Element Pointer. • This usage is only expected post a PreemptToIdle message, and is independent of C6 entry-exit in between PreemptToIdle and Load. • Load+UseHWElementPointer on the first load (out of reset) will cause execution to begin at the first valid element in the Q • Load+UseHWElementPointer without a preceding PreemptToIdle (i.e. when trying to Preempt a currently running Q) will cause undefined behavior.

EXECLIST_CONTROL - Execlist Control Register		
		○ UseHWElementPointer = 0 : HW begins execution at the first valid element in the Q.
	1	**Preempt to Idle** When SW writes a 1 to this bit, HW will immediately copy the contents of the Execution queue into the Submission queue. HW will preempt the executing context on appropriate preemption boundary, saves state and invalidates all the pending elements of the execution queue to be executed. HW saves the element pointer of the EQ on which it got preempted (indicates the element number of the preempted context in the EQ), element pointer is power context save/restored by HW. This forces HW to go idle triggering idle sequence for power management. A Preempt-to-idle message must be followed by a Load message to resume operation. This Load message may occur before or after a power gating/C6 sequence
	0	**Load** Writing to the Load bit triggers HW to sample Submission Queue (SQ) to Execution Queue (EQ) and start executing from Element-0 of Execution Queue. Doing a Load during an ongoing execution of an context will result in preemption and the new submission queue gets sampled to Execution Queue, however HW will not start executing from the newly updated Execution Queue until the preempted context is completely saved. Multiple loads occurring during the preemption of an executing context will result in EQ getting updated multiple times with the SQ and engine will only execute the most up to date EQ available upon completion of the preemption.

Execlist Status

EXECLIST_STATUS - Execlist Status	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02234h-0223Bh
Name:	RCS Execlist Status
ShortName:	EXECLIST_STATUS_RCSUNIT
Address:	18234h-1823Bh
Name:	RCS Execlist Status
ShortName:	EXECLIST_STATUS_POCSUNIT
Address:	22234h-2223Bh
Name:	RCS Execlist Status
ShortName:	EXECLIST_STATUS_BCSUNIT
Address:	1C0234h-1C023Bh
Name:	RCS Execlist Status
ShortName:	EXECLIST_STATUS_VCSUNIT0
Address:	1C4234h-1C423Bh
Name:	RCS Execlist Status
ShortName:	EXECLIST_STATUS_VCSUNIT1
Address:	1C8234h-1C823Bh
Name:	RCS Execlist Status
ShortName:	EXECLIST_STATUS_VECSUNIT0
Address:	1D0234h-1D023Bh
Name:	RCS Execlist Status
ShortName:	EXECLIST_STATUS_VCSUNIT2
Address:	1D4234h-1D423Bh
Name:	RCS Execlist Status
ShortName:	EXECLIST_STATUS_VCSUNIT3
Address:	1D8234h-1D823Bh
Name:	RCS Execlist Status
ShortName:	EXECLIST_STATUS_VECSUNIT1
Address:	1E0234h-1E023Bh
Name:	RCS Execlist Status

EXECLIST_STATUS - Execlist Status

ShortName:	EXECLIST_STATUS_VCSUNIT4
Address:	1E4234h-1E423Bh
Name:	RCS Execlist Status
ShortName:	EXECLIST_STATUS_VCSUNIT5
Address:	1E8234h-1E823Bh
Name:	RCS Execlist Status
ShortName:	EXECLIST_STATUS_VECSUNIT2
Address:	1F0234h-1F023Bh
Name:	RCS Execlist Status
ShortName:	EXECLIST_STATUS_VCSUNIT6
Address:	1F4234h-1F423Bh
Name:	RCS Execlist Status
ShortName:	EXECLIST_STATUS_VCSUNIT7
Address:	1F8234h-1F823Bh
Name:	RCS Execlist Status
ShortName:	EXECLIST_STATUS_VECSUNIT3
Address:	1A234h-1A23Bh
Name:	RCS Execlist Status
ShortName:	EXECLIST_STATUS_CCSUNIT0

This register contains the pointers and full indicator for the Execlist Queue and the context ID of the currently running context.DefaultValue = UUUU UUU1h (4:0 default to 00001b, others UNDEFINED).

Programming Notes

This register functionality is not supported and must not be programmed for Position command streamer.

DWord	Bit	Description
0	63:32	**Current Context ID** Format: U32 Contains the context ID of the currently running context.
	31	**Reserved** Access: RO Format: MBZ
	30	**Pending Load** Format: U1 When set indicates the Load of SQ to EQ is complete. Hardware is in the process of making the first valid element of the EQ to be active (executing).

EXECLIST_STATUS - Execlist Status

29:28	**Reserved** Access: RO Format: MBZ
27	**Arbitration Enable** Format: U1 This field reflects the Arbitration Flag set by the MI_ARB_ON_OFF command in Command Streamer.
26:12	**Last Context Switch Reason** Access: R/W Format: U15

Description

This field contains the switch reason for the last context to switch away.

Bit	Defintion
12	Idle to Active (special case of Switch to New Queue)
13	Switch To New Queue
14	Element Switch
15	Active to Idle (special case of Element Switch)
16	Context Complete
17	Wait on Sync Flip
18	Wait on V-Blank
19	Wait on Semaphore
20	Wait on Scan Line
26:21	Reserved

Programming Notes

This field should not written by SW.

11:8	**Active Context Offset** When Active Context field is set, this field indicates the active context offset within the execution queue.
7	**Active Context** When set indicates there is an active context being executed in hardware.
6:5	**Reserved** Access: RO Format: MBZ
4	**Valid Execution Queue Duplicate** Indicates there is an active context or a pending context or a pending load in progress.

EXECLIST_STATUS - Execlist Status		
	3	**Valid Execution Queue** Indicates there is an active context or a pending context or a pending load in progress.
	2	**Preempt to Idle Pending** Preempt to Idle Pending: HW has received Preempt to Idle load request from the scheduler and hardware is in the process of switching out the active context if any to force HW go IDLE.
	1	**Two Pending Load's** Indicates there are two pending loads in HW, (n-1)th load's first valid element is being pursued by HW to be made active and the Nth loads first valid element will be considered once (N-1)th is active. This situation will arise when Nth load happens while (N-1)th load is pending in hardware. Any further Load (N+1) occurring while this bit is set will result in overwriting the Nth SQ load, that is Nth SQ load contents will never be seen by hardware.
	0	**Execution Queue Invalid** Default Value: 1h There are no contexts available in Execution Queue to be executed. There are no pending loads (including Preempt To Idle) to be processed by the hardware. Scheduler can look for this bit to be set before doing a load of SQ to avoid preemption of any active contexts in hardware.

Execlist Submission Queue Contents

EXECLIST_SQ_CONTENTS - Execlist Submission Queue Contents	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	512
_Custom_GTIReset:	DEV
Address:	02510h-0254Fh
Name:	EXECLIST SQ CONTENTS
ShortName:	EXECLIST_SQ_CONTENTS_RCSUNIT
Address:	18510h-1854Fh
Name:	EXECLIST SQ CONTENTS
ShortName:	EXECLIST_SQ_CONTENTS_POCSUNIT
Address:	22510h-2254Fh
Name:	EXECLIST SQ CONTENTS
ShortName:	EXECLIST_SQ_CONTENTS_BCSUNIT
Address:	1C0510h-1C054Fh
Name:	EXECLIST SQ CONTENTS
ShortName:	EXECLIST_SQ_CONTENTS_VCSUNIT0
Address:	1C4510h-1C454Fh
Name:	EXECLIST SQ CONTENTS
ShortName:	EXECLIST_SQ_CONTENTS_VCSUNIT1
Address:	1C8510h-1C854Fh
Name:	EXECLIST SQ CONTENTS
ShortName:	EXECLIST_SQ_CONTENTS_VECSUNIT0
Address:	1D0510h-1D054Fh
Name:	EXECLIST SQ CONTENTS
ShortName:	EXECLIST_SQ_CONTENTS_VCSUNIT2
Address:	1D4510h-1D454Fh
Name:	EXECLIST SQ CONTENTS
ShortName:	EXECLIST_SQ_CONTENTS_VCSUNIT3
Address:	1D8510h-1D854Fh
Name:	EXECLIST SQ CONTENTS
ShortName:	EXECLIST_SQ_CONTENTS_VECSUNIT1
Address:	1E0510h-1E054Fh
Name:	EXECLIST SQ CONTENTS

EXECLIST_SQ_CONTENTS - Execlist Submission Queue Contents

ShortName:	EXECLIST_SQ_CONTENTS_VCSUNIT4
Address:	1E4510h-1E454Fh
Name:	EXECLIST SQ CONTENTS
ShortName:	EXECLIST_SQ_CONTENTS_VCSUNIT5
Address:	1E8510h-1E854Fh
Name:	EXECLIST SQ CONTENTS
ShortName:	EXECLIST_SQ_CONTENTS_VECSUNIT2
Address:	1F0510h-1F054Fh
Name:	EXECLIST SQ CONTENTS
ShortName:	EXECLIST_SQ_CONTENTS_VCSUNIT6
Address:	1F4510h-1F454Fh
Name:	EXECLIST SQ CONTENTS
ShortName:	EXECLIST_SQ_CONTENTS_VCSUNIT7
Address:	1F8510h-1F854Fh
Name:	EXECLIST SQ CONTENTS
ShortName:	EXECLIST_SQ_CONTENTS_VECSUNIT3
Address:	1A510h-1A54Fh
Name:	EXECLIST SQ CONTENTS
ShortName:	EXECLIST_SQ_CONTENTS_CCSUNIT0

Contents of submission queue from Element-0 to Element-7.

All "Element* Low Dword" have the format of the Bits[31:0] of the "Context Descriptor" definition.
All "Element* HighDword" have the format of the Bits[63:32] of the "Context Descriptor" definition.

DWord	Bit	Description
0	31:0	**Element 0 Low DWord** Format: U32
1	31:0	**Element 0 High DWord** Format: U32
2	31:0	**Element 1 Low DWord** Format: U32
3	31:0	**Element 1 High DWord** Format: U32
4	31:0	**Element 2 Low DWord** Format: U32
5	31:0	**Element 2 High DWord** Format: U32

EXECLIST_SQ_CONTENTS - Execlist Submission Queue Contents		
6	31:0	**Element 3 Low DWord** Format: U32
7	31:0	**Element 3 High DWord** Format: U32
8	31:0	**Element 4 Low DWord** Format: U32
9	31:0	**Element 4 High DWord** Format: U32
10	31:0	**Element 5 Low DWord** Format: U32
11	31:0	**Element 5 High DWord** Format: U32
12	31:0	**Element 6 Low DWord** Format: U32
13	31:0	**Element 6 High DWord** Format: U32
14	31:0	**Element 7 Low DWord** Format: U32
15	31:0	**Element 7 High DWord** Format: U32

Execlist Submit Port Register

EXECLIST_SUBMITPORT - Execlist Submit Port Register	
Register Space:	MMIO: 0/2/0
Access:	WO
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02230h-02233h
Name:	Execlist Submit Port Register
ShortName:	EXECLIST_SUBMITPORT_RCSUNIT
Address:	18230h-18233h
Name:	Execlist Submit Port Register
ShortName:	EXECLIST_SUBMITPORT_POCSUNIT
Address:	22230h-22233h
Name:	Execlist Submit Port Register
ShortName:	EXECLIST_SUBMITPORT_BCSUNIT
Address:	1C0230h-1C0233h
Name:	Execlist Submit Port Register
ShortName:	EXECLIST_SUBMITPORT_VCSUNIT0
Address:	1C4230h-1C4233h
Name:	Execlist Submit Port Register
ShortName:	EXECLIST_SUBMITPORT_VCSUNIT1
Address:	1C8230h-1C8233h
Name:	Execlist Submit Port Register
ShortName:	EXECLIST_SUBMITPORT_VECSUNIT0
Address:	1D0230h-1D0233h
Name:	Execlist Submit Port Register
ShortName:	EXECLIST_SUBMITPORT_VCSUNIT2
Address:	1D4230h-1D4233h
ShortName:	EXECLIST_SUBMITPORT_VCSUNIT3
Address:	1D8230h-1D8233h
Name:	Execlist Submit Port Register
ShortName:	EXECLIST_SUBMITPORT_VECSUNIT1
Address:	1E0230h-1E0233h
Name:	Execlist Submit Port Register
ShortName:	EXECLIST_SUBMITPORT_VCSUNIT4

EXECLIST_SUBMITPORT - Execlist Submit Port Register

Address:	1E4230h-1E4233h
Name:	Execlist Submit Port Register
ShortName:	EXECLIST_SUBMITPORT_VCSUNIT5

Address:	1E8230h-1E8233h
Name:	Execlist Submit Port Register
ShortName:	EXECLIST_SUBMITPORT_VECSUNIT2

Address:	1F0230h-1F0233h
Name:	Execlist Submit Port Register
ShortName:	EXECLIST_SUBMITPORT_VCSUNIT6

Address:	1F4230h-1F4233h
Name:	Execlist Submit Port Register
ShortName:	EXECLIST_SUBMITPORT_VCSUNIT7

Address:	1F8230h-1F8233h
Name:	Execlist Submit Port Register
ShortName:	EXECLIST_SUBMITPORT_VECSUNIT3

Address:	1A230h-1A233h
Name:	Execlist Submit Port Register
ShortName:	EXECLIST_SUBMITPORT_CCSUNIT0

ELSP provides a mechanism to load the elements of the Submission Queue in a cyclic order starting from Element-0 to Element-7. Consecutive writes to ELSP results in progressively updating lower dword followed by upper dword of successive elements starting from Element-0 to Element7. On reaching upper dword of Element-7 it wraps back to lower dword of Element-0.
Example: The first dword write to ELSP results in updating the lower dword of Element-0 and the following write updates the upper dword of Element-0 and the following write updates the lower dword of Element-1 and so on, on updating upper dword of Element-7 it wraps back to lower dword of Element-0.

DWord	Bit	Description
0	31:0	**Context Descriptor DW** Format: U32 See "Context Descriptor Format" for format. The element that this DW is submitted as and whether it is the high DW or the low DW is determined by order. This register must be written to 16 times in order to write to all the eight elements of an Submission Queue. .

Execute Condition Code Register

EXCC - Execute Condition Code Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02028h-0202Bh
Name:	Execute Condition Code Register
ShortName:	EXCC_RCSUNIT
Address:	18028h-1802Bh
Name:	Execute Condition Code Register
ShortName:	EXCC_POCSUNIT
Address:	22028h-2202Bh
Name:	Execute Condition Code Register
ShortName:	EXCC_BCSUNIT
Address:	1C0028h-1C002Bh
Name:	Execute Condition Code Register
ShortName:	EXCC_VCSUNIT0
Address:	1C4028h-1C402Bh
Name:	Execute Condition Code Register
ShortName:	EXCC_VCSUNIT1
Address:	1C8028h-1C802Bh
Name:	Execute Condition Code Register
ShortName:	EXCC_VECSUNIT0
Address:	1D0028h-1D002Bh
Name:	Execute Condition Code Register
ShortName:	EXCC_VCSUNIT2
Address:	1D4028h-1D402Bh
Name:	Execute Condition Code Register
ShortName:	EXCC_VCSUNIT3
Address:	1D8028h-1D802Bh
Name:	Execute Condition Code Register
ShortName:	EXCC_VECSUNIT1
Address:	1E0028h-1E002Bh
Name:	Execute Condition Code Register

EXCC - Execute Condition Code Register

ShortName:	EXCC_VCSUNIT4
Address:	1E4028h-1E402Bh
Name:	Execute Condition Code Register
ShortName:	EXCC_VCSUNIT5
Address:	1E8028h-1E802Bh
Name:	Execute Condition Code Register
ShortName:	EXCC_VECSUNIT2
Address:	1F0028h-1F002Bh
Name:	Execute Condition Code Register
ShortName:	EXCC_VCSUNIT6
Address:	1F4028h-1F402Bh
Name:	Execute Condition Code Register
ShortName:	EXCC_VCSUNIT7
Address:	1F8028h-1F802Bh
Name:	Execute Condition Code Register
ShortName:	EXCC_VECSUNIT3
Address:	1A028h-1A02Bh
Name:	Execute Condition Code Register
ShortName:	EXCC_CCSUNIT0

This register contains user defined and hardware generated conditions that are used by MI_WAIT_FOR_EVENT commands. An MI_WAIT_FOR_EVENT instruction excludes the executing ring from arbitration if the selected event evaluates to a 1, while instruction is discarded if the condition evaluates to a 0. Once excluded a ring is enabled into arbitration when the selected condition evaluates to a 0. This register also contains control for the invalidation of indirect state pointers on context restore.

DWord	Bit	Description
0	31:16	**Mask** Access: WO Format: Mask These bits serves as a write enable for bits 15:0. If this register is written with any of these bits clear the corresponding bit in the field 15:0 will not be modified. Reading these bits always returns 0s.
	15	**Context Wait for V-blank on Pipe-D** This field when set indicates the corresponding context has executed MI_WAIT_FOR_EVENT with "Display Pipe C Vertical Blank Wait Enable" set. This is an internal HW flag and should not be accessed by SW.

EXCC - Execute Condition Code Register

	14	**Context Wait for V-blank on Pipe-C** Source: RenderCS, BlitterCS This field when set indicates the corresponding context has executed MI_WAIT_FOR_EVENT with "Display Pipe C Vertical Blank Wait Enable" set. This is an internal HW flag and should not be accessed by SW.
	13	**Context Wait for V-blank on Pipe-B** Source: RenderCS, BlitterCS This field when set indicates the corresponding context has executed MI_WAIT_FOR_EVENT with "Display Pipe B Vertical Blank Wait Enable" set. This is an internal HW flag and should not be accessed by SW.
	12	**Context Wait for V-blank on Pipe-A** Source: RenderCS, BlitterCS This field when set indicates the corresponding context has executed MI_WAIT_FOR_EVENT with "Display Pipe A Vertical Blank Wait Enable" set. This is an internal HW flag and should not be accessed by SW.
	11:5	**Reserved** Access: RO Format: MBZ
	4:0	**User Defined Condition Codes** Source: RenderCS The software may signal a Stream Semaphore by setting the Mask bit and Signal Bit together to match the bit field specified in a WAIT_FOR_EVENT (Semaphore).

Execution Queue Element Mask

EQ_ELEMENT_MASK - Execution Queue Element Mask	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0256Ch-0256Fh
Name:	EQ_ELEMENT_MASK
ShortName:	EQ_ELEMENT_MASK_RCSUNIT
Address:	1856Ch-1856Fh
Name:	EQ_ELEMENT_MASK
ShortName:	EQ_ELEMENT_MASK_POCSUNIT
Address:	2256Ch-2256Fh
Name:	EQ_ELEMENT_MASK
ShortName:	EQ_ELEMENT_MASK_BCSUNIT
Address:	1C056Ch-1C056Fh
Name:	EQ_ELEMENT_MASK
ShortName:	EQ_ELEMENT_MASK_VCSUNIT0
Address:	1C456Ch-1C456Fh
Name:	EQ_ELEMENT_MASK
ShortName:	EQ_ELEMENT_MASK_VCSUNIT1
Address:	1C856Ch-1C856Fh
Name:	EQ_ELEMENT_MASK
ShortName:	EQ_ELEMENT_MASK_VECSUNIT0
Address:	1D056Ch-1D056Fh
Name:	EQ_ELEMENT_MASK
ShortName:	EQ_ELEMENT_MASK_VCSUNIT2
Address:	1D456Ch-1D456Fh
Name:	EQ_ELEMENT_MASK
ShortName:	EQ_ELEMENT_MASK_VCSUNIT3
Address:	1D856Ch-1D856Fh
Name:	EQ_ELEMENT_MASK
ShortName:	EQ_ELEMENT_MASK_VECSUNIT1
Address:	1E056Ch-1E056Fh
Name:	EQ_ELEMENT_MASK

EQ_ELEMENT_MASK - Execution Queue Element Mask

ShortName:	EQ_ELEMENT_MASK_VCSUNIT4
Address:	1E456Ch-1E456Fh
Name:	EQ_ELEMENT_MASK
ShortName:	EQ_ELEMENT_MASK_VCSUNIT5
Address:	1E856Ch-1E856Fh
Name:	EQ_ELEMENT_MASK
ShortName:	EQ_ELEMENT_MASK_VECSUNIT2
Address:	1F056Ch-1F056Fh
Name:	EQ_ELEMENT_MASK
ShortName:	EQ_ELEMENT_MASK_VCSUNIT6
Address:	1F456Ch-1F456Fh
Name:	EQ_ELEMENT_MASK
ShortName:	EQ_ELEMENT_MASK_VCSUNIT7
Address:	1F856Ch-1F856Fh
Name:	EQ_ELEMENT_MASK
ShortName:	EQ_ELEMENT_MASK_VECSUNIT3
Address:	1A56Ch-1A56Fh
Name:	EQ_ELEMENT_MASK
ShortName:	EQ_ELEMENT_MASK_CCSUNIT0

This register captures the status of the elements executed from an Execution Queue on Preempt to Idle. HW looks at this register on Load with Use HW Element Pointer to pick the right element (instead of default Element-0) to execute in order to resume from where it got preempted on Preempt To Idle. This register must not be programmed by SW. This register is power context save/restored by HW.

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7:0	**Load** Each bit in the mask corresponds to an element in the execution queue. Bit[0] corresponds to Element-0 and Bit[7] corresponds to Element-7 of the EQ. Bit set for an element indicates the element is executed. Bit Reset for an element indicates either the element is invalid or pending execution (a preempted context is treated as pending execution).

FAULT_TLB_RD_DATA1 Register

FAULT_TLB_RD_DATA1 - FAULT_TLB_RD_DATA1 Register

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0CEBCh

DWord	Bit	Description
0	31:9	**Reserved** Access: RO Format: MBZ
	8:7	**TLB Entry Page Size** Default Value: 00b Access: RO 2'b00 - 4k, 2'b01 - 64k, 2'b10 - 2M, 2'b11 - 1G
	6	**TLB Entry Present** Default Value: 0h Access: RO 1'b1 - Present, 1'b0 - Not Present
	5	**TLB Entry Valid** Default Value: 0h Access: RO 1'b1 - Valid, 1'b0 - Not Valid
	4	**Cycle GTT Sel** Default Value: 0h Access: RO Cycle GTT SEL (1-GGTT Cycle, 0-PPGTT Cycle)
	3:0	**Address** Default Value: 0000b Access: RO Bit[3:0] Fault cycle Virtual address [47:44]

FBC_CFB_BASE

FBC_CFB_BASE	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	43200h-43203h
Name:	FBC Compressed Buffer Address
ShortName:	FBC_CFB_BASE_A
Reset:	soft

Restriction
The contents of this register must not be changed while compression is enabled.

DWord	Bit	Description
0	31:28	**Reserved** Access: RO Format: MBZ
	27:12	**CFB Offset Address** This register specifies bits 27:12 of the offset of the Compressed Frame Buffer from the base of stolen memory. A programmed value of 0x0001 in this field corresponds to an offset of 0x1000 (4K) bytes. **Restriction** The buffer must be 4K byte aligned, which is enforced by reserving bits 11:0. The offset must be greater than 4K bytes, avoiding the first 4KB of stolen memory.
	11:0	**Reserved** Access: RO Format: MBZ

FBC_CTL

FBC_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	43208h-4320Bh
Name:	FBC Control
ShortName:	FBC_CTL_A
Reset:	soft

Description
FBC is tied to Plane 1 A.

Programming Notes
Frame Buffer Compression is not supported with per-pixel alpha. Software must disable FBC for per-pixel alpha formats.
Frame Buffer Compression is supported with surfaces of up to 8192 pixels x 4096 lines and plane sizes up to 5120 pixels x 4096 lines.
The FBC compressed vertical limit is 2560 lines, after which the remaining lines will be displayed correctly, but will not be compressed.
Compression ratios cannot be changed dynamically. FBC must be disabled and reenabled when changing compression ratios.

Restriction
The contents of this register must not be changed, except the enable bit, while compression is enabled.
Frame Buffer Compression is only supported with 16bpp and 32bpp 8:8:8 RGB plane source pixel formats. It is not supported with any other format. The 16bpp format requires the compression ratio to be set to 2:1 or 4:1. FBC is not supported with RGB 16bpp plane formats when plane 90/270 rotation is enabled. Frame Buffer Compression is only supported while the plane it is tied to has a source size of at least 200 pixels x 32 lines. Software must disable FBC for the non-supported pixel formats and sizes.
Frame Buffer Compression is not supported with interlaced fetch. With plane 90/270 rotation, all frame buffer modifications will result in full frame invalidation and recompression.

DWord	Bit	Description
0	31	**Enable FBC** This bit is used to globally enable FBC function at the next Vertical Blank start. FBC should not be enabled when the pipe is disabled.

Value	Name
0b	Disable

FBC_CTL

	1b	Enable

30	**Reserved** Access: RO Format: MBZ
29	**Reserved** Access: RO Format: MBZ
28	**Reserved**
27:19	**Reserved** Access: RO Format: MBZ
18	**Reserved**
17	**Reserved**
16	**Reserved**
15	**Reserved**
14:13	**Reserved** Access: RO Format: MBZ
12:11	**Reserved** Access: RO Format: MBZ
10	**Reserved**
9:8	**Reserved**
7:6	**Compression Limit** This register sets a minimum limit on compression. This determines the maximum size of the compressed frame buffer. Display lines that do not meet the compression limit will not be compressed, so the best compression will be achieved with a 1:1 ratio. FBC processes received pixels as 32bpp irrespective of pixel format, so that CFB buffer sizing for 16bpp is indicated below: Compression Ratio 1, Pixel Format 16 bpp - Not Supported Compression Ratio 1, Pixel Format 32 bpp - Supported (CFB=FB) Compression Ratio 1/2, Pixel Format 16 bpp - Supported (CFB=FB) Compression Ratio 1/2, Pixel Format 32 bpp - Supported (CFB=1/2 FB) Compression Ratio 1/4, Pixel Format 16 bpp - Supported (CFB=1/2 FB) Compression Ratio 1/4, Pixel Format 32 bpp - Supported (CFB=1/4 FB) FB = Frame Buffer Size CFB = Compressed Frame Buffer Size

Value	Name	Description
00b	1:1	Compressed buffer is the same size as the uncompressed buffer.

01b	2:1	Compressed buffer is one half the size of the uncompressed buffer.
10b	4:1	Compressed buffer is one quarter the size of the uncompressed buffer.
11b	Reserved	Reserved

5:4 Write Back Watermark

The compressed data write back engine waits for this number of entries to be ready before writing the data out to memory.

Value	Name	Description
00b	4	4 entries
01b	8	8 entries
10b	16	16 entries
11b	32	32 entries

3:0 Reserved

FBC_RT_BASE_ADDR_REGISTER

FBC_RT_BASE_ADDR_REGISTER - FBC_RT_BASE_ADDR_REGISTER

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	07020h

This Register is saved and restored as part of Context.

DWord	Bit	Description
0	31:12	**FBC RT Base Address** Access: R/W Format: GraphicsAddress[31:12] _Custom_GTIReset: DEV 4KB aligned Base Address as mapped in the PPGTT or in the GGTT for the render target. This base address must be the one that is either front buffer or the back-buffer (a flip target). It must be programmed before any draw call binding that render target base address.
	11:2	**Reserved** Access: R/W Format: PBC _Custom_GTIReset: DEV
	1	**FBC Front Buffer Target** Access: R/W _Custom_GTIReset: DEV Value / Name / Description: 0h / **[Default]** / FBC is targeting the Back Buffer for compression. This buffer can be cached in the MLC/LLC, so a GFDT flush is required before FBC can begin compression. 1h / / FBC is targeting the Font Buffer for compression. This buffer cannot be cached in the MLC/LLC. FBC compression can begin after any RC flush.
	0	**PPGTT Render Target Base Address Valid for FBC** Access: R/W _Custom_GTIReset: DEV Value / Name / Description: 0h / **[Default]** / Base address in this register [31:12] is not valid and therefore FBC will not get any modifications from rendering. 1h / / Base address in this register [31:12] is valid and HW needs to compare the

FBC_RT_BASE_ADDR_REGISTER - FBC_RT_BASE_ADDR_REGISTER				
				current render target base address with this base address to provide modifications to FBC.

FBC_RT_BASE_ADDR_REGISTER_UPPER

FBC_RT_BASE_ADDR_REGISTER_UPPER - FBC_RT_BASE_ADDR_REGISTER_UPPER

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	07024h

This Register is saved and restored as part of Context.

Programming Notes

"Render Tracking with Nuke" is the only FBC functional mode supported by render engine. SW must always program the FBC_RT_BASE_ADDR_REGISTER_* register in Render Engine to a reserved value (0xFFFF_FFFF) such that the programmed value doesn't match the render target surface address programmed. This would disable render engine from generating modify messages to FBC unit in display. Refer "Frame Buffer Compression" section for more details related to FBC functionality and programming.

DWord	Bit	Description
0	31:16	**Reserved** Access: R/W Format: PBC _Custom_GTIReset: DEV
	15:0	**FBC RT Base Address Upper DWORD** Access: R/W Format: GraphicsAddress[47:32] _Custom_GTIReset: DEV Must be set to modify corresponding data bit. Reads to this field returns zero.Upper 4KB aligned Base Address as mapped in the PPGTT or in the GGTT for the render target. This base address must be the one that is either front buffer or the back-buffer (a flip target). It can be only programmed once per context. **Programming Notes** It must be programmed before any draw call binding that render target base address.

FBC LLC Config Read Control Register

FBC_LLC_READ_CTRL - FBC LLC Config Read Control Register

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	09044h

FBC LLC Config Read Control Register

DWord	Bit	Description
0	31	**FBC LLC Config Read Control Register Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of FBC_CTRL Register are R/W. 1 = All bits of FBC_CTRL Register are RO (including this lock bit). Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 does not clear the lock). These bits are not reset on FLR.
	30	**FBC LLC Config Start Value** Default Value: 1b Access: R/W Lock _Custom_GTIReset: BUS PCU_CR_LLC_CONFIG Read Cycle Interval in microseconds (1333ns normal). 1'b0 - Treat LLC as partially open on reset (boot or C6 exit) (Default). 1'b1 - Treat LLC as fully open on reset (boot or C6 exit). This must not be set unless coordinated with Uncore.
	29:16	**Reserved** Access: RO Format: MBZ
	15:0	**FBC LLC Config Read Interval** Default Value: 00FFh Access: R/W Lock _Custom_GTIReset: BUS PCU_CR_LLC_CONFIG Read Cycle Interval in microseconds (1333ns normal). 0x0000: Do not read PCU_CR_LLC_CONFIG (use Start Value only). 0x0001-0xFFFF : Read PCU_CR_LLC_CONFIG at the specified interval, until LLC_FULLY_OPEN=1. Default: 0xFF (approx 170us).

FENCE_LSB

FENCE_LSB	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	100020h
Name:	FENCE4_LSB
ShortName:	FENCE4_LSB
Address:	100028h
Name:	FENCE5_LSB
ShortName:	FENCE5_LSB
Address:	100030h
Name:	FENCE6_LSB
ShortName:	FENCE6_LSB
Address:	100038h
Name:	FENCE7_LSB
ShortName:	FENCE7_LSB
Address:	100040h
Name:	FENCE8_LSB
ShortName:	FENCE8_LSB
Address:	100048h
Name:	FENCE9_LSB
ShortName:	FENCE9_LSB
Address:	100050h
Name:	FENCE10_LSB
ShortName:	FENCE10_LSB
Address:	100058h
Name:	FENCE11_LSB
ShortName:	FENCE11_LSB
Address:	100060h
Name:	FENCE12_LSB
ShortName:	FENCE12_LSB
Address:	100068h
Name:	FENCE13_LSB
ShortName:	FENCE13_LSB

FENCE_LSB	
Address:	100070h
Name:	FENCE14_LSB
ShortName:	FENCE14_LSB
Address:	100078h
Name:	FENCE15_LSB
ShortName:	FENCE15_LSB
Address:	100080h
Name:	FENCE16_LSB
ShortName:	FENCE16_LSB
Address:	100088h
Name:	FENCE17_LSB
ShortName:	FENCE17_LSB
Address:	100090h
Name:	FENCE18_LSB
ShortName:	FENCE18_LSB
Address:	100098h
Name:	FENCE19_LSB
ShortName:	FENCE19_LSB
Address:	1000A0h
Name:	FENCE20_LSB
ShortName:	FENCE20_LSB
Address:	1000A8h
Name:	FENCE21_LSB
ShortName:	FENCE21_LSB
Address:	1000B0h
Name:	FENCE22_LSB
ShortName:	FENCE22_LSB
Address:	1000B8h
Name:	FENCE23_LSB
ShortName:	FENCE23_LSB
Address:	1000C0h
Name:	FENCE24_LSB
ShortName:	FENCE24_LSB
Address:	1000C8h
Name:	FENCE25_LSB

FENCE_LSB

ShortName:	FENCE25_LSB
Address:	1000D0h
Name:	FENCE26_LSB
ShortName:	FENCE26_LSB
Address:	1000D8h
Name:	FENCE27_LSB
ShortName:	FENCE27_LSB
Address:	1000E0h
Name:	FENCE28_LSB
ShortName:	FENCE28_LSB
Address:	1000E8h
Name:	FENCE29_LSB
ShortName:	FENCE29_LSB
Address:	1000F0h
Name:	FENCE30_LSB
ShortName:	FENCE30_LSB
Address:	1000F8h
Name:	FENCE31_LSB
ShortName:	FENCE31_LSB

Fence Registers LSBs

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="3">0</td><td>31:12</td><td>FENCELO<table><tr><td>Default Value:</td><td>000000h</td></tr><tr><td>Access:</td><td>R/W</td></tr><tr><td>_Custom_GTIReset:</td><td>BUS</td></tr></table>Bits 31:12 of the ending Graphics Address of the fence region. Fence regions must be aligned to a 4KB page.
This address represents the last 4KB page of the fence region (Lower Bound is included in the fence region).
Graphics Address is the offset within GMADR space.</td></tr>
<tr><td>11:2</td><td>Reserved<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>1</td><td>TILE<table><tr><td>Default Value:</td><td>0b</td></tr><tr><td>Access:</td><td>R/W</td></tr></table></td></tr>
</table>

FENCE_LSB

		_Custom_GTIReset: BUS This field specifies the spatial ordering of QW within tiles. 0b - Consecutive SWords (32B) sequenced in the X direction 1b - Consecutive OWords (16B) sequenced in the Y direction
	0	**FENCEVAL** Default Value: 0b Access: R/W _Custom_GTIReset: BUS This field specifies whether or not this fence register defines a fence region. 0b - FENCE INVALID 1b - FENCE VALID

FENCE_MSB

FENCE_MSB	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	100024h
Name:	FENCE4_MSB
ShortName:	FENCE4_MSB
Address:	10002Ch
Name:	FENCE5_MSB
ShortName:	FENCE5_MSB
Address:	100034h
Name:	FENCE6_MSB
ShortName:	FENCE6_MSB
Address:	10003Ch
Name:	FENCE7_MSB
ShortName:	FENCE7_MSB
Address:	100044h
Name:	FENCE8_MSB
ShortName:	FENCE8_MSB
Address:	10004Ch
Name:	FENCE9_MSB
ShortName:	FENCE9_MSB
Address:	100054h
Name:	FENCE10_MSB
ShortName:	FENCE10_MSB
Address:	10005Ch
Name:	FENCE11_MSB
ShortName:	FENCE11_MSB
Address:	100064h
Name:	FENCE12_MSB
ShortName:	FENCE12_MSB
Address:	10006Ch
Name:	FENCE13_MSB
ShortName:	FENCE13_MSB

FENCE_MSB	
Address:	100074h
Name:	FENCE14_MSB
ShortName:	FENCE14_MSB
Address:	10007Ch
Name:	FENCE15_MSB
ShortName:	FENCE15_MSB
Address:	100084h
Name:	FENCE16_MSB
ShortName:	FENCE16_MSB
Address:	10008Ch
Name:	FENCE17_MSB
ShortName:	FENCE17_MSB
Address:	100094h
Name:	FENCE18_MSB
ShortName:	FENCE18_MSB
Address:	10009Ch
Name:	FENCE19_MSB
ShortName:	FENCE19_MSB
Address:	1000A4h
Name:	FENCE20_MSB
ShortName:	FENCE20_MSB
Address:	1000ACh
Name:	FENCE21_MSB
ShortName:	FENCE21_MSB
Address:	1000B4h
Name:	FENCE22_MSB
ShortName:	FENCE22_MSB
Address:	1000BCh
Name:	FENCE23_MSB
ShortName:	FENCE23_MSB
Address:	1000C4h
Name:	FENCE24_MSB
ShortName:	FENCE24_MSB
Address:	1000CCh
Name:	FENCE25_MSB

FENCE_MSB

ShortName:	FENCE25_MSB
Address:	1000D4h
Name:	FENCE26_MSB
ShortName:	FENCE26_MSB
Address:	1000DCh
Name:	FENCE27_MSB
ShortName:	FENCE27_MSB
Address:	1000E4h
Name:	FENCE28_MSB
ShortName:	FENCE28_MSB
Address:	1000ECh
Name:	FENCE29_MSB
ShortName:	FENCE29_MSB
Address:	1000F4h
Name:	FENCE30_MSB
ShortName:	FENCE30_MSB
Address:	1000FCh
Name:	FENCE31_MSB
ShortName:	FENCE31_MSB

Fence Registers MSBs

DWord	Bit	Description
0	31:12	**FENCEUP** Default Value: 00000000h Access: R/W _Custom_GTIReset: BUS Bits 31:12 of the ending Graphics Address of the fence region. Fence regions must be aligned to a 4KB page. This address represents the last 4KB page of the fence region (Upper Bound is included in the fence region). Graphics Address is the offset within GMADR space.
	11	**Reserved** Access: RO Format: MBZ

FENCE_MSB

	10:0	**Pitch** Default Value: 000h Access: R/W _Custom_GTIReset: BUS This field specifies the width (pitch) of the fence region in multiple of tile widths. For Tile X this field must be programmed to a multiple of 512B (003 is the minimum value) and for Tile Y this field must be programmed to a multiple of 128B (000 is the minimum value). 000h = 128B 001h = 256B 002h = 384B 003h = 512B 004h = 640B 005h = 768B 006h = 896B 007h = 1024B ... 3FFh = 128KB 4FFh = 160KB 5FFh = 192KB 6FFh = 224KB 7FFh = 256KB

Fence Control Register

MFCR - Fence Control Register

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	09070h

Fence Control Register

DWord	Bit	Description
0	31	**Fuse Override Lock** Access: R/W Lock _Custom_GTIReset: BUS SW Fuse Override Lock Bit
	30:28	**ECORSVD** Access: R/W _Custom_GTIReset: BUS ECO purposes Reserved
	27:26	**GT VBOX DISABLE FUSE OVERRIDE** Access: R/W Lock _Custom_GTIReset: BUS S/W GT Vbox Disable Fuse Override Bits
	25:22	**GT SUBSLICE DISABLE FUSE OVERRIDE** Access: R/W Lock _Custom_GTIReset: BUS SW GT SubSlice Disable Fuse Override Bits
	21:16	**GT SLICE ENABLE FUSE OVERRIDE** Default Value: 111111b Access: R/W Lock _Custom_GTIReset: BUS SW GT Slice Enable Fuse Override Bits
	15:5	**Reserved** Access: RO Format: MBZ
	4	**Reserved**
	3	**Reserved**

MFCR - Fence Control Register

	2	**Write/Read Port Block** Access: R/W _Custom_GTIReset: BUS 0 - Dont Block the R/W port when Query is started. 1 - Block the R/W port until the Memory Fence is completed. This is applicable for only Memory Fence.
	1	**LLC Query Enable** Access: R/W _Custom_GTIReset: BUS 0 - Query for 16 Ways. 1 - Query for 32 Ways. No Flexing.
	0	**Fence Controller GFDT Mode** Access: R/W _Custom_GTIReset: BUS Fence Controller GFDT Mode. 0 - Single bit GFDT mode. 1 - Two bit GFDT mode.

FF Performance

FF_PERF - FF Performance

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	06B1Ch
Name:	Render CS FF Performance
ShortName:	RCS_FF_PERF
Address:	17B1Ch
Name:	Position CS POSH Pipeline FF Performance
ShortName:	PCS_FF_PERF

DWord	Bit	Description
0	31:16	**Mask** Access: WO Format: Mask _Custom_GTIReset: DEV Must be set to modify corresponding bit in Bits 15:0. (All implemented bits)
	15:11	**Reserved** Access: R/W Format: PBC _Custom_GTIReset: DEV
	10:8	**Throttle counter value** Access: R/W _Custom_GTIReset: DEV Counter value defining how many clocks the interface needs to be slowed down. Value: 0h — Name: **[Default]** — Description: Masked by default.
	7:3	**Reserved** Access: R/W Format: PBC _Custom_GTIReset: DEV
	2	**Enable throttling for SF-WM interface** Access: R/W _Custom_GTIReset: DEV

FF_PERF - FF Performance

Value	Name	Description
0h	Disable_2	No throttling
1h	Enable_2	Enable throttling in all SF-WM interfaces

Programming Notes
This field must not be programmed for SVGR unit.

1 **Enable throttling for SF-SBE interface**

Access:	R/W
_Custom_GTIReset:	DEV

Value	Name	Description
0h	Disable_1	No throttling
1h	Enable_1	Enable throttling in all SF-SBE interfaces

Programming Notes
This field must not be programmed for SVGR unit.

0 **Enable throttling for CL-SF interface**

Access:	R/W
_Custom_GTIReset:	DEV

Value	Name	Description
0h	Disable_0	No throttling
1h	Enable_0	Enable throttling in all CL-SF interfaces

Restriction
This bit must not be set. SW may choose to use SF-SBE throttle interface(bit 1) to achieve the same effect.

First MGSR read done

FIRST_MGSR_READ_DONE - First MGSR read done	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	BUS
Address:	1C2DA4h-1C2DA7h
Name:	First MGSR read done
ShortName:	FIRST_MGSR_READ_DONE_VDENC_REG0
Address:	1C6DA4h-1C6DA7h
Name:	First MGSR read done
ShortName:	FIRST_MGSR_READ_DONE_VDENC_REG1
Address:	1D2DA4h-1D2DA7h
Name:	First MGSR read done
ShortName:	FIRST_MGSR_READ_DONE_VDENC_REG2
Address:	1D6DA4h-1D6DA7h
Name:	First MGSR read done
ShortName:	FIRST_MGSR_READ_DONE_VDENC_REG3
Address:	1E2DA4h-1E2DA7h
Name:	First MGSR read done
ShortName:	FIRST_MGSR_READ_DONE_VDENC_REG4
Address:	1E6DA4h-1E6DA7h
Name:	First MGSR read done
ShortName:	FIRST_MGSR_READ_DONE_VDENC_REG5
Address:	1F2DA4h-1F2DA7h
Name:	First MGSR read done
ShortName:	FIRST_MGSR_READ_DONE_VDENC_REG6
Address:	1F6DA4h-1F6DA7h
Name:	First MGSR read done
ShortName:	FIRST_MGSR_READ_DONE_VDENC_REG7
This register is used to enable wrap around cases in the current testbench. There is not good way to identify the delay that should be introduced in Primary.bfl file before the wrapped around value is written to MGSR register. RTL will write 1 to this register when the first value is read by ECP and reset at frame_flop_reset. This will indicate that the read has happened and the wrap around can happen. this is introduced, because in real life it should never happen that the tailptr value read by ECP is of the next frame before reading at least one tailptr value of this frame.	

DWord	Bit	Description

FIRST_MGSR_READ_DONE - First MGSR read done		
0	31:16	**mgsr_return_value_zero** Access: RO This field is cleared when a WL is submitted to VDENC and set if the tail pointer value received by VDENC is zero..
	15:8	**VDENC_End_of_frame_processed** Access: RO This field is cleared when a WL is submitted to VDENC and set after VDENC process entire frame.
	7:0	**First_MGSR_read_done** Access: RO This field is cleared when a WL is submitted to VDENC and set after reading first tail pointer from MGSR.

First VF Offset

FIRST_VF_OFFSET_0_2_0_PCI - First VF Offset		
Register Space: PCI: 0/2/0 Size (in bits): 16		
Address: 00334h		
Defines the offset of the function number from the PF to the first VF.		
DWord	**Bit**	**Description**
0	15:0	**FIRST VF OFFSET VALUE** Default Value: 0000000000000001b Access: RO _Custom_GTIReset: BUS Defines the routing ID offset of the first VF that is associated with the PF that contains this Capability structure. The first VFs 16-bit Routing ID is calculated by adding the contents of this field to the Routing ID of the PF containing this field ignoring any carry, using unsigned, 16-bit arithmetic. The value of this field is hardwired to 0001h.

FIX BONUS1 Reg

FIXSPCBONUS1 - FIX BONUS1 Reg

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24314h

Clock Gating Messages Register

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**BONUS1 BIT 7** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	6	**BONUS1 BIT 6** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	5	**BONUS1 BIT 5** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	4	**BONUS1 BIT 4** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

FIXSPCBONUS1 - FIX BONUS1 Reg

	3	**BONUS1 BIT 3** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	2	**BONUS1 BIT 2** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	1	**BONUS1 BIT 1** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**BONUS1 BIT 0** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

FIX BONUS2 Reg

FIXSPCBONUS2 - FIX BONUS2 Reg

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24318h

Clock Gating Messages Register

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**BONUS2 BIT 7** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	6	**BONUS2 BIT 6** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	5	**BONUS2 BIT 5** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	4	**BONUS2 BIT 4** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

FIXSPCBONUS2 - FIX BONUS2 Reg

	3	**BONUS2 BIT 3** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	2	**BONUS2 BIT 2** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	1	**BONUS2 BIT 1** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**BONUS2 BIT 0** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

FIX PGFET control register with lock

FIXSPCPFETCTL - FIX PGFET control register with lock

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24308h

DWord	Bit	Description
0	31	**PFET Control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of Slice 0 PGFETCTL register are R/W 1 = All bits of Slice 0 PGFETCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:24	**Reserved** Access: RO Format: MBZ
	23	**Power Well Status** Access: RO _Custom_GTIReset: BUS 0 = Well is powered Down 1 = Well is powered up Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	22	**Powergood timer error** Access: RO _Custom_GTIReset: BUS 0 = Well is powered Down 1 = Well is powered up Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	21:19	**Delay from enabling secondary PFETs to power good.** Access: R/W Lock _Custom_GTIReset: BUS Delay from enabling secondary PFETs to power good 3'b000: 40ns 3'b001: 80ns 3'b010: 160ns

FIXSPCPFETCTL - FIX PGFET control register with lock

3'b011: 240ns
3'b100: 320ns
3'b101: 480ns
3'b110: 640ns
3'b111: 1280ns

Value	Name
101b	**[Default]**

18:16 **Strobe pulse period**

Default Value:	010b
Access:	R/W Lock
_Custom_GTIReset:	BUS

Time period b/w two adjacent strobes to the primary FETs
3'b000: 10ns (or 1 bclk)
3'b001: 20ns (or 2 bclk)
3'b010: 30ns (or 3 bclk)
3'b111: 80ns (or 8 bclk)

15:0 **PFET Ladder Step Sequence**

Default Value:	1111111111111111b
Access:	R/W Lock
_Custom_GTIReset:	BUS

PFET Ladder STEP sequence
The PFET ladder has 16 steps, each represented by its corresponding bit in pfetlddrstepseq[15:0]. If the bit location is set, it means we must step on that ladder stage before we go to the next stage.
The SPC controller will step through all the steps with asserted bits periodically with period equal to strbpulsprd[2:0]
Programming Rule: pfetldrstepseq[15] and pfetldrstepseq[0] must be '1; A '0 setting for these bits is illegal.
15'FFFFh: Ladder step (ladder_sel) goes 0, 1, 2, ?.15.
15'FFF1h: Ladder step goes 0, 4, 5, 6,?15; Steps 1, 2, 3 are skipped.
15'D555h: Ladder step goes 0, 2, 4, 6, 8, 10, 12, 14, 15; Steps 1, 3, 5, 7, 9, 11, 13 are skipped.
15'80001h: Ladder step goes 0, 15; Steps 1, 2, 3, 4, ?14 are skipped.

Fix Power Context Save request

FIXPGCTXREQ - Fix Power Context Save request

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24304h

DWord	Bit	Description
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask bots for lower 16 bits
	15:10	**Reserved** Access: RO Format: MBZ
	9	**Power context save request** Access: R/W Set _Custom_GTIReset: BUS Power Context Save Request 1'b0 : Power context save is not being requested 1'b1 : Power context save is being requested CPUnit self-clears this bit upon sampling.
	8:0	**Power Context Save request crdit count** Access: R/W _Custom_GTIReset: BUS QWord Credits for Power Context Save Request Minimum Credits = 1 : Unit may send 1 QWord pair (enough for first LRI at least) Maximum Credits = 511 : Unit may send 511 QWord pairs A QWord pair is defined as a 32-bit register address and the corresponding 32-bits of register data. Note that the LRI header and END commands are 64-bits each (32-bit command followed by 32-bit NOOP) and will consume one QWord credit. Only valid with PWRCTX_SAVE_REQ (Bit9).

FIX Power Down FSM control register with lock

FIXSPCPOWERDNFSMCTL - FIX Power Down FSM control register with lock

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24310h

DWord	Bit	Description
0	31	**power down control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of Slice 0 POWERDNFSMCTL register are R/W 1 = All bits of Slice 0 POWERDNFSMCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:13	**Reserved** Access: RO Format: MBZ
	12	**Leave firewall disabled** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not firewall the gated domain for a power down flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e firewall gated domain to ungated domain crossing during power down flows 1 = Leave firewall disabled, i.e dont firewall the gated domain, but complete logical flow
	11	**Leave reset de-asserted** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not assert reset for power off flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e assert resets during power down flows 1 = Leave reset de-asserted mode, i.e dont assert reset, but complete logical flow

FIXSPCPOWERDNFSMCTL - FIX Power Down FSM control register with lock

10	**Leave CLKs ON** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not gate clks for power off flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e gate clocks during power down flows 1 = Leave CLKS ON mode, i.e dont clock gate, but complete logical flow
9	**Leave FET On** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not turn off the PFET eventhough it will complete the flow with PM Encodings: 0 = Default mode, i.e power off fets during power down flows 1 = Leave ON mode, i.e dont power off pfet, but complete logical flow
8:6	**Power Down state 3** Default Value: 010b Access: R/W Lock _Custom_GTIReset: BUS This will be the 3rd state before power is turned OFF in the well Encodings: 000 = Assert Reset 001 = Firewall ON 010 = Gate clocks 1xx = Rsvd for future Default : Gate Clocks
5:3	**Power Down state 2** Default Value: 001b Access: R/W Lock _Custom_GTIReset: BUS This will be the 2nd state before power is turned OFF in the well Encodings: 000 = Assert Reset 001 = Firewall ON 010 = Gate clocks 1xx = Rsvd for future Default :Firewall ON

FIXSPCPOWERDNFSMCTL - FIX Power Down FSM control register with lock

	2:0	**Power Down state 1** Default Value: 000b Access: R/W Lock _Custom_GTIReset: BUS This will be the 1st state before power is turned OFF in the well Encodings: 000 = Assert Reset 001 = Firewall ON 010 = Gate clocks 1xx = Rsvd for future Default : Assert Reset

Fix Power Gate Control Request

FIXPGCTLREQ - Fix Power Gate Control Request

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24300h

Clock Gating Messages Register

DWord	Bit	Description
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask In order to write to bits 15:0, the corresponding message mask bits must be written. For example, for bit 14 to be set, bit 30 needs to be 1 : 40004000
	15:2	**Reserved** Access: RO Format: MBZ
	1	**CLK RST FWE Request** Access: R/W _Custom_GTIReset: BUS SLICE 0 CLK RST FWE request: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**Power Gate Request** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

FIX Power on FSM control register with lock

FIXSPCPOWERUPFSMCTL - FIX Power on FSM control register with lock

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	2430Ch

DWord	Bit	Description
0	31	**power up control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of Slice 0 POWERUPFSMCTL register are R/W 1 = All bits of Slice 0 POWERUPFSMCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:9	**Reserved** Access: RO Format: MBZ
	8:6	**Power UP state 3** Default Value: 010b Access: R/W Lock _Custom_GTIReset: BUS This will be the 3rd state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - De-assert resets 3'b000: 10ns (or 1 bclk)
	5:3	**Power UP state 2** Default Value: 001b Access: R/W Lock _Custom_GTIReset: BUS This will be the 2nd state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF

FIXSPCPOWERUPFSMCTL - FIX Power on FSM control register with lock

		010 = De-assert resets 1xx = Rsvd for future Default - Firewall OFF
	2:0	**Power UP state 1** Default Value: 000b Access: R/W Lock _Custom_GTIReset: BUS This will be the 1st state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - Clock Ungate

Flexible EU Event Control 0

EU_PERF_CNT_CTL0 - Flexible EU Event Control 0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0E458h

This register configures flexible EU event 0/1. Please refer to the description of the flexible EU events for more details on supported events. Please note that this register is render context saved/restored.

DWord	Bit	Description
0	31:26	**Reserved** Access: RO Format: MBZ
	25:24	**Reserved** Access: RO Format: MBZ
	23:20	**Fine Event Filter Select EU event 1** This field controls which fine event filter is applied to the coarsely filtered increment event when creating flexible EU event 1. Note that the fine event filter is logically applied after the coarse event filter. Value: 0xf — Name: Default **[Default]** Value: [0x0-0xA] — Name:
	19:16	**Coarse Event Filter Select EU event 1** This field controls which coarse event filter is applied to the selected increment event when creating flexible EU event 1. Note that the coarse event filter is logically applied before the fine event filter. Value: 0xf — Name: Default **[Default]** Value: [0x0-0x8] — Name:
	15:12	**Increment Event for EU event 1** This field controls which increment event provides the basis for flexible EU event 1. Value: 0xf — Name: Default **[Default]** Value: [0x0-0x8] — Name:
	11:8	**Fine Event Filter Select EU event 0** This field controls which fine event filter is applied to the coarsely filtered increment event when creating flexible EU event 0. Note that the fine event filter is logically applied after the coarse

EU_PERF_CNT_CTL0 - Flexible EU Event Control 0

event filter.

Value	Name
0xf	Default **[Default]**
[0x0-0xA]	

7:4 **Coarse Event Filter Select EU event 0**

This field controls which coarse event filter is applied to the selected increment event when creating flexible EU event 0. Note that the coarse event filter is logically applied before the fine event filter.

Value	Name
0xf	Default **[Default]**
[0x0-0x8]	

3:0 **Increment Event for EU event 0**

This field controls which increment event provides the basis for flexible EU event 0.

Value	Name
0xf	Default **[Default]**
[0x0-0x8]	

Flexible EU Event Control 1

EU_PERF_CNT_CTL1 - Flexible EU Event Control 1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0E558h

This register configures flexible EU event 2/3. Please refer to the description of the flexible EU events for more details on supported events. Please note that this register is render context saved/restored.

DWord	Bit	Description
0	31:26	**Reserved** Access: RO Format: MBZ
	25:24	**Reserved** Access: RO Format: MBZ
	23:20	**Fine Event Filter Select EU event 3** This field controls which fine event filter is applied to the coarsely filtered increment event when creating flexible EU event 3. Note that the fine event filter is logically applied after the coarse event filter. Value: 0xf — Name: Default **[Default]** Value: [0x0-0xA] — Name:
	19:16	**Coarse Event Filter Select EU event 3** This field controls which coarse event filter is applied to the selected increment event when creating flexible EU event 3. Note that the coarse event filter is logically applied before the fine event filter. Value: 0xf — Name: Default **[Default]** Value: [0x0-0x8] — Name:
	15:12	**Increment Event for EU event 3** This field controls which increment event provides the basis for flexible EU event 3. Value: 0xf — Name: Default **[Default]** Value: [0x0-0x8] — Name:
	11:8	**Fine Event Filter Select EU event 2** This field controls which fine event filter is applied to the coarsely filtered increment event when creating flexible EU event 2. Note that the fine event filter is logically applied after the coarse

EU_PERF_CNT_CTL1 - Flexible EU Event Control 1

event filter.

Value	Name
0xf	Default **[Default]**
[0x0-0xA]	

7:4 **Coarse Event Filter Select EU event 2**

This field controls which coarse event filter is applied to the selected increment event when creating flexible EU event 2. Note that the coarse event filter is logically applied before the fine event filter.

Value	Name
0xf	Default **[Default]**
[0x0-0x8]	

3:0 **Increment Event for EU event 2**

This field controls which increment event provides the basis for flexible EU event 2.

Value	Name
0xf	Default **[Default]**
[0x0-0x8]	

Flexible EU Event Control 2

EU_PERF_CNT_CTL2 - Flexible EU Event Control 2

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0E658h

This register configures flexible EU event 4/5. Please refer to the description of the flexible EU events for more details on supported events. Please note that this register is render context saved/restored.

DWord	Bit	Description
0	31:26	**Reserved** Access: RO Format: MBZ
	25:24	**Reserved** Access: RO Format: MBZ
	23:20	**Fine Event Filter Select EU event 5** This field controls which fine event filter is applied to the coarsely filtered increment event when creating flexible EU event 5. Note that the fine event filter is logically applied after the coarse event filter. Value: 0xf — Name: Default **[Default]** Value: [0x0-0xA]
	19:16	**Coarse Event Filter Select EU event 5** This field controls which coarse event filter is applied to the selected increment event when creating flexible EU event 5. Note that the coarse event filter is logically applied before the fine event filter. Value: 0xf — Name: Default **[Default]** Value: [0x0-0x8]
	15:12	**Increment Event for EU event 5** This field controls which increment event provides the basis for flexible EU event 5. Value: 0xf — Name: Default **[Default]** Value: [0x0-0x8]
	11:8	**Fine Event Filter Select EU event 4** This field controls which fine event filter is applied to the coarsely filtered increment event when creating flexible EU event 4. Note that the fine event filter is logically applied after the coarse

EU_PERF_CNT_CTL2 - Flexible EU Event Control 2

event filter.

Value	Name
0xf	Default **[Default]**
[0x0-0xA]	

7:4 Coarse Event Filter Select EU event 4

This field controls which coarse event filter is applied to the selected increment event when creating flexible EU event 4. Note that the coarse event filter is logically applied before the fine event filter.

Value	Name
0xf	Default **[Default]**
[0x0-0x8]	

3:0 Increment Event for EU event 4

This field controls which increment event provides the basis for flexible EU event 4.

Value	Name
0xf	Default **[Default]**
[0x0-0x8]	

Flexible EU Event Control 3

EU_PERF_CNT_CTL3 - Flexible EU Event Control 3

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0E758h

This register configures flexible EU event 6/7. Please refer to the description of the flexible EU events for more details on supported events. Please note that this register is render context saved/restored.

DWord	Bit	Description
0	31:26	**Reserved** Access: RO Format: MBZ
	25:24	**Reserved** Access: RO Format: MBZ
	23:20	**Fine Event Filter Select EU event 7** This field controls which fine event filter is applied to the coarsely filtered increment event when creating flexible EU event 7. Note that the fine event filter is logically applied after the coarse event filter. Value: 0xf — Name: Default **[Default]** Value: [0x0-0xA] — Name:
	19:16	**Coarse Event Filter Select EU event 7** This field controls which coarse event filter is applied to the selected increment event when creating flexible EU event 7. Note that the coarse event filter is logically applied before the fine event filter. Value: 0xf — Name: Default **[Default]** Value: [0x0-0x8] — Name:
	15:12	**Increment Event for EU event 7** This field controls which increment event provides the basis for flexible EU event 7. Value: 0xf — Name: Default **[Default]** Value: [0x0-0x8] — Name:
	11:8	**Fine Event Filter Select EU event 6** This field controls which fine event filter is applied to the coarsely filtered increment event when creating flexible EU event 6. Note that the fine event filter is logically applied after the coarse

EU_PERF_CNT_CTL3 - Flexible EU Event Control 3

event filter.

Value	Name
0xf	Default **[Default]**
[0x0-0xA]	

7:4 **Coarse Event Filter Select EU event 6**

This field controls which coarse event filter is applied to the selected increment event when creating flexible EU event 6. Note that the coarse event filter is logically applied before the fine event filter.

Value	Name
0xf	Default **[Default]**
[0x0-0x8]	

3:0 **Increment Event for EU event 6**

This field controls which increment event provides the basis for flexible EU event 6.

Value	Name
0xf	Default **[Default]**
[0x0-0x8]	

Flexible EU Event Control 4

EU_PERF_CNT_CTL4 - Flexible EU Event Control 4

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0E45Ch

This register configures flexible EU event 8/9. Please refer to the description of the flexible EU events for more details on supported events. Please note that this register is render context saved/restored.

DWord	Bit	Description
0	31:26	**Reserved** Access: RO Format: MBZ
	25:24	**Reserved** Access: RO Format: MBZ
	23:20	**Fine Event Filter Select EU event 9** This field controls which fine event filter is applied to the coarsely filtered increment event when creating flexible EU event 9. Note that the fine event filter is logically applied after the coarse event filter. Value: 0xf — Name: Default **[Default]** Value: [0x0-0xA] — Name:
	19:16	**Coarse Event Filter Select EU event 9** This field controls which coarse event filter is applied to the selected increment event when creating flexible EU event 9. Note that the coarse event filter is logically applied before the fine event filter. Value: 0xf — Name: Default **[Default]** Value: [0x0-0x8] — Name:
	15:12	**Increment Event for EU event 9** This field controls which increment event provides the basis for flexible EU event 9. Value: 0xf — Name: Default **[Default]** Value: [0x0-0x8] — Name:
	11:8	**Fine Event Filter Select EU event 8** This field controls which fine event filter is applied to the coarsely filtered increment event when creating flexible EU event 8. Note that the fine event filter is logically applied after the coarse

EU_PERF_CNT_CTL4 - Flexible EU Event Control 4

event filter.

Value	Name
0xf	Default **[Default]**
[0x0-0xA]	

7:4 **Coarse Event Filter Select EU event 8**

This field controls which coarse event filter is applied to the selected increment event when creating flexible EU event 8. Note that the coarse event filter is logically applied before the fine event filter.

Value	Name
0xf	Default **[Default]**
[0x0-0x8]	

3:0 **Increment Event for EU event 8**

This field controls which increment event provides the basis for flexible EU event 8.

Value	Name
0xf	Default **[Default]**
[0x0-0x8]	

Flexible EU Event Control 5

EU_PERF_CNT_CTL5 - Flexible EU Event Control 5

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0E55Ch

This register configures flexible EU event 10/11. Please refer to the description of the flexible EU events for more details on supported events. Please note that this register is render context saved/restored.

DWord	Bit	Description
0	31:26	**Reserved** Access: RO Format: MBZ
	25:24	**Reserved** Access: RO Format: MBZ
	23:20	**Fine Event Filter Select EU event 11** This field controls which fine event filter is applied to the coarsely filtered increment event when creating flexible EU event 11. Note that the fine event filter is logically applied after the coarse event filter. Value: 0xf — Name: Default **[Default]** Value: [0x0-0xA] — Name:
	19:16	**Coarse Event Filter Select EU event 11** This field controls which coarse event filter is applied to the selected increment event when creating flexible EU event 11. Note that the coarse event filter is logically applied before the fine event filter. Value: 0xf — Name: Default **[Default]** Value: [0x0-0x8] — Name:
	15:12	**Increment Event for EU event 11** This field controls which increment event provides the basis for flexible EU event 11. Value: 0xf — Name: Default **[Default]** Value: [0x0-0x8] — Name:
	11:8	**Fine Event Filter Select EU event 10** This field controls which fine event filter is applied to the coarsely filtered increment event when creating flexible EU event 10. Note that the fine event filter is logically applied after the coarse

EU_PERF_CNT_CTL5 - Flexible EU Event Control 5

event filter.

Value	Name
0xf	Default **[Default]**
[0x0-0xA]	

7:4 **Coarse Event Filter Select EU event 10**

This field controls which coarse event filter is applied to the selected increment event when creating flexible EU event 10. Note that the coarse event filter is logically applied before the fine event filter.

Value	Name
0xf	Default **[Default]**
[0x0-0x8]	

3:0 **Increment Event for EU event 10**

This field controls which increment event provides the basis for flexible EU event 10.

Value	Name
0xf	Default **[Default]**
[0x0-0x8]	

Flexible EU Event Control 6

EU_PERF_CNT_CTL6 - Flexible EU Event Control 6

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0E65Ch

This register configures flexible EU event 12/13. Please refer to the description of the flexible EU events for more details on supported events. Please note that this register is render context saved/restored.

DWord	Bit	Description
0	31:26	**Reserved** Access: RO Format: MBZ
	25:24	**Reserved** Access: RO Format: MBZ
	23:20	**Fine Event Filter Select EU event 13** This field controls which fine event filter is applied to the coarsely filtered increment event when creating flexible EU event 13. Note that the fine event filter is logically applied after the coarse event filter. Value: 0xf — Name: Default **[Default]** Value: [0x0-0xA] — Name:
	19:16	**Coarse Event Filter Select EU event 13** This field controls which coarse event filter is applied to the selected increment event when creating flexible EU event 13. Note that the coarse event filter is logically applied before the fine event filter. Value: 0xf — Name: Default **[Default]** Value: [0x0-0x8] — Name:
	15:12	**Increment Event for EU event 13** This field controls which increment event provides the basis for flexible EU event 13. Value: 0xf — Name: Default **[Default]** Value: [0x0-0x8] — Name:
	11:8	**Fine Event Filter Select EU event 12** This field controls which fine event filter is applied to the coarsely filtered increment event when creating flexible EU event 12. Note that the fine event filter is logically applied after the coarse

EU_PERF_CNT_CTL6 - Flexible EU Event Control 6

event filter.

Value	Name
0xf	Default **[Default]**
[0x0-0xA]	

7:4 **Coarse Event Filter Select EU event 12**

This field controls which coarse event filter is applied to the selected increment event when creating flexible EU event 12. Note that the coarse event filter is logically applied before the fine event filter.

Value	Name
0xf	Default **[Default]**
[0x0-0x8]	

3:0 **Increment Event for EU event 12**

This field controls which increment event provides the basis for flexible EU event 12.

Value	Name
0xf	Default **[Default]**
[0x0-0x8]	

FLT_RPT0

FLT_RPT0 - FLT_RPT0

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	124810h

GT uses this register to post VT-d faults

DWord	Bit	Description
0	31:12	**FI** Default Value: 00000h Access: R/W _Custom_GTIReset: BUS Fault Info.
	11:0	**Reserved** Access: RO Format: MBZ

FLT_RPT1

FLT_RPT1 - FLT_RPT1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
SOC_Consumer:	BIOS	
Address:	124814h	
GT uses this register to post VT-d faults		
DWord	**Bit**	**Description**
0	31:0	**FI** Default Value: 00000000h Access: R/W _Custom_GTIReset: BUS Fault Info.

FLT_RPT2

FLT_RPT2 - FLT_RPT2

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	124818h

GT uses this register to post VT-d faults

DWord	Bit	Description
0	31	**PP** Default Value: 0h Access: R/W _Custom_GTIReset: BUS PASID Present.
	30	**EXE** Default Value: 0h Access: R/W _Custom_GTIReset: BUS Execute Permission Requested.
	29	**PRIV** Default Value: 0h Access: R/W _Custom_GTIReset: BUS Privilege Mode Requested .
	28:16	**Reserved** Access: RO Format: MBZ
	15:0	**Source ID** Default Value: 0010h Access: RO _Custom_GTIReset: BUS Source ID.

FLT_RPT3

FLT_RPT3 - FLT_RPT3

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	12481Ch

GT uses this register to post VT-d faults

DWord	Bit	Description
0	31	**F** Default Value: 0h Access: R/W _Custom_GTIReset: BUS Fault
	30	**T** Default Value: 0h Access: R/W _Custom_GTIReset: BUS Type
	29:28	**AT** Default Value: 0h Access: R/W _Custom_GTIReset: BUS Address Type
	27:8	**PN** Default Value: 00000h Access: R/W _Custom_GTIReset: BUS PASID Number
	7:0	**FR** Default Value: 0h Access: R/W _Custom_GTIReset: BUS Fault Reason

FORCE_TO_NONPRIV

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	024D0h-024D3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_0_RCSUNIT
Address:	024D4h-024D7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_1_RCSUNIT
Address:	024D8h-024DBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_2_RCSUNIT
Address:	024DCh-024DFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_3_RCSUNIT
Address:	024E0h-024E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_4_RCSUNIT
Address:	024E4h-024E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_5_RCSUNIT
Address:	024E8h-024EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_6_RCSUNIT
Address:	024ECh-024EFh
Name:	FORCE_TO_NONPRIV
Address:	024F0h-024F3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_8_RCSUNIT
Address:	024F4h-024F7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_9_RCSUNIT

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
Address:	024F8h-024FBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_10_RCSUNIT
Address:	024FCh-024FFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_11_RCSUNIT
Address:	02010h-02013h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_12_RCSUNIT
Address:	02014h-02017h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_13_RCSUNIT
Address:	02018h-0201Bh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_14_RCSUNIT
Address:	0201Ch-0201Fh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_15_RCSUNIT
Address:	021E0h-021E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_16_RCSUNIT
Address:	021E4h-021E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_17_RCSUNIT
Address:	021E8h-021EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_18_RCSUNIT
Address:	021ECh-021EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_19_RCSUNIT
Address:	184D0h-184D3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_0_POCSUNIT
Address:	184D4h-184D7h
Name:	FORCE_TO_NONPRIV

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
ShortName:	FORCE_TO_NONPRIV_1_POCSUNIT
Address:	184D8h-184DBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_2_POCSUNIT
Address:	184DCh-184DFh
Name:	FORCE_TO_NONPRIV
Address:	184E0h-184E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_4_POCSUNIT
Address:	184E4h-184E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_5_POCSUNIT
Address:	184E8h-184EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_6_POCSUNIT
Address:	184ECh-184EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_7_POCSUNIT
Address:	184F0h-184F3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_8_POCSUNIT
Address:	184F4h-184F7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_9_POCSUNIT
Address:	184F8h-184FBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_10_POCSUNIT
Address:	184FCh-184FFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_11_POCSUNIT
Address:	18010h-18013h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_12_POCSUNIT
Address:	18014h-18017h
Name:	FORCE_TO_NONPRIV

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
ShortName:	FORCE_TO_NONPRIV_13_POCSUNIT
Address:	18018h-1801Bh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_14_POCSUNIT
Address:	1801Ch-1801Fh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_15_POCSUNIT
Address:	181E0h-181E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_16_POCSUNIT
Address:	181E4h-181E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_17_POCSUNIT
Address:	181E8h-181EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_18_POCSUNIT
Address:	181ECh-181EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_19_POCSUNIT
Address:	224D0h-224D3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_0_BCSUNIT
Address:	224D4h-224D7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_1_BCSUNIT
Address:	224D8h-224DBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_2_BCSUNIT
Address:	224DCh-224DFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_3_BCSUNIT
Address:	224E0h-224E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_4_BCSUNIT
Address:	224E4h-224E7h

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_5_BCSUNIT
Address:	224E8h-224EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_6_BCSUNIT
Address:	224ECh-224EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_7_BCSUNIT
Address:	224F0h-224F3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_8_BCSUNIT
Address:	224F4h-224F7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_9_BCSUNIT
Address:	224F8h-224FBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_10_BCSUNIT
Address:	224FCh-224FFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_11_BCSUNIT
Address:	22010h-22013h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_12_BCSUNIT
Address:	22014h-22017h
Name:	FORCE_TO_NONPRIV
Address:	22018h-2201Bh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_14_BCSUNIT
Address:	2201Ch-2201Fh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_15_BCSUNIT
Address:	221E0h-221E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_16_BCSUNIT
Address:	221E4h-221E7h

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_17_BCSUNIT
Address:	221E8h-221EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_18_BCSUNIT
Address:	221ECh-221EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_19_BCSUNIT
Address:	1C04D0h-1C04D3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_0_VCSUNIT0
Address:	1C04D4h-1C04D7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_1_VCSUNIT0
Address:	1C04D8h-1C04DBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_2_VCSUNIT0
Address:	1C04DCh-1C04DFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_3_VCSUNIT0
Address:	1C04E0h-1C04E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_4_VCSUNIT0
Address:	1C04E4h-1C04E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_5_VCSUNIT0
Address:	1C04E8h-1C04EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_6_VCSUNIT0
Address:	1C04ECh-1C04EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_7_VCSUNIT0
Address:	1C04F0h-1C04F3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_8_VCSUNIT0

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
Address:	1C04F4h-1C04F7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_9_VCSUNIT0
Address:	1C04F8h-1C04FBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_10_VCSUNIT0
Address:	1C04FCh-1C04FFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_11_VCSUNIT0
Address:	1C0010h-1C0013h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_12_VCSUNIT0
Address:	1C0014h-1C0017h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_13_VCSUNIT0
Address:	1C0018h-1C001Bh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_14_VCSUNIT0
Address:	1C001Ch-1C001Fh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_15_VCSUNIT0
Address:	1C01E0h-1C01E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_16_VCSUNIT0
Address:	1C01E4h-1C01E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_17_VCSUNIT0
Address:	1C01E8h-1C01EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_18_VCSUNIT0
Address:	1C01ECh-1C01EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_19_VCSUNIT0
Address:	1C44D0h-1C44D3h
Name:	FORCE_TO_NONPRIV

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
ShortName:	FORCE_TO_NONPRIV_0_VCSUNIT1
Address:	1C44D4h-1C44D7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_1_VCSUNIT1
Address:	1C44D8h-1C44DBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_2_VCSUNIT1
Address:	1C44DCh-1C44DFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_3_VCSUNIT1
Address:	1C44E0h-1C44E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_4_VCSUNIT1
Address:	1C44E4h-1C44E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_5_VCSUNIT1
Address:	1C44E8h-1C44EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_6_VCSUNIT1
Address:	1C44ECh-1C44EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_7_VCSUNIT1
Address:	1C44F0h-1C44F3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_8_VCSUNIT1
Address:	1C44F4h-1C44F7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_9_VCSUNIT1
Address:	1C44F8h-1C44FBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_10_VCSUNIT1
Address:	1C44FCh-1C44FFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_11_VCSUNIT1
Address:	1C4010h-1C4013h

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_12_VCSUNIT1
Address:	1C4014h-1C4017h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_13_VCSUNIT1
Address:	1C4018h-1C401Bh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_14_VCSUNIT1
Address:	1C401Ch-1C401Fh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_15_VCSUNIT1
Address:	1C41E0h-1C41E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_16_VCSUNIT1
Address:	1C41E4h-1C41E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_17_VCSUNIT1
Address:	1C41E8h-1C41EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_18_VCSUNIT1
Address:	1C41ECh-1C41EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_19_VCSUNIT1
Address:	1C84D0h-1C84D3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_0_VECSUNIT0
Address:	1C84D4h-1C84D7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_1_VECSUNIT0
Address:	1C84D8h-1C84DBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_2_VECSUNIT0
Address:	1C84DCh-1C84DFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_3_VECSUNIT0

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
Address:	1C84E0h-1C84E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_4_VECSUNIT0
Address:	1C84E4h-1C84E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_5_VECSUNIT0
Address:	1C84E8h-1C84EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_6_VECSUNIT0
Address:	1C84ECh-1C84EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_7_VECSUNIT0
Address:	1C84F0h-1C84F3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_8_VECSUNIT0
Address:	1C84F4h-1C84F7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_9_VECSUNIT0
Address:	1C84F8h-1C84FBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_10_VECSUNIT0
Address:	1C84FCh-1C84FFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_11_VECSUNIT0
Address:	1C8010h-1C8013h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_12_VECSUNIT0
Address:	1C8014h-1C8017h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_13_VECSUNIT0
Address:	1C8018h-1C801Bh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_14_VECSUNIT0
Address:	1C801Ch-1C801Fh
Name:	FORCE_TO_NONPRIV

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
ShortName:	FORCE_TO_NONPRIV_15_VECSUNIT0
Address:	1C81E0h-1C81E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_16_VECSUNIT0
Address:	1C81E4h-1C81E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_17_VECSUNIT0
Address:	1C81E8h-1C81EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_18_VECSUNIT0
Address:	1C81ECh-1C81EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_19_VECSUNIT0
Address:	1D04D0h-1D04D3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_0_VCSUNIT2
Address:	1D04D4h-1D04D7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_1_VCSUNIT2
Address:	1D04D8h-1D04DBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_2_VCSUNIT2
Address:	1D04DCh-1D04DFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_3_VCSUNIT2
Address:	1D04E0h-1D04E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_4_VCSUNIT2
Address:	1D04E4h-1D04E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_5_VCSUNIT2
Address:	1D04E8h-1D04EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_6_VCSUNIT2
Address:	1D04ECh-1D04EFh

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_7_VCSUNIT2
Address:	1D04F0h-1D04F3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_8_VCSUNIT2
Address:	1D04F4h-1D04F7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_9_VCSUNIT2
Address:	1D04F8h-1D04FBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_10_VCSUNIT2
Address:	1D04FCh-1D04FFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_11_VCSUNIT2
Address:	1D0010h-1D0013h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_12_VCSUNIT2
Address:	1D0014h-1D0017h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_13_VCSUNIT2
Address:	1D0018h-1D001Bh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_14_VCSUNIT2
Address:	1D001Ch-1D001Fh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_15_VCSUNIT2
Address:	1D01E0h-1D01E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_16_VCSUNIT2
Address:	1D01E4h-1D01E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_17_VCSUNIT2
Address:	1D01E8h-1D01EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_18_VCSUNIT2

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
Address:	1D01ECh-1D01EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_19_VCSUNIT2
Address:	1D44D0h-1D44D3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_0_VCSUNIT3
Address:	1D44D4h-1D44D7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_1_VCSUNIT3
Address:	1D44D8h-1D44DBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_2_VCSUNIT3
Address:	1D44DCh-1D44DFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_3_VCSUNIT3
Address:	1D44E0h-1D44E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_4_VCSUNIT3
Address:	1D44E4h-1D44E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_5_VCSUNIT3
Address:	1D44E8h-1D44EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_6_VCSUNIT3
Address:	1D44ECh-1D44EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_7_VCSUNIT3
Address:	1D44F0h-1D44F3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_8_VCSUNIT3
Address:	1D44F4h-1D44F7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_9_VCSUNIT3
Address:	1D44F8h-1D44FBh
Name:	FORCE_TO_NONPRIV

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
ShortName:	FORCE_TO_NONPRIV_10_VCSUNIT3
Address:	1D44FCh-1D44FFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_11_VCSUNIT3
Address:	1D4010h-1D4013h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_12_VCSUNIT3
Address:	1D4014h-1D4017h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_13_VCSUNIT3
Address:	1D4018h-1D401Bh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_14_VCSUNIT3
Address:	1D401Ch-1D401Fh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_15_VCSUNIT3
Address:	1D41E0h-1D41E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_16_VCSUNIT3
Address:	1D41E4h-1D41E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_17_VCSUNIT3
Address:	1D41E8h-1D41EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_18_VCSUNIT3
Address:	1D41ECh-1D41EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_19_VCSUNIT3
Address:	1D84D0h-1D84D3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_0_VECSUNIT1
Address:	1D84D4h-1D84D7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_1_VECSUNIT1
Address:	1D84D8h-1D84DBh

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_2_VECSUNIT1
Address:	1D84DCh-1D84DFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_3_VECSUNIT1
Address:	1D84E0h-1D84E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_4_VECSUNIT1
Address:	1D84E4h-1D84E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_5_VECSUNIT1
Address:	1D84E8h-1D84EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_6_VECSUNIT1
Address:	1D84ECh-1D84EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_7_VECSUNIT1
Address:	1D84F0h-1D84F3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_8_VECSUNIT1
Address:	1D84F4h-1D84F7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_9_VECSUNIT1
Address:	1D84F8h-1D84FBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_10_VECSUNIT1
Address:	1D84FCh-1D84FFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_11_VECSUNIT1
Address:	1D8010h-1D8013h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_12_VECSUNIT1
Address:	1D8014h-1D8017h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_13_VECSUNIT1

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
Address:	1D8018h-1D801Bh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_14_VECSUNIT1
Address:	1D801Ch-1D801Fh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_15_VECSUNIT1
Address:	1D81E0h-1D81E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_16_VECSUNIT1
Address:	1D81E4h-1D81E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_17_VECSUNIT1
Address:	1D81E8h-1D81EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_18_VECSUNIT1
Address:	1D81ECh-1D81EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_19_VECSUNIT1
Address:	1E04D0h-1E04D3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_0_VCSUNIT4
Address:	1E04D4h-1E04D7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_1_VCSUNIT4
Address:	1E04D8h-1E04DBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_2_VCSUNIT4
Address:	1E04DCh-1E04DFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_3_VCSUNIT4
Address:	1E04E0h-1E04E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_4_VCSUNIT4
Address:	1E04E4h-1E04E7h
Name:	FORCE_TO_NONPRIV

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
ShortName:	FORCE_TO_NONPRIV_5_VCSUNIT4
Address:	1E04E8h-1E04EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_6_VCSUNIT4
Address:	1E04ECh-1E04EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_7_VCSUNIT4
Address:	1E04F0h-1E04F3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_8_VCSUNIT4
Address:	1E04F4h-1E04F7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_9_VCSUNIT4
Address:	1E04F8h-1E04FBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_10_VCSUNIT4
Address:	1E04FCh-1E04FFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_11_VCSUNIT4
Address:	1E0010h-1E0013h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_12_VCSUNIT4
Address:	1E0014h-1E0017h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_13_VCSUNIT4
Address:	1E0018h-1E001Bh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_14_VCSUNIT4
Address:	1E001Ch-1E001Fh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_15_VCSUNIT4
Address:	1E01E0h-1E01E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_16_VCSUNIT4
Address:	1E01E4h-1E01E7h

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_17_VCSUNIT4
Address:	1E01E8h-1E01EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_18_VCSUNIT4
Address:	1E01ECh-1E01EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_19_VCSUNIT4
Address:	1E44D0h-1E44D3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_0_VCSUNIT5
Address:	1E44D4h-1E44D7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_1_VCSUNIT5
Address:	1E44D8h-1E44DBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_2_VCSUNIT5
Address:	1E44DCh-1E44DFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_3_VCSUNIT5
Address:	1E44E0h-1E44E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_4_VCSUNIT5
Address:	1E44E4h-1E44E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_5_VCSUNIT5
Address:	1E44E8h-1E44EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_6_VCSUNIT5
Address:	1E44ECh-1E44EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_7_VCSUNIT5
Address:	1E44F0h-1E44F3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_8_VCSUNIT5

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
Address:	1E44F4h-1E44F7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_9_VCSUNIT5
Address:	1E44F8h-1E44FBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_10_VCSUNIT5
Address:	1E44FCh-1E44FFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_11_VCSUNIT5
Address:	1E4010h-1E4013h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_12_VCSUNIT5
Address:	1E4014h-1E4017h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_13_VCSUNIT5
Address:	1E4018h-1E401Bh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_14_VCSUNIT5
Address:	1E401Ch-1E401Fh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_15_VCSUNIT5
Address:	1E41E0h-1E41E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_16_VCSUNIT5
Address:	1E41E4h-1E41E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_17_VCSUNIT5
Address:	1E41E8h-1E41EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_18_VCSUNIT5
Address:	1E41ECh-1E41EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_19_VCSUNIT5
Address:	1E84D0h-1E84D3h
Name:	FORCE_TO_NONPRIV

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
ShortName:	FORCE_TO_NONPRIV_0_VECSUNIT2
Address:	1E84D4h-1E84D7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_1_VECSUNIT2
Address:	1E84D8h-1E84DBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_2_VECSUNIT2
Address:	1E84DCh-1E84DFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_3_VECSUNIT2
Address:	1E84E0h-1E84E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_4_VECSUNIT2
Address:	1E84E4h-1E84E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_5_VECSUNIT2
Address:	1E84E8h-1E84EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_6_VECSUNIT2
Address:	1E84ECh-1E84EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_7_VECSUNIT2
Address:	1E84F0h-1E84F3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_8_VECSUNIT2
Address:	1E84F4h-1E84F7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_9_VECSUNIT2
Address:	1E84F8h-1E84FBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_10_VECSUNIT2
Address:	1E84FCh-1E84FFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_11_VECSUNIT2
Address:	1E8010h-1E8013h

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_12_VECSUNIT2
Address:	1E8014h-1E8017h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_13_VECSUNIT2
Address:	1E8018h-1E801Bh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_14_VECSUNIT2
Address:	1E801Ch-1E801Fh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_15_VECSUNIT2
Address:	1E81E0h-1E81E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_16_VECSUNIT2
Address:	1E81E4h-1E81E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_17_VECSUNIT2
Address:	1E81E8h-1E81EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_18_VECSUNIT2
Address:	1E81ECh-1E81EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_19_VECSUNIT2
Address:	1F04D0h-1F04D3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_0_VCSUNIT6
Address:	1F04D4h-1F04D7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_1_VCSUNIT6
Address:	1F04D8h-1F04DBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_2_VCSUNIT6
Address:	1F04DCh-1F04DFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_3_VCSUNIT6

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
Address:	1F04E0h-1F04E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_4_VCSUNIT6
Address:	1F04E4h-1F04E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_5_VCSUNIT6
Address:	1F04E8h-1F04EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_6_VCSUNIT6
Address:	1F04ECh-1F04EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_7_VCSUNIT6
Address:	1F04F0h-1F04F3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_8_VCSUNIT6
Address:	1F04F4h-1F04F7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_9_VCSUNIT6
Address:	1F04F8h-1F04FBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_10_VCSUNIT6
Address:	1F04FCh-1F04FFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_11_VCSUNIT6
Address:	1F0010h-1F0013h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_12_VCSUNIT6
Address:	1F0014h-1F0017h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_13_VCSUNIT6
Address:	1F0018h-1F001Bh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_14_VCSUNIT6
Address:	1F001Ch-1F001Fh
Name:	FORCE_TO_NONPRIV

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
ShortName:	FORCE_TO_NONPRIV_15_VCSUNIT6
Address:	1F01E0h-1F01E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_16_VCSUNIT6
Address:	1F01E4h-1F01E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_17_VCSUNIT6
Address:	1F01E8h-1F01EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_18_VCSUNIT6
Address:	1F01ECh-1F01EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_19_VCSUNIT6
Address:	1F44D0h-1F44D3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_0_VCSUNIT7
Address:	1F44D4h-1F44D7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_1_VCSUNIT7
Address:	1F44D8h-1F44DBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_2_VCSUNIT7
Address:	1F44DCh-1F44DFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_3_VCSUNIT7
Address:	1F44E0h-1F44E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_4_VCSUNIT7
Address:	1F44E4h-1F44E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_5_VCSUNIT7
Address:	1F44E8h-1F44EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_6_VCSUNIT7
Address:	1F44ECh-1F44EFh

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_7_VCSUNIT7
Address:	1F44F0h-1F44F3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_8_VCSUNIT7
Address:	1F44F4h-1F44F7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_9_VCSUNIT7
Address:	1F44F8h-1F44FBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_10_VCSUNIT7
Address:	1F44FCh-1F44FFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_11_VCSUNIT7
Address:	1F4010h-1F4013h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_12_VCSUNIT7
Address:	1F4014h-1F4017h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_13_VCSUNIT7
Address:	1F4018h-1F401Bh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_14_VCSUNIT7
Address:	1F401Ch-1F401Fh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_15_VCSUNIT7
Address:	1F41E0h-1F41E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_16_VCSUNIT7
Address:	1F41E4h-1F41E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_17_VCSUNIT7
Address:	1F41E8h-1F41EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_18_VCSUNIT7

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
Address:	1F41ECh-1F41EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_19_VCSUNIT7
Address:	1F84D0h-1F84D3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_0_VECSUNIT3
Address:	1F84D4h-1F84D7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_1_VECSUNIT3
Address:	1F84D8h-1F84DBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_2_VECSUNIT3
Address:	1F84DCh-1F84DFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_3_VECSUNIT3
Address:	1F84E0h-1F84E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_4_VECSUNIT3
Address:	1F84E4h-1F84E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_5_VECSUNIT3
Address:	1F84E8h-1F84EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_6_VECSUNIT3
Address:	1F84ECh-1F84EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_7_VECSUNIT3
Address:	1F84F0h-1F84F3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_8_VECSUNIT3
Address:	1F84F4h-1F84F7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_9_VECSUNIT3
Address:	1F84F8h-1F84FBh
Name:	FORCE_TO_NONPRIV

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
ShortName:	FORCE_TO_NONPRIV_10_VECSUNIT3
Address:	1F84FCh-1F84FFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_11_VECSUNIT3
Address:	1F8010h-1F8013h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_12_VECSUNIT3
Address:	1F8014h-1F8017h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_13_VECSUNIT3
Address:	1F8018h-1F801Bh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_14_VECSUNIT3
Address:	1F801Ch-1F801Fh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_15_VECSUNIT3
Address:	1F81E0h-1F81E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_16_VECSUNIT3
Address:	1F81E4h-1F81E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_17_VECSUNIT3
Address:	1F81E8h-1F81EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_18_VECSUNIT3
Address:	1F81ECh-1F81EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_19_VECSUNIT3
Address:	1A4D0h-1A4D3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_0_CCSUNIT0
Address:	1A4D4h-1A4D7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_1_CCSUNIT0
Address:	1A4D8h-1A4DBh

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV	
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_2_CCSUNIT0
Address:	1A4DCh-1A4DFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_3_CCSUNIT0
Address:	1A4E0h-1A4E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_4_CCSUNIT0
Address:	1A4E4h-1A4E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_5_CCSUNIT0
Address:	1A4E8h-1A4EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_6_CCSUNIT0
Address:	1A4ECh-1A4EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_7_CCSUNIT0
Address:	1A4F0h-1A4F3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_8_CCSUNIT0
Address:	1A4F4h-1A4F7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_9_CCSUNIT0
Address:	1A4F8h-1A4FBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_10_CCSUNIT0
Address:	1A4FCh-1A4FFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_11_CCSUNIT0
Address:	1A010h-1A013h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_12_CCSUNIT0
Address:	1A014h-1A017h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_13_CCSUNIT0

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV

Address:	1A018h-1A01Bh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_14_CCSUNIT0

Address:	1A01Ch-1A01Fh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_15_CCSUNIT0

Address:	1A1E0h-1A1E3h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_16_CCSUNIT0

Address:	1A1E4h-1A1E7h
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_17_CCSUNIT0

Address:	1A1E8h-1A1EBh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_18_CCSUNIT0

Address:	1A1ECh-1A1EFh
Name:	FORCE_TO_NONPRIV
ShortName:	FORCE_TO_NONPRIV_19_CCSUNIT0

These registers are privilege registers and are not allowed to be written from non-privilege batch buffer. These are global registers and power context save/restored.

Workaround

This register must be programmed as non-privileged to give PPGTT batch buffer access: 3DPRIM_XP0(0x2690), 3DPRIM_XP1(0x2694) and 3DPRIM_XP2(0x2698).

DWord	Bit	Description
0	31	**Virtual Function Privilege Control** Value / Name / Description 0 / **Default]** / MMIO offset programmed in Non Privileged Register Address field will be treated as a non-privilege register by the commands executed from a non-privilege batch buffer. This register provides programmability to add new entries to the User Mode Non-Privileged Registers table mentioned in User Mode Privileged Commands section. 1 / / MMIO offset programmed in Non Privileged Register Address field will be treated as a non-privilege register by the commands executed from a privilege batch buffer or ring buffer of an virtual function. Not that this will not make the register offset as non-privileged for the commands executed from non-privileged batch buffer.
	30	**Denylist Enable**

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV

If this bit is set, then the command stream will not allow access for a PPGTT batch buffer to this register or range. This takes the highest precedence so in the case the register is allow listed in a separate FORCE_TO_NONPRIV register, it will still be deny listed.

Value	Name
0	Allowlist **[Default]**
1	Denylist

29:28 **Access Selection**

This field allows the selection of whether read or write access are impacted. This allows for any permutation to deny list or allow list any combination of reads/writes. For Example, if Deny list Enable is set and the Access if for Reads only, then only Read access will be disallowed based on the Address in this register.

Value	Name
0h	Impacts both Read and Write Access **[Default]**
1h	Impacts only Read Access
2h	Impacts only Write Access
3h	Reserved

27:26 **Reserved**

Access:	RO
Format:	MBZ

25:2 **Non Privilege Register Address**

Format:	MmioAddress[25:2]

This field contains the MMIO offset of a register. MMIO offset programmed in this field will be treated as a non-privilege register by render command streamer while processing register writes from a non-privilege batch buffer. This register provides programmability is to extend the non-privilege register table mentioned in MI_BATCH_BUFFER_START command in render command streamer.

Value	Name
3800h	**[Default]**

1:0 **Offset Range**

This field specifies the range of registers to either be deny listed or allow listed. Note the Non-Privilege Register Address value should be at the same granularity of the Offset Range. The lower bits of the address will be ignored in the comparison as the definition only support base 2 address for range comparison.

Value	Name	Description
0h	Single Register **[Default]**	Offset for allowlist or denylist compare will be 25:2.
1h	Four Registers	Offset for allowlist or denylist compare will be 25:4.
2h	Sixteen Registers	Offset for allowlist or denylist compare will be 25:6.
3h	Sixty Four Registers	Offset for allowlist or denylist compare will be 25:8.

FUSE_STATUS

FUSE_STATUS

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	42000h-42003h
Name:	Fuse Status
ShortName:	FUSE_STATUS
Reset:	global

This register is on the ungated clock and the chip reset, not the FLR.

DWord	Bit	Description
0	31	**Fuse Download Status** Access: RO This field indicates the status of fuse and strap download to the Display Engine. After fuse and strap download, fuses will be distributed within the Display Engine. Value: 0b – Not Done Value: 1b – Done
	30:28	**Reserved** Access: RO Format: MBZ
	27	**Fuse PG0 Distribution Status** Access: RO This field indicates the status of fuse distribution to power well #0. Value: 0b – Not Done Value: 1b – Done
	26	**Fuse PG1 Distribution Status** Access: RO This field indicates the status of fuse distribution to power well #1. Value: 0b – Not Done Value: 1b – Done
	25	**Fuse PG2 Distribution Status** Access: RO This field indicates the status of fuse distribution to power well #2. Value / Name

FUSE_STATUS

	0b	Not Done
	1b	Done

24 Fuse PG3 Distribution Status

Access:	RO

This field indicates the status of fuse distribution to power well #3.

Value	Name
0b	Not Done
1b	Done

23 Fuse PG4 Distribution Status

Access:	RO

This field indicates the status of fuse distribution to power well #4.

Value	Name
0b	Not Done
1b	Done

22 Fuse PG5 Distribution Status

Access:	RO

This field indicates the status of fuse distribution to power well #5.

Value	Name
0b	Not Done
1b	Done

21:16 Reserved

Access:	RO
Format:	MBZ

15:0 Reserved

Access:	RO
Format:	MBZ

GAC_GAM Arbitration Counters Register 0

ARB_GAC_GAM_REQCNTS0 - GAC_GAM Arbitration Counters Register 0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	043A8h

DWord	Bit	Description
0	31:22	**Reserved** Access: RO Format: MBZ
	21:16	**Number of GAC WR requests to be accumulated before applying the arbitration**
	15:14	**Reserved** Access: RO Format: MBZ
	13:8	**Number of GAC R requests to be accumulated before applying the arbitration**
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**Number of GAC RO requests to be accumulated before applying the arbitration**

GAC_GAM Arbitration Counters Register 1

ARB_GAC_GAM_REQCNTS1 - GAC_GAM Arbitration Counters Register 1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	043ACh

DWord	Bit	Description
0	31:22	**Reserved** Access: RO Format: MBZ
	21:16	**Number of GAC WR requests to be accumulated before applying the arbitration**
	15:14	**Reserved** Access: RO Format: MBZ
	13:8	**Number of GAC R requests to be accumulated before applying the arbitration**
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**Number of GAC RO requests to be accumulated before applying the arbitration**

GAC_GAM RO Arbitration Register 0

ARB_RO_GAC_GAM0 - GAC_GAM RO Arbitration Register 0		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	043D0h	
DWord	**Bit**	**Description**
0	31:28	**Reserved** Access: RO Format: MBZ
	27	**Priority for entry 1**
	26:24	**Goto field for entry 1 when request vector is 11b**
	23:21	**Goto field for entry 1 when request vector is 10b**
	20:18	**Goto field for entry 1 when request vector is 01b**
	17:15	**Goto field for entry 1 when request vector is 00b**
	14:13	**Reserved** Access: RO Format: MBZ
	12	**Priority for entry 01**
	11:9	**Goto field for entry 01 when request vector is 11b**
	8:6	**Goto field for entry 01 when request vector is 10b**
	5:3	**Goto field for entry 01 when request vector is 01b**
	2:0	**Goto field for entry 01 when request vector is 00b**

GAC_GAM RO Arbitration Register 1

ARB_RO_GAC_GAM1 - GAC_GAM RO Arbitration Register 1		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	043D4h	

DWord	Bit	Description
0	31:28	**Reserved** Access: RO Format: MBZ
	27	**Priority for entry 3**
	26:24	**Goto field for entry 3 when request vector is 11b**
	23:21	**Goto field for entry 3 when request vector is 10b**
	20:18	**Goto field for entry 3 when request vector is 01b**
	17:15	**Goto field for entry 3 when request vector is 00b**
	14:13	**Reserved** Access: RO Format: MBZ
	12	**Priority for entry 2**
	11:9	**Goto field for entry 2 when request vector is 11b**
	8:6	**Goto field for entry 2 when request vector is 10b**
	5:3	**Goto field for entry 2 when request vector is 01b**
	2:0	**Goto field for entry 2 when request vector is 00b**

GAC_GAM RO Arbitration Register 2

ARB_RO_GAC_GAM2 - GAC_GAM RO Arbitration Register 2		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	043D8h	

DWord	Bit	Description
0	31:28	**Reserved** Access: RO Format: MBZ
	27	**Priority for entry 5**
	26:24	**Goto field for entry 5 when request vector is 11b**
	23:21	**Goto field for entry 5 when request vector is 10b**
	20:18	**Goto field for entry 5 when request vector is 01b**
	17:15	**Goto field for entry 5 when request vector is 00b**
	14:13	**Reserved** Access: RO Format: MBZ
	12	**Priority for entry 4**
	11:9	**Goto field for entry 4 when request vector is 11b**
	8:6	**Goto field for entry 4 when request vector is 10b**
	5:3	**Goto field for entry 4 when request vector is 01b**
	2:0	**Goto field for entry 4 when request vector is 00b**

GAC_GAM RO Arbitration Register 3

ARB_RO_GAC_GAM3 - GAC_GAM RO Arbitration Register 3		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	043DCh	

DWord	Bit	Description
0	31:28	**Reserved** Access: RO Format: MBZ
	27	**Priority for entry 7**
	26:24	**Goto field for entry 7 when request vector is 11b**
	23:21	**Goto field for entry 7 when request vector is 10b**
	20:18	**Goto field for entry 7 when request vector is 01b**
	17:15	**Goto field for entry 7 when request vector is 00b**
	14:13	**Reserved** Access: RO Format: MBZ
	12	**Priority for entry 6**
	11:9	**Goto field for entry 6 when request vector is 11b**
	8:6	**Goto field for entry 6 when request vector is 10b**
	5:3	**Goto field for entry 6 when request vector is 01b**
	2:0	**Goto field for entry 6 when request vector is 00b**

GAF Config Register 0

GAFCFG0 - GAF Config Register 0

Register Space:	MMIO: 0/2/0
Access:	R/W Lock
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0D800h

DWord	Bit	Description
0	31	**gafsrdunit reserved** Default Value: 0b
	30	**Extra 4 slots addition to stopping distance** 0b **[Default]** 1b
	29	**Multi Slice Fuse enable** 0b **[Default]** 1b
	28	**TSG Double min alloc** 0b **[Default]** 1b
	27	**SF Double min alloc** 0b **[Default]** 1b
	26	**CL2 Double min alloc** 0b **[Default]** 1b
	25	**CL1 Double min alloc** 0b **[Default]** 1b

GAFCFG0 - GAF Config Register 0

24 **CL Double min alloc**

Value	Name
0b	**[Default]**
1b	

23 **SOL Double min alloc**

Value	Name
0b	**[Default]**
1b	

22 **GS Double min alloc**

Value	Name
0b	**[Default]**
1b	

21 **TE Double min alloc**

Value	Name
0b	**[Default]**
1b	

20:18 **CL max alloc programming**

Value	Name	Description
000b		Default(64)
001b		56
010b	**[Default]**	48
011b		40
100b		32
101b		24
110b		16
111b		Reserved

17:15 **SOL max alloc programming**

Value	Name	Description
000b		Default(64)
001b		56
010b	**[Default]**	48
011b		40
100b		32
101b		24
110b		16
111b		Reserved

GAFCFG0 - GAF Config Register 0

14:12 **GS max alloc programming**

Value	Name	Description
000b		Default(64)
001b		56
010b	**[Default]**	48
011b		40
100b		32
101b		24
110b		16
111b		Reserved

11:9 **TE max alloc programming**

Value	Name	Description
000b		Default(64)
001b		56
010b	**[Default]**	48
011b		40
100b		32
101b		24
110b		16
111b		Reserved

8:7 **TSG Low priority threshold programming**

Value	Name	Description
00b	**[Default]**	Default Low Priority Threshold
01b		Max Alloc/4
10b		Max Alloc/2
11b		Max Alloc

6:5 **SOL Low priority threshold programming**

Value	Name	Description
00b	**[Default]**	Default Low Priority Threshold
01b		Max Alloc/4
10b		Max Alloc/2
11b		Max Alloc

4:3 **GS Low priority threshold programming**

Value	Name	Description
00b	**[Default]**	Default Low Priority Threshold
01b		Max Alloc/4

GAFCFG0 - GAF Config Register 0

	10b	Max Alloc/2
	11b	Max Alloc

2:1 TE Low priority threshold programming

Value	Name	Description
00b	**[Default]**	Default Low Priority Threshold
01b		Max Alloc/4
10b		Max Alloc/2
11b		Max Alloc

0 GAFS data return policy

Value	Name
0b	**[Default]**
1b	

GAF Config Register 1

GAFCFG1 - GAF Config Register 1

Register Space: MMIO: 0/2/0
Access: R/W Lock
Size (in bits): 32
_Custom_GTIReset: DEV

Address: 0D804h

DWord	Bit	Description
0	31:26	**gafarbunit reserved** Value: 000000b **[Default]**; [0,63]
	25:13	**gafswrunit reserved** Value: 0000h **[Default]**; [0,8191]
	12:11	**sarbunit reserved** Value: 0h **[Default]**; [0,3]
	10	**sarb overfetch cache disable** Value: 0b **[Default]**; 1b
	9:7	**gafdunit reserved** Value: 000b **[Default]**; [0,7]
	6	**ebb signal bypass control** Used to control the bypass ebb signal in gafdunit Value: 0b **[Default]**; 1b
	5:2	**gafmunit reserved** Value: 0000b **[Default]**

GAFCFG1 - GAF Config Register 1

		[0,15]
	1	**POSH nonPOSH priority** Value: 0b — **[Default]** Value: 1b
	0	**VF SOL priority** Value: 0b — **[Default]** Value: 1b

GAMMA_MODE

GAMMA_MODE

Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank

Address:	4A480h-4A483h
Name:	Pipe Gamma Mode
ShortName:	GAMMA_MODE_A
Reset:	soft

Address:	4AC80h-4AC83h
Name:	Pipe Gamma Mode
ShortName:	GAMMA_MODE_B
Reset:	soft

Address:	4B480h-4B483h
Name:	Pipe Gamma Mode
ShortName:	GAMMA_MODE_C
Reset:	soft

Address:	4BC80h-4BC83h
Name:	Pipe Gamma Mode
ShortName:	GAMMA_MODE_D
Reset:	soft

DWord	Bit	Description
0	31	**Pre CSC Gamma Enable** This bit enables the pipe pre color space conversion gamma. Restriction : This bit must not be set when any of the individual plane 'Pipe CSC Enable' bit is set in PLANE_COLOR_CTL register. Value / Name: 1b – Enable 0b – Disable
	30	**Post CSC Gamma Enable** This bit enables the pipe post color space conversion gamma. Restriction : This bit must not be set when any of the individual plane 'Pipe Gamma Enable' bit is set in PLANE_COLOR_CTL register.

GAMMA_MODE

Value	Name
1b	Enable
0b	Disable

29	**Allow Double Buffer Update Disable**
	Access: R/W
	This field controls whether double buffer updates are allowed to be disabled for the Gamma registers that are double buffered. The DOUBLE_BUFFER_CTL register can be configured to globally disable double buffer updates for those resources that allow them to be disabled.

Value	Name
0b	Not Allowed
1b	Allowed **[Default]**

28:25	**Reserved**
	Access: RO
	Format: MBZ
24:16	**Reserved**
	Access: RO
	Format: MBZ
15	**Reserved**
14:5	**Reserved**
	Access: RO
	Format: MBZ
4:3	**Reserved**
	Access: RO
	Format: MBZ
2	**Reserved**
	Access: RO
	Format: MBZ
1:0	**Gamma Mode**
	This field selects which mode the pipe palette/gamma correction logic works in. Other gamma units, such as in the planes, are unaffected by these bits.
	This field applies to post csc gamma. Pre csc gamma mode is fixed and not configurable.

Value	Name	Description
00b	8 bit	8-bit Legacy Palette Mode
01b	10 bit	10-bit Precision Palette Mode
10b	12 bit	12-bit Interpolated Gamma Mode
11b	12 bit Multi Segment	12-bit Multi-segmented Gamma Mode

Gang Timer Register

FF_MODE2 - Gang Timer Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	06604h-06607h
Name:	Ganged Timer Register
ShortName:	FF_MODE2_SVGUNIT
Address:	17604h-17607h
Name:	Ganged Timer Register
ShortName:	FF_MODE2_SVGRUNIT

This register is used to program the FF shader timers.

DWord	Bit	Description
0	31:24	**GS timer value** Format: U8 The GS timer value will be a multiple of 32 clocks. The GS timer value of 0 disables the GS timer. If the number of clocks between ganged threads reaches the GS timer value, the GS will dispatch however many threads it currently has ganged.
	23:16	**TDS timer value** Format: U8 The TDS timer value will be a multiple of 32 clocks. The TDS timer value of 0 disables the TDS timer. If the number of clocks between ganged threads reaches the TDS timer value, the TDS will dispatch however many threads it currently has ganged. **Programming Notes** For best performance, a value of 4 should be programmed.
	15:8	**HS timer value** Format: U8 The HS timer value will be a multiple of 32 clocks. The HS timer value of 0 disables the HS timer. If the number of clocks between ganged threads reaches the HS timer value, the HS will dispatch however many threads it currently has ganged.

FF_MODE2 - Gang Timer Register		
	7:0	**VS timer value** Format: U8 The VS timer value will be a multiple of 32 clocks. The VS timer value of 0 disables the VS timer. If the number of clocks between ganged threads reaches the VS timer value, the VS will dispatch however many threads it currently has ganged.

Gated Clock Counter for DFR Testability

SAMPLER_DFR_GATED_COUNT - Gated Clock Counter for DFR Testability

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0E14Ch

For testability of DFR feature

DWord	Bit	Description
0	31:0	**Counter Bits** Format: U32 Count of edge-skipped sampler clocks

General Purpose Register

CS_GPR - General Purpose Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	1024
_Custom_GTIReset:	DEV
Address:	02600h-0267Fh
Name:	General Purpose Register
ShortName:	CS_GPR_RCSUNIT
Address:	18600h-1867Fh
Name:	General Purpose Register
ShortName:	CS_GPR_POCSUNIT
Address:	22600h-2267Fh
Name:	General Purpose Register
ShortName:	CS_GPR_BCSUNIT
Address:	1C0600h-1C067Fh
Name:	General Purpose Register
ShortName:	CS_GPR_VCSUNIT0
Address:	1C4600h-1C467Fh
Name:	General Purpose Register
ShortName:	CS_GPR_VCSUNIT1
Address:	1C8600h-1C867Fh
Name:	General Purpose Register
ShortName:	CS_GPR_VECSUNIT0
Address:	1D0600h-1D067Fh
Name:	General Purpose Register
ShortName:	CS_GPR_VCSUNIT2
Address:	1D4600h-1D467Fh
Name:	General Purpose Register
ShortName:	CS_GPR_VCSUNIT3
Address:	1D8600h-1D867Fh
Name:	General Purpose Register
ShortName:	CS_GPR_VECSUNIT1
Address:	1E0600h-1E067Fh
Name:	General Purpose Register

CS_GPR - General Purpose Register

ShortName:	CS_GPR_VCSUNIT4
Address:	1E4600h-1E467Fh
Name:	General Purpose Register
ShortName:	CS_GPR_VCSUNIT5
Address:	1E8600h-1E867Fh
Name:	General Purpose Register
ShortName:	CS_GPR_VECSUNIT2
Address:	1F0600h-1F067Fh
Name:	General Purpose Register
ShortName:	CS_GPR_VCSUNIT6
Address:	1F4600h-1F467Fh
Name:	General Purpose Register
ShortName:	CS_GPR_VCSUNIT7
Address:	1F8600h-1F867Fh
Name:	General Purpose Register
ShortName:	CS_GPR_VECSUNIT3
Address:	1A600h-1A67Fh
Name:	General Purpose Register
ShortName:	CS_GPR_CCSUNIT0

This is a General Purpose Register bank of sixteen 64bit registers, which will be used as temporary storage by MI_MATH command to do ALU operations.

GPR Index	MMIO Offset
R_0	0x2600
R_1	0x2608
R_2	0x2610
R_3	0x2618
R_4	0x2620
R_5	0x2628
R_6	0x2630
R_7	0x2638
R_8	0x2640
R_9	0x2648
R_10	0x2650
R_11	0x2658
R_12	0x2660

CS_GPR - General Purpose Register

R_13	0x2668
R_14	0x2670
R_15	0x2678

DWord	Bit	Description
0..1	63:32	**CS_GPR_DATA1** Source: CommandStreamer
	31:0	**CS_GPR_DATA0** Source: CommandStreamer
2..3	63:32	**CS_GPR_DATA3** Source: CommandStreamer
	31:0	**CS_GPR_DATA2** Source: CommandStreamer
4..5	63:32	**CS_GPR_DATA5** Source: CommandStreamer
	31:0	**CS_GPR_DATA4** Source: CommandStreamer
6..7	63:32	**CS_GPR_DATA7** Source: CommandStreamer
	31:0	**CS_GPR_DATA6** Source: CommandStreamer
8..9	63:32	**CS_GPR_DATA9** Source: CommandStreamer
	31:0	**CS_GPR_DATA8** Source: CommandStreamer
10..11	63:32	**CS_GPR_DATA11** Source: CommandStreamer
	31:0	**CS_GPR_DATA10** Source: CommandStreamer
12..13	63:32	**CS_GPR_DATA13** Source: CommandStreamer
	31:0	**CS_GPR_DATA12** Source: CommandStreamer
14..15	63:32	**CS_GPR_DATA15** Source: CommandStreamer
	31:0	**CS_GPR_DATA14**

CS_GPR - General Purpose Register		
		Source: CommandStreamer
16..17	63:32	**CS_GPR_DATA17** Source: CommandStreamer
	31:0	**CS_GPR_DATA16** Source: CommandStreamer
18..19	63:32	**CS_GPR_DATA19** Source: CommandStreamer
	31:0	**CS_GPR_DATA18** Source: CommandStreamer
20..21	63:32	**CS_GPR_DATA21** Source: CommandStreamer
	31:0	**CS_GPR_DATA20** Source: CommandStreamer
22..23	63:32	**CS_GPR_DATA23** Source: CommandStreamer
	31:0	**CS_GPR_DATA22** Source: CommandStreamer
24..25	63:32	**CS_GPR_DATA25** Source: CommandStreamer
	31:0	**CS_GPR_DATA24** Source: CommandStreamer
26..27	63:32	**CS_GPR_DATA27** Source: CommandStreamer
	31:0	**CS_GPR_DATA26** Source: CommandStreamer
28..29	63:32	**CS_GPR_DATA29** Source: CommandStreamer
	31:0	**CS_GPR_DATA28** Source: CommandStreamer
30..31	63:32	**CS_GPR_DATA31** Source: CommandStreamer
	31:0	**CS_GPR_DATA30** Source: CommandStreamer

GFX_FLSH_CNT

GFX_FLSH_CNT - GFX_FLSH_CNT

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	101008h

Used to flush Gunit TLB

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**GfxFlshCntl** Default Value: 0b Access: WO _Custom_GTIReset: BUS Access type of this register is WO. A write to this bit flushes the Gfx TLB in GUNIT. The data associated with the write is discarded and a read return all 0s.

GFX_VTDBAR_LSB

GFXVTDBAR_LSB_0_2_0_PCI - GFX_VTDBAR_LSB

Register Space:	PCI: 0/2/0
Size (in bits):	32
Address:	000C8h

This is the base address for the Graphics VTD configuration space. There is no physical memory within this 4KB window that can be addressed. The 4KB reserved by this register does not alias to any PCI 2.3 compliant memory mapped space. On reset, the GFX-VTD configuration space is disabled and must be enabled by writing a 1 to GFXVTDBAREN.
None of the bits in this register are writeable in Intel TXT mode. This enforcement is based on SAI policy registers.
BIOS programs this register, after which the register cannot be altered.

DWord	Bit	Description
0	31:12	**GFXVTDBAR** Default Value: 000000h Access: R/W _Custom_GTIReset: BUS This field corresponds to bits 31 to 12 of the base address GFX-VTD configuration space. BIOS will program this register resulting in a base address for a 4KB block of contiguous memory address space. This register ensures that a naturally aligned 4KB space is allocated within the first 512GB of addressable memory space. System Software uses this base address to program the GFX-VTD register set. All the Bits in this register are included in an SAI policy group for Intel TXT mode purposes.
	11:2	**Reserved** Access: RO Format: MBZ
	1	**SPARE** Default Value: 000h Access: R/W _Custom_GTIReset: BUS This was a lock bit prior.
	0	**GFXVTDBAREN** Default Value: 0b Access: R/W _Custom_GTIReset: BUS 0: GFX-VTBAR is disabled and does not claim any memory. 1: GFX-VTBAR memory mapped accesses are claimed and decoded appropriately This bit will remain 0 if VTd capability is disabled.

GFXBDF

GFXBDF - GFXBDF

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	10107Ch

If the associated hardware strap is enabled, hardware will update this register based on the received IOSF P Bus / Device value from Type 0 configuration cycles (per PCIe spec). Also hardware will be updating this register, Punit needs the ability to save/restore the register contents for relevant PKGC state flows and/or S0i3 flows. Therefore, this register will be a RW-able by software and write-able by hardware.

DWord	Bit	Description
0	31:24	**BUS** Default Value: 00000000b Access: R/W _Custom_GTIReset: BUS This field specifies the PCI bus number of theGfx device.
	23:19	**DEVICE** Access: R/W _Custom_GTIReset: BUS This field specifies the PCI device number of theGfx device. Value: 00010b — Name: **[Default]**
	18:16	**FUNCTION** Default Value: 000b Access: RO _Custom_GTIReset: BUS This field specifies the PCI function number of theGfx device.
	15:1	**Reserved** Access: RO Format: MBZ
	0	**HWUPDATEDISABLE** Default Value: 0b Access: R/W _Custom_GTIReset: BUS 0 : Hardware updates Bus/Device value based on received Type 0 configuration cycles. 1 : Disable hardware updates based on received hardware cycles (even if strapped to allow updates).

GGTT Pinned Range Base Register

GGTT_PINNED_BASE - GGTT Pinned Range Base Register

Register Space: MMIO: 0/2/0

Size (in bits): 32

GGTT Pinned Range Base register

DWord	Bit	Description
0	31:20	**Reserved** Access: RO Format: MBZ
	19:0	**Pinned Range Base** Default Value: 00000h Access: R/W _Custom_GTIReset: DEV GGTT Entry that represents the base of the Pinned Range. This corresponds to bits 31:12 of the Global Virtual address of the memory page that corresponds to that entry. This register is a shadow of the primary register in Gunit, and is updated automatically by Gunit HW via IOSF-SB.

GGTT Pinned Range Limit Register

GGTT_PINNED_LIMIT - GGTT Pinned Range Limit Register

Register Space: MMIO: 0/2/0

Size (in bits): 32

GGTT Pinned Range Limit register

DWord	Bit	Description
0	31:20	**Reserved** Access: RO Format: MBZ
	19:0	**Pinned Range Limit** Default Value: 00000h Access: R/W _Custom_GTIReset: DEV GGTT Entry that represents the limit(last entry) of the Pinned Range. This corresponds to bits 31:12 of the Global Virtual address of the memory page that corresponds to that entry. This register is a shadow of the primary register in Gunit, and is updated automatically by Gunit HW via IOSF-SB.

Global MicroController Status

GUC_STATUS - Global MicroController Status

Register Space: MMIO: 0/2/0
Access: RO
Size (in bits): 32

Programming Notes
This register is saved in the power context.
Only write by MinuteIA is allowed to this register.
Host writes to this register have no effect (they are dropped).

DWord	Bit	Description
0	31:30	**Authentication Status** BootROM code shall write a 1 to bit 31, if the uOS was authenticated successfully. BootROM code shall write a 1 to bit 30, if the uOS authentication failed. **Value / Name / Description** 1h / Fail / The uOS authentication failed. 2h / Success / The uOS was authenticated successfully. **Programming Notes** Set Once Bits (Once written, they cannot be overwritten) If bit 30 is set, DMA HW will skip trying to load a uOS into the SRAM thereby preventing the MinuteIA from execution.
	29:24	**uApp Status** This field is software-defined.
	23:16	**Reserved**
	15:8	**uKernel/uOS Status** This field is software-defined.
	7:1	**Boot ROM Code Status** This field is software-defined.
	0	**MinIA Is In Reset** '1' indicates that MinIA is in reset.

Global System Interrupt Routine

EU_GLOBAL_SIP - Global System Interrupt Routine		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
_Custom_GTIReset:	DEV	
Address:	0E42Ch	

DWord	Bit	Description
0	31:3	**Global SIP** Format: GraphicsAddress[31:3] Specifies the base address for System Interrupt Routine that over-rides the SIP set by the state (STATE_SIP).
	2:1	**Reserved** Format: PBC
	0	**Global SIP Enable** The bit specifies if the System Routine starts from the Global SIP provided by the DW OR the SIP provided by the state (STATE_SIP) Value: 0 — SIP used is from STATE_EIP Value: 1 — SIP used is from MMIO register

GMCH Graphics Control

GGC - GMCH Graphics Control

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	108040h

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:8	**GMS** Default Value: 05h Access: R/W _Custom_GTIReset: BUS This field is used to select the amount of Main Memory that is pre-allocated to support the Internal Graphics device in VGA (non-linear) and Native (linear) modes. It corresponds to DSM (Data Stolen Memory region) region. The BIOS ensures that memory is pre-allocated only when Internal graphics is enabled. Hardware does not clear or set any of these bits automatically based on IGD being disabled/enabled. BIOS Requirement: BIOS must not set this field to 0h if IVD (bit 1 of this register) is 0. BIOS Requirement: Given new sizes allow down to 8MB allocation, BIOS has to ensure there is sufficient space for WOPCM and basic GFX Stolen functions. 00h:0MB 01h:32MB 02h:64MB 03h:96MB 04h:128MB 05h:160MB 06h:192MB 07h:224MB 08h:256MB 09h:288MB 0Ah:320MB 0Bh:352MB 0Ch:384MB 0Dh:416MB 0Eh:448MB 0Fh:480MB 10h:512MB 11h - 1Fh: Reserved 20h:1024MB

21h - 2Fh: Reserved
30h:1536MB
31h - 3Fh: Reserved
40h: 2048MB
41h - EFh: Reserved
F0h: 4MB
F1h: 8MB
F2h: 12MB
F3h: 16MB
F4h: 20MB
F5h: 24MB
F6h: 28MB
F7h: 32MB
F8h: 36MB
F9h: 40MB
FAh: 44MB
FBh: 48MB
FCh: 52MB
FDh: 56MB
FEh: 60MB
FFh: Reserved Hardware functionality in case of programming this value to Reserved is not guaranteed.

7:6 **GGMS**

Access:	R/W
_Custom_GTIReset:	BUS

This field is used to select the amount of Main Memory that is pre-allocated to support the Internal Graphics Translation Table. The BIOS ensures that memory is pre-allocated only when Internal graphics is enabled.
Hardware functionality in case of programming this value to Reserved is not guaranteed.
0x0:No Preallocated Memory
0x1:2MB of Preallocated Memory
0x2:4MB of Preallocated Memory
0x3:8MB of Preallocated Memory

Value	Name
00b	**[Default]**

5:3 **Reserved**

Access:	RO
Format:	MBZ

2 **VAMEN**

Default Value:	0b
Access:	R/W
_Custom_GTIReset:	BUS

GGC - GMCH Graphics Control

	1	**IVD** Default Value: 0b Access: R/W _Custom_GTIReset: BUS 0: Enable. Device 2 (IGD) claims VGA memory and IO cycles, the Sub-Class Code within Device 2 Class Code register is 00. 1: Disable. Device 2 (IGD) does not claim VGA cycles (Mem and IO), and the Sub- Class Code field within Device 2 function 0 Class Code register is 80. BIOS Requirement: BIOS must not set this bit to 0 if the GMS field (bits 7:3 of this register) pre-allocates no memory.
	0	**SPARE** Default Value: 0b Access: R/W _Custom_GTIReset: BUS Note: This bit was maintained as a placeholder for compatibility. Prior, it locked the register.

GPGPU Dispatch Dimension X

GPGPU_DISPATCHDIMX - GPGPU Dispatch Dimension X

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV

Address:	02500h-02503h
Name:	GPGPU Dispatch Dimension X
ShortName:	GPGPU_DISPATCHDIMX_RCSUNIT_BE_COMPUTE

Address:	1A500h-1A503h
Name:	GPGPU Dispatch Dimension X
ShortName:	GPGPU_DISPATCHDIMX_CCSUNIT_BE_COMPUTE0

DWord	Bit	Description
0	31:0	**Dispatch Dimension X** Format: U32 The number of thread groups to be dispatched in the X dimension (max x + 1). Value: 0, FFFFFFFFh; Name:

GPGPU Dispatch Dimension Y

GPGPU_DISPATCHDIMY - GPGPU Dispatch Dimension Y

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV

Address:	02504h-02507h
Name:	GPGPU Dispatch Dimension Y
ShortName:	GPGPU_DISPATCHDIMY_RCSUNIT_BE_COMPUTE

Address:	1A504h-1A507h
Name:	GPGPU Dispatch Dimension Y
ShortName:	GPGPU_DISPATCHDIMY_CCSUNIT_BE_COMPUTE0

DWord	Bit	Description
0	31:0	**Dispatch Dimension Y** Format: U32 The number of thread groups to be dispatched in the Y dimension (max y + 1 Value: 0, FFFFFFFFh; Name:

GPGPU Dispatch Dimension Z

GPGPU_DISPATCHDIMZ - GPGPU Dispatch Dimension Z	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02508h-0250Bh
Name:	GPGPU Dispatch Dimension Z
ShortName:	GPGPU_DISPATCHDIMZ_RCSUNIT_BE_COMPUTE
Address:	1A508h-1A50Bh
Name:	GPGPU Dispatch Dimension Z
ShortName:	GPGPU_DISPATCHDIMZ_CCSUNIT_BE_COMPUTE0

DWord	Bit	Description
0	31:0	**Dispatch Dimension Z** Format: U32 The number of thread groups to be dispatched in the Zdimension (max Z + 1) Value: 0, FFFFFFFFh; Name:

GP Thread Time

GP_THREAD_TIME - GP Thread Time	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	053C4h

Reading this register returns the cumulative GP context execution time. This register uses the same clock frequency as CTX_TIMESTAMP, but differs from CTX_TIMESTAMP because it excludes the execution time during preemption save or restore. This register gets context save/restored on a context switch.
The granularity of this toggle is at the rate of the bit 3 in the "Reported Timestamp Count" register(0x2358). The toggle will be 8 times slower that "Reported Timestamp Count". The granularity of the time stamp base unit for "Reported Timestamp Count" is defined in the Timestamp Bases subsection in Power Management chapter.

DWord	Bit	Description
0	31:0	**Timestamp Value** Access: RO Number of clock ticks that the context has run.

Graphics Device Reset Control

GDRST - Graphics Device Reset Control

Register Space:	MMIO: 0/2/0
Size (in bits):	32
CrashLogSaved:	true
CrashLogVisibility:	cspec
Address:	0941Ch

Graphics Device Reset Control Registers

DWord	Bit	Description
0	31:24	**Reserved**
	23:22	**Reserved**
	21	**Reserved**
	20	**Initiate Graphics SFC3 soft reset** Access: R/W Set _Custom_GTIReset: BUS Graphics SFC 3 Soft-Reset Control: '1' : Initiate a graphics SFC1 domain reset. - Cleared by CP once the reset is complete '0' : N/A - Once set, clearing of this bit has no effect on CP. Only CP hardware can reset this bit. Note: This is a non-posted register.
	19	**Initiate Graphics SFC2 soft reset** Access: R/W Set _Custom_GTIReset: BUS Graphics SFC 1 Soft-Reset Control: '1' : Initiate a graphics SFC1 domain reset. - Cleared by CP once the reset is complete '0' : N/A - Once set, clearing of this bit has no effect on CP. Only CP hardware can reset this bit. Note: This is a non-posted register.
	18	**Initiate Graphics SFC1 soft reset** Access: R/W Set _Custom_GTIReset: BUS Graphics SFC 1 Soft-Reset Control: '1' : Initiate a graphics SFC1 domain reset. - Cleared by CP once the reset is complete '0' : N/A - Once set, clearing of this bit has no effect on CP. Only CP hardware can reset this bit. Note: This is a non-posted register.

GDRST - Graphics Device Reset Control

	17	**Initiate Graphics SFC0 soft reset** Access: R/W Set _Custom_GTIReset: BUS Graphics SFC 0 Soft-Reset Control: '1' : Initiate a graphics SFC0 domain reset. - Cleared by CP once the reset is complete '0' : N/A - Once set, clearing of this bit has no effect on CP. Only CP hardware can reset this bit. Note: This is a non-posted register.
	16	**Initiate Graphics Vebox3 Soft Reset** Access: R/W Set _Custom_GTIReset: BUS Graphics VEbox Soft-Reset Control: '1' : Initiate a graphics Vebox domain reset. - Cleared by CP once the reset is complete '0' : N/A - Once set, clearing of this bit has no effect on CP. Only CP can reset this bit. Note: This is a non-posted register.
	15	**Initiate Graphics Vebox2 Soft Reset** Access: R/W Set _Custom_GTIReset: BUS Graphics VEbox Soft-Reset Control: '1' : Initiate a graphics Vebox domain reset. - Cleared by CP once the reset is complete '0' : N/A - Once set, clearing of this bit has no effect on CP. Only CP can reset this bit. Note: This is a non-posted register.
	14	**Initiate Graphics Vebox1 Soft Reset** Access: R/W Set _Custom_GTIReset: BUS Graphics VEbox Soft-Reset Control: '1' : Initiate a graphics Vebox domain reset. - Cleared by CP once the reset is complete '0' : N/A - Once set, clearing of this bit has no effect on CP. Only CP can reset this bit. Note: This is a non-posted register.
	13	**Initiate Graphics Vebox0 Soft Reset** Access: R/W Set _Custom_GTIReset: BUS Graphics VEbox Soft-Reset Control: '1' : Initiate a graphics Vebox domain reset. - Cleared by CP once the reset is complete

		'0' : N/A - Once set, clearing of this bit has no effect on CP. Only CP can reset this bit. Note: This is a non-posted register.
	12	**Initiate Graphics Media 7 Soft Reset** Access: R/W Set _Custom_GTIReset: BUS Graphics Media Soft-Reset Control: '1' : Initiate a graphics media 0 domain reset. - Cleared by CP once the reset is complete '0' : N/A - Once set, clearing of this bit has no effect on CP. Only CP can reset this bit. Note: This is a non-posted register.
	11	**Initiate Graphics Media 6 Soft Reset** Access: R/W Set _Custom_GTIReset: BUS Graphics Media Soft-Reset Control: '1' : Initiate a graphics media 0 domain reset. - Cleared by CP once the reset is complete '0' : N/A - Once set, clearing of this bit has no effect on CP. Only CP can reset this bit. Note: This is a non-posted register.
	10	**Initiate Graphics Media 5 Soft Reset** Access: R/W Set _Custom_GTIReset: BUS Graphics Media Soft-Reset Control: '1' : Initiate a graphics media 0 domain reset. - Cleared by CP once the reset is complete '0' : N/A - Once set, clearing of this bit has no effect on CP. Only CP can reset this bit. Note: This is a non-posted register.
	9	**Initiate Graphics Media 4 Soft Reset** Access: R/W Set _Custom_GTIReset: BUS Graphics Media Soft-Reset Control: '1' : Initiate a graphics media 0 domain reset. - Cleared by CP once the reset is complete '0' : N/A - Once set, clearing of this bit has no effect on CP. Only CP can reset this bit. Note: This is a non-posted register.
	8	**Initiate Graphics Media 3 Soft Reset** Access: R/W Set _Custom_GTIReset: BUS

GDRST - Graphics Device Reset Control

		Graphics Media Soft-Reset Control: '1' : Initiate a graphics media 0 domain reset. - Cleared by CP once the reset is complete '0' : N/A - Once set, clearing of this bit has no effect on CP. Only CP can reset this bit. Note: This is a non-posted register.
	7	**Initiate Graphics Media 2 Soft Reset** Access: R/W Set _Custom_GTIReset: BUS Graphics Media Soft-Reset Control: '1' : Initiate a graphics media 0 domain reset. - Cleared by CP once the reset is complete '0' : N/A - Once set, clearing of this bit has no effect on CP. Only CP can reset this bit. Note: This is a non-posted register.
	6	**Initiate Graphics Media 1 Soft Reset** Access: R/W Set _Custom_GTIReset: BUS Graphics Media Soft-Reset Control: '1' : Initiate a graphics media 0 domain reset. - Cleared by CP once the reset is complete '0' : N/A - Once set, clearing of this bit has no effect on CP. Only CP can reset this bit. Note: This is a non-posted register.
	5	**Initiate Graphics Media 0 Soft Reset** Access: R/W Set _Custom_GTIReset: BUS Graphics Media Soft-Reset Control: '1' : Initiate a graphics media 0 domain reset. - Cleared by CP once the reset is complete '0' : N/A - Once set, clearing of this bit has no effect on CP. Only CP can reset this bit. Note: This is a non-posted register.
	4	**Reserved** Access: RO Format: MBZ
	3	**Initiate Graphics GUC soft reset** Access: R/W Set _Custom_GTIReset: BUS Graphics GUC Soft-Reset Control: '1' : Initiate a graphics GUC domain reset(cgucrst_b). - Cleared by CP once the reset is complete

GDRST - Graphics Device Reset Control

		'0' : N/A - Once set, clearing of this bit has no effect on CP. Only CP can reset this bit. Note: This is a non-posted register.
	2	**Initiate Graphics Blitter Soft Reset** Access: R/W Set _Custom_GTIReset: BUS **Description** Graphics Blitter Soft-Reset Control: '1' : Initiate a graphics blitter domain reset. - Cleared by CP once the reset is complete '0' : N/A - Once set, clearing of this bit has no effect on CP. Only CP can reset this bit. Note: This is a non-posted register.
	1	**Initiate Graphics Render Soft Reset** Access: R/W Set _Custom_GTIReset: BUS **Description** Graphics Render Soft-Reset Control: '1' : Initiate a graphics render domain reset. - Cleared by CP once the reset is complete '0' : N/A - Once set, clearing of this bit has no effect on CP. Only CP can reset this bit. Note: This is a non-posted register. Note that both RenderEngine (hedaed by RenderCS) and ComputeEngine (headed by ComputeCS) are in Graphics Render Reset domain.
	0	**Initiate Graphics Full Soft Reset** Access: R/W Set _Custom_GTIReset: BUS Graphics Full Soft-Reset Control: '1' : Initiate a full graphics reset (i.e., graphics render, media, and blitter reset). - Cleared by CP once the reset is complete '0' : N/A - Once set, clearing of this bit has no effect on CP. Only CP can reset this bit. Note: This is a non-posted register.

Graphics Memory Range Address

GMADR_0_2_0_PCI - Graphics Memory Range Address

Register Space:	PCI: 0/2/0
Size (in bits):	64
Address:	00018h

GMADR is the PCI aperture used by S/W to access tiled GFX surfaces in a linear fashion.

DWord	Bit	Description
0	63:32	**Memory Base Address** Default Value: 00000000h Access: R/W _Custom_GTIReset: BUS Set by the OS, these bits correspond to address signals [63:32].
	31	**4096 MB Address Mask** Default Value: 0b Access: R/W Lock _Custom_GTIReset: BUS This bit is either part of the Memory Base Address (R/W) or part of Address Mask (RO) depending on the value of MSAC.APSZ.RO and forced to 0 when MSAC.APSZ >= 4096MB. (i.e. MSAC.APSZ[4]=1)
	30	**2048 MB Address Mask** Default Value: 0b Access: R/W Lock _Custom_GTIReset: BUS This bit is either part of the Memory Base Address (R/W) or part of the Address Mask (RO) depending on the value of MSAC.APSZ.RO and forced to 0 when MSAC.APSZ >= 2048MB. (i.e. MSAC.APSZ[3]=1)
	29	**1024 MB Address Mask** Default Value: 0b Access: R/W Lock _Custom_GTIReset: BUS This bit is either part of the Memory Base Address (R/W) or part of the Address Mask (RO) depending on the value of MSAC.APSZ.RO and forced to 0 when MSAC.APSZ >= 1024MB. (i.e. MSAC.APSZ[2]=1)
	28	**512MB Address Mask** Default Value: 0b Access: R/W Lock _Custom_GTIReset: BUS This bit is either part of the Memory Base Address (R/W) or part of the Address Mask (RO) depending on the value of MSAC.APSZ.RO and forced to 0 when MSAC.APSZ >= 512MB. (i.e.

GMADR_0_2_0_PCI - Graphics Memory Range Address

		MSAC.APSZ[1]=1)
	27	**256 MB Address Mask** Default Value: 0b Access: R/W Lock _Custom_GTIReset: BUS This bit is either part of the Memory Base Address (R/W) or part of the Address Mask (RO) depending on the value of MSAC.APSZ.RO and forced to 0 when MSAC.APSZ >= 256MB. (i.e. MSAC.APSZ[0]=1)
	26:4	**Address Mask** Default Value: 00000000000000000000000b Access: RO _Custom_GTIReset: BUS Hardwired to 0s to indicate at least 128MB address range.
	3	**Prefetchable Memory** Default Value: 1b Access: RO _Custom_GTIReset: BUS Hardwired to 1 to enable prefetching.
	2:1	**Memory Type** Default Value: 10b Access: RO _Custom_GTIReset: BUS Hardwired to 2h to indicate 64 bit base address.
	0	**Memory/IO Space** Default Value: 0b Access: RO _Custom_GTIReset: BUS Hardwired to 0 to indicate memory space.

Graphics Mode Register

GFX_MODE - Graphics Mode Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0229Ch-0229Fh
Name:	Graphics Mode Register
ShortName:	GFX_MODE_RCSUNIT
Address:	1829Ch-1829Fh
Name:	Graphics Mode Register
ShortName:	GFX_MODE_POCSUNIT
Address:	2229Ch-2229Fh
Name:	Graphics Mode Register
ShortName:	GFX_MODE_BCSUNIT
Address:	1C029Ch-1C029Fh
Name:	Graphics Mode Register
ShortName:	GFX_MODE_VCSUNIT0
Address:	1C429Ch-1C429Fh
Name:	Graphics Mode Register
ShortName:	GFX_MODE_VCSUNIT1
Address:	1C829Ch-1C829Fh
Name:	Graphics Mode Register
ShortName:	GFX_MODE_VECSUNIT0
Address:	1D029Ch-1D029Fh
Name:	Graphics Mode Register
ShortName:	GFX_MODE_VCSUNIT2
Address:	1D429Ch-1D429Fh
Name:	Graphics Mode Register
ShortName:	GFX_MODE_VCSUNIT3
Address:	1D829Ch-1D829Fh
Name:	Graphics Mode Register
ShortName:	GFX_MODE_VECSUNIT1
Address:	1E029Ch-1E029Fh
Name:	Graphics Mode Register
ShortName:	GFX_MODE_VCSUNIT4

GFX_MODE - Graphics Mode Register

Address:	1E429Ch-1E429Fh
Name:	Graphics Mode Register
ShortName:	GFX_MODE_VCSUNIT5

Address:	1E829Ch-1E829Fh
Name:	Graphics Mode Register
ShortName:	GFX_MODE_VECSUNIT2

Address:	1F029Ch-1F029Fh
Name:	Graphics Mode Register
ShortName:	GFX_MODE_VCSUNIT6

Address:	1F429Ch-1F429Fh
Name:	Graphics Mode Register
ShortName:	GFX_MODE_VCSUNIT7

Address:	1F829Ch-1F829Fh
Name:	Graphics Mode Register
ShortName:	GFX_MODE_VECSUNIT3

Address:	1A29Ch-1A29Fh
Name:	Graphics Mode Register
ShortName:	GFX_MODE_CCSUNIT0

This register contains a control bit for the new execlist and 2-level PPGTT functions.

Programming Notes

"Privilege Check Disable" are the only programmable bits in GFX_MODE register for PositionCS, functionality for the rest of the bits is not supported by Position command streamer.

DWord	Bit	Description
0	31:16	**Mask** Access: WO Format: Mask Must be set to modify corresponding bit in Bits 15:0. (All implemented bits)
	15	**Reserved** Format: PBC
	14	**Reserved** Format: PBC
	13	**Reserved** Format: PBC
	12	**Reserved**

12 **Reserved**

Source:	RenderCS, BlitterCS, ComputeCS
Format:	PBC

11 **Virtual Function MMIO Read Access Control**

This bit controls the disabling and enabling of MMIO read access of an virtual function context running on an engine.

Value	Name	Description
1		A VF context running on an engine can do MMIO read access to other engines. Ex: VF context running on RenderCS can do MMIO read access to VideoCS.
0	**Default]**	A VF context running on an engine cant do MMIO read access to other engines. Ex: VF context running on RenderCS can't do MMIO read access to VideoCS.

10 **Prefetch Disable**

This field allows software to enable or disable pre-fetch mechanism for command buffers in hardware.

Value	Name	Description
0	**[Default]**	When reset pre-fetch of command buffers is enabled in hardware. However software can enable/disable pre-fetch functionality locally from within a command sequence using MI_ARB_CHK command on per context basis.
1		When set pre-fetch of command buffers is disabled in hardware.

9 **Per-Process GTT Enable**

Per-Process GTT Enable

Value	Name	Description
0h	PPGTT Disable **[Default]**	When clear, the Global GTT will be used to translate memory access from designated commands and for commands that select the PPGTT as their translation space.
1h	PPGTT Enable	When set, the PPGTT will be used to translate memory access from designated commands and for commands that select the PPGTT as their translation space.

Programming Notes

This bit is used for enabling PPGTT access in Ring Buffer mode of scheduling. Privilege field in context descriptor states the same in Execlist Mode of scheduling. This field should be set before programming PDP0/1/2/3 registers in order to set the PPGTT translation of memory access.
Programming this bit doesn't enable or disable the PPGTT translation of memory access immediately; the change comes in to affect only when the Page Directory registers are programmed. Programming this bit must be followed by programming Page Directory Registers in order to enable or disable the PPGTT translation of memory access.

GFX_MODE - Graphics Mode Register

8 — Reserved

Source:	BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS
Format:	PBC

8 — Reserved

7 — 64Bit Virtual Addressing Enable

64Bit Virtual Addressing Enable

Value	Name	Description
0h	64Bit Virtual Addressing Disable **[Default]**	When clear indicates GFX operating in 32bit Virtual Addressing for PPGTT based memory access.
1h	64Bit Virtual Addressing Enable	When Set indicates GFX operating in 64bit (48bit Canonical) Virtual Addressing for PPGTT based memory access.

Programming Notes

This bit is only valid when PPGTT is enabled in ring buffer mode of scheduling. Context Descriptor has a similar bit to control 64bit virtual addressing in execlist mode of scheduling. Whether this field is set or clear virtual addresses translated through GGTT are always 32Bit. This field should be programmed before enabling PPGTT access. When this field is not set or for GGTT virtual addresses, Graphics Address [47:32] field of any commands or register exercised by SW should be programmed to 0x0.

6:5 — Reserved

4 — Reserved

3 — Disable Legacy Mode

This bit must be set to disable Legacy behavior to support features added for the current project.

When set the size of the CSB status FIFO is 12 deep.

Value	Name	Description
0h	Enable Legacy **[Default]**	Any features using this bit will be compatible with legacy drivers.
1h	Disable Legacy	HW will not be compatible with legacy drivers.

Programming Notes

A graphics reset is required prior to changing the value of this bit.

2 — Reserved

Format:	PBC

1 — MMIO Read Privilege Check Disable

Format:	Enable

This field when set, disables MMIO Read Privilege Violation checks on non-privileged batch buffers. When set Privileged MMIO read requests are allowed to be executed from non-privileged batch buffers.

GFX_MODE - Graphics Mode Register		
	0	**Privilege Check Disable** This field when set, disables Privilege Violation checks on non-privileged batch buffers. When set Privileged commands are allowed to be executed from non-privileged batch buffers.

Graphics Primary Interrupt

GFX_MSTR_INTR - Graphics Primary Interrupt

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 190010h

Top level register that indicates interrupt from hardware.
Bits in this register are set interrupts are pending in the underlying PCU, display or GT interrupts

DWord	Bit	Description
0	31	**Primary Interrupt** Default Value: 0b Access: R/W _Custom_GTIReset: BUS This is the primary control for graphics interrupts. This must be enabled for any of these interrupts to propagate to PCI device interrupt .
	30	**PCU** Default Value: 0b Access: RO Variant _Custom_GTIReset: BUS
	29	**GU_MISC** Default Value: 0b Access: RO Variant _Custom_GTIReset: BUS
	28:26	**Reserved** Access: RO Format: MBZ
	25:17	**Reserved** Access: RO Format: MBZ
	16	**Display** Default Value: 0b Access: RO Variant _Custom_GTIReset: BUS
	15:2	**Reserved** Access: RO Format: MBZ

GFX_MSTR_INTR - Graphics Primary Interrupt

	1	**GT DW1**
		Default Value: 0b
		Access: RO Variant
		_Custom_GTIReset: BUS
	0	**GT DW0**
		Default Value: 0b
		Access: RO Variant
		_Custom_GTIReset: BUS

Graphics System Event

GSE_0_2_0_PCI - Graphics System Event		
Register Space:	PCI: 0/2/0	
Size (in bits):	32	
Address:	000E4h	
This register can be accessed by either Byte, Word, or Dword PCI config cycles. A write to this register will cause the Graphics System Event display interrupt if it is enabled and unmasked in the display interrupt registers.		
DWord	**Bit**	**Description**
0	31:24	**GSE Scratch Trigger 3** Default Value: 00000000b Access: R/W _Custom_GTIReset: BUS
	23:16	**GSE Scratch Trigger 2** Default Value: 00000000b Access: R/W _Custom_GTIReset: BUS
	15:8	**GSE Scratch Trigger 1** Default Value: 00000000b Access: R/W _Custom_GTIReset: BUS
	7:0	**GSE Scratch Trigger 0** Default Value: 00000000b Access: R/W _Custom_GTIReset: BUS

Graphics Translation Table Memory Mapped Range Address

GTTMMADR_0_2_0_PCI - Graphics Translation Table Memory Mapped Range Address

Register Space:	PCI: 0/2/0
Size (in bits):	64
Address:	00010h

This register requests allocation for the combined Graphics Translation Table Modification Range and Memory Mapped Range. The range requires 16 MB combined for MMIO and Global GTT aperture, with 2MB of that used by MMIO, 6MB reserved, and 8MB used by GTT. GTTADR will begin at (GTTMMADR + 8 MB) while the MMIO base address will be the same as GTTMMADR. The region between (GTTMMADR + 2MB) - (GTTMMADR + 8MB) is reserved. For the Global GTT, this range is defined as a memory BAR in graphics device config space. It is an alias into which software is required to write Page Table Entry values (PTEs). Software may read PTE values from the global Graphics Translation Table (GTT). PTEs cannot be written directly into the global GTT memory area. The device snoops writes to this region in order to invalidate any cached translations within the various TLB's implemented on-chip. The allocation is for 16MB and the base address is defined by bits [38:24].

DWord	Bit	Description
0	63:24	**Memory Base Address** Default Value: 0000000000h Access: R/W _Custom_GTIReset: BUS Set by the OS, these bits correspond to address signals [63:24].
	23:4	**Address Mask** Default Value: 00000000000000000000b Access: RO _Custom_GTIReset: BUS Hardwired to 0s to indicate at least 16MB address range.
	3	**Prefetchable Memory** Default Value: 0b Access: RO _Custom_GTIReset: BUS Hardwired to 0 to prevent prefetching.
	2:1	**Memory Type** Default Value: 10b Access: RO _Custom_GTIReset: BUS Hardwired to 2h to indicate 64 bit base address.

GTTMMADR_0_2_0_PCI - Graphics Translation Table Memory Mapped Range Address

	0	**Memory/IO Space** Default Value: 0b Access: RO _Custom_GTIReset: BUS Hardwired to 0 to indicate memory space.

Graphics Virtual Primary Interrupt

GFX_VIRT_MSTR_INTR - Graphics Virtual Primary Interrupt

Register Space:	MMIO: 0/2/0
Size (in bits):	32

Address:	191010h
Name:	VF1_GFX_MSTR_INTR
ShortName:	VF1_GFX_MSTR_INTR

Address:	192010h
Name:	VF2_GFX_MSTR_INTR
ShortName:	VF2_GFX_MSTR_INTR

Address:	193010h
Name:	VF3_GFX_MSTR_INTR
ShortName:	VF3_GFX_MSTR_INTR

Address:	194010h
Name:	VF4_GFX_MSTR_INTR
ShortName:	VF4_GFX_MSTR_INTR

Address:	195010h
Name:	VF5_GFX_MSTR_INTR
ShortName:	VF5_GFX_MSTR_INTR

Address:	196010h
Name:	VF6_GFX_MSTR_INTR
ShortName:	VF6_GFX_MSTR_INTR

Address:	197010h
Name:	VF7_GFX_MSTR_INTR
ShortName:	VF7_GFX_MSTR_INTR

Top level register that indicates interrupt from hardware.
Bits in this register are set interrupts are pending in the underlying PCU, display or GT interrupts

DWord	Bit	Description
0	31	**Primary Interrupt** Access: R/W
	30	**PCU** Access: RO
	29:17	**Reserved** Access: RO Format: MBZ

GFX_VIRT_MSTR_INTR - Graphics Virtual Primary Interrupt

	16	**Display** Access: RO
	15:6	**Reserved** Access: RO Format: MBZ
	5:4	**Reserved** Access: RO Format: MBZ
	3:2	**Reserved** Access: RO Format: MBZ
	1	**GT DW1** Access: R/W
	0	**GT DW0** Access: R/W

GSCPowerGoodDelay

GSC_PG_DLY - GSCPowerGoodDelay		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
IA/Context savable by MGSR/punit		
DWord	**Bit**	**Description**
0	31:16	**Power Good Delay** Default Value: 00C8h PWRGOOD_DLY Value
	15:0	**RAMP Delay** Default Value: 00C8h RAMP_DLY Value

GS Invocation Counter

GS_INVOCATION_COUNT - GS Invocation Counter	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02328h
Name:	GS Invocation Counter
ShortName:	GS_INVOCATION_COUNT

This register stores the number of objects that are part of geometry shader threads. This register is part of the context save and restore.

DWord	Bit	Description
0	63:32	**GS Invocation Count UDW** Number of objects that are dispatched as a geometry shader threads invoked by the GS stage. Updated only when **Statistics Enable** is set in GS_STATE (see the Geometry Shader Chapter in the 3D Volume.)
	31:0	**GS Invocation Count LDW** Number of objects that are dispatched as a geometry shader threads invoked by the GS stage. Updated only when **Statistics Enable** is set in GS_STATE (see the Geometry Shader Chapter in the 3D Volume.)

GSI Power Good Delay

GSI_PG_DLY - GSI Power Good Delay		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
IA/Context savable by MGSR/punit		
DWord	**Bit**	**Description**
0	31:16	**Power Good Delay** Default Value: 00C8h PWRGOOD_DLY Value
	15:0	**RAMP Delay** Default Value: 00C8h RAMP_DLY Value

GSMBASE

GSMBASE - GSMBASE

Register Space:	MMIO: 0/2/0
Size (in bits):	64
SOC_Consumer:	BIOS
Address:	108100h

This register contains the base address of stolen DRAM memory for the GTT. BIOS determines the base of GTT stolen memory. BIOS is now able to allocate Gfx Stolen Memory above the 4GB.

DWord	Bit	Description
0..1	63:32	**BGSM_MSB** Default Value: 00000000h Access: R/W _Custom_GTIReset: BUS This BitField contains bits 63 to 32 of the base address of stolen DRAM memory.
	31:20	**BGSM_LSB** Default Value: 001h Access: R/W _Custom_GTIReset: BUS This BitField contains bits 31 to 20 of the base address of stolen DRAM memory.
	19:1	**Reserved** Access: RO Format: MBZ
	0	**SPARE** Default Value: 0b Access: R/W _Custom_GTIReset: BUS This was a lock bit prior.

GS Primitives Counter

GS_PRIMITIVES_COUNT - GS Primitives Counter	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02330h
Name:	GS Primitives Counter
ShortName:	GS_PRIMITIVES_COUNT

This register reflects the total number of primitives that have been output by the Geometry Shader stage. This register is part of the context save and restore.

DWord	Bit	Description
0	63:32	**GS Primitives Count UDW** Total number of primitives output by the geometry stage. Updated only when Statistics Enable is set in GS_STATE (see the Geometry Shader Chapter in the 3D Volume.)
	31:0	**GS Primitives Count LDW** Total number of primitives output by the geometry stage. Updated only when Statistics Enable is set in GS_STATE (see the Geometry Shader Chapter in the 3D Volume.)

GT_FLUSH_BCLD_ACK

GT_FLUSH_BCLD_ACK - GT_FLUSH_BCLD_ACK

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	030C0h

GT writes a '1 to this bit to acknowledge PRMRR range registers are loaded into GT.
This register is a LOCAL CR register and not an MMIO register

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**ACK** Default Value: 0b Access: R/W _Custom_GTIReset: BUS GT Boot Context Load Ack

GT_RELOAD_FLUSH

GT_RELOAD_FLUSH - GT_RELOAD_FLUSH

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	030B0h

Ucode writes a '1 to bits 0 which triggers GT to flush and re-load PRMRR range registers.
This register is a LOCAL CR register and not an MMIO register

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**BCLD_REQ** Default Value: 0b Access: R/W _Custom_GTIReset: BUS GT Boot Context Load Request. Write to this bit will initiate Mcheck Complete Routine (PPPE flow).

GTACK

GTACK - GTACK

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	120004h

This register is written to by GT for various device 2 sequencer flows.

DWord	Bit	Description
0	31:5	**Reserved** Access: RO Format: MBZ
	4	**RTPACK** Default Value: 0h Access: R/W _Custom_GTIReset: BUS GT indicates that the set root table pointer flow is complete from its point of view by writing a 1b to this field. Once SW is notified with the appropriate status bit, this bit is cleared by the HW.
	3	**DESCRACK** Default Value: 0h Access: R/W _Custom_GTIReset: BUS GT indicates that the generic descriptor flow is complete from its point of view by writing a 1b to this field Once SW is notified with the appropriate status bit, this bit is cleared by the HW.
	2	**VTDACK** Default Value: 0h Access: R/W _Custom_GTIReset: BUS GT indicates that the Translation Enable/Disable or IOTLB Invalidation flow is complete from its point of view by writing a 1b to this field. Once SW is notified with the appropriate status bit, this bit is cleared by the HW.
	1	**DPRACK** Default Value: 0h Access: R/W _Custom_GTIReset: BUS GT indicates that DPR Update flow is complete from its point of view by writing a 1b to this field. Once SW is notified with the appropriate status bit, this bit is cleared by the HW.

GTACK - GTACK

	0	**PMRACK** Default Value: 0h Access: R/W _Custom_GTIReset: BUS GT indicates that PMR Enable/Disable flow is complete from its point of view by writing a 1b to this field. Once SW is notified with the appropriate status bit, this bit is cleared by the HW.

GTC_CTL

GTC_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	67000h-67003h
Name:	Global Time Code Control
ShortName:	GTC_CTL
Reset:	soft

DWord	Bit	Description
0	31	**GTC Function Enable** This bit enables the GTC counter. Value: 0b = Disable; 1b = Enable Restriction: Enable this bit before enabling GTC controller operation on a port with a GTC capable device.
	30:29	**Reserved** Access: RO Format: MBZ
	28:13	**Reserved**
	12:1	**Reserved** Access: RO Format: MBZ
	0	**Reserved**

GTC_DDA_M

GTC_DDA_M		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	67010h-67013h	
Name:	Global Time Code DDA M	
ShortName:	GTC_DDA_M	
Reset:	soft	
DWord	**Bit**	**Description**
0	31:24	**Reserved**
	23:0	**GTC DDA M** This field is used to program the M value of the GTC DDA. The ratio of M to N programmed depends on the GTC reference clock. The DDA programmed values are related by the following formula: 1/(accumulator increment) = Reference Clock * DDA_M / DDA_N

GTC_DDA_N

GTC_DDA_N		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	67014h-67017h	
Name:	Global Time Code DDA N	
ShortName:	GTC_DDA_N	
Reset:	soft	
DWord	**Bit**	**Description**
0	31:24	**GTC Accum Inc** Format: U7.1 This field is the GTC accumulator increment value in nanoseconds each time the DDA trips. It is programmed in 7.1 fixed point binary format where the LSB represents 0.5ns increment.
	23:0	**GTC DDA N** This field is used to program the N value of the GTC DDA. The ratio of M to N programmed depends on the GTC reference clock and should not result in any accumulation error in any 10ms interval period. The DDA programmed values are related by the following formula: 1/(accumulator increment) = Reference Clock * DDA_M / DDA_N

GTC_IIR

GTC_IIR	
Register Space:	MMIO: 0/2/0
Access:	R/WC
Size (in bits):	32
Address:	67058h-6705Bh
Name:	Global Time Code Interrupt Identity
ShortName:	GTC_IIR
Reset:	soft

See the GTC interrupt bit definition to find the source event for each interrupt bit.

DWord	Bit	Description
0	31:0	**Interrupt Identity Bits** This field holds the persistent values of the GTC interrupt bits which are unmasked by the GTC_IMR.Bits set in this register will propagate to the GTC interrupt in the Display Engine Miscellaneous Interrupts. Bits set in this register will remain set (persist) until the interrupt condition is cleared by writing a '1' to the appropriate bits.

Value	Name
0b	Condition Not Detected
1b	Condition Detected

GTC_IMR

GTC_IMR	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	67054h-67057h
Name:	Global Time Code Interrupt Mask
ShortName:	GTC_IMR
Reset:	soft

See the GTC interrupt bit definition to find the source event for each interrupt bit.

DWord	Bit	Description
0	31:0	**Interrupt Mask Bits** This field contains a bit mask which selects which GTC events are reported int the GTC IIR.

Value	Name
0b	Not Masked
1b	Masked
FFFFFFFFh	All interrupts masked **[Default]**

GTC_LIVE

GTC_LIVE		
Register Space:	MMIO: 0/2/0	
Access:	RO	
Size (in bits):	32	
Address:	67020h-67023h	
Name:	Global Time Code Live	
ShortName:	GTC_LIVE	
Reset:	soft	
DWord	**Bit**	**Description**
0	31:0	**GTC Live Value** This field contains the live current value of the GTC. It is inactive when the GTC controller function is disabled. This register also samples and holds the live GTC value following a Audio Time Capture (ATC) event until software reads this register. A subsequent read of this register will reflect the live value.

GTC_PORT_CTL

GTC_PORT_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	64370h-64373h
Name:	DDI GTC Port Control
ShortName:	GTC_PORT_CTL_USBC1
Reset:	soft
Address:	64470h-64473h
Name:	DDI GTC Port Control
ShortName:	GTC_PORT_CTL_USBC2
Reset:	soft
Address:	64570h-64573h
Name:	DDI GTC Port Control
ShortName:	GTC_PORT_CTL_USBC3
Reset:	soft
Address:	64670h-64673h
Name:	DDI GTC Port Control
ShortName:	GTC_PORT_CTL_USBC4
Reset:	soft
Address:	64770h-64773h
Name:	DDI GTC Port Control
ShortName:	GTC_PORT_CTL_USBC5
Reset:	soft
Address:	64870h-64873h
Name:	DDI GTC Port Control
ShortName:	GTC_PORT_CTL_USBC6
Reset:	soft
Address:	64070h-64073h
Name:	DDI GTC Port Control
ShortName:	GTC_PORT_CTL_A
Reset:	soft
Address:	64170h-64173h
Name:	DDI GTC Port Control
ShortName:	GTC_PORT_CTL_B

GTC_PORT_CTL

Reset: soft

Address: 64270h-64273h
Name: DDI GTC Port Control
ShortName: GTC_PORT_CTL_C
Reset: soft

DWord	Bit	Description
0	31	**Port Global Time Code Enable** This bit enables the GTC controller to start lock acquisition phase with remote GTC sink connected to this port. This bit has no effect if the GTC controller is disabled. **Value** / **Name** 0b – Disable 1b – Enable **Restriction** The Maintenance Phase Enable bit must be initially written as '0' when this bit is set.
	30:25	**Reserved**
	24	**Maintenance Phase Enable** This bit is used by software to transition from lock acquisition to lock maintenance phase. The GTC controller generates an interrupt at the end of the lock phase as determined by lock acquisition duration field. Software shall read the sink device GTC lock done bit. If set, software shall set this bit to '1' after first writing the GTC skew value to the RX GTC skew DPCD offset with GTC skew enable bit set to '1'. **Value** / **Name** / **Description** 0b – Lock – Lock acquisition phase. The controller writes or reads GTC every 1ms. 1b – Maintain – Lock maintenance phase. The controller writes or reads GTC every 10ms.
	23:1	**Reserved**
	0	**Port RX Lock Done** This bit indicates the remote GTC sink has achieved lock. This bit shall be written by software after reading remote GTC sink DPCD register. This bit shall be cleared by software when GTC controller is reset from lock maintenance mode to lock acquisition mode or when the controller is disabled. **Value** / **Name** 0b – Not Locked 1b – Locked

GTC_PORT_MISC

GTC_PORT_MISC	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	64394h-64397h
Name:	DDI GTC Port Miscellaneous
ShortName:	GTC_PORT_MISC_USBC1
Reset:	soft
Address:	64494h-64497h
Name:	DDI GTC Port Miscellaneous
ShortName:	GTC_PORT_MISC_USBC2
Reset:	soft
Address:	64594h-64597h
Name:	DDI GTC Port Miscellaneous
ShortName:	GTC_PORT_MISC_USBC3
Reset:	soft
Address:	64694h-64697h
Name:	DDI GTC Port Miscellaneous
ShortName:	GTC_PORT_MISC_USBC4
Reset:	soft
Address:	64794h-64797h
Name:	DDI GTC Port Miscellaneous
ShortName:	GTC_PORT_MISC_USBC5
Reset:	soft
Address:	64894h-64897h
Name:	DDI GTC Port Miscellaneous
ShortName:	GTC_PORT_MISC_USBC6
Reset:	soft
Address:	64094h-64097h
Name:	DDI GTC Port Miscellaneous
ShortName:	GTC_PORT_MISC_A
Reset:	soft
Address:	64194h-64197h
Name:	DDI GTC Port Miscellaneous
ShortName:	GTC_PORT_MISC_B

GTC_PORT_MISC

Reset:	soft
Address:	64294h-64297h
Name:	DDI GTC Port Miscellaneous
ShortName:	GTC_PORT_MISC_C
Reset:	soft

DWord	Bit	Description
0	31:22	**Reserved** Access: RO Format: MBZ
	21:12	**GTC Update Message Delay** Default Value: 00110100b 52 nanoseconds This field programs the absolute delay in nanoseconds between the GTC at the aux sync point event and the corresponding GTC value at the capture point. It represents the delay between the GTC values at the aux sync point and capture point introduced due to synchronization and glitch suppression.
	11:8	**Min Lock Duration** Default Value: 1010b 10ms This field determines the minimum duration in milliseconds of lock acquisition and maintenance phase after which software is notified through interrupt. The GTC interrupt enable and mask register must be enabled beforehand. Software may also poll the interrupt identity bit in IIR.
	7:0	**Reserved** Access: RO Format: MBZ

GTC_PORT_TX_CURR

GTC_PORT_TX_CURR	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	64378h-6437Bh
Name:	DDI GTC Port TX Current
ShortName:	GTC_PORT_TX_CURR_USBC1
Reset:	soft
Address:	64478h-6447Bh
Name:	DDI GTC Port TX Current
ShortName:	GTC_PORT_TX_CURR_USBC2
Reset:	soft
Address:	64578h-6457Bh
Name:	DDI GTC Port TX Current
ShortName:	GTC_PORT_TX_CURR_USBC3
Reset:	soft
Address:	64678h-6467Bh
Name:	DDI GTC Port TX Current
ShortName:	GTC_PORT_TX_CURR_USBC4
Reset:	soft
Address:	64778h-6477Bh
Name:	DDI GTC Port TX Current
ShortName:	GTC_PORT_TX_CURR_USBC5
Reset:	soft
Address:	64878h-6487Bh
Name:	DDI GTC Port TX Current
ShortName:	GTC_PORT_TX_CURR_USBC6
Reset:	soft
Address:	64078h-6407Bh
Name:	DDI GTC Port TX Current
ShortName:	GTC_PORT_TX_CURR_A
Reset:	soft
Address:	64178h-6417Bh
Name:	DDI GTC Port TX Current
ShortName:	GTC_PORT_TX_CURR_B

GTC_PORT_TX_CURR		
Reset:	soft	
Address:	64278h-6427Bh	
Name:	DDI GTC Port TX Current	
ShortName:	GTC_PORT_TX_CURR_C	
Reset:	soft	
DWord	**Bit**	**Description**
0	31:0	**Global Time Code Port TX Current** This field contains the local GTC value sampled at the Aux sync point of the response message from the remote GTC sink following software read of the remote sink GTC DPCD register.

GTC_PORT_TX_PREV

GTC_PORT_TX_PREV	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	64380h-64383h
Name:	DDI GTC Port TX Previous
ShortName:	GTC_PORT_TX_PREV_USBC1
Reset:	soft
Address:	64480h-64483h
Name:	DDI GTC Port TX Previous
ShortName:	GTC_PORT_TX_PREV_USBC2
Reset:	soft
Address:	64580h-64583h
Name:	DDI GTC Port TX Previous
ShortName:	GTC_PORT_TX_PREV_USBC3
Reset:	soft
Address:	64680h-64683h
Name:	DDI GTC Port TX Previous
ShortName:	GTC_PORT_TX_PREV_USBC4
Reset:	soft
Address:	64780h-64783h
Name:	DDI GTC Port TX Previous
ShortName:	GTC_PORT_TX_PREV_USBC5
Reset:	soft
Address:	64880h-64883h
Name:	DDI GTC Port TX Previous
ShortName:	GTC_PORT_TX_PREV_USBC6
Reset:	soft
Address:	64080h-64083h
Name:	DDI GTC Port TX Previous
ShortName:	GTC_PORT_TX_PREV_A
Reset:	soft
Address:	64180h-64183h
Name:	DDI GTC Port TX Previous
ShortName:	GTC_PORT_TX_PREV_B

GTC_PORT_TX_PREV

Reset:	soft
Address:	64280h-64283h
Name:	DDI GTC Port TX Previous
ShortName:	GTC_PORT_TX_PREV_C
Reset:	soft

DWord	Bit	Description
0	31:0	**Global Time Code Port TX Previous** This field contains the previous local GTC value sampled at Aux sync point. It is transferred from the GTC_PORT_TX_CURR register when the current value is updated.

GT C6 Entry TSC LSB

GTC6_ENTRY_TSC_LSB - GT C6 Entry TSC LSB		
Register Space:	MMIO: 0/2/0	
Source:	BSpec	
Size (in bits):	32	
Address:	00C28h	
C6 Entry TSC LSB		
DWord	**Bit**	**Description**
0	31:0	**Count for C6 entry TSC LSB** Access: RO _Custom_GTIReset: BUS C6 Entry TSC LSB

GT C6 Entry TSC MSB

GTC6_ENTRY_TSC_MSB - GT C6 Entry TSC MSB		
Register Space:	MMIO: 0/2/0	
Source:	BSpec	
Size (in bits):	32	
Address:	00C2Ch	
C6 Entry TSC MSB		
DWord	**Bit**	**Description**
0	31:24	**RSVD** Access: RO _Custom_GTIReset: BUS RSVD
	23:0	**Count GT C6 Entry TSC MSB** Access: RO _Custom_GTIReset: BUS C6 Entry TSC MSB

GT C6 Residency LSB

GTC6_RESIDENCY_LSB - GT C6 Residency LSB		
Register Space:	MMIO: 0/2/0	
Source:	BSpec	
Size (in bits):	32	
Address:	00C20h	
C6 Residency LSB		
DWord	**Bit**	**Description**
0	31:0	**Count for C6 Residency** Access: RO _Custom_GTIReset: BUS C6 Resindency LSB

GT C6 Residency MSB

GTC6_RESIDENCY_MSB - GT C6 Residency MSB		
Register Space:	MMIO: 0/2/0	
Source:	BSpec	
Size (in bits):	32	
Address:	00C24h	
C6 Residency MSB		
DWord	**Bit**	**Description**
0	31:24	**RSVD** Access: RO _Custom_GTIReset: BUS RSVD
	23:0	**Count for C6 Residency** Access: RO _Custom_GTIReset: BUS C6 Residency MSB

GT Correctable Err Status Register

ERR_STAT_FUSA_GT_COR - GT Correctable Err Status Register

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	100160h

This register captures GT Correctable Errors for FuSa functionality.

DWord	Bit	Description
0	31:16	**Error Mask** Default Value: ffffh Access: R/W _Custom_GTIReset: BUS Bits for masking errors defined on bits 15:0 of ERR_STAT_FUSA_GT_COR. These bits are independent of the lower 16 bits -i.e a write to one of these bits is not required to set/clear the lower 16.
	15:1	**Reserved**
	0	**L3 Single Error Corrected** Default Value: 0b Access: R/WC _Custom_GTIReset: BUS L3 single error corrected

GTDRIVER_MAILBOX_DATA1

GTDRIVER_MAILBOX_DATA1 - GTDRIVER_MAILBOX_DATA1		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	13812Ch	
Data register for the GFX-DRIVER-to-PCODE mailbox. This mailbox is implemented as a means for tuning parameters for specific GFX workloads. This register is used in conjunction with GTDRIVER_MAILBOX_INTERFACE. THIS REGISTER IS DUPLICATED IN THE PCU I/O SPACE, XML CHANGES MUST BE MADE IN BOTH PLACES.		
DWord	**Bit**	**Description**
0	31:0	**Considerations**

GTDRIVER_MAILBOX_INTERFACE

GTDRIVER_MAILBOX_INTERFACE - GTDRIVER_MAILBOX_INTERFACE	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	138124h

Control and Status register for the GFX-DRIVER-to-PCODE mailbox. This mailbox is implemented as a means for tuning parameters for specific GFX workloads. This register is used in conjunction with GTDRIVER_MAILBOX_DATA. THIS REGISTER IS DUPLICATED IN THE PCU.

DWord	Bit	Description
0	31:0	**Considerations**

GTDRIVER_P2G_EVENTS

GTDRIVER_P2G_EVENTS - GTDRIVER_P2G_EVENTS	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	138160h

This extended capability allows PCODE to send an interrupt notification upon completion of a mailbox command. It is enabled via the GFX Driver Mailbox. PCODE will set the appropriate bit in this register to 1b, and will then write to 0.2.0.GTTMMADR.PIM[PCU_MBOXE]. The GFX Driver will clear the appropriate bit in this register by writing a 1 to the bit. THIS REGISTER IS DUPLICATED IN THE PCU I/O SPACE, XML CHANGES MUST BE MADE IN BOTH PLACES.

DWord	Bit	Description
0	31:0	**Considerations**

GT Engine Interrupt Enable

GT_ENG_INTR_ENABLE - GT Engine Interrupt Enable

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	190030h
ShortName:	RENDER_COPY_INTR_ENABLE
Address:	190034h
ShortName:	VIDEODECODE_VIDEOENHANCE_INTR_ENABLE
Address:	190038h
ShortName:	GUC_SCATTERGATHER_INTR_ENABLE
Address:	19003Ch
ShortName:	GPM_WGBOXPERF_INTR_ENABLE
Address:	190044h
ShortName:	GUNIT_CSME_INTR_ENABLE
Address:	190048h
ShortName:	CCS_RSVD_INTR_ENABLE

SW/FW programs this register to control interrupt events that are to be ignored (dropped).
Register content is saved/restored during RC6.
Bits in the registers described above are in the order: Engine1_Engine0_INTR_ENABLE.

Register Address	Engine 1	Engine 0	Structure defining bits
190030	Render	Copy	Render: Render Engine Interrupt Vector Copy: Blitter Interrupt Vector
190034	Vide Decode	Video Enhace	Video Decode: VideoDecoder Interrupt Vector Video Enhance: VideEnhancement Interupt Vector
190038	GuC	Scatter Gather	GuC: GUC Interrupt Vector Scatter Gather: Scatter Gather Interrupt Vector
19003C	GPM	WGBoxPerf	GPM: GTPM Interrupt Vector WGBoxPerf: WDBoxOAInterrupt Vector
190040	Reserved	Reserved	Reserved:
190044	GUnit	CSME	GUnit: G-Unit Interrupt Vector CSME: Manageability Engine Interrupt Vector
190048	CCS	Reserved	CCS: Compute CS Reserved

DWord	Bit	Description
0	31:16	**Engine1 Interrupt Enable** Default Value: 0000h Access: R/W

GT_ENG_INTR_ENABLE - GT Engine Interrupt Enable			
		_Custom_GTIReset:	BUS
	15:0	**Engine0 Interrupt Enable**	
		Default Value:	0000h
		Access:	R/W
		_Custom_GTIReset:	BUS

GT Engine Interrupt Mask

GT_ENG_INTR_MASK - GT Engine Interrupt Mask			
Register Space:	MMIO: 0/2/0		
Size (in bits):	32		
Address:	190090h		
ShortName:	RCS0_RSVD_INTR_MASK		
Address:	1900A0h		
ShortName:	BCS_RSVD_INTR_MASK		
Address:	1900A8h		
ShortName:	VCS0_VCS1_INTR_MASK		
Address:	1900ACh		
ShortName:	VCS2_VCS3_INTR_MASK		
Address:	1900B0h		
ShortName:	VCS4_VCS5_INTR_MASK		
Address:	1900B4h		
ShortName:	VCS6_VCS7_INTR_MASK		
Address:	1900D0h		
ShortName:	VECS0_VECS1_INTR_MASK		
Address:	1900D4h		
ShortName:	VECS2_VECS3_INTR_MASK		
Address:	1900E8h		
ShortName:	GUC_SCATTERGATHER_INTR_MASK		
Address:	1900ECh		
ShortName:	GPM_WGBOXPERF_INTR_MASK		
Address:	1900F4h		
ShortName:	GUNIT_CSME_INTR_MASK		
Address:	190100h		
ShortName:	CCS0_CCS1_INTR_MASK		
DWord	**Bit**	**Description**	
0	31:16	**Engine1 Interrupt Mask**	
		Default Value:	0000h
		Access:	R/W
		_Custom_GTIReset:	BUS

GT_ENG_INTR_MASK - GT Engine Interrupt Mask		
	15:0	**Engine0 Interrupt Mask** Default Value: 0000h Access: R/W _Custom_GTIReset: BUS

GT Fatal Err Status Register

ERR_STAT_FUSA_GT_FATAL - GT Fatal Err Status Register

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	100168h

This register captures GT Fatal Errors for FuSa functionality.

DWord	Bit	Description
0	31:16	**Error Mask** Default Value: ffffh Access: R/W _Custom_GTIReset: BUS Bits for masking errors defined on bits 15:0 of ERR_STAT_FUSA_GT_FATAL. These bits are independent of the lower 16 bits -i.e a write to one of these bits is not required to set/clear the lower 16.
	15:9	**Reserved**
	8	**IDI Parity Error** Default Value: 0b Access: R/WC _Custom_GTIReset: BUS IDI Parity Error
	7:6	**Reserved HW Error2** Access: R/WC _Custom_GTIReset: BUS
	5	**L3 ECC Checker Error** Default Value: 0b Access: R/WC _Custom_GTIReset: BUS L3 ECC Checker Error
	4	**L3 Double Error** Default Value: 0b Access: R/WC _Custom_GTIReset: BUS L3 double error
	3:2	**Reserved HW Error1** Access: R/WC _Custom_GTIReset: BUS

ERR_STAT_FUSA_GT_FATAL - GT Fatal Err Status Register

	1	**Array Bist Error** Default Value: 0b Access: R/WC _Custom_GTIReset: BUS Array BIST Error
	0	**Reserved HW Error0** Access: R/WC _Custom_GTIReset: BUS

GT Function Level Reset Control Message

FLRCTLMSG - GT Function Level Reset Control Message

Register Space:	MMIO: 0/2/0
Size (in bits):	32
CrashLogSaved:	true
CrashLogVisibility:	cspec
Address:	08100h

GT Reset Control Register

DWord	Bit	Description
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask In order to write to bits 15:0, the corresponding message mask bits must be written. For example, for bit 14 to be set, bit 30 needs to be 1 : 40004000
	15:2	**Reserved** Access: RO Format: MBZ
	1	**Reserved** Access: RO Format: MBZ
	0	**Initiate GT Function Level Reset Message** Access: R/W Set _Custom_GTIReset: BUS GT Function Level Reset (FLR) 1: Initiate GT FLR - This is a Non-Posted message to reset Render, Media, Blitter and GTI-Device domains. - This bit is cleared by the CPunit upon completion of the reset.

GTICP BONUS1 Reg

GTICPBONUS1 - GTICP BONUS1 Reg

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 24014h

Clock Gating Messages Register

DWord	Bit	Description
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask In order to write to bits 15:0, the corresponding message mask bits must be written. For example, for bit 14 to be set, bit 30 needs to be 1 : 40004000
	15:8	**Reserved** Access: RO Format: MBZ
	7	**BONUS BIT 7** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	6	**BONUS BIT 6** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	5	**BONUS BIT 5** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	4	**BONUS BIT 4** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request

GTICPBONUS1 - GTICP BONUS1 Reg

		'1' : Initiate Power UP req
	3	**BONUS BIT 3** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	2	**BONUS BIT 2** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	1	**BONUS BIT 1** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**BONUS BIT 0** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

GTICP BONUS2 Reg

GTICPBONUS2 - GTICP BONUS2 Reg

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24018h

Clock Gating Messages Register

DWord	Bit	Description
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask In order to write to bits 15:0, the corresponding message mask bits must be written. For example, for bit 14 to be set, bit 30 needs to be 1 : 40004000
	15:8	**Reserved** Access: RO Format: MBZ
	7	**BONUS BIT 7** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	6	**BONUS BIT 6** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	5	**BONUS BIT 5** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)

GTICPBONUS2 - GTICP BONUS2 Reg

	4	**BONUS BIT 4** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	3	**BONUS BIT 3** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	2	**BONUS BIT 2** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	1	**BONUS BIT 1** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**BONUS BIT 0** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

GT Interrupt DW0

GT_INTR_DW0 - GT Interrupt DW0

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 190018h

Engine bits in this register are set if any of the unmasked bits in the underlying engine 16b interrupt vector is non-zero

DWord	Bit	Description
0	31	**CSME** Access: R/W One Clear
	30	**Reserved** Access: RO Format: MBZ
	29	**Reserved** Access: RO Format: MBZ
	28	**GUNIT** Access: R/W One Clear
	27:26	**Reserved** Access: RO Format: MBZ
	25	**GUC** Access: R/W One Clear
	24	**Reserved** Access: RO Format: MBZ
	23	**SCTRGTHR** Access: R/W One Clear
	22	**Reserved** Access: RO Format: MBZ
	21	**Reserved** Access: RO Format: MBZ

GT_INTR_DW0 - GT Interrupt DW0

	20	**WDPERF** Access: R/W One Clear
	19	**KCR** Access: R/W One Clear
	18	**Reserved** Access: RO Format: MBZ
	17	**Reserved** Access: RO Format: MBZ
	16	**GTPM** Access: R/W One Clear
	15	**BCS** Access: R/W One Clear
	14:10	**Reserved**
	9:8	**Reserved** Access: RO Format: MBZ
	7	**CCS3** Access: R/W One Clear
	6	**CCS2** Access: R/W One Clear
	5	**CCS1** Access: R/W One Clear
	4	**CCS0** Access: R/W One Clear
	3:1	**Reserved** Access: RO Format: MBZ
	0	**RCS0** Access: R/W One Clear

GT Interrupt DW1

GT_INTR_DW1 - GT Interrupt DW1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	19001Ch	
Interrupt bits indicating one of theunderlying engine interrupts is non-zero		
DWord	**Bit**	**Description**
0	31	**VECS0** Access: R/W One Clear
	30	**VECS1** Access: R/W One Clear
	29	**VECS2** Access: R/W One Clear
	28	**VECS3** Access: R/W One Clear
	27:8	**Reserved** Access: RO Format: MBZ
	7	**VCS7** Access: R/W One Clear
	6	**VCS6** Access: R/W One Clear
	5	**VCS5** Access: R/W One Clear
	4	**VCS4** Access: R/W One Clear
	3	**VCS3** Access: R/W One Clear
	2	**VCS2** Access: R/W One Clear
	1	**VCS1** Access: R/W One Clear
	0	**VCS0** Access: R/W One Clear

GT Interrupt Identity

GT_INTR_IDENTITY - GT Interrupt Identity

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	190060h
ShortName:	INTR_IDENTITY_REG0
Address:	190064h
Name:	INTR_IDENTITY_REG1
ShortName:	INTR_IDENTITY_REG1

HW displays the interrupt bits for engine chosen using Selector.
Only unmasked interrupts are displayed. Masked interrupts continue to accumulate behind the mask.
After processing, SW shall write 1's to clear. (Bit 31 must be cleared by SW)
Write to Clear indicates to HW that processing is complete (for displayed interrupts).

DWord	Bit	Description
0	31	**Data Valid** Access: R/W
	30:26	**Reserved** Access: RO Format: MBZ
	25:20	**Engine Instance ID** Engine Instance ID format is defined in structure "Engine ID Definition"
	19	**Reserved** Access: RO Format: MBZ
	18:16	**Engine Class ID** Engine class is defined in structure "Engine ID Definition"
	15:0	**Engine Interrupt** Access: R/W Format is specific to the engine that is sending the interrupt. Format is defined in structure "EngineInterrupt Vector" (where engine isBlitter/G-Unit/GTPM/GuC/Render Engine/Video Decoder/VideoEnhancement).

GT Interrupt IIR Selector

GT_INTR_IIR_SELECTOR - GT Interrupt IIR Selector

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	190070h
ShortName:	IIR_REG0_SELECTOR
Address:	190074h
ShortName:	IIR_REG1_SELECTOR

This is a basic register template

DWord	Bit	Description
0	31:0	**Engine ID** Access: R/W SW/FW shall program the appropriate Engine ID to view the interrupts from an engine Engine_ID is a one-hot encoding that follows the bit definition of GuC GT Interrupt DW register. Register: IIR_REG0_SELECTOR — Bit Definiition: Format: GT Interrupt DW0 Register: IIR_REG1_SELECTOR — Bit Definiition: Format: GT Interrupt DW1

GTI PGFET control register with lock

GTIPFETCTL - GTI PGFET control register with lock

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 24008h

DWord	Bit	Description
0	31	**PFET Control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of MEDIA1 PGFETCTL register are R/W 1 = All bits of MEDIA1 PGFETCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:26	**Reserved** Access: RO Format: MBZ
	25	**Leave firewall disabled** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not firewall the gated domain for a power down flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e firewall gated domain to ungated domain crossing during power down flows 1 = Leave firewall disabled, i.e dont firewall the gated domain, but complete logical flow
	24	**Leave FET On** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not turn off the PFET eventhough it will complete the flow with PM Encodings: 0 = Default mode, i.e power off fets during power down flows 1 = Leave ON mode, i.e dont power off pfet, but complete logical flow
	23	**Power Well Status** Access: RO _Custom_GTIReset: BUS 0 = Well is powered Down 1 = Well is powered up

GTIPFETCTL - GTI PGFET control register with lock

	Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
22	**Reserved** Access: RO Format: MBZ
21:19	**Delay from enabling secondary PFETs to power good.** Default Value: 110b Access: R/W Lock _Custom_GTIReset: BUS Delay from enabling secondary PFETs to power good 3'b000: 40ns 3'b001: 80ns 3'b010: 160ns 3'b011: 240ns 3'b100: 320ns 3'b101: 480ns 3'b110: 640ns 3'b111: 1280ns
18:16	**Strobe pulse period** Access: R/W Lock _Custom_GTIReset: BUS Time period b/w two adjacent strobes to the primary FETs 3'b000: 10ns (or 1 bclk) 3'b001: 20ns (or 2 bclk) 3'b010: 30ns (or 3 bclk) 3'b111: 80ns (or 8 bclk) Value: 001b — Name: **[Default]**
15:0	**PFET Ladder Step Sequence** Access: R/W Lock _Custom_GTIReset: BUS PFET Ladder STEP sequence The PFET ladder has 16 steps, each represented by its corresponding bit in pfetlddrstepseq[15:0]. If the bit location is set, it means we must step on that ladder stage before we go to the next stage. The SPC controller will step through all the steps with asserted bits periodically with period equal to strbpulsprd[2:0] Programming Rule: pfetldrstepseq[15] and pfetldrstepseq[0] must be '1; A '0 setting for these bits is illegal. 15'FFFFh: Ladder step (ladder_sel) goes 0, 1, 2, ?.15.

GTIPFETCTL - GTI PGFET control register with lock

15'FFF1h: Ladder step goes 0, 4, 5, 6,?15; Steps 1, 2, 3 are skipped.
15'D555h: Ladder step goes 0, 2, 4, 6, 8, 10, 12, 14, 15; Steps 1, 3, 5, 7, 9, 11, 13 are skipped.
15'80001h: Ladder step goes 0, 15; Steps 1, 2, 3, 4, ?14 are skipped.

Value	Name
1000011111111001b	**[Default]**

GTI Power Gate Control Request

GTIPGCTLREQ - GTI Power Gate Control Request

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24000h

Clock Gating Messages Register

DWord	Bit	Description
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask In order to write to bits 15:0, the corresponding message mask bits must be written. For example, for bit 14 to be set, bit 30 needs to be 1 : 40004000
	15:1	**Reserved** Access: RO Format: MBZ
	0	**Power Gate Request** Access: R/W _Custom_GTIReset: BUS Media1 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

GT Mode Register

GT_MODE - GT Mode Register

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	07008h
Name:	GT Mode Register
ShortName:	GT_MODE

This Register is used to control the 6EU and 12EU configuration for GT. Writing 0x01FF01FF to this register enables the 6EU mode.

DWord	Bit	Description
0	31:16	**Mask** Access: WO Format: Mask _Custom_GTIReset: DEV Must be set to modify corresponding bit in Bits 15:0. (All implemented bits)
	15	**EU Local Thread Checking Enable** Access: R/W _Custom_GTIReset: DEV This field configures the EU local thread checking. If enable the stateless access will be checked against the local thread's scratch space size and start address. Value / Name / Description: 0h / Disable **[Default]** / EU local thread checking is disabled. 1h / Enable / EU local thread checking is enabled.
	14:13	**SFR mode** Access: R/W Format: U2 _Custom_GTIReset: DEV This field must be zero when not in GT4(SFR) configuration i.e GTB_rendermode fuse set to SFR.
	12:11	**Reserved** Access: R/W Format: PBC _Custom_GTIReset: DEV

GT_MODE - GT Mode Register

10 HW Binding Table Alignment

Access:	R/W
Format:	Disable
_Custom_GTIReset:	DEV

This bit changes the format of the binding table pointer.

When this bit is set, the format of the binding table is SW binding table format whether Binding Table Pool is enabled or disabled.

Value	Name	Description
0h	Legacy **[Default]**	Binding table pointer 15:5 Maps to 15:5 for INTERFACE_DESCRIPTOR DATA. Binding table pointer 15:5 Maps to 15:5 for 3DSTATE_BINDING_TABLE_POINTER_* if Binding Table Pool is disabled. Binding table pointer 15:5 Maps to 16:6 for 3DSTATE_BINDING_TABLE_POINTER_* if Binding Table Pool is enabled.
1h	Enable 512KB Binding Table size	Binding table pointer maps to 18:8 for both 3DSTATE_BINDING_TABLE_POINTER_* and INTERFACE_DESCRIPTOR DATA.

9 Reserved

Access:	R/W
Format:	PBC
_Custom_GTIReset:	DEV

8 Reserved

Access:	R/W
Format:	PBC
_Custom_GTIReset:	DEV

7 Reserved

Access:	RO
Format:	MBZ

6 Reserved

Access:	R/W
Format:	PBC
_Custom_GTIReset:	DEV

GT_MODE - GT Mode Register

5:4 **Slice2 IZ Hashing: 7 EU subslice encoding**

Access:	R/W
_Custom_GTIReset:	DEV

These bits control 3-way sub-slice hashing by conveying which sub-slice has 7 EUs.

Value	Name	Description
0h	**[Default]**	All subslices have equal number of EUs.
1h		Subslice 2 has 7 EUs.
2h		Subslice 1 has 7 EU.
3h		Subslice 0 has 7 EUs.

Programming Notes
SW must program these bits based on EU Disable Fuses in Slice 2.

3:2 **Slice1 IZ Hashing: 7 EU subslice encoding**

Access:	R/W
_Custom_GTIReset:	DEV

These bits control 3-way sub-slice hashing by conveying which sub-slice has 7 EUs.

Value	Name	Description
0h	**[Default]**	All subslices have equal number of EUs.
1h		Subslice 2 has 7 EUs.
2h		Subslice 1 has 7 EUs.
3h		Subslice 0 has 7 EUs.

Programming Notes
SW must program these bits based on EU Disable Fuses in Slice 1.

1:0 **Slice 0 IZ Hashing: 7 EU subslice encoding**

Access:	R/W
_Custom_GTIReset:	DEV

These bits control 3-way sub-slice hashing by conveying which sub-slice has 7 EUs.

Value	Name	Description
0h	**[Default]**	All subslices have equal number of EUs.
1h		Subslice 2 has 7 EUs.
2h		Subslice 1 has 7 EUs.
3h		Subslice0 has 7 EUs.

Programming Notes
SW must program these bits based on EU Disable Fuses in Slice 0.

GT Non Fatal Err Status Register

ERR_STAT_FUSA_GT_NONFATAL - GT Non Fatal Err Status Register

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	100164h

This register captures GT Non-Fatal errors for FuSa functionality

DWord	Bit	Description
0	31:16	**Error Mask** Default Value: ffffh Access: R/W _Custom_GTIReset: BUS bits for masking errors defined on bit 15:0 of ERR_STAT_FUSA_DISP_NONFATAL. These bits are independent of the lower 16 bits -i.e a write to one of these bits is not required to set/clear the lower 16.
	15:0	**Reserved**

GTPO Triggering Block 1 Mask A

GTPO_TRIGGERING_BLOCK_1_MASK_A - GTPO Triggering Block 1 Mask A

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
Address:	0D214h-0D21Bh

Primary 64bit Mask Value for GTPO Triggering Block 1

DWord	Bit	Description
0..1	63:32	**GTPO TRIGGERING MASK A UPPER** Access: R/W _Custom_GTIReset: BUS Lower 32bit Mask Value for the value programmed in the corresponding match register
	31:0	**GTPO TRIGGERING MASK A LOWER** Access: R/W _Custom_GTIReset: BUS Lower 32bit Mask Value for the value programmed in the corresponding match register

GTPO Triggering Block 1 Mask B

GTPO_TRIGGERING_BLOCK_1_MASK_B - GTPO Triggering Block 1 Mask B	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
Address:	0D21Ch-0D223h
Secondary 64bit Mask Value for GTPO Triggering Block 1	

DWord	Bit	Description
0..1	63:32	**GTPO TRIGGERING Mask B Upper** Access: R/W _Custom_GTIReset: BUS Upper 32bit Mask Value for the value programmed in the corresponding match register
	31:0	**GTPO TRIGGERING Mask B Lower** Access: R/W _Custom_GTIReset: BUS Lower 32bit Mask Value for the value programmed in the corresponding match register

GTPO Triggering Block 1 Match A

GTPO_TRIGGERING_BLOCK1_MATCHA - GTPO Triggering Block 1 Match A

Register Space:	MMIO: 0/2/0
Size (in bits):	64
Address:	0D224h-0D22Bh

Primary 64bit Match Value for GTPO Triggering Block 1

DWord	Bit	Description
0..1	63:32	**Match A Upper** Access: R/W _Custom_GTIReset: BUS Upper 32bit match value for use by GTPO triggering block
	31:0	**Macth A Lower** _Custom_GTIReset: BUS Lower 32bit match value for use by GTPO triggering block

GTPO Triggering Block 1 Match B

GTPO_TRIGGERING_BLOCK_1_MATCH_B - GTPO Triggering Block 1 Match B

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
Address:	0D22Ch-0D233h

Secondary 64bit Match Value for GTPO Triggering Block 1

DWord	Bit	Description
0..1	63:32	**GTPO TRIGGERING Match B Upper** Access: R/W _Custom_GTIReset: BUS Upper 32bit match value for use by GTPO triggering block
	31:0	**GTPO Triggering Match B Lower** Access: R/W _Custom_GTIReset: BUS Lower 32bit match value for use by GTPO triggering block

GTPO Triggering Block Mode Enable

GTPO_TRIGGERING_BLOCK_MODE - GTPO Triggering Block Mode Enable

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	0D234h

Register to enable and set the mode for 4 GTPO triggering blocks

DWord	Bit	Description
0	31:3	**GTPO BLOCK MODE ENABLE RSVD** Access: R/W _Custom_GTIReset: BUS
	2:1	**GTPO Triggering Block 1 Mode** Default Value: 00b Access: R/W _Custom_GTIReset: BUS 01 - Match; 10 - Match Secondary; 11 - Match Content;
	0	**GTPO Triggering Block 1 Enable** Default Value: 0b _Custom_GTIReset: BUS

GTSCRATCH

GTSCRATCH	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	4F100h-4F11Fh
Name:	GT SCRATCH
ShortName:	GTSCRATCH_*

There are 8 instances of this register format.

Restriction
These registers are used by hardware and must not be used by software.

DWord	Bit	Description
0	31:0	**GT Sratchpad** Default Value: 00000000000000000000000000000000b Access: R/W _Custom_GTIReset: BUS GT Scratchpad

GTSP

GTSP	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	130040h-13005Fh
Name:	GT SCRATCHPAD
ShortName:	GTSP_*

There are 8 instances of this register format.

DWord	Bit	Description
0	31:0	**GT Scratchpad** Default Value: 00000000000000000000000000000000b Access: R/W _Custom_GTIReset: BUS GT Scratchpad

GT Virtual Function Engine Interrupt Enable

GT_ENG_INTR_ENABLE - GT Virtual Function Engine Interrupt Enable	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	191030h
ShortName:	VF1_RENDER_COPY_INTR_ENABLE
Address:	192030h
ShortName:	VF2_RENDER_COPY_INTR_ENABLE
Address:	193030h
ShortName:	VF3_RENDER_COPY_INTR_ENABLE
Address:	194030h
ShortName:	VF4_RENDER_COPY_INTR_ENABLE
Address:	195030h
ShortName:	VF5_RENDER_COPY_INTR_ENABLE
Address:	196030h
ShortName:	VF6_RENDER_COPY_INTR_ENABLE
Address:	197030h
ShortName:	VF7_RENDER_COPY_INTR_ENABLE
Address:	191034h
ShortName:	VF1_VIDEODECODE_VIDEOENHANCE_INTR_ENABLE
Address:	192034h
ShortName:	VF2_VIDEODECODE_VIDEOENHANCE_INTR_ENABLE
Address:	193034h
ShortName:	VF3_VIDEODECODE_VIDEOENHANCE_INTR_ENABLE
Address:	194034h
ShortName:	VF4_VIDEODECODE_VIDEOENHANCE_INTR_ENABLE
Address:	195034h
ShortName:	VF5_VIDEODECODE_VIDEOENHANCE_INTR_ENABLE
Address:	196034h
ShortName:	VF6_VIDEODECODE_VIDEOENHANCE_INTR_ENABLE
Address:	197034h
ShortName:	VF7_VIDEODECODE_VIDEOENHANCE_INTR_ENABLE
Address:	191038h

GT_ENG_INTR_ENABLE - GT Virtual Function Engine Interrupt Enable

ShortName:	VF1_GUC_RESVD_INTR_ENABLE
Address:	192038h
ShortName:	VF2_GUC_RESVD_INTR_ENABLE
Address:	193038h
ShortName:	VF3_GUC_RESVD_INTR_ENABLE
Address:	194038h
ShortName:	VF4_GUC_RESVD_INTR_ENABLE
Address:	195038h
ShortName:	VF5_GUC_RESVD_INTR_ENABLE
Address:	196038h
ShortName:	VF6_GUC_RESVD_INTR_ENABLE
Address:	197038h
ShortName:	VF7_GUC_RESVD_INTR_ENABLE
Address:	191048h
ShortName:	VF1_CCS_RSVD_INTR_ENABLE
Address:	192048h
ShortName:	VF2_CCS_RSVD_INTR_ENABLE
Address:	193048h
ShortName:	VF3_CCS_RSVD_INTR_ENABLE
Address:	194048h
ShortName:	VF4_CCS_RSVD_INTR_ENABLE
Address:	195048h
ShortName:	VF5_CCS_RSVD_INTR_ENABLE
Address:	196048h
ShortName:	VF6_CCS_RSVD_INTR_ENABLE
Address:	197048h
ShortName:	VF7_CCS_RSVD_INTR_ENABLE

DWord	Bit	Description
0	31:16	**Engine1 Interrupt Enable** Access: R/W
	15:0	**Engine0 Interrupt Enable** Access: R/W

GT Virtual Function Engine Interrupt Mask

GT Virtual Function Engine Interrupt Mask	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	191090h
ShortName:	VF1_RCS0_RSVD_INTR_MASK
Address:	192090h
ShortName:	VF2_RCS0_RSVD_INTR_MASK
Address:	193090h
ShortName:	VF3_RCS0_RSVD_INTR_MASK
Address:	194090h
ShortName:	VF4_RCS0_RSVD_INTR_MASK
Address:	195090h
ShortName:	VF5_RCS0_RSVD_INTR_MASK
Address:	196090h
ShortName:	VF6_RCS0_RSVD_INTR_MASK
Address:	197090h
ShortName:	VF7_RCS0_RSVD_INTR_MASK
Address:	1910A0h
ShortName:	VF1_BCS_RSVD_INTR_MASK
Address:	1920A0h
ShortName:	VF2_BCS_RSVD_INTR_MASK
Address:	1930A0h
ShortName:	VF3_BCS_RSVD_INTR_MASK
Address:	1940A0h
ShortName:	VF4_BCS_RSVD_INTR_MASK
Address:	1950A0h
ShortName:	VF5_BCS_RSVD_INTR_MASK
Address:	1960A0h
ShortName:	VF6_BCS_RSVD_INTR_MASK
Address:	1970A0h
ShortName:	VF7_BCS_RSVD_INTR_MASK
Address:	1910A8h
ShortName:	VF1_VCS0_VCS1_INTR_MASK

GT Virtual Function Engine Interrupt Mask	
Address:	1920A8h
ShortName:	VF2_VCS0_VCS1_INTR_MASK
Address:	1930A8h
ShortName:	VF3_VCS0_VCS1_INTR_MASK
Address:	1940A8h
ShortName:	VF4_VCS0_VCS1_INTR_MASK
Address:	1950A8h
ShortName:	VF5_VCS0_VCS1_INTR_MASK
Address:	1960A8h
ShortName:	VF6_VCS0_VCS1_INTR_MASK
Address:	1970A8h
ShortName:	VF7_VCS0_VCS1_INTR_MASK
Address:	1910ACh
ShortName:	VF1_VCS2_VCS3_INTR_MASK
Address:	1920ACh
ShortName:	VF2_VCS2_VCS3_INTR_MASK
Address:	1930ACh
ShortName:	VF3_VCS2_VCS3_INTR_MASK
Address:	1940ACh
ShortName:	VF4_VCS2_VCS3_INTR_MASK
Address:	1950ACh
ShortName:	VF5_VCS2_VCS3_INTR_MASK
Address:	1960ACh
ShortName:	VF6_VCS2_VCS3_INTR_MASK
Address:	1970ACh
ShortName:	VF7_VCS2_VCS3_INTR_MASK
Address:	1910B0h
ShortName:	VF1_VCS4_VCS5_INTR_MASK
Address:	1920B0h
ShortName:	VF2_VCS4_VCS5_INTR_MASK
Address:	1930B0h
ShortName:	VF3_VCS4_VCS5_INTR_MASK
Address:	1940B0h
ShortName:	VF4_VCS4_VCS5_INTR_MASK

GT Virtual Function Engine Interrupt Mask

Address:	1950B0h
ShortName:	VF5_VCS4_VCS5_INTR_MASK
Address:	1960B0h
ShortName:	VF6_VCS4_VCS5_INTR_MASK
Address:	1970B0h
ShortName:	VF7_VCS4_VCS5_INTR_MASK
Address:	1910B4h
ShortName:	VF1_VCS6_VCS7_INTR_MASK
Address:	1920B4h
ShortName:	VF2_VCS6_VCS7_INTR_MASK
Address:	1930B4h
ShortName:	VF3_VCS6_VCS7_INTR_MASK
Address:	1940B4h
ShortName:	VF4_VCS6_VCS7_INTR_MASK
Address:	1950B4h
ShortName:	VF5_VCS6_VCS7_INTR_MASK
Address:	1960B4h
ShortName:	VF6_VCS6_VCS7_INTR_MASK
Address:	1970B4h
ShortName:	VF7_VCS6_VCS7_INTR_MASK
Address:	1910D0h
ShortName:	VF1_VECS0_VECS1_INTR_MASK
Address:	1920D0h
ShortName:	VF2_VECS0_VECS1_INTR_MASK
Address:	1930D0h
ShortName:	VF3_VECS0_VECS1_INTR_MASK
Address:	1940D0h
ShortName:	VF4_VECS0_VECS1_INTR_MASK
Address:	1950D0h
ShortName:	VF5_VECS0_VECS1_INTR_MASK
Address:	1960D0h
ShortName:	VF6_VECS0_VECS1_INTR_MASK
Address:	1970D0h
ShortName:	VF7_VECS0_VECS1_INTR_MASK

GT Virtual Function Engine Interrupt Mask	
Address:	1910D4h
ShortName:	VF1_VECS2_VECS3_INTR_MASK
Address:	1920D4h
ShortName:	VF2_VECS2_VECS3_INTR_MASK
Address:	1930D4h
ShortName:	VF3_VECS2_VECS3_INTR_MASK
Address:	1940D4h
ShortName:	VF4_VECS2_VECS3_INTR_MASK
Address:	1950D4h
ShortName:	VF5_VECS2_VECS3_INTR_MASK
Address:	1960D4h
ShortName:	VF6_VECS2_VECS3_INTR_MASK
Address:	1970D4h
ShortName:	VF7_VECS2_VECS3_INTR_MASK
Address:	1910E8h
ShortName:	VF1_GUC_RESVD_INTR_MASK
Address:	1920E8h
ShortName:	VF2_GUC_RESVD_INTR_MASK
Address:	1930E8h
ShortName:	VF3_GUC_RESVD_INTR_MASK
Address:	1940E8h
ShortName:	VF4_GUC_RESVD_INTR_MASK
Address:	1950E8h
ShortName:	VF5_GUC_RESVD_INTR_MASK
Address:	1960E8h
ShortName:	VF6_GUC_RESVD_INTR_MASK
Address:	1970E8h
ShortName:	VF7_GUC_RESVD_INTR_MASK
Address:	191100h
ShortName:	VF1_GPGPU1_GPGPU2_INTR_MASK
SW/FW programs this register to mask interrupt events to GuC. Register content is saved/restored during RC6. Bits in the registers described above are in the order: Engine1_Engine0_INTR_MASK. For e.g: Engine1 houses bits for: RCS0, BCS, VCS0,... Engine0 houses bits for: RSVD, RSVD, VCS1,...	

GT Virtual Function Engine Interrupt Mask

Register Address	Engine 1	Engine 0	Structure defining bits
190090	RCS0	Reserved	Format: Render Engine Interrupt Vector
1900A0	BCS	Reserved	Format: Blitter Interrupt Vector
1900A8 1900AC 1900B0 1900B4	VCS0 VCS2 VCS4 VCS6	VCS1 VCS3 VCS5 VCS7	Format: VideoDecoder Interrupt Vector
1900D0 1900D4	VECS0 VECS2	VECS1 VECS3	Format: VideEnhancement Interrupt Vector
1900E8	GuC	Scatter Gather	GuC: GUC Interrupt Vector Scatter Gather: Scatter Gather Interrupt Vector
1900EC	GPM	WGBoxPerf	GPM: GTPM Interrupt Vector WGBoxPerf: WDBoxOAInterrupt Vector
1900F0		Reserved	Reserved:
1900F4	GUnit	CSME	GUnit: G-Unit Interrupt Vector CSME: Manageability Engine Interrupt Vector

DWord	Bit	Description
0	31:16	**Engine1 Interrupt Mask** Access: R/W
	15:0	**Engine0 Interrupt Mask** Access: R/W

GT Virtual Function IIR Selector

GT_VF_INTR_IIR_SELECTOR - GT Virtual Function IIR Selector	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	191070h
ShortName:	VF1_IIR_REG0_SELECTOR
Address:	191074h
ShortName:	VF1_IIR_REG1_SELECTOR
Address:	192070h
ShortName:	VF2_IIR_REG0_SELECTOR
Address:	192074h
ShortName:	VF2_IIR_REG1_SELECTOR
Address:	193070h
ShortName:	VF3_IIR_REG0_SELECTOR
Address:	193074h
ShortName:	VF3_IIR_REG1_SELECTOR
Address:	194070h
ShortName:	VF4_IIR_REG0_SELECTOR
Address:	194074h
ShortName:	VF4_IIR_REG1_SELECTOR
Address:	195070h
ShortName:	VF5_IIR_REG0_SELECTOR
Address:	195074h
ShortName:	VF5_IIR_REG1_SELECTOR
Address:	196070h
ShortName:	VF6_IIR_REG0_SELECTOR
Address:	196074h
ShortName:	VF6_IIR_REG1_SELECTOR
Address:	197070h
ShortName:	VF7_IIR_REG0_SELECTOR
Address:	197074h
ShortName:	VF7_IIR_REG1_SELECTOR
This is a basic register template	

DWord	Bit	Description

GT_VF_INTR_IIR_SELECTOR - GT Virtual Function IIR Selector

0	31:0	**Engine ID** Access: R/W SW/FW shall program the appropriate Engine ID to view the interrupts from an engine Engine_ID is a one-hot encoding that follows the bit definition of GuC GT Interrupt DW register. Register: IIR_REG0_SELECTOR — Bit Definiition: Format: GT Interrupt DW0 Register: IIR_REG1_SELECTOR — Bit Definiition: Format: GT Interrupt DW1

GT Virtual Function Interrupt DW0

GT_VF_INTR_DW0 - GT Virtual Function Interrupt DW0	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	191018h
Name:	VF1_INTR_DW0
ShortName:	VF1_INTR_DW0
Address:	192018h
Name:	VF2_INTR_DW0
ShortName:	VF2_INTR_DW0
Address:	193018h
Name:	VF3_INTR_DW0
ShortName:	VF3_INTR_DW0
Address:	194018h
Name:	VF4_INTR_DW0
ShortName:	VF4_INTR_DW0
Address:	195018h
Name:	VF5_INTR_DW0
ShortName:	VF5_INTR_DW0
Address:	196018h
Name:	VF6_INTR_DW0
ShortName:	VF6_INTR_DW0
Address:	197018h
Name:	VF7_INTR_DW0
ShortName:	VF7_INTR_DW0

Bits set in this register indicate if an engine has an interrupt that requires servicing.

DWord	Bit	Description
0	31	**CSME** Access: R/W
	30:29	**Reserved** Access: RO Format: MBZ
	28	**GUNIT** Access: R/W

GT_VF_INTR_DW0 - GT Virtual Function Interrupt DW0

	27:26	**Reserved** Access: RO Format: MBZ
	25	**GUC** Access: R/W
	24	**Reserved** Access: RO Format: MBZ
	23	**SCTRGTHR** Access: R/W
	22:21	**Reserved** Access: RO Format: MBZ
	20	**WDPERF** Access: R/W
	19	**KCR** Access: R/W
	18:17	**Reserved** Access: RO Format: MBZ
	16	**GTPM** Access: R/W
	15	**BCS** Access: R/W
	14:8	**Reserved** Access: RO Format: MBZ
	7	**CCS3**
	6	**CCS2**
	5	**CCS1**
	4	**CCS0**
	3:1	**Reserved** Access: RO Format: MBZ
	0	**RCS0** Access: R/W

GT Virtual Function Interrupt DW1

GT_VF_INTR_DW1 - GT Virtual Function Interrupt DW1

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	19101Ch
ShortName:	VF1_INTR_DW1
Address:	19201Ch
ShortName:	VF2_INTR_DW1
Address:	19301Ch
ShortName:	VF3_INTR_DW1
Address:	19401Ch
ShortName:	VF4_INTR_DW1
Address:	19501Ch
ShortName:	VF5_INTR_DW1
Address:	19601Ch
ShortName:	VF6_INTR_DW1
Address:	19701Ch
ShortName:	VF7_INTR_DW1

Interrupt bits indicating one of theunderlying engine interrupts is non-zero

DWord	Bit	Description
0	31	**VECS0** Access: R/W
	30	**VECS1** Access: R/W
	29	**VECS2** Access: R/W
	28	**VECS3** Access: R/W
	27:8	**Reserved** Access: RO Format: MBZ
	7	**VCS7** Access: R/W
	6	**VCS6** Access: R/W

GT_VF_INTR_DW1 - GT Virtual Function Interrupt DW1		
	5	**VCS5** Access: R/W
	4	**VCS4** Access: R/W
	3	**VCS3** Access: R/W
	2	**VCS2** Access: R/W
	1	**VCS1** Access: R/W
	0	**VCS0** Access: R/W

GT Virtual Function Interrupt Identity

GT_VF_INTR_IDENTITY - GT Virtual Function Interrupt Identity	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	191060h
ShortName:	VF1_INTR_IDENTITY_REG0
Address:	191064h
ShortName:	VF1_INTR_IDENTITY_REG1
Address:	192060h
ShortName:	VF2_INTR_IDENTITY_REG0
Address:	192064h
ShortName:	VF2_INTR_IDENTITY_REG1
Address:	193060h
ShortName:	VF3_INTR_IDENTITY_REG0
Address:	193064h
ShortName:	VF3_INTR_IDENTITY_REG1
Address:	194060h
ShortName:	VF4_INTR_IDENTITY_REG0
Address:	194064h
ShortName:	VF4_INTR_IDENTITY_REG1
Address:	195060h
ShortName:	VF5_INTR_IDENTITY_REG0
Address:	195064h
ShortName:	VF5_INTR_IDENTITY_REG1
Address:	196060h
ShortName:	VF6_INTR_IDENTITY_REG0
Address:	196064h
ShortName:	VF6_INTR_IDENTITY_REG1
Address:	197060h
ShortName:	VF7_INTR_IDENTITY_REG0
Address:	197064h
ShortName:	VF7_INTR_IDENTITY_REG1
HW displays the interrupt bits for engine chosen using Selector. Only unmasked interrupts are displayed. Masked interrupts continue to accumulate behind the mask.	

GT_VF_INTR_IDENTITY - GT Virtual Function Interrupt Identity

After processing, SW shall write 1's to clear. (Bit 31 must be cleared by SW)
Write to Clear indicates to HW that processing is complete (for displayed interrupts).

DWord	Bit	Description
0	31	**Data Valid** Access: R/W
	30:26	**Reserved** Access: RO Format: MBZ
	25:20	**Engine Instance ID**
	19	**Reserved** Access: RO Format: MBZ
	18:16	**Engine Class ID**
	15:0	**Engine Interrupt** Access: R/W

GUC_HOST_INTR_IIR

GUC_HOST_INTR_IIR - GUC_HOST_INTR_IIR

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	1901F0h

This register provides a capability for a Host SW to communicate with GuC FW. The Host SW, whether Host OS/VMM or a Guest VM, writes to this register at MMIO offset 1901F0h with a 32-bit payload, which results in the Gunit sending an interrupt to GuC, which can then retrieve the payload from this register using the appropriate Gunit unique address.

DWord	Bit	Description
0	31:0	**Data** Default Value: 00000000h Access: R/W _Custom_GTIReset: BUS Data

GuC DMA Interrupt Input

GUC_DMA_IIR - GuC DMA Interrupt Input

Register Space: MMIO: 0/2/0
Access: R/W
Size (in bits): 32

DWord	Bit	Description
0	31:2	**Reserved** Access: RO Format: MBZ
	1	**DMA Service Interrupt Routing** 0h – Host **[Default]** – Interrupts will be routed to the host CPU. 1h – Reserved – Reserved
	0	**DMA Interrupt** There can only be one DMA workload outstanding at any time. Set by DMA hardware. Cleared by Host. 0h – No DMA Interrupt Pending **[Default]** 1h – DMA Interrupt Pending

GuC Doorbell Group 0 Interrupt Status

GUC_DB_ISR_0 - GuC Doorbell Group 0 Interrupt Status	
Register Space:	MMIO: 0/2/0
Access:	R/WC
Size (in bits):	32
This is a "shadow" of registers in GTI. The GuC takes a snapshot when it services them. Access: copy-write by the GuC hardware, read and cleared by the uKernel and host CPU.	

DWord	Bit	Description
0	31:0	**Interrupt Status** GuC shadow copy of the interrupt status mask for doorbells 0 through 31.

GuC Doorbell Group 1 Interrupt Status

GUC_DB_ISR_1 - GuC Doorbell Group 1 Interrupt Status	
Register Space:	MMIO: 0/2/0
Access:	R/WC
Size (in bits):	32

Interrupt status shadow copy. Access: copy-write by the GuC hardware, read and cleared by the uKernel and host CPU.

DWord	Bit	Description
0	31:0	**Interrupt Status** GuC shadow copy of the interrupt status mask for doorbells 32 through 63.

GuC Doorbell Group 2 Interrupt Status

GUC_DB_ISR_2 - GuC Doorbell Group 2 Interrupt Status		
Register Space:	MMIO: 0/2/0	
Access:	R/WC	
Size (in bits):	32	
Interrupt status shadow copy. Access: copy-write by the GuC hardware, read and cleared by the uKernel and host CPU.		
DWord	**Bit**	**Description**
0	31:0	**Interrupt Status** GuC shadow copy of the interrupt status mask for doorbells 64 through 95.

GuC Doorbell Group 3 Interrupt Status

GUC_DB_ISR_3 - GuC Doorbell Group 3 Interrupt Status	
Register Space:	MMIO: 0/2/0
Access:	R/WC
Size (in bits):	32

Interrupt status shadow copy. Access: copy-write by the GuC hardware, read and cleared by the uKernel and host CPU.

DWord	Bit	Description
0	31:0	**Interrupt Status** GuC shadow copy of the interrupt status mask for doorbells 96 through 127.

GuC Doorbell Group 4 Interrupt Status

GUC_DB_ISR_4 - GuC Doorbell Group 4 Interrupt Status		
Register Space:	MMIO: 0/2/0	
Access:	R/WC	
Size (in bits):	32	
Interrupt status shadow copy. Access: copy-write by the GuC hardware, read and cleared by the uKernel and host CPU.		
DWord	**Bit**	**Description**
0	31:0	**Interrupt Status** GuC shadow copy of the interrupt status mask for doorbells 128 through 159.

GuC Doorbell Group 5 Interrupt Status

GUC_DB_ISR_5 - GuC Doorbell Group 5 Interrupt Status	
Register Space:	MMIO: 0/2/0
Access:	R/WC
Size (in bits):	32

Interrupt status shadow copy. Access: copy-write by the GuC hardware, read and cleared by the uKernel and host CPU.

DWord	Bit	Description
0	31:0	**Interrupt Status** GuC shadow copy of the interrupt status mask for doorbells 160 through 191.

GuC Doorbell Group 6 Interrupt Status

GUC_DB_ISR_6 - GuC Doorbell Group 6 Interrupt Status		
Register Space:	MMIO: 0/2/0	
Access:	R/WC	
Size (in bits):	32	
Interrupt status shadow copy. Access: copy-write by the GuC hardware, read and cleared by the uKernel and host CPU.		
DWord	**Bit**	**Description**
0	31:0	**Interrupt Status** GuC shadow copy of the interrupt status mask for doorbells 192 through 223.

GuC Doorbell Group 7 Interrupt Status

GUC_DB_ISR_7 - GuC Doorbell Group 7 Interrupt Status	
Register Space:	MMIO: 0/2/0
Access:	R/WC
Size (in bits):	32

Interrupt status shadow copy. Access: copy-write by the GuC hardware, read and cleared by the uKernel and host CPU.

DWord	Bit	Description
0	31:0	**Interrupt Status** GuC shadow copy of the interrupt status mask for doorbells 224 through 255.

GuC Engine Interrupt Mask

GUC_ENG_INTR_MASK - GuC Engine Interrupt Mask		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
DWord	**Bit**	**Description**
0	31:16	**Engine1 Interrupt Mask** Access: R/W
	15:0	**Engine0 Interrupt Mask** Access: R/W

Guc GT Interrupt DW1

GUC_GT_INTR_DW1 - Guc GT Interrupt DW1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
This is a basic register template		
DWord	**Bit**	**Description**
0	31	**VECS0** Access: R/W
	30	**VECS1** Access: R/W
	29	**VECS2** Access: R/W
	28	**VECS3** Access: R/W
	27:8	**Reserved** Access: RO Format: MBZ
	7	**VCS7** Access: R/W
	6	**VCS6** Access: R/W
	5	**VCS5** Access: R/W
	4	**VCS4** Access: R/W
	3	**VCS3** Access: R/W
	2	**VCS2** Access: R/W
	1	**VCS1** Access: R/W
	0	**VCS0** Access: R/W

GuC Host Interrupt Interrupt Input 0

GUC_HOST_INTR_IIR_0 - GuC Host Interrupt Interrupt Input 0		
Register Space:	MMIO: 0/2/0	
Access:	R/WC	
Size (in bits):	32	
The format of this register is defined by software. The host CPU is expected to write this register and the uKernel to read it.		
DWord	**Bit**	**Description**
0	31:0	**Scratch**

GuC Host Interrupt Mask 0

GUC_HOST_INTR_MASK_0 - GuC Host Interrupt Mask 0		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Interrupt Masks for the host KMD that can generate interrupts. When virtualization is enabled, bits 1 through 31 may indicated interrupts		
DWord	**Bit**	**Description**
0	31:8	**VF N Intr Mask**
	7	**VF7 Intr Mask**
	6	**VF6 Intr Mask**
	5	**VF5 Intr Mask**
	4	**VF4 Intr Mask**
	3	**VF3 Intr Mask**
	2	**VF2 Intr Mask**
	1	**VF1 Intr Mask**
	0	**Host Intr Mask**

GuC Host Interupt Enable 0

GUC_HOST_INTR_ENA_0 - GuC Host Interupt Enable 0

Register Space:	MMIO: 0/2/0
Access:	R/WC
Size (in bits):	32

Interrupt Enables for the host KMD that can generate interrupts.
When virtualization is enabled, bits 1 through 31 may indicated interrupts
Contents of the register are saved and restored across RC6 transitions.

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**VF7 Intr Enable**
	6	**VF6 Intr Enable**
	5	**VF5 Intr Enable**
	4	**VF4 Intr Enable**
	3	**VF3 Intr Enable**
	2	**VF2 Intr Enable**
	1	**VF1 Intr Enable**
	0	**Host Intr Enable**

GuC Interupt IIR Selector

INTR_IIR_SELECTOR - GuC Interupt IIR Selector		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
DWord	**Bit**	**Description**
0	31:0	**Engine ID** Access: R/W SW/FW shall program the appropriate Engine ID to view the interrupts from an engine. Engine_ID is a one-hot encoding that follows the bit definition of GuC GT Interrupt DW register. GUC_IIR_REG0_SELECTOR uses GuC GT Interrupt DW0. GUC_IIR_REG1_SELECTOR uses GuC GT Interrupt DW1. **Programming Notes** A write to the selector is dropped if there are no interrupts to be serviced - because the State Machine is not in the correct state to accept the selector write

Guc Peek Register 0

DRB0PEEK - Guc Peek Register 0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	01920h	

DWord	Bit	Description
0	31	**Doorbell #(0*32+31) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+31) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	30	**Doorbell #(0*32+30) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+30) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	29	**Doorbell #(0*32+29) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+29) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	28	**Doorbell #(0*32+28) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+28) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	27	**Doorbell #(0*32+27) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+27) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB0PEEK - Guc Peek Register 0

26	**Doorbell #(0*32+26) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+26) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
25	**Doorbell #(0*32+25) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+25) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
24	**Doorbell #(0*32+24) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+24) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
23	**Doorbell #(0*32+23) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+23) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
22	**Doorbell #(0*32+22) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+22) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
21	**Doorbell #(0*32+21) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+21) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB0PEEK - Guc Peek Register 0

	20	**Doorbell #(0*32+20) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+20) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	19	**Doorbell #(0*32+19) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+19) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	18	**Doorbell #(0*32+18) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+18) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	17	**Doorbell #(0*32+17) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+17) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	16	**Doorbell #(0*32+16) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+16) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	15	**Doorbell #(0*32+15) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+15) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB0PEEK - Guc Peek Register 0

14 Doorbell #(0*32+14) Guc Peek

Access:	RO
_Custom_GTIReset:	BUS

Doorbell #(0*32+14) has been rung.
Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

13 Doorbell #(0*32+13) Guc Peek

Access:	RO
_Custom_GTIReset:	BUS

Doorbell #(0*32+13) has been rung.
Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

12 Doorbell #(0*32+12) Guc Peek

Access:	RO
_Custom_GTIReset:	BUS

Doorbell #(0*32+12) has been rung.
Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

11 Doorbell #(0*32+11) Guc Peek

Access:	RO
_Custom_GTIReset:	BUS

Doorbell #(0*32+11) has been rung.
Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

10 Doorbell #(0*32+10) Guc Peek

Access:	RO
_Custom_GTIReset:	BUS

Doorbell #(0*32+10) has been rung.
Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

9 Doorbell #(0*32+9) Guc Peek

Access:	RO
_Custom_GTIReset:	BUS

Doorbell #(0*32+9) has been rung.
Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB0PEEK - Guc Peek Register 0

	8	**Doorbell #(0*32+8) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+8) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	7	**Doorbell #(0*32+7) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+7) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	6	**Doorbell #(0*32+6) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+6) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	5	**Doorbell #(0*32+5) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+5) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	4	**Doorbell #(0*32+4) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+4) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	3	**Doorbell #(0*32+3) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+3) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB0PEEK - Guc Peek Register 0

	2	**Doorbell #(0*32+2) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+2) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	1	**Doorbell #(0*32+1) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+1) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	0	**Doorbell #(0*32+0) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(0*32+0) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

Guc Peek Register 1

DRB1PEEK - Guc Peek Register 1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	01924h	

DWord	Bit	Description
0	31	**Doorbell #(1*32+31) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+31) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	30	**Doorbell #(1*32+30) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+30) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	29	**Doorbell #(1*32+29) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+29) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	28	**Doorbell #(1*32+28) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+28) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	27	**Doorbell #(1*32+27) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+27) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB1PEEK - Guc Peek Register 1

26	**Doorbell #(1*32+26) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+26) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
25	**Doorbell #(1*32+25) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+25) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
24	**Doorbell #(1*32+24) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+24) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
23	**Doorbell #(1*32+23) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+23) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
22	**Doorbell #(1*32+22) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+22) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
21	**Doorbell #(1*32+21) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+21) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB1PEEK - Guc Peek Register 1

	20	**Doorbell #(1*32+20) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+20) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	19	**Doorbell #(1*32+19) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+19) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	18	**Doorbell #(1*32+18) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+18) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	17	**Doorbell #(1*32+17) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+17) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	16	**Doorbell #(1*32+16) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+16) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	15	**Doorbell #(1*32+15) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+15) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB1PEEK - Guc Peek Register 1

14	**Doorbell #(1*32+14) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+14) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
13	**Doorbell #(1*32+13) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+13) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
12	**Doorbell #(1*32+12) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+12) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
11	**Doorbell #(1*32+11) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+11) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
10	**Doorbell #(1*32+10) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+10) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
9	**Doorbell #(1*32+9) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+9) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB1PEEK - Guc Peek Register 1

	8	**Doorbell #(1*32+8) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+8) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	7	**Doorbell #(1*32+7) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+7) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	6	**Doorbell #(1*32+6) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+6) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	5	**Doorbell #(1*32+5) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+5) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	4	**Doorbell #(1*32+4) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+4) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	3	**Doorbell #(1*32+3) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+3) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB1PEEK - Guc Peek Register 1

	2	**Doorbell #(1*32+2) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+2) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	1	**Doorbell #(1*32+1) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+1) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	0	**Doorbell #(1*32+0) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(1*32+0) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

Guc Peek Register 2

DRB2PEEK - Guc Peek Register 2		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	01928h	

DWord	Bit	Description
0	31	**Doorbell #(2*32+31) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+31) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	30	**Doorbell #(2*32+30) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+30) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	29	**Doorbell #(2*32+29) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+29) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	28	**Doorbell #(2*32+28) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+28) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	27	**Doorbell #(2*32+27) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+27) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB2PEEK - Guc Peek Register 2

26 Doorbell #(2*32+26) Guc Peek

Access:	RO
_Custom_GTIReset:	BUS

Doorbell #(2*32+26) has been rung.
Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

25 Doorbell #(2*32+25) Guc Peek

Access:	RO
_Custom_GTIReset:	BUS

Doorbell #(2*32+25) has been rung.
Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

24 Doorbell #(2*32+24) Guc Peek

Access:	RO
_Custom_GTIReset:	BUS

Doorbell #(2*32+24) has been rung.
Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

23 Doorbell #(2*32+23) Guc Peek

Access:	RO
_Custom_GTIReset:	BUS

Doorbell #(2*32+23) has been rung.
Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

22 Doorbell #(2*32+22) Guc Peek

Access:	RO
_Custom_GTIReset:	BUS

Doorbell #(2*32+22) has been rung.
Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

21 Doorbell #(2*32+21) Guc Peek

Access:	RO
_Custom_GTIReset:	BUS

Doorbell #(2*32+21) has been rung.
Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB2PEEK - Guc Peek Register 2

20	**Doorbell #(2*32+20) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+20) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
19	**Doorbell #(2*32+19) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+19) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
18	**Doorbell #(2*32+18) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+18) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
17	**Doorbell #(2*32+17) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+17) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
16	**Doorbell #(2*32+16) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+16) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
15	**Doorbell #(2*32+15) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+15) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB2PEEK - Guc Peek Register 2

	14	**Doorbell #(2*32+14) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+14) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	13	**Doorbell #(2*32+13) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+13) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	12	**Doorbell #(2*32+12) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+12) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	11	**Doorbell #(2*32+11) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+11) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	10	**Doorbell #(2*32+10) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+10) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	9	**Doorbell #(2*32+9) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+9) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB2PEEK - Guc Peek Register 2

	8	**Doorbell #(2*32+8) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+8) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	7	**Doorbell #(2*32+7) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+7) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	6	**Doorbell #(2*32+6) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+6) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	5	**Doorbell #(2*32+5) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+5) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	4	**Doorbell #(2*32+4) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+4) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	3	**Doorbell #(2*32+3) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+3) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB2PEEK - Guc Peek Register 2

	2	**Doorbell #(2*32+2) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+2) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	1	**Doorbell #(2*32+1) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+1) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	0	**Doorbell #(2*32+0) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(2*32+0) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

Guc Peek Register 3

DRB3PEEK - Guc Peek Register 3		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	0192Ch	
DWord	**Bit**	**Description**
0	31	**Doorbell #(3*32+31) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+31) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	30	**Doorbell #(3*32+30) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+30) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	29	**Doorbell #(3*32+29) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+29) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	28	**Doorbell #(3*32+28) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+28) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	27	**Doorbell #(3*32+27) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+27) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB3PEEK - Guc Peek Register 3

26 Doorbell #(3*32+26) Guc Peek

Access:	RO
_Custom_GTIReset:	BUS

Doorbell #(3*32+26) has been rung.
Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

25 Doorbell #(3*32+25) Guc Peek

Access:	RO
_Custom_GTIReset:	BUS

Doorbell #(3*32+25) has been rung.
Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

24 Doorbell #(3*32+24) Guc Peek

Access:	RO
_Custom_GTIReset:	BUS

Doorbell #(3*32+24) has been rung.
Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

23 Doorbell #(3*32+23) Guc Peek

Access:	RO
_Custom_GTIReset:	BUS

Doorbell #(3*32+23) has been rung.
Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

22 Doorbell #(3*32+22) Guc Peek

Access:	RO
_Custom_GTIReset:	BUS

Doorbell #(3*32+22) has been rung.
Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

21 Doorbell #(3*32+21) Guc Peek

Access:	RO
_Custom_GTIReset:	BUS

Doorbell #(3*32+21) has been rung.
Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB3PEEK - Guc Peek Register 3

	20	**Doorbell #(3*32+20) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+20) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	19	**Doorbell #(3*32+19) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+19) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	18	**Doorbell #(3*32+18) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+18) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	17	**Doorbell #(3*32+17) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+17) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	16	**Doorbell #(3*32+16) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+16) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	15	**Doorbell #(3*32+15) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+15) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB3PEEK - Guc Peek Register 3

	14	**Doorbell #(3*32+14) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+14) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	13	**Doorbell #(3*32+13) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+13) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	12	**Doorbell #(3*32+12) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+12) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	11	**Doorbell #(3*32+11) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+11) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	10	**Doorbell #(3*32+10) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+10) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	9	**Doorbell #(3*32+9) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+9) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB3PEEK - Guc Peek Register 3

8	**Doorbell #(3*32+8) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+8) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
7	**Doorbell #(3*32+7) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+7) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
6	**Doorbell #(3*32+6) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+6) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
5	**Doorbell #(3*32+5) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+5) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
4	**Doorbell #(3*32+4) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+4) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
3	**Doorbell #(3*32+3) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+3) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB3PEEK - Guc Peek Register 3

	2	**Doorbell #(3*32+2) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+2) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	1	**Doorbell #(3*32+1) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+1) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	0	**Doorbell #(3*32+0) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(3*32+0) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

Guc Peek Register 4

DRB4PEEK - Guc Peek Register 4		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	01930h	
DWord	**Bit**	**Description**
0	31	**Doorbell #(4*32+31) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+31) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	30	**Doorbell #(4*32+30) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+30) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	29	**Doorbell #(4*32+29) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+29) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	28	**Doorbell #(4*32+28) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+28) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	27	**Doorbell #(4*32+27) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+27) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB4PEEK - Guc Peek Register 4

	26	**Doorbell #(4*32+26) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+26) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	25	**Doorbell #(4*32+25) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+25) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	24	**Doorbell #(4*32+24) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+24) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	23	**Doorbell #(4*32+23) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+23) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	22	**Doorbell #(4*32+22) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+22) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	21	**Doorbell #(4*32+21) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+21) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB4PEEK - Guc Peek Register 4

20	**Doorbell #(4*32+20) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+20) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
19	**Doorbell #(4*32+19) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+19) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
18	**Doorbell #(4*32+18) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+18) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
17	**Doorbell #(4*32+17) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+17) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
16	**Doorbell #(4*32+16) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+16) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
15	**Doorbell #(4*32+15) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+15) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB4PEEK - Guc Peek Register 4

14	**Doorbell #(4*32+14) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+14) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
13	**Doorbell #(4*32+13) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+13) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
12	**Doorbell #(4*32+12) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+12) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
11	**Doorbell #(4*32+11) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+11) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
10	**Doorbell #(4*32+10) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+10) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
9	**Doorbell #(4*32+9) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+9) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB4PEEK - Guc Peek Register 4

	8	**Doorbell #(4*32+8) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+8) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	7	**Doorbell #(4*32+7) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+7) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	6	**Doorbell #(4*32+6) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+6) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	5	**Doorbell #(4*32+5) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+5) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	4	**Doorbell #(4*32+4) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+4) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	3	**Doorbell #(4*32+3) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+3) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB4PEEK - Guc Peek Register 4

	2	**Doorbell #(4*32+2) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+2) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	1	**Doorbell #(4*32+1) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+1) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	0	**Doorbell #(4*32+0) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(4*32+0) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

Guc Peek Register 5

DRB5PEEK - Guc Peek Register 5		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	01934h	
DWord	**Bit**	**Description**
0	31	**Doorbell #(5*32+31) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+31) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	30	**Doorbell #(5*32+30) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+30) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	29	**Doorbell #(5*32+29) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+29) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	28	**Doorbell #(5*32+28) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+28) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	27	**Doorbell #(5*32+27) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+27) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB5PEEK - Guc Peek Register 5

26	**Doorbell #(5*32+26) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+26) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
25	**Doorbell #(5*32+25) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+25) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
24	**Doorbell #(5*32+24) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+24) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
23	**Doorbell #(5*32+23) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+23) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
22	**Doorbell #(5*32+22) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+22) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
21	**Doorbell #(5*32+21) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+21) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB5PEEK - Guc Peek Register 5

	20	**Doorbell #(5*32+20) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+20) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	19	**Doorbell #(5*32+19) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+19) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	18	**Doorbell #(5*32+18) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+18) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	17	**Doorbell #(5*32+17) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+17) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	16	**Doorbell #(5*32+16) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+16) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	15	**Doorbell #(5*32+15) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+15) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB5PEEK - Guc Peek Register 5

14	**Doorbell #(5*32+14) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+14) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
13	**Doorbell #(5*32+13) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+13) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
12	**Doorbell #(5*32+12) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+12) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
11	**Doorbell #(5*32+11) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+11) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
10	**Doorbell #(5*32+10) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+10) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
9	**Doorbell #(5*32+9) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+9) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB5PEEK - Guc Peek Register 5

	8	**Doorbell #(5*32+8) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+8) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	7	**Doorbell #(5*32+7) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+7) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	6	**Doorbell #(5*32+6) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+6) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	5	**Doorbell #(5*32+5) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+5) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	4	**Doorbell #(5*32+4) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+4) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	3	**Doorbell #(5*32+3) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+3) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB5PEEK - Guc Peek Register 5

	2	**Doorbell #(5*32+2) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+2) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	1	**Doorbell #(5*32+1) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+1) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	0	**Doorbell #(5*32+0) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(5*32+0) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

Guc Peek Register 6

DRB6PEEK - Guc Peek Register 6		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	01938h	
DWord	**Bit**	**Description**
0	31	**Doorbell #(6*32+31) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+31) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	30	**Doorbell #(6*32+30) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+30) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	29	**Doorbell #(6*32+29) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+29) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	28	**Doorbell #(6*32+28) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+28) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	27	**Doorbell #(6*32+27) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+27) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB6PEEK - Guc Peek Register 6

	26	**Doorbell #(6*32+26) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+26) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	25	**Doorbell #(6*32+25) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+25) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	24	**Doorbell #(6*32+24) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+24) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	23	**Doorbell #(6*32+23) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+23) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	22	**Doorbell #(6*32+22) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+22) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	21	**Doorbell #(6*32+21) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+21) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB6PEEK - Guc Peek Register 6

	20	**Doorbell #(6*32+20) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+20) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	19	**Doorbell #(6*32+19) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+19) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	18	**Doorbell #(6*32+18) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+18) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	17	**Doorbell #(6*32+17) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+17) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	16	**Doorbell #(6*32+16) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+16) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	15	**Doorbell #(6*32+15) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+15) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB6PEEK - Guc Peek Register 6

14	**Doorbell #(6*32+14) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+14) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
13	**Doorbell #(6*32+13) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+13) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
12	**Doorbell #(6*32+12) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+12) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
11	**Doorbell #(6*32+11) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+11) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
10	**Doorbell #(6*32+10) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+10) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
9	**Doorbell #(6*32+9) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+9) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB6PEEK - Guc Peek Register 6

	8	**Doorbell #(6*32+8) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+8) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	7	**Doorbell #(6*32+7) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+7) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	6	**Doorbell #(6*32+6) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+6) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	5	**Doorbell #(6*32+5) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+5) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	4	**Doorbell #(6*32+4) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+4) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	3	**Doorbell #(6*32+3) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+3) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB6PEEK - Guc Peek Register 6

	2	**Doorbell #(6*32+2) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+2) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	1	**Doorbell #(6*32+1) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+1) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	0	**Doorbell #(6*32+0) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(6*32+0) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

Guc Peek Register 7

DRB7PEEK - Guc Peek Register 7		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	0193Ch	

DWord	Bit	Description
0	31	**Doorbell #(7*32+31) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+31) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	30	**Doorbell #(7*32+30) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+30) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	29	**Doorbell #(7*32+29) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+29) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	28	**Doorbell #(7*32+28) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+28) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	27	**Doorbell #(7*32+27) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+27) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB7PEEK - Guc Peek Register 7

	26	**Doorbell #(7*32+26) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+26) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	25	**Doorbell #(7*32+25) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+25) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	24	**Doorbell #(7*32+24) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+24) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	23	**Doorbell #(7*32+23) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+23) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	22	**Doorbell #(7*32+22) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+22) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	21	**Doorbell #(7*32+21) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+21) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB7PEEK - Guc Peek Register 7

	20	**Doorbell #(7*32+20) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+20) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	19	**Doorbell #(7*32+19) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+19) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	18	**Doorbell #(7*32+18) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+18) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	17	**Doorbell #(7*32+17) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+17) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	16	**Doorbell #(7*32+16) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+16) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	15	**Doorbell #(7*32+15) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+15) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB7PEEK - Guc Peek Register 7

	14	**Doorbell #(7*32+14) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+14) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	13	**Doorbell #(7*32+13) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+13) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	12	**Doorbell #(7*32+12) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+12) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	11	**Doorbell #(7*32+11) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+11) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	10	**Doorbell #(7*32+10) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+10) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	9	**Doorbell #(7*32+9) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+9) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB7PEEK - Guc Peek Register 7

	8	**Doorbell #(7*32+8) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+8) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	7	**Doorbell #(7*32+7) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+7) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	6	**Doorbell #(7*32+6) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+6) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	5	**Doorbell #(7*32+5) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+5) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	4	**Doorbell #(7*32+4) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+4) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	3	**Doorbell #(7*32+3) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+3) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

DRB7PEEK - Guc Peek Register 7

	2	**Doorbell #(7*32+2) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+2) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	1	**Doorbell #(7*32+1) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+1) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.
	0	**Doorbell #(7*32+0) Guc Peek** Access: RO _Custom_GTIReset: BUS Doorbell #(7*32+0) has been rung. Shadow copy of matching register in DRB*ACT, which is not cleared after a read and only echoes the register value.

GuC PM Time Stamp Counter 0

GUC_PM_TSCVALUE0 - GuC PM Time Stamp Counter 0	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32

This register value shadows the GPM time stamp counter value.

Programming Notes
This value is provided locally in the GuC Shim to avoid overhead of retrieving a time stamp value from GPM unit (for e.g: latency incurred for reading the timestamp value makes the value stale).
This register is saved in the power context.

DWord	Bit	Description
0	31:0	**Time Stamp Value** Holds the lower 32 bits of the incoming time stamp counter value.

GuC PM Time Stamp Counter 1

GUC_PM_TSCVALUE1 - GuC PM Time Stamp Counter 1

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32

This register value shadows the GPM time stamp counter value.

Programming Notes
This value is provided locally in the GuC Shim to avoid overhead of retrieving a time stamp value from GPM unit (for e.g: latency incurred for reading the timestamp value makes the value stale).
This register is saved in the power context.

DWord	Bit	Description
0	31:4	**Reserved** Access: RO Format: MBZ
	3:0	**Time Stamp Value** Holds the upper 4 bits of the incoming time stamp counter value.

GU Misc Interrupt Definition

GU Misc Interrupt Definition

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS

Address:	444F0h-444FFh
Name:	Gunit Miscellaneous Interrupts
ShortName:	GU_MISC_INTERRUPT

This table indicates which events are mapped to each bit of the Gunit Miscellaneous Interrupt registers.
0x444F0 = ISR
0x444F4 = IMR
0x444F8 = IIR
0x444FC = IER

DWord	Bit	Description
0	31	**Spare_31** _Custom_GTIReset: BUS Spare bit
	30	**Spare_30** _Custom_GTIReset: BUS Spare bit
	29	**Invalid GTT page table entry** _Custom_GTIReset: BUS The ISR is an active high pulse on receiving the iMPH invalid GTT page table entry indication.
	28	**Spare_28** _Custom_GTIReset: BUS Spare bit
	27	**GSE** _Custom_GTIReset: BUS The ISR is an active high pulse on the GSE system level event.
	26	**Spare_26** _Custom_GTIReset: BUS Spare bit

GU Misc Interrupt Definition

	25	**Spare_25** _Custom_GTIReset: BUS Spare bit
	24	**Spare_24** _Custom_GTIReset: BUS Spare bit
	23	**Spare_23** _Custom_GTIReset: BUS Spare bit
	22	**SVM Device Mode PRQ Event** _Custom_GTIReset: BUS The ISR is an active high pulse on receiving the iMPH SVM Device Mode PRQ event indication. This event indicates that a GT advanced context encountered a recoverable page fault.
	21	**SVM Device Mode VTD Fault** _Custom_GTIReset: BUS The ISR is an active high pulse on receiving the iMPH SVM Device Mode VT-d fault indication. This event indicates GT encountered a non-recoverable translation fault.
	20	**SVM Device Mode Wait Descriptor Completion** _Custom_GTIReset: BUS The ISR is an active high pulse on receiving the iMPH SVM Device Mode Wait Descriptor Completion indication.This event indicates that IMPH completed Invalidation Wait Descriptor.
	19	**Spare_19** _Custom_GTIReset: BUS Spare bit
	18	**Spare_18** _Custom_GTIReset: BUS Spare bit
	17	**Spare_17** _Custom_GTIReset: BUS Spare bit

GU Misc Interrupt Definition		
	16	**Spare_16** _Custom_GTIReset: BUS Spare bit
	15	**Spare_15** _Custom_GTIReset: BUS Spare bit
	14	**Spare_14** _Custom_GTIReset: BUS Spare bit
	13	**Spare_13** _Custom_GTIReset: BUS Spare bit
	12	**Spare_12** _Custom_GTIReset: BUS Spare bit
	11	**Spare_11** _Custom_GTIReset: BUS Spare bit
	10	**Spare_10** _Custom_GTIReset: BUS Spare bit
	9	**Spare_9** _Custom_GTIReset: BUS Spare bit
	8	**Spare_8** _Custom_GTIReset: BUS Spare bit
	7	**Spare_7** _Custom_GTIReset: BUS Spare bit

GU Misc Interrupt Definition		
	6	**Spare_6** _Custom_GTIReset: BUS Spare bit
	5	**Spare_5** _Custom_GTIReset: BUS Spare bit
	4	**Spare_4** _Custom_GTIReset: BUS Spare bit
	3	**Spare_3** _Custom_GTIReset: BUS Spare bit
	2	**Spare_2** _Custom_GTIReset: BUS Spare bit
	1	**Spare_1** _Custom_GTIReset: BUS Spare bit
	0	**Spare_0** _Custom_GTIReset: BUS Spare bit

Hardware Status Mask Register

HWSTAM - Hardware Status Mask Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02098h-0209Bh
Name:	Hardware Status Mask Register
ShortName:	HWSTAM_RCSUNIT
Address:	18098h-1809Bh
Name:	Hardware Status Mask Register
ShortName:	HWSTAM_POCSUNIT
Address:	22098h-2209Bh
Name:	Hardware Status Mask Register
ShortName:	HWSTAM_BCSUNIT
Address:	1C0098h-1C009Bh
Name:	Hardware Status Mask Register
ShortName:	HWSTAM_VCSUNIT0
Address:	1C4098h-1C409Bh
Name:	Hardware Status Mask Register
ShortName:	HWSTAM_VCSUNIT1
Address:	1C8098h-1C809Bh
Name:	Hardware Status Mask Register
ShortName:	HWSTAM_VECSUNIT0
Address:	1D0098h-1D009Bh
Name:	Hardware Status Mask Register
ShortName:	HWSTAM_VCSUNIT2
Address:	1D4098h-1D409Bh
Name:	Hardware Status Mask Register
ShortName:	HWSTAM_VCSUNIT3
Address:	1D8098h-1D809Bh
Name:	Hardware Status Mask Register
ShortName:	HWSTAM_VECSUNIT1
Address:	1E0098h-1E009Bh
Name:	Hardware Status Mask Register

HWSTAM - Hardware Status Mask Register

ShortName:	HWSTAM_VCSUNIT4
Address:	1E4098h-1E409Bh
Name:	Hardware Status Mask Register
ShortName:	HWSTAM_VCSUNIT5
Address:	1E8098h-1E809Bh
Name:	Hardware Status Mask Register
ShortName:	HWSTAM_VECSUNIT2
Address:	1F0098h-1F009Bh
Name:	Hardware Status Mask Register
ShortName:	HWSTAM_VCSUNIT6
Address:	1F4098h-1F409Bh
Name:	Hardware Status Mask Register
ShortName:	HWSTAM_VCSUNIT7
Address:	1F8098h-1F809Bh
Name:	Hardware Status Mask Register
ShortName:	HWSTAM_VECSUNIT3
Address:	1A098h-1A09Bh
Name:	Hardware Status Mask Register
ShortName:	HWSTAM_CCSUNIT0

The HWSTAM register has the same format as the Interrupt Control Registers. The bits in this register are mask bits that prevent the corresponding bits in the Interrupt Status Register from generating a Hardware Status Write (PCI write cycle). Any unmasked interrupt bit (HWSTAM bit set to 0) will allow the Interrupt Status Register to be written to the ISR location (within the memory page specified by the Hardware Status Page Address Register) when that Interrupt Status Register bit changes state.

Programming Notes

- To write the interrupt to the HWSP, the corresponding IMR bit must also be clear (enabled).
- At most 1 bit can be unmasked at any given time.

DWord	Bit	Description
0	31:0	**Hardware Status Mask** Default Value: 00000000h Refer to the Interrupt Control Register section for bit definitions. Reserved bits are RO.

Hardware Status Page Address Register

HWS_PGA - Hardware Status Page Address Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02080h-02083h
Name:	Hardware Status Page Address Register
ShortName:	HWS_PGA_RCSUNIT
Address:	18080h-18083h
Name:	Hardware Status Page Address Register
ShortName:	HWS_PGA_POCSUNIT
Address:	22080h-22083h
Name:	Hardware Status Page Address Register
ShortName:	HWS_PGA_BCSUNIT
Address:	1C0080h-1C0083h
Name:	Hardware Status Page Address Register
ShortName:	HWS_PGA_VCSUNIT0
Address:	1C4080h-1C4083h
Name:	Hardware Status Page Address Register
ShortName:	HWS_PGA_VCSUNIT1
Address:	1C8080h-1C8083h
Name:	Hardware Status Page Address Register
ShortName:	HWS_PGA_VECSUNIT0
Address:	1D0080h-1D0083h
Name:	Hardware Status Page Address Register
ShortName:	HWS_PGA_VCSUNIT2
Address:	1D4080h-1D4083h
Name:	Hardware Status Page Address Register
ShortName:	HWS_PGA_VCSUNIT3
Address:	1D8080h-1D8083h
Name:	Hardware Status Page Address Register
ShortName:	HWS_PGA_VECSUNIT1
Address:	1E0080h-1E0083h
Name:	Hardware Status Page Address Register

HWS_PGA - Hardware Status Page Address Register

ShortName:	HWS_PGA_VCSUNIT4
Address:	1E4080h-1E4083h
Name:	Hardware Status Page Address Register
ShortName:	HWS_PGA_VCSUNIT5
Address:	1E8080h-1E8083h
Name:	Hardware Status Page Address Register
ShortName:	HWS_PGA_VECSUNIT2
Address:	1F0080h-1F0083h
Name:	Hardware Status Page Address Register
ShortName:	HWS_PGA_VCSUNIT6
Address:	1F4080h-1F4083h
Name:	Hardware Status Page Address Register
ShortName:	HWS_PGA_VCSUNIT7
Address:	1F8080h-1F8083h
Name:	Hardware Status Page Address Register
ShortName:	HWS_PGA_VECSUNIT3
Address:	1A080h-1A083h
Name:	Hardware Status Page Address Register
ShortName:	HWS_PGA_CCSUNIT0

This register is used to program the 4 KB-aligned System Memory address of the Hardware Status Page used to report hardware status into (typically cacheable) System Memory.

DWord	Bit	Description
0	31:12	**Address** Format: GraphicsAddress[31:12] This field is used by SW to specify Bits 31:12 of the 4 KB-aligned System Memory address of the 4 KB page known as the Hardware Status Page. The Global GTT is used to map this page from the graphics virtual address to physical address. **Programming Notes** If the Per-Process Virtual Address Space and Exec List Enable bit is set, HW requires that the status page is programmed to allow for the context switch status to be reported.
	11:0	**Reserved** Access: RO Format: MBZ

HCP Bitstream Output Minimal Size Padding Count Report Register

HCP_MINSIZE_PADDING_COUNT - HCP Bitstream Output Minimal Size Padding Count Report Register

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1E9B4h

This register stores the count in bytes of **minimal size padding insertion. It is primarily provided forstatistical data gathering.** This register is part of the context save and restore.

DWord	Bit	Description
0	31:0	**HCP MinSize Padding Count** Format: U32 Total number of bytes in the bitstream output contributing to minimal size padding operation. Thiscount is updated each time when the padding count is incremented.

HCP CABAC Status

HCP_CABAC_STATUS - HCP CABAC Status	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	1C2804h
ShortName:	HCP_CABAC_STATUS_VCS0
Description:	For VDBox0
Address:	1C6804h
ShortName:	HCP_CABAC_STATUS_VCS1
Description:	For VDBox1
Address:	1D2804h
ShortName:	HCP_CABAC_STATUS_VCS2
Description:	For VDBox2
Address:	1D6804h
ShortName:	HCP_CABAC_STATUS_VCS3
Description:	For VDBox3
Address:	1E2804h
ShortName:	HCP_CABAC_STATUS_VCS4
Description:	For VDBox4
Address:	1E6804h
ShortName:	HCP_CABAC_STATUS_VCS5
Description:	For VDBox5
Address:	1F2804h
ShortName:	HCP_CABAC_STATUS_VCS6
Description:	For VDBox6
Address:	1F6804h
ShortName:	HCP_CABAC_STATUS_VCS7
Description:	For VDBox7
HCP CABAC status or VP9 Decode status	

DWord	Bit	Description
0	31:18	**Reserved** Access: RO Format: MBZ

HCP_CABAC_STATUS - HCP CABAC Status

	17:0	**VP9 SuperBlock Concealment Counter** Exists If: // VP9 decode = 1 Format: U18 Indicate the number of Superblock (SB) concealed by VP9 decoder (not decoded from bitstream due to error)
	11	**Temporal Direction Motion Vector Out-of-Bound Error** Default Value: 0 Access: RO Exists If: // HEVC decode = 1 Format: U1 This flag indicates motion vectors calculated from the Temporal Direct Motion vector is larger than the allowed range for HEVC decode.
	6	**Motion Vector Delta SE** Default Value: 0 Access: RO Exists If: // HEVC decode = 1 Format: U1 This flag indicates out-of-bound motion vector delta SEs coded in the bit-stream for HEVC decode.
	5	**Delta QP SE** Default Value: 0 Access: RO Exists If: // HEVC decode = 1 Format: U1 This flag indicates leading-one overflow during CABAC decode of cu_qp_delta_abs for HEVC decode.
	4	**Residual Error** Default Value: 0 Access: RO Exists If: // HEVC decode = 1 Format: U1 This flag indicates out-of-bound absolute coefficient level SEs coded in the bit-streamfor HEVC decode.

HCP_CABAC_STATUS - HCP CABAC Status

	3	**Slice and Error** Default Value: 0 Access: RO Exists If: // HEVC decode = 1 Format: U1 This flag indicates a pre-mature end to the slice or an inconsistent end of slice on the last Ctb of a slice for HEVC decode.
	0	**HEVC Ctb Concealment Flag** Default Value: 0 Access: RO Exists If: // HEVC decode = 1 Format: U1 Each pulse from this flag indicates one Ctb is concealed by the HEVC decode.

HCP Decode Status

HCP_DEC_STATUS - HCP Decode Status

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32

Address:	1C2800h
ShortName:	HCP_DEC_STATUS_VCS0
Description:	For VDBox0

Address:	1C6800h
ShortName:	HCP_DEC_STATUS_VCS1
Description:	For VDBox1

Address:	1D2800h
ShortName:	HCP_DEC_STATUS_VCS2
Description:	For VDBox2

Address:	1D6800h
ShortName:	HCP_DEC_STATUS_VCS3
Description:	For VDBox3

Address:	1E2800h
ShortName:	HCP_DEC_STATUS_VCS4
Description:	For VDBox4

Address:	1E6800h
ShortName:	HCP_DEC_STATUS_VCS5
Description:	For VDBox5

Address:	1F2800h
ShortName:	HCP_DEC_STATUS_VCS6
Description:	For VDBox6

Address:	1F6800h
ShortName:	HCP_DEC_STATUS_VCS7
Description:	For VDBox7

HCP Decode status.

DWord	Bit	Description
0	31:18	**Number of Ctbs Concealed** Default Value: 0 Format: U14 This 14-bit field indicates the number of Ctbs concealed during the decoding of the current frame. This field is cleared with the HCP_PIPE_MODE_SELECT command.

HCP_DEC_STATUS - HCP Decode Status

	17	**Frame Dec Active** Default Value: 0 Format: U1 This flag indicates that the decoder hardware is actively decoding a picture.
	16	**Indirect Bitstream ObjectAccess Upper Bound Error** Default Value: 0 Format: U1 This flag indicates that the upper bound bit-stream address was reached.
	15:0	**Bit-stream Error Flags** Default Value: 0 Format: U16 This 16-bit field indicates the number of bit stream errors detected for each bit fieldindicated in the CABAC Status register.

HCP Frame BitStream BIN Count

HCP_BIN_CT - HCP Frame BitStream BIN Count

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1E980h

This register stores the number of BINs decoded in a frame. This register is not part of hardware context save and restore.

DWord	Bit	Description
0	31:0	**HCP Frame Bit-stream BIN Count** Default Value: 723ba5c0h Format: U32 Total number of BINs decoded/ in current frame. This number is used with frame performance count to derive Bin/clk.

HCP Image Status Control

HCP_IMAGE_STATUS_CONTROL - HCP Image Status Control

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	1C28BCh

DWord	Bit	Description
0	31:24	**Cumulative Frame Delta QP/QIndex** Format: S7 Used for Frame Level Multi-pass Rate Control.HEVC: cu_qp = input (first pass) cu_qp + Cumulative Frame Delta Qp. Pak does clamping to max value based on bitdepth. Bit31 is the sign bit. VP9: cu_qindex = input (first pass) cu_qindex + Cumulative Frame Delta Qindex. Pak does clamping to -127..127 after adding. Bit31 is the sign bit. VDENC: In VDenc mode this value is added even in first pass (always)so the recommendation is to set this value to zero in first pass
	23	**Reserved** Access: RO Format: MBZ
	22:16	**Cumulative Frame Delta LF** Access: RO Format: S6 Used for Frame Level Multi-pass Rate Control. LF_level = input (first pass) LF_level + Cumulative Frame Delta LF level. Pak does clamping to -63..63 after adding.
	15:12	**Reserved** Access: RO Format: MBZ
	11:8	**Total Num-Pass** Format: U4
	7:3	**Reserved** Access: RO Format: MBZ
	2	**Frame Bit Count Violate - under run** Access: RO Format: U1 This can trigger Frame Level Multi-pass Rate Control.Set to 1 if frame bit count is less than or equal to FrameBitRateMin

HCP_IMAGE_STATUS_CONTROL - HCP Image Status Control		
	1	**Frame Bit Count Violate - over run** Access: RO Format: U1 This can trigger Frame Level Multi-pass Rate Control.Set to 1 if frame bit count is less than or equal to FrameBitRateMax
	0	**LCU Bit Count Violate- overrun**

HCP Image Status Mask

HCP_IMAGE_STATUS_MASK - HCP Image Status Mask		
Register Space:	MMIO: 0/2/0	
Access:	RO	
Size (in bits):	32	
Address:	1E9B8h	
This register stores the image status(flags).		

DWord	Bit	Description
0	31:3	**Reserved** Access: RO Format: MBZ
	2	**FrameBitRateMinReportMask** Same as **FrameSzUnderStatusEn** in HCP_PIC_STATE.
	1	**FrameBitRateMaxReportMask** Same as **FrameSzOverStatusEn** in HCP_PIC_STATE.
	0	**FrameLcuMaxReportMask**

HCP Last Position

HCP_LAST_POSITION - HCP Last Position

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	1C2808h
ShortName:	HCP_LAST_POSITION_VCS0
Description:	For VDBox0
Address:	1C6808h
ShortName:	HCP_LAST_POSITION_VCS1
Description:	For VDBox1
Address:	1D2808h
ShortName:	HCP_LAST_POSITION_VCS2
Description:	For VDBox2
Address:	1D6808h
ShortName:	HCP_LAST_POSITION_VCS3
Description:	For VDBox3
Address:	1E2808h
ShortName:	HCP_LAST_POSITION_VCS4
Description:	For VDBox4
Address:	1E6808h
ShortName:	HCP_LAST_POSITION_VCS5
Description:	For VDBox5
Address:	1F2808h
ShortName:	HCP_LAST_POSITION_VCS6
Description:	For VDBox6
Address:	1F6808h
ShortName:	HCP_LAST_POSITION_VCS7
Description:	For VDBox7

Last row and column position of the decoder.

- The HCP Last Position register reports the position of the last Ctb to be decoded by the HCP hardware.
- It can be reset to 0H with the HCP_PIPE_MODE_SELECT command.

DWord	Bit	Description
0	31:25	**Reserved** Access: RO

HCP_LAST_POSITION - HCP Last Position

		Format: MBZ
	24:16	**Last Row Position in Ctbs** Default Value: 0 Access: RO Format: U9
	15:9	**Reserved** Access: RO Format: MBZ
	8:0	**Last Column Position in Ctbs** Default Value: 0 Access: RO Format: U9

HCP Picture Checksum cIdx0

HCP_PICTURE_CHECKSUM_CIDX0 - HCP Picture Checksum cIdx0	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1C281Ch
ShortName:	HCP_PICTURE_CHECKSUM_CIDX0_VCS0
Description:	For VDBox0
Address:	1C681Ch
ShortName:	HCP_PICTURE_CHECKSUM_CIDX0_VCS1
Description:	For VDBox1
Address:	1D281Ch
ShortName:	HCP_PICTURE_CHECKSUM_CIDX0_VCS2
Description:	For VDBox2
Address:	1D681Ch
ShortName:	HCP_PICTURE_CHECKSUM_CIDX0_VCS3
Description:	For VDBox3
Address:	1E281Ch
ShortName:	HCP_PICTURE_CHECKSUM_CIDX0_VCS4
Description:	For VDBox4
Address:	1E681Ch
ShortName:	HCP_PICTURE_CHECKSUM_CIDX0_VCS5
Description:	For VDBox5
Address:	1F281Ch
ShortName:	HCP_PICTURE_CHECKSUM_CIDX0_VCS6
Description:	For VDBox6
Address:	1F681Ch
ShortName:	HCP_PICTURE_CHECKSUM_CIDX0_VCS7
Description:	For VDBox7

- The HCP Picture Checksum cIdx0 register reports the 32-bit unsigned picture checksum for cIdx=0 calculatedby the HCP hardware and whose algorithm is defined in Annex D of the HEVC standard specification.
- This calculated value is updated at the end of the frame.

DWord	Bit	Description
0	31:0	**Picture checksum cIdx0**

HCP_PICTURE_CHECKSUM_CIDX0 - HCP Picture Checksum cIdx0			
		Default Value:	0
		Format:	U32

HCP Picture Checksum cIdx1

HCP_PICTURE_CHECKSUM_CIDX1 - HCP Picture Checksum cIdx1	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1C2820h
ShortName:	HCP_PICTURE_CHECKSUM_CIDX1_VCS0
Description:	For VDBox0
Address:	1C6820h
ShortName:	HCP_PICTURE_CHECKSUM_CIDX1_VCS1
Description:	For VDBox1
Address:	1D2820h
ShortName:	HCP_PICTURE_CHECKSUM_CIDX1_VCS2
Description:	For VDBox2
Address:	1D6820h
ShortName:	HCP_PICTURE_CHECKSUM_CIDX1_VCS3
Description:	For VDBox3
Address:	1E2820h
ShortName:	HCP_PICTURE_CHECKSUM_CIDX1_VCS4
Description:	For VDBox4
Address:	1E6820h
ShortName:	HCP_PICTURE_CHECKSUM_CIDX1_VCS5
Description:	For VDBox5
Address:	1F2820h
ShortName:	HCP_PICTURE_CHECKSUM_CIDX1_VCS6
Description:	For VDBox6
Address:	1F6820h
ShortName:	HCP_PICTURE_CHECKSUM_CIDX1_VCS7
Description:	For VDBox7

- The HCP Picture Checksum cIdx1 register reports the 32-bit unsigned picture checksum for cIdx=1 calculatedby the HCP hardware and whose algorithm is defined in Annex D of the HEVC standard specification.
- This calculated value is updated at the end of the frame.

DWord	Bit	Description
0	31:0	**Picture checksum cIdx1**

HCP_PICTURE_CHECKSUM_CIDX1 - HCP Picture Checksum cIdx1			
		Default Value:	0
		Format:	U32

HCP Picture Checksum cIdx2

HCP_PICTURE_CHECKSUM_CIDX2 - HCP Picture Checksum cIdx2	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1C2824h
ShortName:	HCP_PICTURE_CHECKSUM_CIDX2_VCS0
Description:	For VDBox0
Address:	1C6824h
ShortName:	HCP_PICTURE_CHECKSUM_CIDX2_VCS1
Description:	For VDBox1
Address:	1D2824h
ShortName:	HCP_PICTURE_CHECKSUM_CIDX2_VCS2
Description:	For VDBox2
Address:	1D6824h
ShortName:	HCP_PICTURE_CHECKSUM_CIDX2_VCS3
Description:	For VDBox3
Address:	1E2824h
ShortName:	HCP_PICTURE_CHECKSUM_CIDX2_VCS4
Description:	For VDBox4
Address:	1E6824h
ShortName:	HCP_PICTURE_CHECKSUM_CIDX2_VCS5
Description:	For VDBox5
Address:	1F2824h
ShortName:	HCP_PICTURE_CHECKSUM_CIDX2_VCS6
Description:	For VDBox6
Address:	1F6824h
ShortName:	HCP_PICTURE_CHECKSUM_CIDX2_VCS7
Description:	For VDBox7

- The HCP Picture Checksum cIdx2 register reports the 32-bit unsigned picture checksum for cIdx=2 calculatedby the HCP hardware and whose algorithm is defined in Annex D of the HEVC standard specification.
- This calculated value is updated at the end of the frame.

DWord	Bit	Description
0	31:0	**Picture checksum cIdx2**

HCP_PICTURE_CHECKSUM_CIDX2 - HCP Picture Checksum cIdx2			
		Default Value:	0
		Format:	U32

HCP PMU Status

HCP_PMU_STATUS - HCP PMU Status	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	1C280Ch
ShortName:	HCP_PMU_STATUS_VCS0
Description:	For VDBox0
Address:	1C680Ch
ShortName:	HCP_PMU_STATUS_VCS1
Description:	For VDBox1
Address:	1D280Ch
ShortName:	HCP_PMU_STATUS_VCS2
Description:	For VDBox2
Address:	1D680Ch
ShortName:	HCP_PMU_STATUS_VCS3
Description:	For VDBox3
Address:	1E280Ch
ShortName:	HCP_PMU_STATUS_VCS4
Description:	For VDBox4
Address:	1E680Ch
ShortName:	HCP_PMU_STATUS_VCS5
Description:	For VDBox5
Address:	1F280Ch
ShortName:	HCP_PMU_STATUS_VCS6
Description:	For VDBox6
Address:	1F680Ch
ShortName:	HCP_PMU_STATUS_VCS7
Description:	For VDBox7

PMU counter overflow status.

- The HCP PMU Status register reports the overflow status of the HCP PMU Luma Cache Miss Counter,the HCP PMU Chroma cache Miss Counter and the HCP Frame Decode Active Counter.
- It can be reset to 0H with the HCP_PIPE_MODE_SELECT command.

DWord	Bit	Description
0	31:3	**Reserved**

HCP_PMU_STATUS - HCP PMU Status

Access:	RO
Format:	MBZ

2 Event Counter Overflow - Frame Decode Active

Access:	RO
Format:	U1

Value	Name
1	Counter overflow
0	Non-Active **[Default]**

1 Event Counter Overflow - Chroma Cache Miss

Access:	RO
Format:	U1

Value	Name
1	Counter overflow
0	Non-Active **[Default]**

0 Event Counter Overflow - Luma Cache Miss

Access:	RO
Format:	U1

Value	Name
1	Counter overflow
0	Non-Active **[Default]**

HCP Power Context Save request

HCPPGCTXREQ - HCP Power Context Save request

Register Space: MMIO: 0/2/0

Size (in bits): 32

DWord	Bit	Description
0	31:16	**Reserved**
	15:10	**Reserved** Access: RO Format: MBZ
	9	**Power context save request** Access: R/W Set _Custom_GTIReset: BUS Power Context Save Request 1'b0 : Power context save is not being requested 1'b1 : Power context save is being requested CPUnit self-clears this bit upon sampling.
	8:0	**Power Context Save request crdit count** Access: R/W _Custom_GTIReset: BUS QWord Credits for Power Context Save Request Minimum Credits = 1 : Unit may send 1 QWord pair (enough for first LRI at least) Maximum Credits = 511 : Unit may send 511 QWord pairs A QWord pair is defined as a 32-bit register address and the corresponding 32-bits of register data. Note that the LRI header and END commands are 64-bits each (32-bit command followed by 32-bit NOOP) and will consume one QWord credit. Only valid with PWRCTX_SAVE_REQ (Bit9).

HCP Power Down FSM control register with lock

HCPSPCPOWERDNFSMCTL - HCP Power Down FSM control register with lock

Register Space: MMIO: 0/2/0

Size (in bits): 32

DWord	Bit	Description
0	31	**Reserved**
	30:13	**Reserved** Access: RO Format: MBZ
	12	**Leave firewall disabled** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not firewall the gated domain for a power down flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e firewall gated domain to ungated domain crossing during power down flows 1 = Leave firewall disabled, i.e don't firewall the gated domain, but complete logical flow
	11	**Leave reset de-asserted** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not assert reset for power off flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e assert resets during power down flows 1 = Leave reset de-asserted mode, i.e don't assert reset, but complete logical flow
	10	**Leave CLKs ON** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not gate clks for power off flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e gate clocks during power down flows 1 = Leave CLKS ON mode, i.e don't clock gate, but complete logical flow

HCPSPCPOWERDNFSMCTL - HCP Power Down FSM control register with lock

	9	**Leave FET On** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not turn off the PFET even though it will complete the flow with PM Encodings: 0 = Default mode, i.e power off fets during power down flows 1 = Leave ON mode, i.e don't power off pfet, but complete logical flow
	8:0	**Reserved** Access: RO Format: MBZ

HCP Power Gate Control Request

HCPPGCTLREQ - HCP Power Gate Control Request

Register Space: MMIO: 0/2/0

Size (in bits): 32

Clock Gating Messages Register

DWord	Bit	Description
0	31:16	**Reserved**
	15:2	**Reserved** Access: RO Format: MBZ
	1	**CLK RST FWE Request** Access: R/W _Custom_GTIReset: BUS HCP CLK RST FWE request: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**Power Gate Request** Access: R/W _Custom_GTIReset: BUS HCP power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

HCP Qp Status Count

HCP_QP_STATUS_COUNT - HCP Qp Status Count		
Register Space:	MMIO: 0/2/0	
Access:	RO	
Size (in bits):	64	
Address:	1E9C0h	

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23:0	**Cumulative QP** Format: U24 Cumulative QP for all LCU of a Frame (Can be used for computing average QP).
1	31:15	**Reserved** Access: RO Format: MBZ
	14:8	**Frame Max CU QP** Format: U7 Valid Range: 0-51 for 8bit, 0-63 for 10bit and 0-75 for 12bit
	7	**Reserved** Access: RO Format: MBZ
	6:0	**Frame Min CU QP** Format: U7 Valid Range: 0-51 for 8bit and 0-63 for 10bit range 0-75 for 12bit

HCP Reported Bitstream Output Byte Count with header per Frame Register

HCP_BITSTREAM_BYTECOUNT_FRAME_WITH_HEADER - HCP Reported Bitstream Output Byte Count with header per Frame Register

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1E9A0h

DWord	Bit	Description
0	31:0	**HCP Bitstream Byte Count per Frame With header** Format: U32 Total number of bytes in the bitstream output per frame from the encoder. This includes header, optional tail, byte alignment, data bytes, EMU (emulation) bytes, capac-zero word insertion, and padding insertion. The optional header/optional tail includes all bits accumulated for PAK_INSERT_COMMAND when HeaderLenghtExcludeFrmSize is set to 0 and it does NOT include all bits generated by PAK_INSERT_COMMAND when HeaderLenghtExcludeFrmSize is set to 1. This count is updated for every time the internal bitstream counter is incremented and its reset at image start.

HCP Reported Bitstream Output Byte Count without header per Frame Register

HCP_BITSTREAM_BYTECOUNT_FRAME_NO_HEADER - HCP Reported Bitstream Output Byte Count without header per Frame Register

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1E9A4h

DWord	Bit	Description
0	31:0	**HCP Bitstream Byte Count per Frame Without header** Format: U32 Total number of bytes in the bitstream output per frame from the encoder. This includes optional header, optional tail, byte alignment, data bytes, EMU (emulation) bytes, capac-zero word insertion, and padding insertion. The optional header/optional tail includes all bits accumulated for PAK_INSERT_COMMAND when HeaderLenghtExcludeFrmSize is set to 0 and it does NOT include all bits generated by PAK_INSERT_COMMAND when HeaderLenghtExcludeFrmSize is set to 1. This count is updated for every time the internal bitstream counter is incremented and its reset at image start.

HCP Reported Bitstream Output CABAC Bin Count Register

HCP_CABAC_BIN_COUNT_FRAME - HCP Reported Bitstream Output CABAC Bin Count Register

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1E9ACh

This register stores the count of number of bins per frame.

DWord	Bit	Description
0	31:0	**HCP Cabac Bin Count** Default Value: 0 Format: U32 Total number of BINs in the bitstream output per frame from the encoder. This count is updated for every time the bin counter is incremented and its reset at image start.

HCP SLICE COUNT

HCP_SLICE_COUNT - HCP SLICE COUNT

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1E9C8h

DWord	Bit	Description
0	31:20	**Reserved** Access: RO Format: MBZ
	19:0	**Slice Count** Format: U20 Indicates Number of Slices in a Frame.

HCP Unit Done

HCP_UNIT_DONE - HCP Unit Done	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1C28D8h
ShortName:	HCP_UNIT_DONE_VCS0
Description:	For VDBox0
Address:	1C68D8h
ShortName:	HCP_UNIT_DONE_VCS1
Description:	For VDBox1
Address:	1D28D8h
ShortName:	HCP_UNIT_DONE_VCS2
Description:	For VDBox2
Address:	1D68D8h
ShortName:	HCP_UNIT_DONE_VCS3
Description:	For VDBox3
Address:	1E28D8h
ShortName:	HCP_UNIT_DONE_VCS4
Description:	For VDBox4
Address:	1E68D8h
ShortName:	HCP_UNIT_DONE_VCS5
Description:	For VDBox5
Address:	1F28D8h
ShortName:	HCP_UNIT_DONE_VCS6
Description:	For VDBox6
Address:	1F68D8h
ShortName:	HCP_UNIT_DONE_VCS7
Description:	For VDBox7

Unit Done Signals.

DWord	Bit	Description
0	31:26	**Reserved** Access: RO Format: MBZ

HCP_UNIT_DONE - HCP Unit Done

	25	**SFI unit Done** Format: U1
	24	**HFC unit Done** Format: U1
	23	**HSF unit Done** Format: U1
	22	**VHLF unit Done** Format: U1
	21	**HHLF unit Done** Format: U1
	20	**HED unit Done** Format: U1
	19	**HVD unit Done** Format: U1
	18	**HPP unit Done** Format: U1
	17	**HMC unit Done** Format: U1
	16	**HIT unit Done** Format: U1
	15	**HPR unit Done** Format: U1
	14	**HFE unit Done** Format: U1
	13	**HBE unit Done** Format: U1
	12	**HMXF unit Done** Format: U1
	11	**HMXB unit Done** Format: U1
	10	**HTQunit Done** Format: U1
	9	**HSSE unit Done** Format: U1
	8	**VNC unit done** Format: U1

HCP_UNIT_DONE - HCP Unit Done		
	7	**VNE unit done** Format: U1
	6	**HSAO Unit Done** Format: U1
	5	**HLC unit done** Format: U1
	4	**HLE unit done** Format: U1
	3	**HFQ unit done** Format: U1
	2	**HFT unit done** Format: U1
	1	**HRS unit done** Format: U1
	0	**HPO unit done** Format: U1

Header Type

HDR2_0_2_0_PCI - Header Type		
Register Space: PCI: 0/2/0 Size (in bits): 8		
Address: 0000Eh		
This register contains the Header Type of the IGD.		
DWord	**Bit**	**Description**
0	7	**Multi Function Status** Default Value: 0b Access: RO _Custom_GTIReset: BUS Indicates if the device is a Multi-Function Device. The Value of this register is hardwired to 0, internal graphics is a single function.
	6:0	**Header Code** Default Value: 0000000b Access: RO _Custom_GTIReset: BUS This is a 7-bit value that indicates the Header Code for the IGD. This code is hardwired to the value 00h, indicating a type 0 configuration space format.

Head pointer update

HEAD_PTR_UPDATE - Head pointer update	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	BUS
Address:	1C2D90h-1C2D93h
Name:	Head pointer Update
ShortName:	HEAD_PTR_UPDATE_VDENC_REG0
Address:	1C6D90h-1C6D93h
Name:	Head pointer Update
ShortName:	HEAD_PTR_UPDATE_VDENC_REG1
Address:	1D2D90h-1D2D93h
Name:	Head pointer Update
ShortName:	HEAD_PTR_UPDATE_VDENC_REG2
Address:	1D6D90h-1D6D93h
Name:	Head pointer Update
ShortName:	HEAD_PTR_UPDATE_VDENC_REG3
Address:	1E2D90h-1E2D93h
Name:	Head pointer Update
ShortName:	HEAD_PTR_UPDATE_VDENC_REG4
Address:	1E6D90h-1E6D93h
Name:	Head pointer Update
ShortName:	HEAD_PTR_UPDATE_VDENC_REG5
Address:	1F2D90h-1F2D93h
Name:	Head pointer Update
ShortName:	HEAD_PTR_UPDATE_VDENC_REG6
Address:	1F6D90h-1F6D93h
Name:	Head pointer Update
ShortName:	HEAD_PTR_UPDATE_VDENC_REG7

A write to this register with a value of 1 would result in head pointer update to Display.

DWord	Bit	Description
0	31:1	**Head pointer value** Access: R/W Hardware Clear The head pointer value is read only value. It is updated by HW.

HEAD_PTR_UPDATE - Head pointer update

	0	**Head pointer update enable** Access: R/W Hardware Clear This bit can be written with a value of 0 or 1. A write with 0 is ignored. A write with 1 would result in head pointer update to display if VDEnc is in WiDi session. While reading this bit always results zero. Head pointer update is achieved by performing config write to address 2005_0500 for widi session 0 when Capture Mode = 0, VDENC_PIPE_MODE_SELECT, DW 5. Head pointer update is achieved by performing config write to address 2005_0600 for widi session 1 when Capture Mode = 0, VDENC_PIPE_MODE_SELECT, DW 5. Head pointer update is achieved by performing config write to address 2024_2000 for widi session 0 when Capture Mode = 1, VDENC_PIPE_MODE_SELECT, DW 5. Head pointer update is achieved by performing config write to address 2024_2004 for widi session 1 when Capture Mode = 1, VDENC_PIPE_MODE_SELECT, DW 5. Head pointer update is achieved by performing config write to address 2024_2008 for widi session 2 when Capture Mode = 1, VDENC_PIPE_MODE_SELECT, DW 5. Head pointer update is achieved by performing config write to address 2024_200C for widi session 3 when Capture Mode = 1, VDENC_PIPE_MODE_SELECT, DW 5. Data format: 31:28 Zeros. 27:20 Tile number 19:16 Frame number (only bit 0 is used by HW) 15:0 Tail pointer pointing to pixel row.

HEVC Local APIC Retry Vector

HEVC_LAPIC_RETRY_VECT - HEVC Local APIC Retry Vector

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	0D594h

Holds the 4 last retry interrupt vectors. The retry vector register holds the last 4 values acknowledged as an interrupt retry. Retries are errors in hardware and are not expected. HUCINT handles retries by logging the interrupt vector in this register. No interrupt is actually retried, and the interrupt stimulus will be lost if a retry occurs. The system will hang eventually. A 2-bit counter (starting at reset value of 0) is used to point to the slot/byte location from which to load the next retry vector (into the 4 available slots) in sequence. This means if a 5th retry vector shows up, it will be loaded into slot 0 again (as the counter wraps around), over-writing the retry vector which existed there in slot_0.

DWord	Bit	Description
0	31:24	**Vector Slot 3** Format: U8
	23:16	**Vector Slot 2** Format: U8
	15:8	**Vector Slot 1** Format: U8
	7:0	**Vector Slot 0** Format: U8

HIP_INDEX_REG0

HIP_INDEX_REG0 - HIP_INDEX_REG0		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
SOC_Consumer:	BIOS	
Address:	1010A0h	
DWord	**Bit**	**Description**
0	31:24	**HIP_16B_Index** _Custom_GTIReset: BUS Bits 27:24 provide a 4KB index window for the GTTMMADR[16_B000h to 16_BFFFh] decode range. Bits 31:28 are reserved.
	23:16	**HIP_16A_Index** _Custom_GTIReset: BUS Bits 19:16 provide a 4KB index window for the GTTMMADR[16_A000h to 16_AFFFh] decode range. Bits 23:20 are reserved.
	15:8	**HIP_169_Index** _Custom_GTIReset: BUS Bits 11:8 provide a 4KB index window for the GTTMMADR[16_9000h to 16_9FFFh] decode range. Bits 15:12 are reserved.
	7:0	**HIP_168_Index** _Custom_GTIReset: BUS Bits 3:0 provide a 4KB index window for the GTTMMADR[16_8000h to 16_8FFFh] decode range. Bits 7:4 are reserved.

HIP_INDEX_REG1

HIP_INDEX_REG1 - HIP_INDEX_REG1		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
SOC_Consumer:	BIOS	
Address:	1010A4h	
DWord	**Bit**	**Description**
0	31:24	**HIP_16F_Index** _Custom_GTIReset: BUS Bits 27:24 provide a 4KB index window for the GTTMMADR[16_F000h to 16_FFFFh] decode range. Bits 31:28 are reserved.
	23:16	**HIP_16E_Index** _Custom_GTIReset: BUS Bits 19:16 provide a 4KB index window for the GTTMMADR[16_E000h to 16_EFFFh] decode range. Bits 23:20 are reserved.
	15:8	**HIP_16D_Index** _Custom_GTIReset: BUS Bits 11:8 provide a 4KB index window for the GTTMMADR[16_D000h to 16_DFFFh] decode range. Bits 15:12 are reserved.
	7:0	**HIP_16C_Index** _Custom_GTIReset: BUS Bits 3:0 provide a 4KB index window for the GTTMMADR[16_C000h to 16_CFFFh] decode range. Bits 7:4 are reserved.

HOTPLUG_CTL

HOTPLUG_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	44030h-44033h
Name:	Thunderbolt Hot Plug Control
ShortName:	TBT_HOTPLUG_CTL
Reset:	soft
Address:	44038h-4403Bh
Name:	Type-C Hot Plug Control
ShortName:	TC_HOTPLUG_CTL
Reset:	soft

DWord	Bit	Description
0	31	**Port8 HPD Enable** Value: 0b – Disable Value: 1b – Enable
	30	**Reserved** Access: RO Format: MBZ
	29:28	**Port8 HPD Status** Access: R/WC Value: 00b – Hot plug event not detected Value: 01b – Short pulse detected Value: 10b – Long pulse detected Value: 11b – Short and long pulses detected
	27	**Port7 HPD Enable** Value: 0b – Disable Value: 1b – Enable
	26	**Reserved** Access: RO Format: MBZ

HOTPLUG_CTL

25:24	**Port7 HPD Status** Access: R/WC Value: 00b — Hot plug event not detected Value: 01b — Short pulse detected Value: 10b — Long pulse detected Value: 11b — Short and long pulses detected
23	**Port6 HPD Enable** Value: 0b — Disable Value: 1b — Enable
22	**Reserved** Access: RO Format: MBZ
21:20	**Port6 HPD Status** Access: R/WC Value: 00b — Hot plug event not detected Value: 01b — Short pulse detected Value: 10b — Long pulse detected Value: 11b — Short and long pulses detected
19	**Port5 HPD Enable** Value: 0b — Disable Value: 1b — Enable
18	**Reserved** Access: RO Format: MBZ
17:16	**Port5 HPD Status** Access: R/WC Value: 00b — Hot plug event not detected Value: 01b — Short pulse detected Value: 10b — Long pulse detected

HOTPLUG_CTL

		11b: Short and long pulses detected
	15	**Port4 HPD Enable** 0b: Disable 1b: Enable
	14	**Reserved** Access: RO Format: MBZ
	13:12	**Port4 HPD Status** Access: R/WC 00b: Hot plug event not detected 01b: Short pulse detected 10b: Long pulse detected 11b: Short and long pulses detected
	11	**Port3 HPD Enable** 0b: Disable 1b: Enable
	10	**Reserved** Access: RO Format: MBZ
	9:8	**Port3 HPD Status** Access: R/WC 00b: Hot plug event not detected 01b: Short pulse detected 10b: Long pulse detected 11b: Short and long pulses detected
	7	**Port2 HPD Enable** 0b: Disable 1b: Enable

HOTPLUG_CTL

	6	**Reserved** Access: RO Format: MBZ
	5:4	**Port2 HPD Status** Access: R/WC 00b: Hot plug event not detected 01b: Short pulse detected 10b: Long pulse detected 11b: Short and long pulses detected
	3	**Port1 HPD Enable** 0b: Disable 1b: Enable
	2	**Reserved** Access: RO Format: MBZ
	1:0	**Port1 HPD Status** Access: R/WC 00b: Hot plug event not detected 01b: Short pulse detected 10b: Long pulse detected 11b: Short and long pulses detected

HS Invocation Counter

HS_INVOCATION_COUNT - HS Invocation Counter	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02300h
Name:	HS Invocation Counter
ShortName:	HS_INVOCATION_COUNT

This register stores the number of patch objects processed by the HS unit. E.g., A PATCHLIST_2 topology with 6 vertices would cause this counter to increment by 3 (there are 3 2-vertex patch objects in that topology).This register is part of the context save and restore.

DWord	Bit	Description
0	63:32	**HS Invocation Count UDW** Number of patch objects processed by the HS stage. Updated only when HS Enable and HS Statistics Enable are set in 3DSTATE_HS
	31:0	**HS Invocation Count LDW** Number of patch objects processed by the HS stage. Updated only when HS Enable and HS Statistics Enable are set in 3DSTATE_HS

IA Vertices Count

IA_VERTICES_COUNT - IA Vertices Count	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02310h-02317h
Name:	IA Vertices Count
ShortName:	IA_VERTICES_COUNT_RCSUNIT_BE_GEOMETRY
Address:	18310h-18317h
Name:	IA Vertices Count
ShortName:	IA_VERTICES_COUNT_POCSUNIT_BE_GEOMETRY

This register stores the count of vertices processed by VF. This register is part of the context save and restore.

DWord	Bit	Description
0	63:32	**IA Vertices Count Report UDW** Total number of vertices fetched by the VF stage. This count is updated for every input vertex as long as Statistics Enable is set in VF_STATE (see the Vertex Fetch Chapter in the 3D Volume.)
	31:0	**IA Vertices Count Report LDW** Total number of vertices fetched by the VF stage. This count is updated for every input vertex as long as Statistics Enable is set in VF_STATE (see the Vertex Fetch Chapter in the 3D Volume.)

Idle Switch Delay

IDLEDLY - Idle Switch Delay	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0223Ch-0223Fh
Name:	Idle Switch Delay
ShortName:	IDLEDLY_RCSUNIT
Address:	1823Ch-1823Fh
Name:	Idle Switch Delay
ShortName:	IDLEDLY_POCSUNIT
Address:	2223Ch-2223Fh
Name:	Idle Switch Delay
ShortName:	IDLEDLY_BCSUNIT
Address:	1C023Ch-1C023Fh
Name:	Idle Switch Delay
ShortName:	IDLEDLY_VCSUNIT0
Address:	1C423Ch-1C423Fh
Name:	Idle Switch Delay
ShortName:	IDLEDLY_VCSUNIT1
Address:	1C823Ch-1C823Fh
Name:	Idle Switch Delay
ShortName:	IDLEDLY_VECSUNIT0
Address:	1D023Ch-1D023Fh
Name:	Idle Switch Delay
ShortName:	IDLEDLY_VCSUNIT2
Address:	1D423Ch-1D423Fh
Name:	Idle Switch Delay
ShortName:	IDLEDLY_VCSUNIT3
Address:	1D823Ch-1D823Fh
Name:	Idle Switch Delay
ShortName:	IDLEDLY_VECSUNIT1
Address:	1E023Ch-1E023Fh
Name:	Idle Switch Delay

IDLEDLY - Idle Switch Delay

ShortName:	IDLEDLY_VCSUNIT4
Address:	1E423Ch-1E423Fh
Name:	Idle Switch Delay
ShortName:	IDLEDLY_VCSUNIT5
Address:	1E823Ch-1E823Fh
Name:	Idle Switch Delay
ShortName:	IDLEDLY_VECSUNIT2
Address:	1F023Ch-1F023Fh
Name:	Idle Switch Delay
ShortName:	IDLEDLY_VCSUNIT6
Address:	1F423Ch-1F423Fh
Name:	Idle Switch Delay
ShortName:	IDLEDLY_VCSUNIT7
Address:	1F823Ch-1F823Fh
Name:	Idle Switch Delay
ShortName:	IDLEDLY_VECSUNIT3
Address:	1A23Ch-1A23Fh
Name:	Idle Switch Delay
ShortName:	IDLEDLY_CCSUNIT0

The IDLEDLY register contains an Idle Delay field which specifies eight times the time stamp base units allowed for command streamer to wait before a context is switched out leading to IDLE state in Execlist mode, i.e following this context switch there is no active element available in HW to execute. Refer Time Stamp Bases subsection in Power Management chapter for time stamp base unit granularity. Example: An IDLE Delay count of 2 with Time stamp base unit value of 80ns would mean an idle delay wait of 1280ns (2*8*80). A default value of 0, means that by default, there is no restriction to wait on a context switch leading to IDLE. This register has no significance when Execlists are not enabled.

DWord	Bit	Description
0	31:21	**Reserved** Access: RO Format: MBZ
	20:0	**IDLE Delay** Format: U21 Eight times the time stamp base units allowed. Refer Time Stamp Bases subsection in Power Management chapter for time stamp base unit granularity. Example: An IDLE Delay count of 2 with Time stamp base unit value of 80ns would mean an idle delay wait of 1280ns (2*8*80).

Indirect Context Offset Pointer

INDIRECT_CTX_OFFSET - Indirect Context Offset Pointer	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	021C8h-021CBh
Name:	Indirect Context Offset Pointer
ShortName:	INDIRECT_CTX_OFFSET_RCSUNIT
Address:	181C8h-181CBh
Name:	Indirect Context Offset Pointer
ShortName:	INDIRECT_CTX_OFFSET_POCSUNIT
Address:	221C8h-221CBh
Name:	Indirect Context Offset Pointer
ShortName:	INDIRECT_CTX_OFFSET_BCSUNIT
Address:	1C01C8h-1C01CBh
Name:	Indirect Context Offset Pointer
ShortName:	INDIRECT_CTX_OFFSET_VCSUNIT0
Address:	1C41C8h-1C41CBh
Name:	Indirect Context Offset Pointer
ShortName:	INDIRECT_CTX_OFFSET_VCSUNIT1
Address:	1C81C8h-1C81CBh
Name:	Indirect Context Offset Pointer
ShortName:	INDIRECT_CTX_OFFSET_VECSUNIT0
Address:	1D01C8h-1D01CBh
Name:	Indirect Context Offset Pointer
ShortName:	INDIRECT_CTX_OFFSET_VCSUNIT2
Address:	1D41C8h-1D41CBh
Name:	Indirect Context Offset Pointer
ShortName:	INDIRECT_CTX_OFFSET_VCSUNIT3
Address:	1D81C8h-1D81CBh
Name:	Indirect Context Offset Pointer
ShortName:	INDIRECT_CTX_OFFSET_VECSUNIT1
Address:	1E01C8h-1E01CBh
Name:	Indirect Context Offset Pointer

INDIRECT_CTX_OFFSET - Indirect Context Offset Pointer

ShortName:	INDIRECT_CTX_OFFSET_VCSUNIT4
Address:	1E41C8h-1E41CBh
Name:	Indirect Context Offset Pointer
ShortName:	INDIRECT_CTX_OFFSET_VCSUNIT5
Address:	1E81C8h-1E81CBh
Name:	Indirect Context Offset Pointer
ShortName:	INDIRECT_CTX_OFFSET_VECSUNIT2
Address:	1F01C8h-1F01CBh
Name:	Indirect Context Offset Pointer
ShortName:	INDIRECT_CTX_OFFSET_VCSUNIT6
Address:	1F41C8h-1F41CBh
Name:	Indirect Context Offset Pointer
ShortName:	INDIRECT_CTX_OFFSET_VCSUNIT7
Address:	1F81C8h-1F81CBh
Name:	Indirect Context Offset Pointer
ShortName:	INDIRECT_CTX_OFFSET_VECSUNIT3
Address:	1A1C8h-1A1CBh
Name:	Indirect Context Offset Pointer
ShortName:	INDIRECT_CTX_OFFSET_CCSUNIT0

This register is used to program the offset where commands RCS_INDIRECT_CTX points to will be executed as part of engine context restore.

Programming Notes

- BlitterCS/VideoCS/VideoCS2/VideoEnhancementCS/PositionCS: This register functionality is not supported and must not be programmed for these command streamers.
- Offset of Indirect CS context must be always programmed to a command boundary and cacheline boundary inside the context image.
- Indirect context pointer itself is restored during context restore and hence Indirect Context Offset must not be programmed with value less than 0x5.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:6	**Offset of Indirect CS Context** Format: U10 This is the cache line offset for the Indirect CS context. This defaults to execute between CS and SVG context. It is not valid to program this to a value that is greater or equal to the starting offset for RS context. If context must be programmed at the end of engine context then program

INDIRECT_CTX_OFFSET - Indirect Context Offset Pointer

		then use BB_PER_CTX_PTR. **Value** / **Name** 0Dh / **[Default]**
	5:0	**Reserved** Access: RO Format: MBZ

Indirect Context Pointer

INDIRECT_CTX - Indirect Context Pointer	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	021C4h-021C7h
Name:	Indirect Context Pointer
ShortName:	INDIRECT_CTX_RCSUNIT
Address:	181C4h-181C7h
Name:	Indirect Context Pointer
ShortName:	INDIRECT_CTX_POCSUNIT
Address:	221C4h-221C7h
Name:	Indirect Context Pointer
ShortName:	INDIRECT_CTX_BCSUNIT
Address:	1C01C4h-1C01C7h
Name:	Indirect Context Pointer
ShortName:	INDIRECT_CTX_VCSUNIT0
Address:	1C41C4h-1C41C7h
Name:	Indirect Context Pointer
ShortName:	INDIRECT_CTX_VCSUNIT1
Address:	1C81C4h-1C81C7h
Name:	Indirect Context Pointer
ShortName:	INDIRECT_CTX_VECSUNIT0
Address:	1D01C4h-1D01C7h
Name:	Indirect Context Pointer
ShortName:	INDIRECT_CTX_VCSUNIT2
Address:	1D41C4h-1D41C7h
Name:	Indirect Context Pointer
ShortName:	INDIRECT_CTX_VCSUNIT3
Address:	1D81C4h-1D81C7h
Name:	Indirect Context Pointer
ShortName:	INDIRECT_CTX_VECSUNIT1
Address:	1E01C4h-1E01C7h
Name:	Indirect Context Pointer

INDIRECT_CTX - Indirect Context Pointer

ShortName:	INDIRECT_CTX_VCSUNIT4
Address:	1E41C4h-1E41C7h
Name:	Indirect Context Pointer
ShortName:	INDIRECT_CTX_VCSUNIT5
Address:	1E81C4h-1E81C7h
Name:	Indirect Context Pointer
ShortName:	INDIRECT_CTX_VECSUNIT2
Address:	1F01C4h-1F01C7h
Name:	Indirect Context Pointer
ShortName:	INDIRECT_CTX_VCSUNIT6
Address:	1F41C4h-1F41C7h
Name:	Indirect Context Pointer
ShortName:	INDIRECT_CTX_VCSUNIT7
Address:	1F81C4h-1F81C7h
Name:	Indirect Context Pointer
ShortName:	INDIRECT_CTX_VECSUNIT3
Address:	1A1C4h-1A1C7h
Name:	Indirect Context Pointer
ShortName:	INDIRECT_CTX_CCSUNIT0

This register is used to program the indirect address to be executed between CS and SVG engine context if enabled. This will only get executed due to regular context save/restore and not during power restore. This register is part of the execution list context and will be executed per context. Only supported if execution list is enabled. There is no preempting workloads within this context.

Programming Notes

BlitterCS/VideoCS/VideoCS2/VideoEnhancementCS/PositionCS: This register functionality is not supported and must not be programmed for these command streamers.

The following commands are not supported within Render CS indirect context:

Command Name
MI_WAIT_FOR_EVENT
MI_ARB_CHECK
MI_REPORT_HEAD
MI_TOPOLOGY_FILTER
MI_SET_CONTEXT
MI_SEMAPHORE_WAIT in Memory Poll Mode is not supported. MI_SEMAPHORE_WAIT in register poll mode is supported.
MI_BATCH_BUFFER_START
MI_CONDITIONAL_BATCH_BUFFER_END

INDIRECT_CTX - Indirect Context Pointer

GPGPU_WALKER
3DPRIMITIVE
MI_BATCH_BUFFER_END

DWord	Bit	Description
0	31:6	**Indirect CS Context Address** Format: GraphicsAddress[31:6] Pointer to the Context in memory to be executed as a batch.
	5:0	**Size of Indirect CS Context** Format: U6 This is the size of the Indirect Context for CS. This size supports up to 63 cache lines worth of commands where a cache line is 64B. If programmed to zero then the indirect fetch of the CS context is disabled. Value: [0,63] Name:

infcp Vdbox unit Level Clock Gating override during rstflow

INFMISCCP9568 - infcp Vdbox unit Level Clock Gating override during rstflow

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	09568h

Micellaneous Clocking Config Bits

DWord	Bit	Description
0	31	**Reserved**
	30:0	**Reserved** Access: RO Format: MBZ

INF Power Context Save request

INFCGCTL9564 - INF Power Context Save request

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	09564h

DWord	Bit	Description
0	31:16	**Reserved**
	15:10	**Reserved** Access: RO Format: MBZ
	9	**Power context save request** Access: R/W Set _Custom_GTIReset: BUS Power Context Save Request 1'b0 : Power context save is not being requested 1'b1 : Power context save is being requested CPUnit self-clears this bit upon sampling.
	8:0	**Power Context Save request crdit count** Access: R/W _Custom_GTIReset: BUS QWord Credits for Power Context Save Request Minimum Credits = 1 : Unit may send 1 QWord pair (enough for first LRI at least) Maximum Credits = 511 : Unit may send 511 QWord pairs A QWord pair is defined as a 32-bit register address and the corresponding 32-bits of register data. Note that the LRI header and END commands are 64-bits each (32-bit command followed by 32-bit NOOP) and will consume one QWord credit. Only valid with PWRCTX_SAVE_REQ (Bit9).

INF unit Level Clock Gating Control 9560

INFCGCTL9560 - INF unit Level Clock Gating Control 9560

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 09560h

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31:17	**Reserved** Access: RO Format: MBZ
	16:10	**Reserved** Access: RO Format: MBZ
	9	**Reserved**
	8	**SnoopFilter Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS **Description** Snoop filter Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always) **Value** / **Name** 1b / **[Default]**
	7:0	**Reserved** Access: RO Format: MBZ

Instruction Parser Mode Register

INSTPM - Instruction Parser Mode Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	020C0h-020C3h
Name:	Instruction Parser Mode Register
ShortName:	INSTPM_RCSUNIT
Address:	180C0h-180C3h
Name:	Instruction Parser Mode Register
ShortName:	INSTPM_POCSUNIT
Address:	220C0h-220C3h
Name:	Instruction Parser Mode Register
ShortName:	INSTPM_BCSUNIT
Address:	1C00C0h-1C00C3h
Name:	Instruction Parser Mode Register
ShortName:	INSTPM_VCSUNIT0
Address:	1C40C0h-1C40C3h
Name:	Instruction Parser Mode Register
ShortName:	INSTPM_VCSUNIT1
Address:	1C80C0h-1C80C3h
Name:	Instruction Parser Mode Register
ShortName:	INSTPM_VECSUNIT0
Address:	1D00C0h-1D00C3h
Name:	Instruction Parser Mode Register
ShortName:	INSTPM_VCSUNIT2
Address:	1D40C0h-1D40C3h
Name:	Instruction Parser Mode Register
ShortName:	INSTPM_VCSUNIT3
Address:	1D80C0h-1D80C3h
Name:	Instruction Parser Mode Register
ShortName:	INSTPM_VECSUNIT1
Address:	1E00C0h-1E00C3h
Name:	Instruction Parser Mode Register

INSTPM - Instruction Parser Mode Register

ShortName:	INSTPM_VCSUNIT4
Address:	1E40C0h-1E40C3h
Name:	Instruction Parser Mode Register
ShortName:	INSTPM_VCSUNIT5
Address:	1E80C0h-1E80C3h
Name:	Instruction Parser Mode Register
ShortName:	INSTPM_VECSUNIT2
Address:	1F00C0h-1F00C3h
Name:	Instruction Parser Mode Register
ShortName:	INSTPM_VCSUNIT6
Address:	1F40C0h-1F40C3h
Name:	Instruction Parser Mode Register
ShortName:	INSTPM_VCSUNIT7
Address:	1F80C0h-1F80C3h
Name:	Instruction Parser Mode Register
ShortName:	INSTPM_VECSUNIT3
Address:	1A0C0h-1A0C3h
Name:	Instruction Parser Mode Register
ShortName:	INSTPM_CCSUNIT0

The INSTPM register is used to control the operation of the Instruction Parser. Certain classes of instructions can be disabled (ignored) - often useful for detecting performance bottlenecks. Also, Synchronizing Flush operations can be initiated - useful for ensuring the completion (vs. only parsing) of rendering instructions.

Programming Notes

- If an instruction type is disabled, the parser will read those instructions but not process them.
- Error checking will be performed even if the instruction is ignored.
- All Reserved bits are implemented.
- This Register is saved and restored as part of Context.

DWord	Bit	Description
0	31:16	**Mask** Access: WO Format: Mask Masks: These bits serve as write enables for bits 15:0. If this register is written with any of these bits clear the corresponding bit in the field 15:0 will not be modified. Reading these bits always returns 0s.

INSTPM - Instruction Parser Mode Register

15 — Register Poll Mode Semaphore Wait Event IDLE message Disable

This bit controls the DOP CG behavior of CS while waiting for pending semaphore wait for event to be satisfied in register poll mode of operation.

Value	Name	Description
0	**[Default]**	When Rest, CS trigger DOP CG on an unsuccessful semaphore wait in register poll mode of operation.
1		When Set, CS doesn't trigger DOP CG on an unsuccessful semaphore wait in register poll mode of operation.

14 — Ignore Posh Start and Posh Enable fields in Batch Start command

Source:	RenderCS, PositionCS

This bit controls the execution of ring buffers and batch buffer by PositionCS and RenderCS.

Value	Name	Description
0	**[Default]**	PositonCS and RenderCS consider the Posh Start and Posh Enable fields programmed in the MI_BATCH_BUFFER_START command and execute accordingly. PositionCS parses (doesnt execute) the commands programmed in the ring buffer.
1		PositonCS and RenderCS ignores the Posh Start and Posh Enable fields programmed in the MI_BATCH_BUFFER_START command. PositionCS executes all the commands programmed in the ring buffer and the batch buffers.

13 — Enable Semaphore Register Poll Mask

This bit enables masking of the register data red prior to semaphore comparison on a register poll mode.

Value	Name	Description
1		In register poll mode of operation Semaphore Address Upper Dword will be used as mask and applied to the data red form the register prior to comparison. Mask Bit Set to 0 indicate the corresponding bit red from the register is considered as it is unmodified for comparison. Mask Bit Set to 1 indicate the corresponding bit red from the register is forced to 0 for comparison.
0	**[Default]**	Regular comparison with no mask applied.

12 — PreFetch Disable Status

This bit gets programmed on executing MI_ARB_CHK command with mask bit set for Pre-Fetch Disable. This bit is used to context save/restore the Pre-Fetch Disable status on a context switch. This bit must not be directly written by software.

11 — CLFLUSH Toggle

Source:	RenderCS, PositionCS
Access:	RO
Format:	U1

INSTPM - Instruction Parser Mode Register		
		This bit changes polarity each time the MI_CLFLUSH command completes. This bit is Read Only.
	10	**Reserved**
	9:0	**Reserved** Access: RO Format: MBZ

Interrupt Group 0

INTR_GROUP_0 - Interrupt Group 0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

The two Interrupt Group Registers (0 and 1) use their total 64 bits to map 16 groups to 16 base values (14 of which are currently used, 13 used prior) within the 256-bit interrupt vector. Each group (8 per register) is assigned a 4-bit value that specifies its position in the 256 vector.
The position of an interrupt within a group of 16 interrupts (as assigned by the group value) is HW determined. Groups with a single interrupt use the highest possible encoding (0XF), groups with multiple interrupts use a HW encoding as specified. Since Doorbell interrupts may use the same group value for more than one group of interrupts, each doorbell interrupt is assigned (by HW) a unique position in each group of 16. In other words, even though Doorbell groups have a single interrupt each, these do not all use the highest possible encoding, the encoding is as specified.
A value of 0 turns off that entire group of interrupts.

Programming Notes
When programmed to send doorbell interrupts to Host, only one group of 32 doorbells may be enabled. All other (seven) doorbell groups MUST be disabled by making their group value 0.
This register is saved in the power context

DWord	Bit	Description
0	31:28	**Doorbell Group 7** Interrupt vector group (dword index) for doorbell group 7, which covers input doorbells 224 through 255 Encoding: 0XF
	27:24	**Doorbell Group 6** Interrupt vector group (dword index) for doorbell group 6, which covers input doorbells 192 through 223 Encoding: 0XE
	23:20	**Doorbell Group 5** Interrupt vector group (dword index) for doorbell group 5, which covers input doorbells 160 through 191 Encoding: 0XD
	19:16	**Doorbell Group 4** Interrupt vector group (dword index) for doorbell group 4, which covers input doorbells 128 through 159 Encoding: 0XC
	15:12	**Doorbell Group 3** Interrupt vector group (dword index) for doorbell group 3, which covers input doorbells 96 through 127 Encoding: 0XB

INTR_GROUP_0 - Interrupt Group 0		
	11:8	**Doorbell Group 2** Interrupt vector group (dword index) for doorbell group 2, which covers input doorbells 64 through 95 Encoding: 0XA
	7:4	**Doorbell Group 1** Interrupt vector group (dword index) for doorbell group 1, which covers input doorbells 32 through 63 Encoding: 0X9
	3:0	**Doorbell Group 0** Interrupt vector group (dword index) for doorbell group 0, which covers input doorbells 0 through 31 Encoding: 0X8

Interrupt Group 1

INTR_GROUP_1 - Interrupt Group 1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

The two Interrupt Group Registers (0 and 1) use their total 64 bits to map 16 groups to 16 base values (14 of which are currently used, 13 used prior) within the 256-bit interrupt vector. Each group (8 per register) is assigned a 4-bit value that specifies its position in the 256 vector.
The position of an interrupt within a group of 16 interrupts (as assigned by the group value) is HW determined. Groups with a single interrupt use the highest possible encoding (0XF), groups with multiple interrupts use a HW encoding as specified.
Since Doorbell interrupts may use the same group value for more than one group of interrupts, each doorbell interrupt is assigned (by HW) a unique position in each group of 16. In other words, even though Doorbell groups have a single interrupt each, these do not all use the highest possible encoding, the encoding is as specified.
A value of 0 turns off that entire group of interrupts.

Programming Notes
This register is saved in the power context

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23:20	**Host Group** Interrupt vector group for Host. Encoding (see description above) used: 0xF
	19:16	**DMA/Timer/FLR Group** Interrupt vector group (dword index) for the DMA, Timer, FLR group. Encoding (see description above): FLR= 0XD, Timer = 0xE, DMA = 0xF
	15:12	**Display Events Group** Interrupt vector group (dword index) for display events Encoding (see description above): Solicited = 0XF Unsolicited = 0XE
	11:8	**IOMMU Interrupts Group** Interrupt vector group (dword index) for IOMMU interrupts Encoding (see description above): GuC Catastrophic Error = 0xF, Non GuC Catastrophic errors = 0XC, Page Faults = 0XB, Page Response = 0XA
	7:4	**Semaphore Group** Description Interrupt vector group (dword index) for semaphores Encoding of interrupts: VF0 = 0xF, VF1 = 0xE, VF2 = 0xD, VF3 = 0xC, VF4 = 0xB, VF5 = 0xA, VF6 = 0x9, VF7 = 0x8

INTR_GROUP_1 - Interrupt Group 1		
	3:0	**Engine Interrupts Group** Interrupt vector group (dword index) for engine interrupts Encoding: INTDW0 = 0XF, INTDW1 = 0XE

Interrupt Line

INTRLINE_0_2_0_PCI - Interrupt Line		
Register Space:	PCI: 0/2/0	
Size (in bits):	8	
Address:	0003Ch	
This register is used to communicate interrupt line routing information. The device itself does not use this value, rather it is used by device drivers and operating systems to determine priority and vector information.		
DWord	**Bit**	**Description**
0	7:0	**Interrupt Connection** Default Value: 00000000b Access: R/W _Custom_GTIReset: BUS Used to communicate interrupt line routing information. POST software writes the routing information into this register as it initializes and configures the system. The value in this register indicates to which input of the system interrupt controller the device's interrupt pin is connected.

Interrupt Pin

INTRPIN_0_2_0_PCI - Interrupt Pin		
Register Space:	PCI: 0/2/0	
Size (in bits):	8	
Address:	0003Dh	
This register tells which interrupt pin the device uses.		
DWord	**Bit**	**Description**
0	7:0	**Interrupt Pin Value** Default Value: 00000001b Access: RO _Custom_GTIReset: BUS As a single function device, the IGD specifies INTA# as its interrupt pin. Hardwired to 01h = INTA#.

I/O Base Address

IOBAR_0_2_0_PCI - I/O Base Address

Register Space:	PCI: 0/2/0
Size (in bits):	32
Address:	00020h

This register provides the Base offset of the I/O registers within Device #2. Bits 15:6 are programmable allowing the I/O Base to be located anywhere in 16bit I/O Address Space. Bits 2:1 are fixed and return zero; bit 0 is hardwired to a one indicating that 8 bytes of I/O space are decoded. Access to the 8Bs of IO space is allowed in PM state D0 when IO Enable (PCICMD bit 0) set. Access is disallowed in PM states D1-D3 or if IO Enable is clear or if Device #2 is turned off or if Internal graphics is disabled through the fuse or fuse override mechanisms.Note that access to this IO BAR is independent of VGA functionality within Device #2. If accesses to this IO bar is allowed then all 8, 16 or 32 bit IO cycles from IA cores that falls within the 8B are claimed.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:6	**IO Base Address** Default Value: 0000000000b Access: R/W _Custom_GTIReset: BUS Set by the OS, these bits correspond to address signals [15:6].
	5:3	**Reserved** Access: RO Format: MBZ
	2:1	**Memory Type** Default Value: 00b Access: RO _Custom_GTIReset: BUS Hardwired to 0s to indicate 32-bit address.
	0	**Memory/IO Space** Default Value: 1b Access: RO _Custom_GTIReset: BUS Hardwired to "1" to indicate IO space.

Jump Location

JMP_DEST - Jump Location

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32

The Jump Location used by HW shall be in DW1 of the last Cacheline. The 32 bits are split up into two fields: one to specify the uOS load location, another to specify where in the uOS image to jump to. BootROMs shall use the 2 fields to compute uOS jump point.

Programming Notes

This register is saved in the power context

DWord	Bit	Description
0	31:18	**uOS Intended Base** Default Value: 0h Indicate the location at which uOS shall be loaded. BootROM shall verify that the ukernel was loaded at this intended location (by inspecting DMA_OUS_BASE register) before transferring control to the uOS. These 14 bits represent a cacheline aligned address [19:6] for the uOS location, allowing a uOS to be loaded at an address of upto 1MB in the MinuteIA address space.
	17:0	**uOS Offset** Default Value: 4000h Indicate the byte Offset (from the start of the uOS image) of the first executable instruction within the uOS image. This field allows the BootROM to determine where in the uOS image to transfer control to.

KVMR_SPR_COLOR_CTL

KVMR_SPR_COLOR_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	45030h-45033h
Name:	Kvmr Sprite Color Control
ShortName:	KVMR_SPR_COLOR_CTL
Reset:	soft

DWord	Bit	Description
0	31	**Enable Color Processing** This field enables the sRGB de-gamma, color space conversion to BT2020 and tone mapping with the programmed tone mapping factor. Value: 1b — Name: Enable Value: 0b — Name: Disable
	30:10	**Reserved** Access: RO Format: MBZ
	9:0	**Tone Mapping Factor** This field specifies the tone mapping factor. Each color component gets corrected with this programmed 10 bit fractional value.

L3 Allocation Control Register

L3ALLOCREG - L3 Allocation Control Register

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B134h
Name:	L3 Allocation Control Register
ShortName:	L3ALLOCREG
Address:	0B234h
ShortName:	L3ALLOCREG_CCS0

This register controls the allocation of various sections of the L3 cache

DWord	Bit	Description
0 L3CacheLayout: 2x2 array	31:25	**All L3 Client Pool** Access: R/W Number of ways allocated for the all client pool. This is a combined pool for all clients. Value: 68h — Name: default **[Default]** Value: [0h,68h] — Name: **Programming Notes** When this field is non-zero, **DC Way Assignment** and **Read Only Client Pool** should be 0KB.
	24:18	**DC Way Assignment** Access: R/W Number of ways allocated for HDC. Value: 0h — Name: **[Default]** Value: [0h,68h] — Name: **Programming Notes** Note: This field must be 0KB if All L3 Client Pool is non-zero.
	17:11	**Read Only Client Pool** Access: R/W Number of ways allocated for Read Only L3 clients. This is a combined pool for all Read Only clients.

L3ALLOCREG - L3 Allocation Control Register

Value	Name
0h	**[Default]**
[0h,68h]	

Programming Notes
Note: This field must be 0KB if All L3 Client Pool is non-zero.

10 **Reserved**

Access:	RO
Format:	MBZ

9 **Reserved**

Access:	RO
Format:	MBZ

8 **Reserved**

Access:	RO
Format:	MBZ

7:1 **URB Allocation**

Access:	R/W

Number of ways allocated for URB

Value	Name
10h	**[Default]**
20h	

0 **Allocation Error status**

Access:	RO

This bit indicates a programming error in L3 allocation registers (L3ALLOCREG and L3TCCNTRLREG).
0 : Indicates no error in programming
1 : Indicates that the L3 allocation programming is incorrect.L3 will use the most recent valid allocation OR default programming.

L3 Cache Runtime ECC capture Register

L3_ECC_CAPTURE_REG - L3 Cache Runtime ECC capture Register

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B144h

This register shows the captured value of the ECC calculated by the internal logic for L3 writes.

DWord	Bit	Description
0	31:20	**Reserved** Access: RO Format: MBZ
	19:0	**L3 Cache Latched ECC Value** Access: RO This field holds the value of the ECC generated by the L3 cache pipeline for the preceding write cycle.

L3 Cache Runtime ECC Test Control Register

L3_ECC_TEST_CTL - L3 Cache Runtime ECC Test Control Register

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B13Ch

This register is used to control the run time testing of the ECC logic in the L3 cache.

DWord	Bit	Description
0	31:22	**Reserved** Access: RO Format: MBZ
	21	**L3 ECC Latch Enable** Access: R/W This bit enables the latching of the ECC value generated for any write access made to the L3. The latched value of the ECC can be read from the "L3 Cache Runtime ECC capture Register". 0 – **[Default]** – Disable latching. 1 – Enable the latching
	20	**L3 ECC Override Enable** Access: R/W This bit enables the overriding of the ECC value generated for any write access made to the L3. The value in the field below will be used to override the ECC internally generated. 0 – **[Default]** – Disable override of the ECC value 1 – Enable the override of the ECC value
	19:0	**L3 ECC Override Value** Default Value: 00h Access: R/W This is the value to be used as the override for the ECC when enabled in the bit field above.

L3 Control Register

L3CNTLREG - L3 Control Register		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	07034h	
Name:	L3 Control Register	
ShortName:	L3CNTLREG	

DWord	Bit	Description
0	31:0	**Reserved** Access: R/W Format: PBC _Custom_GTIReset: DEV

L3 Multi Context Reserved1

L3RCS0RSVD1 - L3 Multi Context Reserved1		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
_Custom_GTIReset:	DEV	
Address:	0B168h	
ShortName:	L3RCS0RSVD1	
Address:	0B268h	
ShortName:	L3CCS0RSVD1	

DWord	Bit	Description
0	31:0	**SCRATCH** Access: R/W

L3 Multi Context Reserved2

L3 Multi Context Reserved2			
Register Space:	MMIO: 0/2/0		
Access:	R/W		
Size (in bits):	32		
_Custom_GTIReset:	DEV		
Address:	0B16Ch		
ShortName:	L3RCS0RSVD2		
Address:	0B26Ch		
ShortName:	L3CCS0RSVD2		
DWord	**Bit**	**Description**	
0	31:0	**SCRATCH**	
		Access:	R/W

L3 Node Units Idle Status

L3_NODE_IDLE - L3 Node Units Idle Status			
Register Space:	MMIO: 0/2/0		
Access:	RO Variant		
Size (in bits):	32		
_Custom_GTIReset:	DEV		
Address:	0B0B0h		
Status register for l3 node units idle indication. Value 1 means idle			
DWord	**Bit**	**Description**	
0	31:10	**Reserved**	
		Access:	RO
		Format:	MBZ
	9	**IDLE_LTISEQSL**	
	8	**IDLE_GAPL3**	
	7	**IDLE_LNS**	
	6	**IDLE_LSN3**	
	5	**IDLE_LSN2**	
	4	**IDLE_LSN1**	
	3	**IDLE_LSN0**	
	2	**IDLE_LNIC**	
	1	**IDLE_LNIB**	
	0	**IDLE_LNE**	

L3 Parameter Information Register

L3PARAMINFO - L3 Parameter Information Register

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B164h

Programming Notes

GT_L3_NUM_WAYS_NOTAG + GT_L3_NUM_WAYS gives total number of ways for one L3 bank. Storage per way can be calculated as Total Storage per L3 bank / (GT_L3_NUM_WAYS_NOTAG + GT_L3_NUM_WAYS). L3ALLOCREG and L3TCCNTLREG can be programmed appropriately using storage/way information.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:8	**GT_L3_NUM_WAYS_NOTAG** This field indicates number of ways in L3 bank without tag storage. These number of ways must be allocated for URB.
	7:0	**GT_L3_NUM_WAYS** This field indicates number of tagged ways in L3 bank which is maximum number of ways that can be allocated for cachable data.

L3 SQC register 4

L3SQCREG4 - L3 SQC register 4

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B118h
Address:	0B218h
ShortName:	L3SQCREG4_CCS0

DWord	Bit	Description
0	31:21	**Reserved** Access: RO Format: MBZ
	20:15	**Reserved** Access: RO Format: MBZ
	14:11	**Clean Evict Regulation Window** Access: R/W This field determines the time of the clean-eviction evaluation window. Value: 0000b — Name: **[Default]** **Programming Notes** The value provided in this window is multiplied by 16 in hardware. This allows for a range of 0 to 240 clocks for the evaluation window.
	10:7	**Clean Evict Regulation Count** Access: R/W Within the evaluation window, this field defines the number of allowable clean evictions to be sent by the L3BANK. If the limit is reached in a given window, L3BANK will downgrade any further clean evictions (within that window) to silent evictions. This evaluation mechanism is only applicable if clean-evicts are enabled. Value: 0000b — Name: lp_default **[Default]** **Programming Notes** The value programmed in to this register is multiplied by 2 in hardware. This allows for a limit of 0 to 30 clean evictions within the programmable window.

L3SQCREG4 - L3 SQC register 4

	6	**Clean Evict Disable** Access: R/W Clean evicts are disabled when set Value: 1 — Name: **[Default]**
	5:0	**Reserved** Access: RO Format: MBZ

L3 SQC register 5

L3SQCREG5 - L3 SQC register 5		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
_Custom_GTIReset:	DEV	
Address:	0B158h	
Address:	0B258h	
ShortName:	L3SQCREG5_CCS0	

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23	**Reserved** Access: RO Format: MBZ
	22	**Coherent Partial Write Merge Enable** Default Value: 1 Access: R/W Format: Enable _Custom_GTIReset: DEV Enables partial write merge optimization for coherent surfaces.
	21	**Compressible Partial Write Merge Enable** Default Value: 1 Access: R/W Format: Enable _Custom_GTIReset: DEV Enables partial write merge optimization for compressible surfaces.
	20	**LRU update control for detecting oldest partial writes** Default Value: 0 Access: R/W '0' : (default) - Update LRU bit for parent and also for the child merge event '1' : Update LRU bit only for the parent

L3SQCREG5 - L3 SQC register 5

	19:10	**Reserved** Access: RO Format: MBZ
	9:0	**L3 cache partial Write merge timer initial value** Default Value: 1FFh Default Access: R/W If partial write merging is enabled, this timer controls the window to allow partial writes to merge together. The value is first loaded when an eligible request arrives, and is reloaded every time another partial write is collapsed in to the parent **Programming Notes** The value of this register must be nonzero if any partial write merging is enabled, consisting of the CZ + L3 Partial Write Merge Enables in TCCNTLREG*, and the Coherent + Compressible Partial Write Merge Enables in LSCREG5*

L3 SQC register 6

L3SQCREG6 - L3 SQC register 6		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	DEV	
Address:	0B15Ch	
Address:	0B25Ch	
ShortName:	L3SQCREG6_CCS0	

DWord	Bit	Description
0	31:14	**Reserved** Access: RO Format: MBZ
	13:7	**Maximum number of partial write chains** Access: R/W Value / Name / Description: 10h **[Default]** 4-64 Upper bound is limited by the number of L3 SuperQueue slots
	6:0	**Maximum number of slots that can wait for timer to expire** Access: R/W Value / Name / Description: 20h **[Default]** 4-64 Upper bound is limited by the number of L3 SuperQueue slots

L3 SQC registers 1

L3SQCREG1 - L3 SQC registers 1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B100h
Address:	0B200h
ShortName:	L3SQCREG1_CCS0

Programming Notes

Workaround Credits between LNI/LSQC are not updated in case of Render DOP gating condition- DOP Render clock ungating needs to happen before L3SQCREG1 is programmed and it should happen through the driver.

DWord	Bit	Description
0	31:30	**Lookup Fifo2or3 over Fifo1 counter** Default Value: 10b Access: R/W Fifo2 or Fifo3 will be selected as winner as many times as described by this counter before fifo1 that is also available can be selected as winner. Valid values are as follows: ^M 0 = 1 cnt. 1 = 2 cnt. 2 = 4 cnt (Default). 3 = 8 cnt. lbcf_csr_lsqc_lookup_f2_f3_cnt[1:0]
	29:28	**Lookup Fifo1 over Fifo2or3 counter** Default Value: 11b Access: R/W Fifo1 will be selected as winner as many times as described by this counter before some other fifo (any between Fifo2/Fifo3) that is also available can be selected as winner. Valid values are as follows: ^M 0 = 1 cnt. 1 = 2 cnt. 2 = 4 cnt 3 = 8 cnt (Default). lbcf_csr_lsqc_lookup_f1_cnt[1:0]

L3SQCREG1 - L3 SQC registers 1

27:26 **Lookup fifo3 counter**

Default Value:	00b
Access:	R/W

Fifo3 will be selected as winner as many times as described by this counter before Fifo2 that is also available can be selected as winner.
Valid values are as follows:
^M
0 = 1 cnt (Default).
1 = 2 cnt.
2 = 4 cnt
3 = 8 cnt.
lbcf_csr_lsqc_lookup_f3_cnt[1:0]

25:24 **Lookup fifo2 counter**

Default Value:	11b
Access:	R/W

Fifo2 will be selected as winner as many times as described by this counter before Fifo3 that is also available can be selected as winner.
Valid values are as follows:
^M
0 = 1 cnt.
1 = 2 cnt.
2 = 4 cnt
3 = 8 cnt (Default).
lbcf_csr_lsqc_lookup_f2_cnt[1:0]

23:17 **L3SQ General Priority Credit Initialization**

Access:	R/W

Description
Number of general priority credits that SQ presents to L3 Arbiter blocks. This inherently also determines the depth of the SQ; reduce the number of credits and SQ uses fewer slots. The number of general priority credits is always greater than that of high priority credits. Total number of general and high priority credits should be less than or equal to 64. This implies that the sum of the programmed general priority and high priority values should be less than or equal to 32 The number of general priority credits is equal to double the value written in this register. Example: [00001b = 2 credits; 00100b = 8 credits] Valid values for general priority credits can range from: [1 to 32]. Other values are reserved Signal name: lbcf_csr_lsqc_gen_credit_init

Value	Name
0100000b	**[Default]**

L3SQCREG1 - L3 SQC registers 1

		[2h,20h]

Programming Notes

The number of general priority credits is always greater than that of high priority credits. Total number of general and high priority credits should be less than or equal to 64. This implies that the sum of the programmed general priority and high priority values should be less than or equal to 32 lbcf_csr_lsqc_gen_credit_init + lbcf_csr_lsqc_hp_credit_init less than or equal to 32

16:11 **L3SQ High Priority Credit Initialization**

Access:	R/W

Number of high priority credits that SQ presents to L3 Arbiter blocks. This inherently also determines the depth of the SQ; reduce the number of credits and SQ uses fewer slots.
The number of high priority credits is always lesser than that of general priority credits. Total number of general and high priority credits should be less than or equal to 64. This implies that the sum of the programmed general priority and high priority values should be less than or equal to 32
The number of high priority credits is equal to double the value written in this register. Example: [00001b = 2 credits; 00100b = 8 credits]
Valid values for high priority credits can range from: [0 to 15]. Other values are reserved
Signal name: lbcf_csr_lsqc_gen_credit_init

Value	Name
000000b	**[Default]**

Programming Notes

The number of high priority credits is always lesser than that of general priority credits. Total number of general and high priority credits should be less than or equal to 64. This implies that the sum of the programmed general priority and high priority values should be less than or equal to 32 lbcf_csr_lsqc_gen_credit_init + lbcf_csr_lsqc_hp_credit_init less than or equal to 32

10 **Reserved**

Access:	RO
Format:	MBZ

9 **L3SQ Read Once Enable for Sampler Client**

Access:	R/W

L3SQ Read Once Enable for Sampler Client (SQROE):
Enables Read Once indications to L3 Cache from SQ. Once enabled, any reads from Sampler client (MT) are sent as Read Once.
0 = (default) Reads from Sampler clients issue Read to L3 Cache.
1 = Reads from Sampler clients issue Read Once to L3 Cache.
lbcf_csr_sampler_readonce_en.

L3SQCREG1 - L3 SQC registers 1

8 — Error Detection Behavior Control

Access:	R/W

The L3 error detection can be enabled to hang the GPU on a non-recoverable error due to SER type events. Such option will be used when corresponding context has data consistency requirements. Once error detection is enabled, s/w has to initialize URB or SLM to all 0's (based on usage model) prior to execution of the workload. Initialization is required to clean up the error detection logic and syndrome tracking.

Value	Name	Description
0	**[Default]**	RTL does not hang on parity errors or double bit error
1		RTL enforces a hang on parity errors or double bit error

7 — GPGPU L3 Credit Mode Enable

Default Value:	0
Access:	R/W

This bit is required to be enabled under GPGPU workloads to provide the MAX latency coverage form L3 cache. It will override the registers 0xB100[18:14] and 0xB100[23:19], to 0 and the maximum value respectively.

6 — Reserved

Access:	RO
Format:	MBZ

5 — Flush all non coherent lines

The L3 will support destination based flushing for the unified sections of the cache. i.e., if we get a DC flush, even if we have a unified cache, only the lines brought in by DC will be flushed. Similarly, tile cache flushes only clean out lines brought in by the C/Z clients into the unified section.

When a pipeline flush or a tile cache flush is received and if this bit is set, then all the modified non-coherent lines irrespective of DC/C/Z in the unified cache section will be flushed. Also, all the Shared lines in the unified cache section will be downgraded to invalid for all Non- Coherent lines.

This bit will have no impact when DC and tile cache are in separate sections. Also note that this bit will not initiate the invalidation of RO lines in the cache. That is only controlled by the RO invalidation commands sent to the L3 on a per context basis.

Value	Name	Description
0	**[Default]**	By default, the full flush mode will be disabled.

4:0 — Reserved

Access:	RO
Format:	MBZ

L3 Unslice IDLE Status Register

L3_UNSL_IDLE - L3 Unslice IDLE Status Register		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	DEV	
Address:	0B414h	
DWord	**Bit**	**Description**
0	31:2	**Reserved** Access: RO Format: MBZ
	1	**LSN Endpoint Unit IDLE Indication** Access: RO Variant LSN Endpoint Unit IDLE. This bit is not used and assigned to zero.
	0	**LTISEQunit Unit IDLE Indication** Access: RO Variant LTISEQunit IDLE

LBCF Render config save msg

LBCFRCSR - LBCF Render config save msg

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B3FCh

This register is not context saved and is written by CS unit

DWord	Bit	Description
0	31:17	**Reserved** Access: RO Format: MBZ
	16	**Render Context Save Request Mask** Access: R/W Hardware Clear Value / Name / Description 0 / default **[Default]** / Render context save request is ignored, and LBCFRCSR[0] is invalid. 1 / / LBCFRCSR[0] is valid.
	15:1	**Reserved** Access: RO Format: MBZ
	0	**Render Context save Request** Access: R/W Hardware Clear Value / Name / Description 0 / default **[Default]** / Render context save is not being requested. 1 / / Render context save is being requested. This bit is self-cleared upon sampling.

LINKM

LINKM	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60040h-60043h
Name:	Transcoder Link M Value 1
ShortName:	TRANS_LINKM1_A
Reset:	soft
Address:	61040h-61043h
Name:	Transcoder Link M Value 1
ShortName:	TRANS_LINKM1_B
Reset:	soft
Address:	62040h-62043h
Name:	Transcoder Link M Value 1
ShortName:	TRANS_LINKM1_C
Reset:	soft
Address:	63040h-63043h
Name:	Transcoder Link M Value 1
ShortName:	TRANS_LINKM1_D
Reset:	soft

Description
This register is double buffered to update on the next MSA after LINKN is written.

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23:0	**Link M value** This field is the link M value for external transmission in the Main Stream Attributes.

LINKN

LINKN	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60044h-60047h
Name:	Transcoder Link N Value 1
ShortName:	TRANS_LINKN1_A
Reset:	soft
Address:	61044h-61047h
Name:	Transcoder Link N Value 1
ShortName:	TRANS_LINKN1_B
Reset:	soft
Address:	62044h-62047h
Name:	Transcoder Link N Value 1
ShortName:	TRANS_LINKN1_C
Reset:	soft
Address:	63044h-63047h
Name:	Transcoder Link N Value 1
ShortName:	TRANS_LINKN1_D
Reset:	soft

Description

This register is double buffered to update on the next MSA after written. **Writes to this register arm M/N registers for this transcoder.**

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23:0	**Link N value** This field is the link N value for external transmission in the Main Stream Attributes and VB-ID.

LNCF MOCS Register 0

LNCFCMOCS0 - LNCF MOCS Register 0

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B020h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Value	Name	Description
0h	UPPER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC **[Default]**	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

LNCFCMOCS0 - LNCF MOCS Register 0

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	10 ^ 16
1	11 ^ 17
2	12 ^ 18

16 **Upper MOCS Index - Enable Skip Caching**

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 **Lower MOCS Index Mask Bit**

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 **Reserved**

Access:	RO
Format:	MBZ

7:6 **Reserved**

Access:	RO
Format:	MBZ

5:4 **Lower MOCS Index - L3 Cacheability Control**

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT **[Default]**	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS0 - LNCF MOCS Register 0

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 10 ^ 16 1 / 11 ^ 17 2 / 12 ^18
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE **[Default]** 1h / LOWER_ESC_ENABLE

LNCF MOCS Register 1

LNCFCMOCS1 - LNCF MOCS Register 1

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B024h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 3h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Value	Name	Description
0h	UPPER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC **[Default]**	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

LNCFCMOCS1 - LNCF MOCS Register 1

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	10 ^ 16
1	11 ^ 17
2	12 ^ 18

16 **Upper MOCS Index - Enable Skip Caching**

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE
1h	UPPER_ESC_ENABLE **[Default]**

15 **Lower MOCS Index Mask Bit**

Access:	WO

In order to prevent overwriting the lower MOCS index of this register,this bit must be set as part of the write.

14:8 **Reserved**

Access:	RO
Format:	MBZ

7:6 **Reserved**

Access:	RO
Format:	MBZ

5:4 **Lower MOCS Index - L3 Cacheability Control**

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS1 - LNCF MOCS Register 1

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 1h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 10 ^ 16 1 / 11 ^ 17 2 / 12 ^ 18
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE 1h / LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 2

LNCFCMOCS2 - LNCF MOCS Register 2

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B028h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls **Value / Name / Description** 0h / UPPER_DIRECT / Use binding table index for direct EU accesses - for the rest it is reserved. 1h / UPPER_UC **[Default]** / Uncacheable 2h / UPPER_RESERVED / Reserved 3h / UPPER_WB / Writeback
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

LNCFCMOCS2 - LNCF MOCS Register 2

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS2 - LNCF MOCS Register 2

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 7h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 9 1 / 10 2 / 11
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE 1h / LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 3

LNCFCMOCS3 - LNCF MOCS Register 3

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B02Ch

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 3h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Value	Name	Description
0h	UPPER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC **[Default]**	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

LNCFCMOCS3 - LNCF MOCS Register 3

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE
1h	UPPER_ESC_ENABLE **[Default]**

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS3 - LNCF MOCS Register 3

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 1h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 9 1 / 10 2 / 11
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE 1h / LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 4

LNCFCMOCS4 - LNCF MOCS Register 4

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B030h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls **Value / Name / Description** 0h / UPPER_DIRECT / Use binding table index for direct EU accesses - for the rest it is reserved. 1h / UPPER_UC / Uncacheable 2h / UPPER_RESERVED / Reserved 3h / UPPER_WB **[Default]** / Writeback
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

LNCFCMOCS4 - LNCF MOCS Register 4

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS4 - LNCF MOCS Register 4

3:1 Lower MOCS Index - Skip Caching Control

Default Value:	7h
Access:	R/W Lock

Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

0 Lower MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	LOWER_ESC_DISABLE
1h	LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 5

LNCFCMOCS5 - LNCF MOCS Register 5

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B034h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls **Value / Name / Description** 0h / UPPER_DIRECT / Use binding table index for direct EU accesses - for the rest it is reserved. 1h / UPPER_UC **[Default]** / Uncacheable 2h / UPPER_RESERVED / Reserved 3h / UPPER_WB / Writeback
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 3h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

LNCFCMOCS5 - LNCF MOCS Register 5

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE
1h	UPPER_ESC_ENABLE **[Default]**

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS5 - LNCF MOCS Register 5

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 1h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 9 1 / 10 2 / 11
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE 1h / LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 6

LNCFCMOCS6 - LNCF MOCS Register 6

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B038h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Value	Name	Description
0h	UPPER_DIRECT **[Default]**	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

LNCFCMOCS6 - LNCF MOCS Register 6

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS6 - LNCF MOCS Register 6

3:1 Lower MOCS Index - Skip Caching Control

Default Value:	7h
Access:	R/W Lock

Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

0 Lower MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	LOWER_ESC_DISABLE
1h	LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 7

LNCFCMOCS7 - LNCF MOCS Register 7

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B03Ch

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Value	Name	Description
0h	UPPER_DIRECT **[Default]**	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

LNCFCMOCS7 - LNCF MOCS Register 7

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 **Upper MOCS Index - Enable Skip Caching**

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 **Lower MOCS Index Mask Bit**

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 **Reserved**

Access:	RO
Format:	MBZ

7:6 **Reserved**

Access:	RO
Format:	MBZ

5:4 **Lower MOCS Index - L3 Cacheability Control**

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT **[Default]**	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC	Uncacheable
2h	LOWER_WT	Writethrough
3h	LOWER_WB	Writeback

LNCFCMOCS7 - LNCF MOCS Register 7

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism

Value	Name
0h	LOWER_ESC_DISABLE **[Default]**
1h	LOWER_ESC_ENABLE

LNCF MOCS Register 8

LNCFCMOCS8 - LNCF MOCS Register 8

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B040h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Value	Name	Description
0h	UPPER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC **[Default]**	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

LNCFCMOCS8 - LNCF MOCS Register 8

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 **Upper MOCS Index - Enable Skip Caching**

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 **Lower MOCS Index Mask Bit**

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 **Reserved**

Access:	RO
Format:	MBZ

7:6 **Reserved**

Access:	RO
Format:	MBZ

5:4 **Lower MOCS Index - L3 Cacheability Control**

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT **[Default]**	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS8 - LNCF MOCS Register 8

3:1 Lower MOCS Index - Skip Caching Control

Default Value:	0h
Access:	R/W Lock

Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

0 Lower MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	LOWER_ESC_DISABLE **[Default]**
1h	LOWER_ESC_ENABLE

LNCF MOCS Register 9

LNCFCMOCS9 - LNCF MOCS Register 9

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B044h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 3h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Value	Name	Description
0h	UPPER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC **[Default]**	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

LNCFCMOCS9 - LNCF MOCS Register 9

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE
1h	UPPER_ESC_ENABLE **[Default]**

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS9 - LNCF MOCS Register 9

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 1h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 9 1 / 10 2 / 11
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE 1h / LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 10

LNCFCMOCS10 - LNCF MOCS Register 10

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B048h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Value	Name	Description
0h	UPPER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC **[Default]**	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

LNCFCMOCS10 - LNCF MOCS Register 10

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS10 - LNCF MOCS Register 10

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 7h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 9 1 / 10 2 / 11
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE 1h / LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 11

LNCFCMOCS11 - LNCF MOCS Register 11

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B04Ch

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 3h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Value	Name	Description
0h	UPPER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC **[Default]**	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

LNCFCMOCS11 - LNCF MOCS Register 11

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE
1h	UPPER_ESC_ENABLE **[Default]**

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS11 - LNCF MOCS Register 11

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 1h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 9 1 / 10 2 / 11
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE 1h / LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 12

LNCFCMOCS12 - LNCF MOCS Register 12

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B050h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register,this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Value	Name	Description
0h	UPPER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB **[Default]**	Writeback

LNCFCMOCS12 - LNCF MOCS Register 12

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS12 - LNCF MOCS Register 12

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 7h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 9 1 / 10 2 / 11
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE 1h / LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 13

LNCFCMOCS13 - LNCF MOCS Register 13

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B054h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 3h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Value	Name	Description
0h	UPPER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC **[Default]**	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

LNCFCMOCS13 - LNCF MOCS Register 13

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE
1h	UPPER_ESC_ENABLE **[Default]**

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register,this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS13 - LNCF MOCS Register 13

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 1h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism

Value	Name
0h	LOWER_ESC_DISABLE
1h	LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 14

LNCFCMOCS14 - LNCF MOCS Register 14

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B058h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	UPPER_DIRECT **[Default]**	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

DWord	Bit	Description
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

LNCFCMOCS14 - LNCF MOCS Register 14

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS14 - LNCF MOCS Register 14

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 7h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 9 1 / 10 2 / 11
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE 1h / LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 15

LNCFCMOCS15 - LNCF MOCS Register 15

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B05Ch

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Value	Name	Description
0h	UPPER_DIRECT **[Default]**	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

LNCFCMOCS15 - LNCF MOCS Register 15

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT **[Default]**	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS15 - LNCF MOCS Register 15

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism

Value	Name
0h	LOWER_ESC_DISABLE **[Default]**
1h	LOWER_ESC_ENABLE

LNCF MOCS Register 16

LNCFCMOCS16 - LNCF MOCS Register 16

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B060h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Value	Name	Description
0h	UPPER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC **[Default]**	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

LNCFCMOCS16 - LNCF MOCS Register 16

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT **[Default]**	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS16 - LNCF MOCS Register 16

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 9 1 / 10 2 / 11
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE **[Default]** 1h / LOWER_ESC_ENABLE

LNCF MOCS Register 17

LNCFCMOCS17 - LNCF MOCS Register 17

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B064h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 3h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Value	Name	Description
0h	UPPER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC **[Default]**	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

LNCFCMOCS17 - LNCF MOCS Register 17

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE
1h	UPPER_ESC_ENABLE **[Default]**

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS17 - LNCF MOCS Register 17

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 1h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 9 1 / 10 2 / 11
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE 1h / LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 18

LNCFCMOCS18 - LNCF MOCS Register 18

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B068h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Value	Name	Description
0h	UPPER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC **[Default]**	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

LNCFCMOCS18 - LNCF MOCS Register 18

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS18 - LNCF MOCS Register 18

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 7h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 9 1 / 10 2 / 11
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE 1h / LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 19

LNCFCMOCS19 - LNCF MOCS Register 19

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B06Ch

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	UPPER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC **[Default]**	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 3h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

LNCFCMOCS19 - LNCF MOCS Register 19

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 **Upper MOCS Index - Enable Skip Caching**

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE
1h	UPPER_ESC_ENABLE **[Default]**

15 **Lower MOCS Index Mask Bit**

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 **Reserved**

Access:	RO
Format:	MBZ

7:6 **Reserved**

Access:	RO
Format:	MBZ

5:4 **Lower MOCS Index - L3 Cacheability Control**

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS19 - LNCF MOCS Register 19

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 1h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 9 1 / 10 2 / 11
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE 1h / LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 20

LNCFCMOCS20 - LNCF MOCS Register 20

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B070h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls **Value / Name / Description** 0h / UPPER_DIRECT / Use binding table index for direct EU accesses - for the rest it is reserved. 1h / UPPER_UC / Uncacheable 2h / UPPER_RESERVED / Reserved 3h / UPPER_WB **[Default]** / Writeback
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

LNCFCMOCS20 - LNCF MOCS Register 20

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS20 - LNCF MOCS Register 20

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 7h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 9 1 / 10 2 / 11
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE 1h / LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 21

LNCFCMOCS21 - LNCF MOCS Register 21

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B074h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	UPPER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC **[Default]**	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

DWord	Bit	Description
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 3h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

LNCFCMOCS21 - LNCF MOCS Register 21

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE
1h	UPPER_ESC_ENABLE **[Default]**

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS21 - LNCF MOCS Register 21

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 1h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 9 1 / 10 2 / 11
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE 1h / LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 22

LNCFCMOCS22 - LNCF MOCS Register 22

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B078h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Value	Name	Description
0h	UPPER_DIRECT **[Default]**	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

LNCFCMOCS22 - LNCF MOCS Register 22

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS22 - LNCF MOCS Register 22

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 7h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 9 1 / 10 2 / 11
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE 1h / LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 23

LNCFCMOCS23 - LNCF MOCS Register 23

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B07Ch

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Value	Name	Description
0h	UPPER_DIRECT **[Default]**	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

LNCFCMOCS23 - LNCF MOCS Register 23

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 **Upper MOCS Index - Enable Skip Caching**

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 **Lower MOCS Index Mask Bit**

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 **Reserved**

Access:	RO
Format:	MBZ

7:6 **Reserved**

Access:	RO
Format:	MBZ

5:4 **Lower MOCS Index - L3 Cacheability Control**

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT **[Default]**	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS23 - LNCF MOCS Register 23

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism

Value	Name
0h	LOWER_ESC_DISABLE **[Default]**
1h	LOWER_ESC_ENABLE

LNCF MOCS Register 24

LNCFCMOCS24 - LNCF MOCS Register 24

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B080h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Value	Name	Description
0h	UPPER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC **[Default]**	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

LNCFCMOCS24 - LNCF MOCS Register 24

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT **[Default]**	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS24 - LNCF MOCS Register 24

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 9 1 / 10 2 / 11
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE **[Default]** 1h / LOWER_ESC_ENABLE

LNCF MOCS Register 25

LNCFCMOCS25 - LNCF MOCS Register 25

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B084h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 3h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Value	Name	Description
0h	UPPER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC **[Default]**	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

LNCFCMOCS25 - LNCF MOCS Register 25

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE
1h	UPPER_ESC_ENABLE **[Default]**

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS25 - LNCF MOCS Register 25

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 1h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism

Value	Name
0h	LOWER_ESC_DISABLE
1h	LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 26

LNCFCMOCS26 - LNCF MOCS Register 26

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B088h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Value	Name	Description
0h	UPPER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC **[Default]**	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

LNCFCMOCS26 - LNCF MOCS Register 26

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS26 - LNCF MOCS Register 26

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 7h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 9 1 / 10 2 / 11
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE 1h / LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 27

LNCFCMOCS27 - LNCF MOCS Register 27

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B08Ch

Programming Notes
WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls **Value / Name / Description** 0h / UPPER_DIRECT / Use binding table index for direct EU accesses - for the rest it is reserved. 1h / UPPER_UC **[Default]** / Uncacheable 2h / UPPER_RESERVED / Reserved 3h / UPPER_WB / Writeback
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 3h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

LNCFCMOCS27 - LNCF MOCS Register 27

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE
1h	UPPER_ESC_ENABLE **[Default]**

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS27 - LNCF MOCS Register 27

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 1h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 9 1 / 10 2 / 11
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE 1h / LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 28

LNCFCMOCS28 - LNCF MOCS Register 28

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B090h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls **Value / Name / Description** 0h / UPPER_DIRECT / Use binding table index for direct EU accesses - for the rest it is reserved. 1h / UPPER_UC / Uncacheable 2h / UPPER_RESERVED / Reserved 3h / UPPER_WB **[Default]** / Writeback
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

LNCFCMOCS28 - LNCF MOCS Register 28

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS28 - LNCF MOCS Register 28

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 7h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 9 1 / 10 2 / 11
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE 1h / LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 29

LNCFCMOCS29 - LNCF MOCS Register 29

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B094h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 3h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

Value	Name	Description
0h	UPPER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC **[Default]**	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

LNCFCMOCS29 - LNCF MOCS Register 29

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 **Upper MOCS Index - Enable Skip Caching**

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE
1h	UPPER_ESC_ENABLE **[Default]**

15 **Lower MOCS Index Mask Bit**

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 **Reserved**

Access:	RO
Format:	MBZ

7:6 **Reserved**

Access:	RO
Format:	MBZ

5:4 **Lower MOCS Index - L3 Cacheability Control**

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS29 - LNCF MOCS Register 29

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 1h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 9 1 / 10 2 / 11
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE 1h / LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 30

LNCFCMOCS30 - LNCF MOCS Register 30

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B098h

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	UPPER_DIRECT **[Default]**	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

DWord	Bit	Description
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

LNCFCMOCS30 - LNCF MOCS Register 30

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC **[Default]**	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS30 - LNCF MOCS Register 30

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 7h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target. **Bit Offset** / **Corresponding Address Bit** 0 / 9 1 / 10 2 / 11
	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism **Value** / **Name** 0h / LOWER_ESC_DISABLE 1h / LOWER_ESC_ENABLE **[Default]**

LNCF MOCS Register 31

LNCFCMOCS31 - LNCF MOCS Register 31

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B09Ch

Programming Notes

WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	UPPER_DIRECT **[Default]**	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

DWord	Bit	Description
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

LNCFCMOCS31 - LNCF MOCS Register 31

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.
If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

16 Upper MOCS Index - Enable Skip Caching

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 Lower MOCS Index Mask Bit

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 Reserved

Access:	RO
Format:	MBZ

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 Lower MOCS Index - L3 Cacheability Control

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.
For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT **[Default]**	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC	Uncacheable
2h	LOWER_RESERVED	Reserved
3h	LOWER_WB	Writeback

LNCFCMOCS31 - LNCF MOCS Register 31

	3:1	**Lower MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface. **Programming Notes** If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	9
1	10
2	11

	0	**Lower MOCS Index - Enable Skip Caching** Access: R/W Lock Enable the skip cache mechanism

Value	Name
0h	LOWER_ESC_DISABLE **[Default]**
1h	LOWER_ESC_ENABLE

Load Indirect Base Vertex

3DPRIM_BASE_VERTEX - Load Indirect Base Vertex	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02440h-02443h
Name:	Load Indirect Base Vertex
ShortName:	3DPRIM_BASE_VERTEX_RCSUNIT_BE_GEOMETRY
Address:	18440h-18443h
Name:	Load Indirect Base Vertex
ShortName:	3DPRIM_BASE_VERTEX_POCSUNIT_BE_GEOMETRY

DWord	Bit	Description
0	31:0	**Base Vertex** Format: S31 This register is used to store the Base Vertex of the 3D_PRIMITIVE command when Load Indirect Enable is set.

Load Indirect Extended Parameter 0

3DPRIM_XP0 - Load Indirect Extended Parameter 0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02690h-02693h
Name:	Load Indirect Extended Parameter 0
ShortName:	3DPRIM_XP0_RCSUNIT_BE_GEOMETRY
Address:	18690h-18693h
Name:	Load Indirect Extended Parameter 0
ShortName:	3DPRIM_XP0_POCSUNIT_BE_GEOMETRY

DWord	Bit	Description
0	31:0	**Extended Parameter 0** Format: U32 **Description** This register is used to store the Extended Parameter 0 of the 3D_PRIMITIVE command when Load Indirect Enable is set.

Load Indirect Extended Parameter 1

3DPRIM_XP1 - Load Indirect Extended Parameter 1	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02694h-02697h
Name:	Load Indirect Extended Parameter 1
ShortName:	3DPRIM_XP1_RCSUNIT_BE_GEOMETRY
Address:	18694h-18697h
Name:	Load Indirect Extended Parameter 1
ShortName:	3DPRIM_XP1_POCSUNIT_BE_GEOMETRY

DWord	Bit	Description
0	31:0	**Extended Parameter 1** Format: U32 **Description** This register is used to store the Extended Parameter 1 of the 3D_PRIMITIVE command when Load Indirect Enable is set.

Load Indirect Extended Parameter 2

3DPRIM_XP2 - Load Indirect Extended Parameter 2	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02698h-0269Bh
Name:	Load Indirect Extended Parameter 2
ShortName:	3DPRIM_XP2_RCSUNIT_BE_GEOMETRY
Address:	18698h-1869Bh
Name:	Load Indirect Extended Parameter 2
ShortName:	3DPRIM_XP2_POCSUNIT_BE_GEOMETRY

DWord	Bit	Description
0	31:0	**Extended Parameter 2** Format: U32 **Description** This register is used to store the Extended Parameter 2 of the 3D_PRIMITIVE command when Load Indirect Enable is set.

Load Indirect Instance Count

3DPRIM_INSTANCE_COUNT - Load Indirect Instance Count	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02438h-0243Bh
Name:	Load Indirect Instance Count
ShortName:	3DPRIM_INSTANCE_COUNT_RCSUNIT_BE_GEOMETRY
Address:	18438h-1843Bh
Name:	Load Indirect Instance Count
ShortName:	3DPRIM_INSTANCE_COUNT_POCSUNIT_BE_GEOMETRY

DWord	Bit	Description
0	31:0	**Instance Count** This register is used to store the Instance Count of the 3D_PRIMITIVE command when Load Indirect Enable is set.

Load Indirect Start Instance

3DPRIM_START_INSTANCE - Load Indirect Start Instance	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0243Ch-0243Fh
Name:	Load Indirect Start Instance
ShortName:	3DPRIM_START_INSTANCE_RCSUNIT_BE_GEOMETRY
Address:	1843Ch-1843Fh
Name:	Load Indirect Start Instance
ShortName:	3DPRIM_START_INSTANCE_POCSUNIT_BE_GEOMETRY

DWord	Bit	Description
0	31:0	**Start Vertex** Format: U32 This register is used to store the Start Instance of the 3D_PRIMITIVE command when Load Indirect Enable is set.

Load Indirect Start Vertex

3DPRIM_START_VERTEX - Load Indirect Start Vertex	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02430h-02433h
Name:	Load Indirect Start Vertex
ShortName:	3DPRIM_START_VERTEX_RCSUNIT_BE_GEOMETRY
Address:	18430h-18433h
Name:	Load Indirect Start Vertex
ShortName:	3DPRIM_START_VERTEX_POCSUNIT_BE_GEOMETRY

DWord	Bit	Description
0	31:0	**Start Vertex** Format: U32 This register is used to store the Start Vertex of the 3D_PRIMITIVE command when Load Indirect Enable is set.

Load Indirect Vertex Count

3DPRIM_VERTEX_COUNT - Load Indirect Vertex Count	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02434h-02437h
Name:	Load Indirect Vertex Count
ShortName:	3DPRIM_VERTEX_COUNT_RCSUNIT_BE_GEOMETRY
Address:	18434h-18437h
Name:	Load Indirect Vertex Count
ShortName:	3DPRIM_VERTEX_COUNT_POCSUNIT_BE_GEOMETRY

DWord	Bit	Description
0	31:0	**Vertex Count** Format: U32 This register is used to store the Vertex Count of the 3D_PRIMITIVE command when Load Indirect Enable is set.

Lock register for Bank

L3BANKLOCK - Lock register for Bank

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B160h

This is a basic register template

DWord	Bit	Description
0	31	**Lockdown L3BANKLOCK for writes** Default Value: 0h Access: R/W 0 : (default) All bits of L3BANKLOCK (offset 0xB160) register are R/W 1 : All bits of L3BANKLOCK (offset 0xB160) register are RO Once written to 1, the lock is set for all of 0B160 and cannot be cleared (i.e., writing a 0 will not clear the lock). Lock Control is 0B160h bit 31
	30:17	**Reserved** Access: RO Format: MBZ
	16:14	**Reserved** Access: RO Format: MBZ
	13	**Lockdown L3_ECC_TEST_CTL for writes** Once Lock Control 0xB160h bit 31 is written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). Value / Name / Description: 0 / **[Default]** / 1 / / All bits of L3_ECC_TEST_CTL (offset 0xB13C) and URB_ECC_TEST_CTL (offset 0xB148) registers are RO.
	12	**Lockdown L3SQCREG3 for writes** Access: R/W Once Lock Control 0xB160h bit 31 is written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). Value / Name / Description: 0 / default **[Default]** / 1 / / All bits of L3SQCREG3 (offset 0xB108) register are RO.

L3BANKLOCK - Lock register for Bank

11 — Lockdown LBSREG for writes

Access:	R/W

Once Lock Control 0xB160h bit 31 is written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock).

Value	Name	Description
0	default **[Default]**	
1		All bits of LBSREG (offset 0xB124) register are RO.

10 — Lockdown LSQCFIFOARB for writes

Access:	R/W

Once Lock Control 0xB160h bit 31 is written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock).

Value	Name	Description
0	default **[Default]**	
1		All bits of LSQCFIFOARB (offset 0xB150) register are RO.

9 — Lockdown SCRATCH1 for writes

Access:	R/W

Once Lock Control 0xB160h bit 31 is written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock).

Value	Name	Description
0	default **[Default]**	
1		All bits of SCRATCH1 (offset 0xB11C) register are RO.

8 — Lockdown LTCDREG2 for writes

Access:	R/W

Once Lock Control 0xB160h bit 31 is written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock).

Value	Name	Description
0	default **[Default]**	
1		All bits of LTCDREG2 (offset 0xB110) register are RO.

7 — Lockdown L3SQCREG2 for writes

Access:	R/W

Once Lock Control 0xB160h bit 31 is written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock).

Value	Name	Description
0	default **[Default]**	
1		All bits of L3SQCREG2 (offset 0xB104) register are RO.

6 — Lockdown L3SCRATCH3 for writes

Access:	R/W

Once Lock Control 0xB160h bit 31 is written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock).

L3BANKLOCK - Lock register for Bank

Value	Name	Description
0	default **[Default]**	All bits of L3SCRATCH3 (offset 0xB154) register are R/W
1		Bits [10:4] of L3SCRATCH3 (offset 0xB154) register are RO.

5 **Lockdown L3SCRATCH2 for writes**

Access:	R/W

Once Lock Control 0xB160h bit 31 is written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock).

Value	Name	Description
0h	**[Default]**	(default) All bits of L3SCRATCH2;(offset 0xB140) register are R/W
1		All bits of L3SCRATCH2 offset 0xB140) register are RO

4 **Lockdown L3SQCREG1 for writes**

Access:	R/W

Once Lock Control 0xB160h bit 31 is written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock).

Value	Name	Description
0h	Default **[Default]**	All valid bits of L3SQREG1 (offset 0xB100) register are R/W
1		All valid bits of L3SQREG1 (offset 0xB100) register are R/W (no locking effect)

3 **Lockdown LTCDREG for writes**

Access:	R/W

Once Lock Control 0xB160h bit 31 is written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock).

Value	Name	Description
0h	**[Default]**	(default) All bits of LTCDREG (offset 0xB120) register are R/W
1		All bits of LTCDREG (offset 0xB120) register are RO

2 **Lockdown L3SQCREG4 for writes**

Access:	R/W

Once Lock Control 0xB160h bit 31 is written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock).

Value	Name	Description
0h	**[Default]**	(default) All bits of L3SQREG4 (offset 0xB118) register are R/W
1		All bits of L3SQREG4 (offset 0xB118) register are R/W (no locking effect)

L3BANKLOCK - Lock register for Bank

	1	**Lockdown L3CHMD for writes** Access: R/W Once Lock Control 0xB160h bit 31 is written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). **Value** / **Name** / **Description** 0h / **[Default]** / (default) All bits of L3CHMD (offset 0xB114) register are R/W 1 / / All bits of L3CHMD (offset 0xB114) register are RO
	0	**Lockdown L3CNTLREG1 for writes** Access: R/W Once Lock Control 0xB160h bit 31 is written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). **Value** / **Name** / **Description** 0h / **[Default]** / (default) - All bits of L3CNTLREG1 (offset 0xB10C) register are R/W 1 / / All bits of L3CNTLREG1 (offset 0xB10C) register are RO

Lock register for LPFC

L3LPFCLOCK - Lock register for LPFC

Register Space: MMIO: 0/2/0
Size (in bits): 32
_Custom_GTIReset: DEV

Address: 0B460h

This is a basic register template

DWord	Bit	Description
0	31	**Lockdown L3LPFCLOCK** Default Value: 0h Access: R/W 0 : (default) All bits of L3LPFCLOCK (offset 0xB460) register are R/W 1 : All bits of L3LPFCLOCK (offset 0xB460) register are RO Once written to 1, the lock is set for all of 0xB460 and cannot be cleared (i.e., writing a 0 will not clear the lock). Lock Control is B460h bit 31
	30:3	**Reserved** Access: RO Format: MBZ
	2	**Reserved**
	1	**Lockdown LSN Unslice Client Virtual Channel Assignment** Default Value: 0h Access: R/W 0 : (default) bits is R/W 1 : bit is RO Once Lock Control B460h bit 31 is written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock).
	0	**Lockdown LPFC hashing disable** Default Value: 0h Access: R/W 0 : (default) bit is R/W 1 : bit is RO Once Lock Control B460h bit 31 is written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock).

Lock register for Node

L3NODELOCK - Lock register for Node

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B0ACh

This is a basic register template

DWord	Bit	Description
0	31	**Lockdown L3NODELOCK** Default Value: 0h Access: R/W 0 : (default) All bits of L3NODELOCK (offset 0xB0AC) register are R/W 1 : All bits of L3NODELOCK (offset 0xB0AC) register are RO Once written to 1, the lock is set for all of 0xB0AC and cannot be cleared (i.e., writing a 0 will not clear the lock). Lock Control is 0B0ACh bit 31
	30:4	**Reserved** Access: RO Format: MBZ
	3	**Lockdown Color client arb pri, Z client Arb pri** Default Value: 0h Access: R/W 0 : (default) bits is R/W 1 : bit is RO Once Lock Control 0B0ACh bit 31 is written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock).
	2	**Lockdown slow client fix dis** Default Value: 0h Access: R/W 0 : (default) bits is R/W 1 : bit is RO Once Lock Control 0B0ACh bit 31 is written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock).

L3NODELOCK - Lock register for Node

	1	**Lockdown TC/vc sel, LSN arb pri sel** Default Value: 0h Access: R/W 0 : (default) bits is R/W 1 : bit is RO Once Lock Control 0B0ACh bit 31 is written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock).
	0	**Lockdown Hashing disable, mod5 hash dis, Non coh completion from node enable** Default Value: 0h Access: R/W 0 : (default) bit is R/W 1 : bit is RO Once Lock Control 0xB0ACh bit 31 is written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock).

LPFC FUSA Register

LPFC_FUSA - LPFC FUSA Register

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B480h

Functional Safety Register. LPFC collects updates of single and double bit errors from all LBCF in ECC Checker Mode.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15	**Double bit error indication from Bank 7** Access: R/W Double Error detection indication from Bank7.
	14	**Single bit error indication from Bank 7** Access: R/W Single Error detection indication from Bank7.
	13	**Double bit error indication from Bank 6** Access: R/W Double Error detection indication from Bank6.
	12	**Single bit error indication from Bank 6** Access: R/W Single Error detection indication from Bank6.
	11	**Double bit error indication from Bank 5** Access: R/W Double Error detection indication from Bank5.
	10	**Single bit error indication from Bank 5** Access: R/W Single Error detection indication from Bank5.

LPFC_FUSA - LPFC FUSA Register

	9	**Double bit error indication from Bank 4** Access: R/W Double Error detection indication from Bank4.
	8	**Single bit error indication from Bank 4** Access: R/W Single Error detection indication from Bank4.
	7	**Double bit error indication from Bank 3** Access: R/W Double Error detection indication from Bank3.
	6	**Single bit error indication from Bank 3** Access: R/W Single Error detection indication from Bank3.
	5	**Double bit error indication from Bank 2** Access: R/W Double Error detection indication from Bank2.
	4	**Single bit error indication from Bank 2** Access: R/W Single Error detection indication from Bank2.
	3	**Double bit error indication from Bank 1** Access: R/W Double Error detection indication from Bank1.
	2	**Single bit error indication from Bank 1** Access: R/W Single Error detection indication from Bank1.
	1	**Double bit error indication from Bank 0** Access: R/W Double Error detection indication from Bank0.
	0	**Single bit error indication from Bank 0** Access: R/W Single Error detection indication from Bank0.

LSN Arbitration Priority Register 0

LSN_ARBPRI0 - LSN Arbitration Priority Register 0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	BUS
Address:	0B0D0h

DWord	Bit	Description
0	31:28	**GAPS Local-Slice Arbitration Priority** This value is 'n' in the m:n priority scheme for L3 vs. GAPS local-slice traffic. Value: 1 — **[Default]** Value: [1,15]
	27:24	**L3 Local-Slice Arbitration Priority** This value is 'm' in the m:n priority scheme for L3 vs. GAPS local-slice traffic. Value: 4 — **[Default]** Value: [1,15]
	23:20	**LNE Traffic Class Channel 1 Priority Value** This value is 'm' in the n:m priority scheme for TCC0 vs. TCC1 arbitration on LNE cycles. Value: 4 — **[Default]** Value: [1,15]
	19:16	**LNE Traffic Class Channel 0 Priority Value** This value is 'n' in the n:m priority scheme for TCC0 vs. TCC1 arbitration on LNE cycles. Value: 1 — **[Default]** Value: [1,15]
	15:12	**LSN Virtual Channel 1 Traffic Class Channel 1 Priority Value** This value is 'm' in the n:m priority scheme for TCC0 vs. TCC1 arbitration on LSN Virtual Channel 1 cycles. Value: 4 — **[Default]** Value: [1,15]

LSN_ARBPRI0 - LSN Arbitration Priority Register 0

	11:8	**LSN Virtual Channel 1 Traffic Class Channel 0 Priority Value** This value is 'n' in the n:m priority scheme for TCC0 vs. TCC1 arbitration on LSN Virtual Channel 1 cycles. **Value** / **Name** 1 / **[Default]** [1,15] /
	7:4	**LSN Virtual Channel 0 Traffic Class Channel 1 Priority Value** This value is 'm' in the n:m priority scheme for TCC0 vs. TCC1 arbitration on LSN Virtual Channel 0 cycles. **Value** / **Name** 4 / **[Default]** [1,15] /
	3:0	**LSN Virtual Channel 0 Traffic Class Channel 0 Priority Value** This value is 'n' in the n:m priority scheme for TCC0 vs. TCC1 arbitration on LSN Virtual Channel 0 cycles. **Value** / **Name** 1 / **[Default]** [1,15] /

LSN Arbitration Priority Register 1

LSN_ARBPRI1 - LSN Arbitration Priority Register 1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	BUS
Address:	0B0D4h

DWord	Bit	Description
0	31:25	**Reserved** Access: RO Format: MBZ
	24	**LSN Opporunistic Arbitration Disable** Default Value: 0 _Custom_GTIReset: DEV 0 (default) - Opportunistic Arbitration mechanism of LSN Cross-Slice, LSN-LNI and LSN-LNE arbiters is enabled. 1- Opportunistic Arbitration mechanism of LSN Cross-Slice, LSN-LNI and LSN-LNE arbiters is disabled. Signal name - lncf_csr_lsn_opprtnstc_arb_dis
	23:20	**LNI Egress Traffic Class Channel 1 Priority Value** This value is 'm' in the n:m priority scheme for TCC0 vs. TCC1 arbitration on completions to LNI's egress port. Value: 4 **[Default]** Value: [1,15]
	19:16	**LNI Egress Traffic Class Channel 0 Priority Value** This value is 'n' in the n:m priority scheme for TCC0 vs. TCC1 arbitration on completions to LNI's egress port. Value: 1 **[Default]** Value: [1,15]
	15:12	**LNI Ingress Traffic Class Channel 1 Priority Value** This value is 'm' in the n:m priority scheme for TCC0 vs. TCC1 arbitration on completions to LNI's ingress port. Value: 4 **[Default]** Value: [1,15]

LSN_ARBPRI1 - LSN Arbitration Priority Register 1

	11:8	**LNI Ingress Traffic Class Channel 0 Priority Value** This value is 'n' in the n:m priority scheme for TCC0 vs. TCC1 arbitration on completions to LNI's ingress port. **Value** / **Name** 1 / **[Default]** [1,15] /
	7:4	**Cross-Slice Completion Traffic Class Channel 1 Priority Value** This value is 'm' in the n:m priority scheme for TCC0 vs. TCC1 arbitration on cross-slice completions. **Value** / **Name** 4 / **[Default]** [1,15] /
	3:0	**Cross-Slice Completion Traffic Class Channel 0 Priority Value** This value is 'n' in the n:m priority scheme for TCC0 vs. TCC1 arbitration on cross-slice completions. **Value** / **Name** 1 / **[Default]** [1,15] /

LSN Miscellaneous Configuration

LSN_MISC - LSN Miscellaneous Configuration

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	BUS
Address:	0B0CCh

DWord	Bit	Description
0	31:6	**Reserved** Access: RO Format: MBZ
	5	**L3 Smart Repeater Bypass Enable** When set, L3 smart repeaters in unslice will bypass all FIFOs for requests. The simple flop bypass path will be used.
	4	**LSN Egress Pipeline Clock Gating Enable** Value / Name / Description: 1 / **[Default]** / Enable fub-level clock gating of the LSN egress pipeline. 0 / / Disable fub-level clock gating of the LSN egress pipeline.
	3	**LSN Smart Gating for Slice-Bound Memory Fill Repeaters Disable** Value / Name / Description: 1 / **[Default]** / Disable the smart gating of slice-bound memory fill repeaters. 0 / / Allow smart-gating for slice-bound memory fill repeaters.
	2	**LSN Conservative Arbitration Control** Controls whether LSN uses a more conservative form of arbitration by removing assumptions about packet arrival Value / Name / Description: 1 / / Cycles are only eligible for arbitration in the pipeline after arrival of all packets of the cycle. 0 / **[Default]** / Cycles are eligible for arbitration after the arrival of the first packet of the cycle.
	1	**GAFS Return Simple Arbitration Control** Enables simple arbitration between upstream LSN busses when returning GAFS-bound read returns. This arbitration uses a continuously running counter to select two busses for GAFS-bound read returns.

LSN_MISC - LSN Miscellaneous Configuration

Value	Name	Description
1	SIMPLE	Enable simple (always running) arbitration
0	PRIORITY **[Default]**	Enable priority-based arbitration

0 **LSN Cross-Slice Virtual Channel Arbitration Control**

Controls whether to use round-robin or fixed priority arbitration between cross-slice virtual channels

Value	Name	Description
1	RR_PRI	Enable round-robin priority between VC1 and VC0
0	FIXED_PRI **[Default]**	Enable fixed priority between VC1 and VC0. VC1 will always take higher priority

LSN Slice Client Virtual Channel Assignment

LSN_SLCVC - LSN Slice Client Virtual Channel Assignment

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	BUS
Address:	0B0C8h

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:14	**Reserved** Access: RO Format: MBZ
	13	**Traffic Class Channel ID for GAFS-Bound L3 Returns in LNE Ingress FIFOs** Default Value: 1
	12	**Traffic Class Channel ID for L3-Bound L3 Returns in LNE Ingress FIFOs** Default Value: 0
	11	**Traffic Class Channel ID for GAFS-Bound Completions in LNI Completion Ingress FIFO** Default Value: 1
	10	**Traffic Class Channel ID for L3-Bound Completions in LNI Completion Ingress FIFO** Default Value: 0
	9	**Traffic Class Channel ID for L3 Cycles** Value: 0 — Name: **[Default]**
	8	**Virtual Channel ID for L3 Cycles** Default Value: 0
	7	**Traffic Class Channel ID for GAFS-Bound Read Returns** Default Value: 1
	6	**Virtual Channel ID for GAFS-Bound Read Returns** Default Value: 1
	5	**Traffic Class Channel ID for Downstream L3-Bound Read Returns** Value: 1 — Name: **[Default]**

LSN_SLCVC - LSN Slice Client Virtual Channel Assignment

	4	**Virtual Channel ID for Downstream L3-Bound Read Returns** Default Value: 1
	3	**Traffic Class Channel ID for Upstream L3-Bound Read Returns** Value: 1 — Name: **[Default]**
	2	**Virtual Channel ID for Upstream L3-Bound Read Returns** Default Value: 1
	1	**Traffic Class Channel ID for GAPS Cycles** Value: 1 — Name: **[Default]**
	0	**Virtual Channel ID for GAPS Cycles** Default Value: 0

LSQC FIFO Arbitration

LSQCFIFOARB - LSQC FIFO Arbitration

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	0B150h

LSQC FIFO Arbitration register enables programmable arbitration used for selecting a winner amongst multiple FIFO's in LSQC

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23:22	**Memory read over memory write** Access: R/W _Custom_GTIReset: DEV This count will determine for how many clocks the memory reads will be selected as winner before allowing memory writes. Needed as Cmd port from lsqc to gapl3 is just 1. Valid values are as follows: 0 = 1 cnt (Default). 1 = 2 cnt. 2 = 4 cnt. 3 = 8 cnt. lbcf_csr_lsqc_memrd_over_memwr_cnt[1:0] Value: 1 — Name: **[Default]**
	21:20	**Memory write over Memory Read** Access: R/W _Custom_GTIReset: DEV This count will determine for how many clocks the memory writes will be selected as winner before allowing memory reads. Needed as Cmd port from lsqc to gapl3 is just 1. Valid values are as follows: 0 = 1 cnt. 1 = 2 cnt. 2 = 4 cnt. 3 = 8 cnt (Default). lbcf_csr_lsqc_memwr_over_memrd_cnt[1:0] Value: 0 — Name: def **[Default]**

LSQCFIFOARB - LSQC FIFO Arbitration

	19:18	**Memory read Fifo2 counter** Default Value: 00b Access: R/W _Custom_GTIReset: DEV This count will determine for how many clocks the memory reads fifo2 will be selected as winner before allowing memory read Fifo1. Valid values are as follows: 0 = 1 cnt (Default). 1 = 2 cnt. 2 = 4 cnt. 3 = 8 cnt. lbcf_csr_lsqc_memrd_f2_cnt[1:0]
	17:16	**Memory read Fifo1 counter** Default Value: 11b Access: R/W _Custom_GTIReset: DEV This count will determine for how many clocks the memory reads fifo1 will be selected as winner before allowing memory read Fifo2. Valid values are as follows: 0 = 1 cnt. 1 = 2 cnt. 2 = 4 cnt. 3 = 8 cnt (Default). lbcf_csr_lsqc_memrd_f1_cnt[1:0]
	15:14	**Memory write Fifo2 counter** Default Value: 00b Access: R/W _Custom_GTIReset: DEV This count will determine for how many clocks the memory writes fifo2 will be selected as winner before allowing read returns. Valid values are as follows: 0 = 1 cnt (Default). 1 = 2 cnt. 2 = 4 cnt. 3 = 8 cnt. lbcf_csr_lsqc_memwr_f2_cnt[1:0]

LSQCFIFOARB - LSQC FIFO Arbitration

	13:12	**Memory write Fifo1 counter** Default Value: 11b Access: R/W _Custom_GTIReset: DEV This count will determine for how many clocks the memory writes fifo1 will be selected as winner before allowing read returns. Valid values are as follows: 0 = 1 cnt. 1 = 2 cnt. 2 = 4 cnt. 3 = 8 cnt (Default). lbcf_csr_lsqc_memwr_f1_cnt[1:0]
	11:10	**Read Return Fifo3 counter** Default Value: 00b Access: R/W _Custom_GTIReset: DEV Fifo3 will be selected as winner as many times as described by this counter before some other fifo that is also available can be selected as winner. Valid values are as follows: 0 = 1 cnt (Default). 1 = 2 cnt. 2 = 4 cnt. 3 = 8 cnt. lbcf_csr_lsqc_rdret_f3_cnt[1:0]
	9:8	**Read Return Fifo2 counter** Default Value: 10b Access: R/W _Custom_GTIReset: DEV Fifo2 will be selected as winner as many times as described by this counter before some other fifo that is also available can be selected as winner. Valid values are as follows: 0 = 1 cnt. 1 = 2 cnt. 2 = 4 cnt (Default). 3 = 8 cnt. lbcf_csr_lsqc_rdret_f2_cnt[1:0]

LSQCFIFOARB - LSQC FIFO Arbitration

	7:6	**Read Return Fifo1 counter** Default Value: 11b Access: R/W _Custom_GTIReset: DEV Fifo1 will be selected as winner as many times as described by this counter before some other fifo that is also available can be selected as winner. Valid values are as follows: 0 = 1 cnt. 1 = 2 cnt. 2 = 4 cnt. 3 = 8 cnt (Default). lbcf_csr_lsqc_rdret_f1_cnt[1:0]
	5:4	**Fill Fifo3 counter** Default Value: 00b Access: R/W _Custom_GTIReset: DEV Fifo3 will be selected as winner as many times as described by this counter before some other fifo that is also available can be selected as winner. Valid values are as follows: 0 = 1 cnt (Default). 1 = 2 cnt. 2 = 4 cnt. 3 = 8 cnt. lbcf_csr_lsqc_fill_f3_cnt[1:0]
	3:2	**Fill Fifo2 counter** Default Value: 10b Access: R/W _Custom_GTIReset: DEV Fifo2 will be selected as winner as many times as described by this counter before some other fifo that is also available can be selected as winner. Valid values are as follows: 0 = 1 cnt. 1 = 2 cnt. 2 = 4 cnt (Default). 3 = 8 cnt. lbcf_csr_lsqc_fill_f2_cnt[1:0]

LSQCFIFOARB - LSQC FIFO Arbitration

	1:0	**Fill Fifo1 counter** Default Value: 11b Access: R/W _Custom_GTIReset: DEV Fifo1 will be selected as winner as many times as described by this counter before some other fifo that is also available can be selected as winner. Valid values are as follows: 0 = 1 cnt. 1 = 2 cnt. 2 = 4 cnt. 3 = 8 cnt (Default). lbcf_csr_lsqc_fill_f1_cnt[1:0]

LUT_3D_CTL

LUT_3D_CTL

Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled

Address:	490A4h-490A7h
Name:	Pipe A 3D LUT Control
ShortName:	LUT_3D_CTL_A
Reset:	soft

Address:	491A4h-491A7h
Name:	Pipe B 3D LUT Control
ShortName:	LUT_3D_CTL_B
Reset:	soft

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="2">0</td><td>31</td><td>LUT 3D Enable
This field enables the 3D LUT.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0b</td><td>Disable</td></tr>
<tr><td>1b</td><td>Enable</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>3D LUT can be enabled/disabled at any time irrespective of when the pipe is enabled/disabled. Program the Bit 10 of register 420b0h to 1b to ensure that the 3D LUT functionality gets enabled on the first frame after the pipe turns on.</td></tr>
</table></td></tr>
<tr><td>30</td><td>New LUT Ready
<table>
<tr><td>Access:</td><td>R/W Set</td></tr>
</table>
This bit must be set to '1' after all the 3D LUT entries are programmed. This bit will get cleared by hardware after the LUT buffer is loaded in to the internal working RAM.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0b</td><td>New LUT not ready</td><td>New LUT is not yet ready/hardware finished loading the LUT buffer in to internal working RAM.</td></tr>
<tr><td>1b</td><td>New LUT Ready</td><td>New LUT is ready.</td></tr>
</table>
<table>
<tr><th>Restriction</th></tr>
<tr><td>Once set, only hardware is allowed to clear this bit. Software cannot clear this bit.</td></tr>
</table></td></tr>
</table>

LUT_3D_CTL

	29	**Allow Double Buffer Update Disable** Access: R/W This field controls whether double buffer updates are allowed to be disabled for the 3D LUT registers that are double buffered. The DOUBLE_BUFFER_CTL register can be configured to globally disable double buffer updates for those resources that allow them to be disabled. Updates to the LUT entries, triggered by New LUT Ready, do not have double buffer updating disabled. **Value** / **Name** 0b: Not Allowed 1b: Allowed **[Default]**
	28:0	**Reserved** Access: RO Format: MBZ

LUT_3D_DATA

LUT_3D_DATA	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	490ACh-490AFh
Name:	Pipe A 3D LUT Data
ShortName:	LUT_3D_DATA_A
Reset:	soft
Address:	491ACh-491AFh
Name:	Pipe B 3D LUT Data
ShortName:	LUT_3D_DATA_B
Reset:	soft

These are the 3D LUT entries. The 3D LUT Index Value indicates the 3D LUT location to be accessed through this register.
Even though this specific register is not double buffered, the 3D LUT table that this register accesses is (i.e. this register updates the table's back buffer). The double buffering point for the table (after DB'ing is armed) is the start of V. Blank or the Pipe is disabled

Restriction
This register must be written only as a full 32 bit dword. Byte or word writes are not supported.

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29:0	**LUT 3D Entry** 3D LUT entry value programmed as R10G10B10.

LUT_3D_INDEX

LUT_3D_INDEX

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

Address:	490A8h-490ABh
Name:	Pipe A 3D LUT Index
ShortName:	LUT_3D_INDEX_A
Reset:	soft

Address:	491A8h-491ABh
Name:	Pipe B 3D LUT Index
ShortName:	LUT_3D_INDEX_B
Reset:	soft

This index controls access to the pre-double buffered array of 3D LUT entries.
Even though this specific register is not double buffered, the 3D LUT table that this register accesses is (i.e. this register updates the table's back buffer). The double buffering point for the table (after DB'ing is armed) is the start of V. Blank or the Pipe is disabled

DWord	Bit	Description
0	31:14	**Reserved** Access: RO Format: MBZ
	13	**Index Auto Increment** This field enables the index value to auto increment on each read or write to the data register. 0b – No Increment 1b – Auto Increment
	12:0	**Index Value** This field indicates the data location to be accessed through the data register. This value can be automatically incremented by a read or a write to the data register if the index auto increment bit is set. When automatically incrementing, the current automatically calculated index value can be read here, and the index will roll over to 0 after reaching the end of the allowed range. [0,4912]

Intel® Iris® Xe and UHD Graphics Open Source

Programmer's Reference Manual

For the 2020-2021 11th Generation Intel Xeon®, Core™, Celeron®, Pentium® Gold Processors based on the "Tiger Lake" Platform

Volume 2c: Command Reference: Registers
Part 2 – Registers M through Z

December 2021, Revision 1.0

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Table of Contents

MAILBOX0 1
MAILBOX1 2
MAILBOX2 3
MAILBOX3 4
Maximum Latency 5
MBC Control Register 6
MBUS_ABOX_CTL 8
MBUS_BBOX_CTL 10
MBUS_DBOX_CTL 12
MBUS_UBOX_CTL 14
MCPG1 Hysteresis Time Free 16
MCR Packet Control 17
MCR Packet Control 0 19
MCRPacketControl1 20
MCR Packet Control 2 21
MDRB Context Base 1 22
MDRB Context Base 2 23
MED0HCP0PowerGoodDelay 24
MED0MFX0PowerGoodDelay 25
MED0PowerGoodDelay 26
MED1HCP0PowerGoodDelay 27
MED1MFX0PowerGoodDelay 28
MED1PowerGoodDelay 29
Media Die Recovery 30
Media FIFO Messaging Register for Shadow Register Unit 31
Media unit Level Clock Gating override during rstflow 94B0 35
MEMRR_BASE_LSB 36
MEMRR_BASE_MSB 37
MEMRR_MASK_LSB 38
MEMRR_MASK_MSB 39
Message Address 40
Message Control 41

Message Data 43
Message Signaled Interrupts Capability ID 44
MFC_AVC_CABAC_INSERTION_COUNT 45
MFC_AVC Bitstream Decoding Front-End Parsing Logic Error Counter 46
MFC Image Status Control 47
MFC Image Status Mask 48
MFC QP Status Count 49
MFD Error Status 50
MFD Picture Parameter 52
MFX Frame BitStream SE/BIN Count 53
MFX Frame Macroblock Count 54
MFX Frame Row-Stored/BitStream Read Count 56
MFX PAK MPEG TS STATUS 57
MFX Pipeline Status Flags 58
MFX SFC LOCK Request 60
MFX SFC LOCK Status 61
MGCMD 62
MGSR GTI PD Control 63
MGSR Media PD Control 64
MGSR Programmable Shadow 0 65
MGSR Programmable Shadow 1 66
Minimum Grant 67
Minute IA Force Fence 68
Minute IA Force TLB Invalidate 69
Mirror FUSE A 70
Mirror GT C6 entry Count 71
Mirror of Base Data of Stolen Memory 72
Mirror of Device Enable 73
Mirror of DPRB LSB 75
Mirror of DPRB MSB 76
Mirror of DSMBASE LSB 77
Mirror of DSMBASE MSB 78
Mirror of EMRR Base LSB 79
Mirror of EMRR Base MSB 80

Mirror of EU Disable Fuses - Register0 81
Mirror of FUSE1 Control DW 82
Mirror of Fuse 3 control DW 84
Mirror of Global Command Register 86
Mirror of GMCH Graphics Control Register 90
Mirror of Graphics Translation Table and Memory Mapped Range Address (31:0) 92
Mirror of Graphics Translation Table and Memory Mapped Range Address UDW 94
Mirror of GSMBASE LSB 95
Mirror of GSMBASE MSB 96
Mirror of GT DSS Enable Fuses 97
Mirror of GT Slice Enable Fuses 98
Mirror of GT VEBOX and VDBOX Disable 99
Mirror of MBC IDIMSG Reg 100
Mirror of PCICMD MAE/BME 101
Mirror of PMR HIGH LIMIT (31-0) 103
Mirror of PMR HIGH LIMIT 63-32 104
Mirror of RC0 Hysterisis free 105
Mirror RC0 Any Engine Busy Free 106
Mirror RC0 wake count free 107
Mirror Rc6 post hysterisis overhead free 108
Mirror rc6 pre Engine wake overhead free 109
Misc Clocking Reset Control Registers 110
MISC CTX control register 113
Miscellaneous Message Register 0 for GTI Doorbell Unit 114
Misc Reset Control Register 116
MMIO_INDEX 117
Mode Register for GAB 118
Mode Register for GAC 119
Mode Register for GAFS 121
Mode Register for Software Interface 123
MSI Mask Bits 129
MSI Pending Bits 130
Multi Context Control Register 131
Multi Size Aperture Control 132

NDE_RSTWRN_OPT 134
NONPIPE_ISOCREQ 135
NOP Identification Register 136
Number Of VFs 138
OAC Aggregate Perf Counter A0 139
OAC Aggregate Perf Counter A0 Upper DWord 140
OAC Aggregate Perf Counter A4 141
OAC Aggregate Perf Counter A4 Upper DWord 142
OAC Aggregate Perf Counter A7 143
OAC Aggregate Perf Counter A7 Upper DWord 144
OAC Aggregate Perf Counter A8 145
OAC Aggregate Perf Counter A8 Upper DWord 146
OAC Aggregate Perf Counter A9 147
OAC Aggregate Perf Counter A9 Upper DWord 148
OAC Aggregate Perf Counter A10 149
OAC Aggregate Perf Counter A10 Upper DWord 150
OAC Aggregate Perf Counter A11 151
OAC Aggregate Perf Counter A11 Upper DWord 152
OAC Aggregate Perf Counter A12 153
OAC Aggregate Perf Counter A12 Upper DWord 154
OAC Aggregate Perf Counter A13 155
OAC Aggregate Perf Counter A13 Upper DWord 156
OAC Aggregate Perf Counter A14 157
OAC Aggregate Perf Counter A14 Upper DWord 158
OAC Aggregate Perf Counter A15 159
OAC Aggregate Perf Counter A15 Upper DWord 160
OAC Aggregate Perf Counter A16 161
OAC Aggregate Perf Counter A16 Upper DWord 162
OAC Aggregate Perf Counter A17 163
OAC Aggregate Perf Counter A17 Upper DWord 164
OAC Aggregate Perf Counter A18 165
OAC Aggregate Perf Counter A18 Upper DWord 166
OAC Aggregate Perf Counter A19 167
OAC Aggregate Perf Counter A19 Upper DWord 168

OAC Aggregate Perf Counter A20 169
OAC Aggregate Perf Counter A20 Upper DWord 170
OAC Aggregate Perf Counter A28 171
OAC Aggregate Perf Counter A28 Upper DWord 172
OAC Aggregate Perf Counter A29 173
OAC Aggregate Perf Counter A29 Upper DWord 174
OAC Aggregate Perf Counter A30 175
OAC Aggregate Perf Counter A30 Upper DWord 176
OAC Aggregate Perf Counter A31 177
OAC Aggregate Perf Counter A31 Upper DWord 178
OAC Aggregate Perf Counter A32 179
OACAggregatePerfCounterA32UpperDWord 180
OAC Aggregate Perf Counter A33 181
OACAggregatePerfCounterA33UpperDWord 182
OAC Aggregate Perf Counter A34 183
OACAggregatePerfCounterA34UpperDWord 184
OAC Aggregate Perf Counter A35 185
OACAggregatePerfCounterA35UpperDWord 186
OAC Boolean_Counter_B0 187
OAC Boolean_Counter_B1 188
OAC Boolean_Counter_B2 189
OAC Boolean_Counter_B3 190
OAC Boolean_Counter_B4 191
OAC Boolean_Counter_B5 192
OAC Boolean_Counter_B6 193
OAC Boolean_Counter_B7 194
OAC GPU Ticks Counter 195
OAC GPU Ticks Counter Upper DWord 196
OAC Observation Architecture Control 197
OAC Observation Architecture Status Register 198
OAG Aggregate Perf Counter A0 200
OAG Aggregate Perf Counter A0 Upper DWord 201
OAG Aggregate Perf Counter A1 202
OAG Aggregate Perf Counter A1 Upper DWord 203

OAG Aggregate Perf Counter A2 204
OAG Aggregate Perf Counter A2 Upper DWord 205
OAG Aggregate Perf Counter A3 206
OAG Aggregate Perf Counter A3 Upper DWord 207
OAG Aggregate Perf Counter A4 208
OAG Aggregate Perf Counter A4 Lower DWord Free 209
OAG Aggregate Perf Counter A4 Upper DWord 210
OAG Aggregate Perf Counter A4 Upper DWord Free 211
OAG Aggregate Perf Counter A5 212
OAG Aggregate Perf Counter A5 Upper DWord 213
OAG Aggregate Perf Counter A6 214
OAG Aggregate Perf Counter A6 Lower DWord Free 215
OAG Aggregate Perf Counter A6 Upper DWord 216
OAG Aggregate Perf Counter A6 Upper DWord Free 217
OAG Aggregate Perf Counter A7 218
OAG Aggregate Perf Counter A7 Upper DWord 219
OAG Aggregate Perf Counter A8 220
OAG Aggregate Perf Counter A8 Upper DWord 221
OAG Aggregate Perf Counter A9 222
OAG Aggregate Perf Counter A9 Upper DWord 223
OAG Aggregate Perf Counter A10 224
OAG Aggregate Perf Counter A10 Upper DWord 225
OAG Aggregate Perf Counter A11 226
OAG Aggregate Perf Counter A11 Upper DWord 227
OAG Aggregate Perf Counter A12 228
OAG Aggregate Perf Counter A12 Upper DWord 229
OAG Aggregate Perf Counter A13 230
OAG Aggregate Perf Counter A13 Upper DWord 231
OAG Aggregate Perf Counter A14 232
OAG Aggregate Perf Counter A14 Upper DWord 233
OAG Aggregate Perf Counter A15 234
OAG Aggregate Perf Counter A15 Upper DWord 235
OAG Aggregate Perf Counter A16 236
OAG Aggregate Perf Counter A16 Upper DWord 237

OAG Aggregate Perf Counter A17 ..... **238**
OAG Aggregate Perf Counter A17 Upper DWord **239**
OAG Aggregate Perf Counter A18 ..... **240**
OAG Aggregate Perf Counter A18 Upper DWord **241**
OAG Aggregate Perf Counter A19 ..... **242**
OAG Aggregate Perf Counter A19 Lower DWord Free **243**
OAG Aggregate Perf Counter A19 Upper DWord **244**
OAG Aggregate Perf Counter A19 Upper DWord Free **245**
OAG Aggregate Perf Counter A20 ..... **246**
OAG Aggregate Perf Counter A20 Lower DWord Free **247**
OAG Aggregate Perf Counter A20 Upper DWord **248**
OAG Aggregate Perf Counter A20 Upper DWord Free **249**
OAG Aggregate Perf Counter A21 ..... **250**
OAG Aggregate Perf Counter A21 Upper DWord **251**
OAG Aggregate Perf Counter A22 ..... **252**
OAG Aggregate Perf Counter A22 Upper DWord **253**
OAG Aggregate Perf Counter A23 ..... **254**
OAG Aggregate Perf Counter A23 Upper DWord **255**
OAG Aggregate Perf Counter A24 ..... **256**
OAG Aggregate Perf Counter A24 Upper DWord **257**
OAG Aggregate Perf Counter A25 ..... **258**
OAG Aggregate Perf Counter A25 Upper DWord **259**
OAG Aggregate Perf Counter A26 ..... **260**
OAG Aggregate Perf Counter A26 Upper DWord **261**
OAG Aggregate Perf Counter A27 ..... **262**
OAG Aggregate Perf Counter A27 Upper DWord **263**
OAG Aggregate Perf Counter A28 ..... **264**
OAG Aggregate Perf Counter A28 Upper DWord **265**
OAG Aggregate Perf Counter A29 ..... **266**
OAG Aggregate Perf Counter A29 Upper DWord **267**
OAG Aggregate Perf Counter A30 ..... **268**
OAG Aggregate Perf Counter A30 Upper DWord **269**
OAG Aggregate Perf Counter A31 ..... **270**
OAG Aggregate Perf Counter A31 Upper DWord **271**

OAG Aggregate Perf Counter A32 ..... **272**
OAG Aggregate Perf Counter A33 ..... **273**
OAG Aggregate Perf Counter A34 ..... **274**
OAG Aggregate Perf Counter A35 ..... **275**
OAG Aggregate Perf Counter SPM0 Lower DWord Free **276**
OAG Aggregate Perf Counter SPM0 Upper DWord Free **277**
OAG Aggregate Perf Counter SPM1 Lower DWord Free **278**
OAG Aggregate Perf Counter SPM1 Upper DWord Free **279**
OAG Aggregate Perf Counter SPM2 Lower DWord Free **280**
OAG Aggregate Perf Counter SPM2 Upper DWord Free **281**
OAG Aggregate Perf Counter SPM3 Lower DWord Free **282**
OAG Aggregate Perf Counter SPM3 Upper DWord Free **283**
OAG Aggregate Perf Counter SPM4 Lower DWord Free **284**
OAG Aggregate Perf Counter SPM4 Upper DWord Free **285**
OAG Aggregate Perf Counter SPM5 Lower DWord Free **286**
OAG Aggregate Perf Counter SPM5 Upper DWord Free **287**
OAG Any Media Fixed Function 0 Wake Count Free **288**
OAG Any Media Fixed Function 1 Wake Count Free **289**
OAG Any Media Fixed Function Busy Free **290**
OAG BLT Busy Free **291**
OAG BLT Wake Count Free **292**
OAG Boolean Counter B0 ..... **293**
OAG Boolean Counter B1 ..... **294**
OAG Boolean Counter B2 ..... **295**
OAG Boolean Counter B3 ..... **296**
OAG Boolean Counter B4 ..... **297**
OAG Boolean Counter B5 ..... **298**
OAG Boolean Counter B6 ..... **299**
OAG Boolean Counter B7 ..... **300**
OAG Context Control **301**
OAG Customizable Event Creation 0-0 ..... **303**
OAG Customizable Event Creation 1-0 ..... **305**
OAG Customizable Event Creation 1-1 ..... **307**
OAG Customizable Event Creation 2-0 ..... **308**

OAG Customizable Event Creation 2-1 310
OAG Customizable Event Creation 3-0 311
OAG Customizable Event Creation 3-1 313
OAG Customizable Event Creation 4-0 314
OAG Customizable Event Creation 5-0 316
OAG Customizable Event Creation 5-1 318
OAG Customizable Event Creation 6-0 319
OAG Customizable Event Creation 6-1 321
OAG Customizable Event Creation 7-0 322
OAG Customizable Event Creation 7-1 324
OAG GPU Ticks Counter 325
OAG GPU Ticks Counter Upper DWord 326
OAG GTI Busy Free 327
OAG GTI Wake Count Free 328
OAG GuC Busy Free 329
OAG GuC Wake Count Free 330
OAG Interrupt Mask Register 331
OAG MCPG0 Hysteresis Time Free 332
OAG Media Sampler Really Busy Free 333
OAG Media Sampler Wake Count Free 334
OAG Observation Architecture Buffer 335
OAG Observation Architecture Control 337
OAG Observation Architecture Report Trigger 2 339
OAG Observation Architecture Report Trigger 6 344
OAG Observation Architecture Report Trigger Counter 348
OAG Observation Architecture Start Trigger 5 349
OAG Observation Architecture Start Trigger Counter 350
OAG Observation Architecture Status Register 351
OAG Observation Architecture Tail Pointer 355
OAG RC0 Any Engine Busy Free 356
OAG RC0 Clocks Free 357
OAG RC0 Hysteresis Free 358
OAG RC0 Wake Count Free 359
OAG RC6 Entry Count Free 360

OAG RC6 Pre Engine Wake Overhead Free 361
OAG RCPG Hysteresis Time Free 362
OAG Render Busy Free 363
OAG Render Wake Count Free 364
OAG Slice Customizable Event Creation 0-0 365
OAG Slice Customizable Event Creation 1-0 367
OAG Slice Customizable Event Creation 1-1 369
OAG Slice Customizable Event Creation 2-0 370
OAG Slice Customizable Event Creation 2-1 372
OAG Slice Customizable Event Creation 3-0 373
OAG Slice Customizable Event Creation 3-1 375
OAG Slice Customizable Event Creation 4-0 376
OAG Slice Customizable Event Creation 5-0 378
OAG Slice Customizable Event Creation 5-1 380
OAG Slice Customizable Event Creation 6-0 381
OAG Slice Customizable Event Creation 6-1 383
OAG Slice Customizable Event Creation 7-0 384
OAG Slice Customizable Event Creation 7-1 386
OAG SQ Full Occupancy count Free 387
OAG SQ Memory Bound Stall count Free 388
OAG Unslice Busy Free 389
OAG Unslice Wake Count Free 390
OAM GPU Ticks Counter 391
OAM GPU Ticks Counter Upper DWord 392
OA Programmable Event Source Select Register 393
OAR Aggregate Perf Counter A0 394
OAR Aggregate Perf Counter A0 Upper DWord 395
OAR Aggregate Perf Counter A1 396
OAR Aggregate Perf Counter A1 Upper DWord 397
OAR Aggregate Perf Counter A2 398
OAR Aggregate Perf Counter A2 Upper DWord 399
OAR Aggregate Perf Counter A3 400
OAR Aggregate Perf Counter A3 Upper DWord 401
OAR Aggregate Perf Counter A4 402

OAR Aggregate Perf Counter A4 Upper DWord **403**
OAR Aggregate Perf Counter A5 ..... **404**
OAR Aggregate Perf Counter A5 Upper DWord **405**
OAR Aggregate Perf Counter A6 ..... **406**
OAR Aggregate Perf Counter A6 Upper DWord **407**
OAR Aggregate Perf Counter A7 ..... **408**
OAR Aggregate Perf Counter A7 Upper DWord **409**
OAR Aggregate Perf Counter A8 ..... **410**
OAR Aggregate Perf Counter A8 Upper DWord **411**
OAR Aggregate Perf Counter A9 ..... **412**
OAR Aggregate Perf Counter A9 Upper DWord **413**
OAR Aggregate Perf Counter A10 ..... **414**
OAR Aggregate Perf Counter A10 Upper DWord **415**
OAR Aggregate Perf Counter A11 ..... **416**
OAR Aggregate Perf Counter A11 Upper DWord **417**
OAR Aggregate Perf Counter A12 ..... **418**
OAR Aggregate Perf Counter A12 Upper DWord **419**
OAR Aggregate Perf Counter A13 ..... **420**
OAR Aggregate Perf Counter A13 Upper DWord **421**
OAR Aggregate Perf Counter A14 ..... **422**
OAR Aggregate Perf Counter A14 Upper DWord **423**
OAR Aggregate Perf Counter A15 ..... **424**
OAR Aggregate Perf Counter A15 Upper DWord **425**
OAR Aggregate Perf Counter A16 ..... **426**
OAR Aggregate Perf Counter A16 Upper DWord **427**
OAR Aggregate Perf Counter A17 ..... **428**
OAR Aggregate Perf Counter A17 Upper DWord **429**
OAR Aggregate Perf Counter A18 ..... **430**
OAR Aggregate Perf Counter A18 Upper DWord **431**
OAR Aggregate Perf Counter A19 ..... **432**
OAR Aggregate Perf Counter A19 Upper DWord **433**
OAR Aggregate Perf Counter A20 ..... **434**
OAR Aggregate Perf Counter A20 Upper DWord **435**
OAR Aggregate Perf Counter A21 ..... **436**

OAR Aggregate Perf Counter A21 Upper DWord **437**
OAR Aggregate Perf Counter A22 ..... **438**
OAR Aggregate Perf Counter A22 Upper DWord **439**
OAR Aggregate Perf Counter A23 ..... **440**
OAR Aggregate Perf Counter A23 Upper DWord **441**
OAR Aggregate Perf Counter A24 ..... **442**
OAR Aggregate Perf Counter A24 Upper DWord **443**
OAR Aggregate Perf Counter A25 ..... **444**
OAR Aggregate Perf Counter A25 Upper DWord **445**
OAR Aggregate Perf Counter A26 ..... **446**
OAR Aggregate Perf Counter A26 Upper DWord **447**
OAR Aggregate Perf Counter A27 ..... **448**
OAR Aggregate Perf Counter A27 Upper DWord **449**
OAR Aggregate Perf Counter A28 ..... **450**
OAR Aggregate Perf Counter A28 Upper DWord **451**
OAR Aggregate Perf Counter A29 ..... **452**
OAR Aggregate Perf Counter A29 Upper DWord **453**
OAR Aggregate Perf Counter A30 ..... **454**
OAR Aggregate Perf Counter A30 Upper DWord **455**
OAR Aggregate Perf Counter A31 ..... **456**
OAR Aggregate Perf Counter A31 Upper DWord **457**
OAR Aggregate Perf Counter A32 ..... **458**
OAR Aggregate Perf Counter A33 ..... **459**
OAR Aggregate Perf Counter A34 ..... **460**
OAR Aggregate Perf Counter A35 ..... **461**
OAR Boolean_Counter_B0 ..... **462**
OAR Boolean_Counter_B1 ..... **463**
OAR Boolean_Counter_B2 ..... **464**
OAR Boolean_Counter_B3 ..... **465**
OAR Boolean_Counter_B4 ..... **466**
OAR Boolean_Counter_B5 ..... **467**
OAR Boolean_Counter_B6 ..... **468**
OAR Boolean_Counter_B7 ..... **469**
OAR GPU Ticks Counter **470**

OAR GPU Ticks Counter Upper DWord 471
OAR Observation Architecture Control 472
OAR Observation Architecture Status Register 473
Observation Architecture Control Context ID 475
Observation Architecture Control per Context 476
Observation Architecture Head Pointer 478
OUTPUT_CSC_COEFF 479
OUTPUT_CSC_POSTOFF 481
OUTPUT_CSC_PREOFF 483
Outstanding Page Request Allocation 485
Outstanding Page Request Capacity 486
PAGE_FAULT_MODE 487
Page Directory Pointer Descriptor - PDP0/PML4/PASID 488
Page Directory Pointer Descriptor - PDP1 491
Page Directory Pointer Descriptor - PDP2 493
Page Directory Pointer Descriptor - PDP3 495
Page Req Queue Tail Shadow Register DW0 497
Page Req Queue Tail Shadow Register DW1 498
Page Request Control 499
Page Request Extended Capability Header 500
Page Request Queue Address Register 0 501
Page Request Queue Address Register 1 502
Page Request Queue Head Register 0 503
Page Request Queue Head Register 1 504
Page Request Queue Tail Register 0 505
Page Request Queue Tail Register 1 506
Page Request Status 507
PAK_NUM_OF_SLICES 509
PAK_Stream-Out Report (Errors) 510
PAK_Stream-Out Report (Warnings) 511
PAK Report Running Status 512
PAL_EXT_GC_MAX 513
PAL_EXT2_GC_MAX 515
PAL_GC_MAX 517

PAL_LGC 519
PAL_PREC_DATA 634
PAL_PREC_INDEX 636
PAL_PREC_MULTI_SEG_DATA 638
PAL_PREC_MULTI_SEG_INDEX 640
PASID Capability 642
PASID Control 643
PASID Extended Capability Header 644
PAT Index 645
PCI Command 647
PCI Express Capability 649
PCI Express Capability Header 650
PCI Express Capability Structure 651
PCI Express Device Control 653
PCI Mirror of GMCH Graphics Control 656
PCI Status 659
PCU Interrupt Definition 661
PHY_MISC 664
Pinned Surface Mapping Size Register 665
Pinned Surface Per Process Virtual Address Base Register 666
Pinned Surface Process ID Register 667
Pinned Surface Virtual Address Base Register 668
PIPE_ARB_CTL 669
PIPE_BOTTOM_COLOR 671
PIPE_DMCSCANLINECOMP 673
PIPE_DSS_CTL1 674
PIPE_DSS_CTL2 677
PIPE_FLIPCNT 679
PIPE_FLIPDONETMSTMP 680
PIPE_FLIPTMSTMP 681
PIPE_FRMCNT 682
PIPE_FRMTMSTMP 683
PIPE_ISOCREQ 684
PIPE_ISOCREQ_OFFSET 686

PIPE_MISC 688
PIPE_MISC2 694
PIPE_MISC3 696
PIPE_SCANLINE 697
PIPE_SCANLINECOMP 699
PIPE_SEAM_EXCESS 702
PIPE_SRCSZ 704
PIPE_STATUS 706
PIPEDMC_CONTROL 709
Pixel Shader Scheduling Mode 710
Pixel Shader Scheduling Mode 2 713
PLANE_AUX_DIST 716
PLANE_BUF_CFG 720
PLANE_CC_VAL 725
PLANE_COLOR_CTL 729
PLANE_CSC_COEFF 736
PLANE_CSC_POSTOFF 739
PLANE_CSC_PREOFF 742
PLANE_CTL 745
PLANE_CUS_CTL 754
PLANE_INPUT_CSC_COEFF 759
PLANE_INPUT_CSC_POSTOFF 762
PLANE_INPUT_CSC_PREOFF 765
PLANE_KEYMAX 768
PLANE_KEYMSK 772
PLANE_KEYVAL 777
PLANE_LEFT_SURF 781
PLANE_OFFSET 785
PLANE_PIXEL_NORMALIZE 790
PLANE_POS 792
PLANE_POST_CSC_GAMC_DATA 797
PLANE_POST_CSC_GAMC_DATA_ENH 800
PLANE_POST_CSC_GAMC_INDEX 803
PLANE_POST_CSC_GAMC_INDEX_ENH 806

PLANE_POST_CSC_GAMC_SEG0_DATA_ENH....809
PLANE_POST_CSC_GAMC_SEG0_INDEX_ENH....811
PLANE_PRE_CSC_GAMC_DATA....814
PLANE_PRE_CSC_GAMC_DATA_ENH....817
PLANE_PRE_CSC_GAMC_INDEX....820
PLANE_PRE_CSC_GAMC_INDEX_ENH....823
PLANE_SIZE....826
PLANE_STRIDE....832
PLANE_SURF....837
PLANE_SURFLIVE....842
PLANE_WM....849
POISON DATA HANDLING ENABLE....883
POISON DATA STATUS....884
PORT_CL_DW5....885
PORT_CL_DW10....887
PORT_CL_DW12....890
PORT_CL_DW15....892
PORT_CL_DW16....894
PORT_COMP_DW0....896
PORT_COMP_DW1....897
PORT_COMP_DW3....899
PORT_COMP_DW8....901
PORT_COMP_DW9....902
PORT_COMP_DW10....903
PORT_PCS_DW1....904
PORT_PCS_DW9....908
PORT_TX_DFLEXDPCSSS....911
PORT_TX_DFLEXDPMLE1....913
PORT_TX_DFLEXDPPMS....916
PORT_TX_DFLEXDPSP....918
PORT_TX_DFLEXNPCPMS....923
PORT_TX_DFLEXPA1....925
PORT_TX_DFLEXPA2....927
PORT_TX_DW0....929

PORT_TX_DW1 932
PORT_TX_DW2 935
PORT_TX_DW4 938
PORT_TX_DW5 941
PORT_TX_DW6 945
PORT_TX_DW7 948
PORT_TX_DW8 951
POSH LRCA 954
Power Clock State Register 956
Power Management Capabilities 957
Power Management Capabilities ID 959
Power Management Control and Status 960
PPPR 961
PPRO 962
PRE_CSC_GAMC_DATA 963
PRE_CSC_GAMC_INDEX 965
Predicate Rendering Data Result 967
Predicate Rendering Data Result 1 968
Predicate Rendering Data Result 2 970
Predicate Rendering Data Storage 972
Predicate Rendering Temporary Register0 973
Predicate Rendering Temporary Register1 974
Preemption Status 975
PRERESETMessagingRegister 978
Primary Latency Timer 981
Primitives Generated By VF 982
PS_ADAPTIVE_CTRL 983
PS_COEF_DATA 986
PS_COEF_INDEX 989
PS_CTRL 992
PS_ECC_STAT 998
PS_HPHASE 1000
PS_HSCALE 1002
PS_PROG_HSCALE 1004

PS_PROG_VSCALE 1006
PS_PWR_GATE 1008
PS_VPHASE 1010
PS_VSCALE 1013
PS_WIN_POS 1015
PS_WIN_SZ 1017
PS Depth Count 1019
PS Depth Count for Slice0 1020
PS Depth Count for Slice1 1021
PS Depth Count for Slice2 1022
PS Depth Count for Slice3 1023
PS Depth Count for Slice4 1024
PS Depth Count for Slice5 1025
PS Depth Count for Slice6 1026
PS Depth Count for Slice7 1027
PS Invocation Count 1028
PS Invocation Count for Slice0 1029
PS Invocation Count for Slice1 1030
PS Invocation Count for Slice2 1031
PS Invocation Count for Slice3 1032
PS Invocation Count for Slice4 1033
PS Invocation Count for Slice5 1034
PS Invocation Count for Slice6 1035
PS Invocation Count for Slice7 1036
PSR_EVENT 1037
PSR_IIR 1041
PSR_IMR 1043
PSR_MASK 1045
PSR2_CTL 1048
PSR2_MAN_TRK_CTL 1052
PSR2_STATUS 1054
PSR2_SU_ECC_STAT 1057
PSR2_SU_STATUS 1058
PTBR Number Of Pages Recorded 1059

PTBR Page Pool Size Register **1060**
PTE SW Fault Repair High **1061**
PTE SW Fault Repair Low **1062**
PWR_WELL_CTL **1063**
PWR_WELL_CTL_AUX **1066**
PWR_WELL_CTL_DDI **1072**
PWRCTXSAVE Message Register for Boot Controller Unit **1076**
RC6 Context Base **1077**
RC6 LOCATION **1078**
Register_HCPBitstreamOutputCABACInsertionCount **1079**
Register_HCPSFCLOCKRequest **1080**
Register_HCPSFCLOCKStatus **1081**
Render and Media Context Restore Needed **1082**
Render Control Unit Mode Register **1083**
Render Control Unit Power Clock State Register **1085**
Reported BitRateControl Convergence Status **1086**
Reported BitRateControl CumulativeDeltaLoopFilter and LoopFilter 01 ..... **1087**
Reported BitRateControl CumulativeDeltaLoopFilter and LoopFilter 23 ..... **1089**
Reported BitRateControl CumulativeDeltaQindex and Qindex 01 ..... **1091**
Reported BitRateControl CumulativeDeltaQindex and Qindex 23 ..... **1092**
Reported BitRateControl DeltaLoopFilter **1093**
Reported BitRateControl DeltaQindex **1095**
Reported BitRateControl parameter Mask **1096**
Reported BitRateControl parameter Status **1098**
Reported Bitstream Output Bit Count for Syntax Elements Only **1100**
Reported Bitstream Output Bit Count for Syntax Elements Only Register **1101**
Reported Bitstream Output Byte Count per Frame Register **1102**
Reported Bitstream Output Byte Count per Tile **1103**
Reported Bitstream Output CABAC Bin Count Register **1104**
Reported Final Bitstream Byte Count **1105**
Reported Frame Zero Padding Byte Count **1106**
Reported Timestamp Count **1107**
Report Queue CFG HI **1110**
Report Queue CFG LO **1111**

Reset Control Register 1112
Reset Flow Control Messages 0 1115
Revision Identification and Class Code register 1116
RING_BUFFER_HEAD_PREEMPT_REG 1117
Ring Buffer Control 1120
Ring Buffer Head 1124
Ring Buffer Start 1127
Ring Buffer Tail 1129
Ring Context Address Register 0 for GTI Doorbell Unit 1132
Ring Context Start Register for GTI Doorbell Unit 1133
RMTIMEOUTREG_CAPTURE 1134
RSA for uOS/Soft Scratch 1135
RTADDR_LSB 1136
RTADDR_MSB 1137
RTT_CR_SPARE 1138
Sampler control register 1139
Sampler Dummy Register 1140
SAMPLER Mode Register 1141
SAMPLER READ DATA 1144
SCRATCH 1 from LPFCunit 1145
SCRATCH 2 for LNCFunit 1146
SCRATCH2 Register 1147
SCRATCH 3 for LNCFunit 1149
SCRATCH3 Register 1150
Second Level Batch Buffer Head Pointer Preemption Register 1152
Second Level Batch Buffer Head Pointer Register 1155
Second Level Batch Buffer State Register 1158
Second Level Batch Buffer Upper Head Pointer Preemption Register 1161
Second Level Batch Buffer Upper Head Pointer Register 1163
SEL_FETCH_PLANE_CTL 1165
Semaphore Interrupt Register 1167
Semaphore Polling Interval on Wait 1168
Semaphore Signal Port 1170
Semaphore Token 1172

Slice 0 BONUS1 Reg **1174**
Slice 0 BONUS2 Reg **1176**
Slice 0 PGFET control register with lock **1178**
Slice 0 Power Context Save request **1180**
Slice 0 Power Down FSM control register with lock **1181**
Slice 0 Power Gate Control Request **1183**
Slice 0 Power on FSM control register with lock **1184**
Slice 0 SubSlice 0 PGFET control register with lock **1185**
Slice 0 SubSlice 1 PGFET control register with lock **1187**
Slice 0 SubSlice 2 PGFET control register with lock **1189**
Slice 1 - 5 BONUS1 Reg **1191**
Slice 1 - 5 BONUS2 Reg **1193**
Slice 1 - 5 PGFET control register with lock **1195**
Slice 1 - 5 Power Down FSM control register with lock **1197**
Slice 1 - 5 Power Gate Control Request **1200**
Slice 1 -5 Power on FSM control register with lock **1201**
Slice unit Level Clock Gating Control 94D0 ..... **1203**
Slice unit Level Clock Gating Control 94D4 ..... **1204**
Slice unit Level Clock Gating Control 94D8 ..... **1210**
Slice unit Level Clock Gating Control 94DC **1216**
Slice unit Level Clock Gating Control 94E0 ..... **1217**
Slice unit Level Clock Gating Control 94E4 ..... **1223**
Slice unit Level Clock Gating override during rstflow 94F0 ..... **1230**
SLM Bank Hash **1231**
SNP_FILTER_Q_STATUS **1234**
Soft Scratch **1235**
Software SCI **1236**
Software SMI **1237**
SRD_CTL **1238**
SRD_PERF_CNT **1243**
SRD_STATUS **1244**
SSM0 BONUS1 Reg **1247**
SSM0 BONUS2 Reg **1249**
SSM1 BONUS1 Reg **1251**

SSM1 BONUS2 Reg 1253
SSM2 BONUS1 Reg 1255
SSM2 BONUS2 Reg 1257
ssmcp Sub Slice Power Context Save request 1259
Stream Output 0 Num Primitives Written Counter 1260
Stream Output 0 Primitive Storage Needed Counter 1261
Stream Output 0 Write Offset 1262
Stream Output 1 Num Primitives Written Counter 1263
Stream Output 1 Primitive Storage Needed Counter 1264
Stream Output 1 Write Offset 1265
Stream Output 2 Num Primitives Written Counter 1266
Stream Output 2 Primitive Storage Needed Counter 1267
Stream Output 2 Write Offset 1268
Stream Output 3 Num Primitives Written Counter 1269
Stream Output 3 Primitive Storage Needed Counter 1270
Stream Output 3 Write Offset 1271
SubSlice0 Power Down FSM control register with lock 1272
SubSlice 0 Power Gate Control Request 1275
SubSlice 0 Power on FSM control register with lock 1276
SubSlice 1 Power Context Save request 1278
SubSlice 1 Power Down FSM control register with lock 1279
SubSlice 1 Power Gate Control Request 1282
SubSlice 1 Power on FSM control register with lock 1283
SubSlice 2 Power Down FSM control register with lock 1285
SubSlice 2 Power Gate Control Request 1288
SubSlice 2 Power on FSM control register with lock 1289
SubSlice unit Level Clock Gating Control 9520 1291
SubSlice unit Level Clock Gating Control 9524 1292
SubSlice unit Level Clock Gating Control 9528 1297
SubSlice unit Level Clock Gating Control 9530 1300
SubSlice unit Level Clock Gating override during rstflow 9540 1303
Subsystem Identification 1304
Subsystem Vendor Identification 1305
Supported Page Sizes 1306

SVG Debug Control 1307
SVG Debug Return Data 1308
SVL Barrier Done 1309
SWF 1310
System Page Sizes 1311
Tailpointer delay counter 1312
Thread Dispatched Count Register 1313
Thread Faulted Count Register 1314
Thread Fault Status Register 0 1315
Thread Fault Status Register 1 1316
Thread Load Status Register 0 1317
Thread Load Status Register 1 1318
Thread Mode Register 1319
Thread Restart Control Register 1322
Tile Cache Control Register 1323
TILECTL 1325
TiledResources Invalid Tile Detection Register 1326
TiledResources VA Detection Registers 1327
Tiled Resources Wrapper Write Data Port arbitration 1328
TIMESTAMP_CTR 1330
TLBInvalidationRegister_BLT 1331
TLBInvalidationRegister_GFX 1332
TLBInvalidationRegister_GUC 1333
TLBInvalidationRegister_OA 1334
Top of Low Usable DRAM Register 1335
TOUUD_LSB_REG 1336
TOUUD_MSB_REG 1337
TRANS_CLK_SEL 1338
TRANS_CONF 1340
TRANS_DDI_FUNC_CTL 1343
TRANS_DDI_FUNC_CTL2 1349
TRANS_DSI_FUNC_CONF 1352
TRANS_EXITLINE 1358
TRANS_HBLANK 1360

TRANS_HSYNC 1362
TRANS_HTOTAL 1364
TRANS_MSA_MISC 1367
TRANS_MULT 1369
TRANS_PUSH 1370
TRANS_SPACE 1371
TRANS_STEREO3D_CTL 1373
TRANS_VBLANK 1377
TRANS_VRR_CTL 1379
TRANS_VRR_FLIPLINE 1381
TRANS_VRR_STATUS 1382
TRANS_VRR_STATUS2 1385
TRANS_VRR_VMAX 1386
TRANS_VRR_VMAXSHIFT 1387
TRANS_VRR_VMIN 1389
TRANS_VRR_VTOTAL_PREV 1390
TRANS_VSYNC 1392
TRANS_VSYNCSHIFT 1394
TRANS_VTOTAL 1396
TRANS_WD_FUNC_CTL 1398
TRNULLDETCT 1401
TSEG Base Memory 1402
Ungated Clock Counter for DFR Testability 1403
Unslice unit Level Clock Gating Control 943C 1404
Unslice unit Level Clock Gating Control 9430 1409
Unslice unit Level Clock Gating Control 9434 1415
Unslice unit Level Clock Gating Control 9438 1421
Unslice unit Level Clock Gating Control 9440 1425
Unslice unit Level Clock Gating Control 9444 1432
Unslice unit Level Clock Gating Control 9448 1433
Unslice unit Level Clock Gating Control 9450 1434
Unslice unit Level Clock Gating Control 9454 1439
uOS Full Hash High 1443
uOS Full Hash Low 1444

URB Runtime ECC capture Register 1445
URB Runtime ECC Test Control Register 1446
UTIL_PIN_BUF_CTL 1447
UTIL_PIN_CTL 1448
UTIL2_PIN_BUF_CTL 1450
UTIL2_PIN_CTL 1451
Valid Bit Vector 3 for RCC Register 1452
VCS CSB 1453
VCS CSB Fifo Status Register 1454
VDBox TLB Invalidation Register 1455
Vdbox unit Level Clock Gating Control 3F0C 1458
Vdbox unit Level Clock Gating Control 3F04 1465
Vdbox unit Level Clock Gating Control 3F08 1473
Vdbox unit Level Clock Gating Control 3F10 1481
Vdbox unit Level Clock Gating Control 3F14 1488
Vdbox unit Level Clock Gating Control 3F18 1497
Vdbox unit Level Clock Gating override during rstflow 1504
VDENC current encoded bitstream quality 1505
VDENC FPAK Tile completion message 1507
VDENC X_Y_POS 1509
VDMBDFBARKVM 1510
VEBOX TLB Invalidation Register 1511
Vebox unit Level Clock Gating Control 3F04 1513
Vebox unit Level Clock Gating Control 3F08 1518
Vebox unit Level Clock Gating override during rstflow 1520
VECS CSB 1521
VECS CSB Fifo Status Register 1522
Vendor Identification 1523
VEO Current Pipe 0 XY Register 1524
VEO DN Pipe 0 XY Register 1525
VEO DN Pipe 1 XY Register 1526
VEO DV Count Register 1527
VEO DV Hold Register 1528
VEO IECP DV Count Register 1531

VEO Previous Pipe 0 XY Register 1532
VEO State Register 1533
VE SFC Forced Lock Acknowledgement Register 1535
VE SFC Forced Lock Register 1536
VE VFW SFC Usage Register 1537
VF_CAPABILITY_REGISTER 1538
VF_SW_FLAG 1539
VF BAR0 LDW 1540
VF BAR0 UDW 1542
VF BAR1 LDW 1543
VF BAR1 UDW 1544
VF BAR2 LDW 1545
VF BAR2 UDW 1546
VF Device ID 1547
VF Migration State Array Offset 1548
VF Scratch Pad 1549
VF Stride 1553
VFW Credit Count Register 1554
VGA_CONTROL 1555
VIDEO_DIP_CTL 1557
VIDEO_DIP_DATA 1561
VIDEO_DIP_DRM_DATA 1580
VIDEO_DIP_DRM_ECC 1584
VIDEO_DIP_ECC 1586
VIDEO_DIP_GCP 1594
VIDEO_DIP_PPS_DATA 1596
VIDEO_DIP_PPS_ECC 1611
Video BIOS ROM Base Address 1616
VIRTUALIZATION CONTROL REGISTER 1617
VSC_EXT_SDP_CONF 1618
VSC_EXT_SDP_CTL 1620
VSC_EXT_SDP_DATA 1623
VSC_EXT_SDP_HEADER 1625
VS Invocation Counter 1627

VSR_PUSH_CONSTANT_BASE 1628
VTd Status 1629
Wait For Event and Display Flip Flags Register 1630
Wait For Event and Display Flip Flags Register 1 1634
Wait For Event and Display Flip Flags Register 2 1639
Wait For Event and Display Flip Flags Register 3 1645
Wait For Event and Display Flip Flags Register 4 1651
Wait For Event and Display Flip Flags Register 5 1657
Wait For Event and Display Flip Flags Register 6 1663
Watchdog Counter 1666
Watchdog Counter Control 1668
Watchdog Counter Threshold 1671
WD_27_M 1673
WD_27_N 1674
WD_FRAME_STATUS 1675
WD_IIR 1677
WD_IMR 1678
WD_PERF_CNT 1679
WD_STRIDE 1680
WD_SURF 1681
WD_TAIL_CFG 1682
WD_TAIL_CFG2 1684
WIDI Session 0 1685
WIDI Session 1 1686
WIDI Session 2 1687
WIDI Session 3 1688
WiDi VDEnc Stall counter 1689
Window Hardware Generated Clear Value 1690
WM_LINETIME 1691
WM_MISC 1692

MAILBOX0

MAILBOX0 - MAILBOX0

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	120800h

This register contains bits 31:0 of the generic descriptor, fetched from the invalidate queue for GT.

DWord	Bit	Description
0	31:0	**DATA** Default Value: 00000000h Access: R/W _Custom_GTIReset: BUS This field contains bits 31:0 of the generic descriptor, fetched from the invalidate queue for GT.

MAILBOX1

MAILBOX1 - MAILBOX1

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	120804h

This register contains bits 63:32 of the generic descriptor, fetched from the invalidate queue for GT.

DWord	Bit	Description
0	31:0	**DATA** Default Value: 00000000h Access: R/W _Custom_GTIReset: BUS This field contains bits 63:32 of the generic descriptor, fetched from the invalidate queue for GT.

MAILBOX2

MAILBOX2 - MAILBOX2		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
SOC_Consumer:	BIOS	
Address:	120808h	
This register contains bits 95:64 of the generic descriptor, fetched from the invalidate queue for GT.		

DWord	Bit	Description
0	31:0	**DATA** Default Value: 00000000h Access: R/W _Custom_GTIReset: BUS This field contains bits 95:64 of the generic descriptor, fetched from the invalidate queue for GT.

MAILBOX3

MAILBOX3 - MAILBOX3

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	12080Ch

This register contains bits 127:96 of the generic descriptor, fetched from the invalidate queue for GT.

DWord	Bit	Description
0	31:0	**DATA** Default Value: 00000000h Access: R/W _Custom_GTIReset: BUS This field contains bits 127:96 of the generic descriptor, fetched from the invalidate queue for GT.

Maximum Latency

MAXLAT_0_2_0_PCI - Maximum Latency		
Register Space:	PCI: 0/2/0	
Size (in bits):	8	
Address:	0003Fh	
The Integrated Graphics Device has no requirement for the settings of Latency Timers.		
DWord	**Bit**	**Description**
0	7:0	**Maximum Latency Value** Default Value: 00000000b Access: RO _Custom_GTIReset: BUS Hardwired to 0s because the IGD has no specific requirements for how often it needs to access the PCI bus.

MBC Control Register

MBCTL - MBC Control Register

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	0907Ch

MBC Control Register

DWord	Bit	Description
0	31:18	**ECORSVD** Access: R/W _Custom_GTIReset: BUS ECO purposes Reserved
	17	**U2C Global PMON Enable Override** Default Value: 1b Access: R/W _Custom_GTIReset: BUS U2C Performance Monitor Global Enable Override 0 - U2C Global PMON needs to be enabled for performance monitors to be enabled (default) 1 - Overrride U2C Global PMON Enable is ignored ine baled performance monitor counters
	16	**VCR Fuse Writes as Posted** Access: R/W _Custom_GTIReset: BUS Non-posted fuse fetching is NOT supported. Only posted is allowed (the default). 0 - MBCunit sends VCR Fuse Writes as Non-posted. 1 - MBCunit sends VCR Fuse Writes as posted.
	15:8	**Reserved** Access: RO Format: MBZ
	7	**Disable Wait for SQemtpy in MAE** Access: R/W _Custom_GTIReset: BUS 0 - Wait for SQempty for MAE update Flow. 1 - MBC MAE update FSM does not wait for the SQempty to complete the FSM.
	6	**Reserved**

MBCTL - MBC Control Register

	5:4	**Reserved** Access: RO Format: MBZ
	3	**Context Fetch Needed** Access: R/W _Custom_GTIReset: BUS Context Fetch Needed for Power Exits. 0 - Context Fetch Not Needed. 1 - Context Fetch Needed for Power Exits (CPD Entry).
	2:0	**Reserved** Access: RO Format: MBZ

MBUS_ABOX_CTL

MBUS_ABOX_CTL

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

Address:	45038h-4503Bh
Name:	MBus ABox Control
ShortName:	MBUS_ABOX_CTL
Reset:	soft

Address:	45048h-4504Bh
Name:	MBus ABox 1 Control
ShortName:	MBUS_ABOX1_CTL
Reset:	soft

Address:	4504Ch-4504Fh
Name:	MBus ABox 2 Control
ShortName:	MBUS_ABOX2_CTL
Reset:	soft

DWord	Bit	Description
0	31	**Status** Access: RO This field indicates if the box is enabled. 0b: Disabled 1b: Enabled
	30:27	**Ring Stop Address** Access: RO This field indicates the address of the box in the ring.
	26:22	**B2B Transactions Max** This fields indicates the number of back to back transactions that can be added to either to top or bottom of the ring when 'Regulate Transactions' bit is 1b. 10: **[Default]** [1-31]
	21:20	**BW Credits** Default Value: 1h BW credits are used by the Arbiter to initiate write cycles when performing VRH read-modified-writes to the display buffer.

MBUS_ABOX_CTL

	19:16	**B Credits** Default Value: 1h B credits are used by the Arbiter to initiate read cycles when performing VRH read-modified-writes to the display buffer.
	15:14	**Reserved** Access: RO Format: MBZ
	13	**Regulate B2B Transactions** This field controls the regulation of back to back transactions from this ring stop. 0b: Disable 1b: Enable **[Default]**
	12:8	**BT Credits Pool2** Default Value: 10h BT credits are used by the Arbiter to request trackers from the Display Buffer.
	7:5	**B2B Transactions Delay** This field indicates the number of wait cycles after the maximum back to back transactions is sent. [0-7] 2 **[Default]**
	4:0	**BT Credits Pool1** Default Value: 10h BT credits are used by the Arbiter to request trackers from the Display Buffer.

MBUS_BBOX_CTL

MBUS_BBOX_CTL

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

Address:	45040h-45043h
Name:	MBus BBox 1 Control
ShortName:	MBUS_BBOX_CTL_S1
Reset:	soft

Address:	45044h-45047h
Name:	MBus BBox 2 Control
ShortName:	MBUS_BBOX_CTL_S2
Reset:	soft

DWord	Bit	Description
0	31	**Status** Access: RO This field indicates if the box is enabled. 0b: Disabled 1b: Enabled
	30:27	**Ring Stop Address** Access: RO This field indicates the address of the box in the ring.
	26:25	**Reserved** Access: RO Format: MBZ
	24:20	**B2B Transactions Max** This fields indicates the number of back to back transactions that can be added to either to top or bottom of the ring when 'Regulate Transactions' bit is 1b. 16: **[Default]** [1-31]
	19:17	**B2B Transactions Delay** This field indicates the number of wait cycles after the maximum back to back transactions is sent.

MBUS_BBOX_CTL

		Value / **Name** [0-7] | 1 | **[Default]**
	16	**Regulate B2B Transactions** This field controls the regulation of back to back transactions from this ring stop. **Value** / **Name** 0b | Disable 1b | Enable **[Default]**
	15:0	**Reserved** Access: | RO Format: | MBZ

MBUS_DBOX_CTL

MBUS_DBOX_CTL	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank OR pipe disabled

Address:	7003Ch-7003Fh
Name:	Pipe MBus DBox Control
ShortName:	PIPE_MBUS_DBOX_CTL_A
Reset:	soft

Address:	7103Ch-7103Fh
Name:	Pipe MBus DBox Control
ShortName:	PIPE_MBUS_DBOX_CTL_B
Reset:	soft

Address:	7203Ch-7203Fh
Name:	Pipe MBus DBox Control
ShortName:	PIPE_MBUS_DBOX_CTL_C
Reset:	soft

Address:	7303Ch-7303Fh
Name:	Pipe MBus DBox Control
ShortName:	PIPE_MBUS_DBOX_CTL_D
Reset:	soft

DWord	Bit	Description
0	31	**Status** Access: RO This field indicates if the box is enabled. Value / Name: 0b Disabled; 1b Enabled
	30:27	**Ring Stop Address** Access: RO This field indicates the address of the box in the ring.
	26:25	**Reserved** Access: RO Format: MBZ

MBUS_DBOX_CTL

24:20	**B2B Transactions Max** This fields indicates the number of back to back transactions that can be added to either to top or bottom of the ring when 'Regulate Transactions' bit is 1b.

Value	Name
16	**[Default]**
[1-31]	

19:17	**B2B Transactions Delay** This field indicates the number of wait cycles after the maximum back to back transactions is sent.

Value	Name
[0-7]	
1	**[Default]**

16	**Regulate B2B Transactions** This field controls the regulation of back to back transactions from this ring stop.

Value	Name
0b	Disable
1b	Enable **[Default]**

15:14	**BW Credits** BW credits are used by the display pipe to write color clear data to DBUF.

Value	Name
1h	**[Default]**

13	**Reserved**

Access:	RO
Format:	MBZ

12:8	**B Credits** B credits are used by the display pipe to request data from display buffer.

Value	Name
06h	**[Default]**

7:5	**Reserved**

Access:	RO
Format:	MBZ

4	**Reserved**

Access:	RO
Format:	MBZ

3:0	**A Credits** A credits are used by the display pipe to make data/TLB/VTd/MCS requests to Arbiter.

Value	Name
2h	2 credits **[Default]**

MBUS_UBOX_CTL

MBUS_UBOX_CTL

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	4503Ch-4503Fh
Name:	MBus UBox Control
ShortName:	MBUS_UBOX_CTL
Reset:	soft

DWord	Bit	Description
0	31	**Status** Access: RO This field indicates if the box is enabled. 0b: Disabled 1b: Enabled
	30:27	**Ring Stop Address** Access: RO This field indicates the address of the box in the ring.
	26:25	**Reserved** Access: RO Format: MBZ
	24:20	**B2B Transactions Max** This fields indicates the number of back to back transactions that can be added to either to top or bottom of the ring when 'Regulate Transactions' bit is 1b. 16: **[Default]** [1-31]
	19:17	**B2B Transactions Delay** This field indicates the number of wait cycles after the maximum back to back transactions is sent. [0-7] 1: **[Default]**
	16	**Regulate B2B Transactions** This field controls the regulation of back to back transactions from this ring stop.

MBUS_UBOX_CTL

Value	Name
0b	Disable
1b	Enable **[Default]**

15:7 Reserved

Access:	RO
Format:	MBZ

6:4 KVM Sprite A Credits

Default Value:	1

A Credits used by KVM to make data requests to Arbiter.

3 Reserved

Access:	RO
Format:	MBZ

2:0 VGA B Credits

Default Value:	4

B credits used by VGA to request data from Display Buffer.

MCPG1 Hysteresis Time Free

OAG_MCPG1_HYSTERESIS_TIME_FREE - MCPG1 Hysteresis Time Free		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DBB4h	
This register counts the time that MCPG hysteresis time is accumulating for media slice 1. This counter is free running, always enabled and counting irrespective of OA enabled or disabled.		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

MCR Packet Control

MCRPKT_CTRL - MCR Packet Control

Register Space: MMIO: 0/2/0
Size (in bits): 32
_Custom_GTIReset: BUS

Address: 00FDCh-00FDFh

DWord	Bit	Description
0	31	**MULTICAST** Access: R/W Value determines the multicast value driven to MCR. 0 - not multicast 1 - multicast The usage model is that the value is returned to the default (1), after completion of the unicast request. Value / Name: 1b: Multicast **[Default]** 0b: Unicast Programming Notes: SLICEID and DUAL_SUBSLICEID are used in Unicast but ignored for Multicast.
	30:27	**SLICEID** Access: R/W Value determines the slice ID driven to MCR. Value / Name: 0000b: **[Default]** [0001b,1111b]: Programming Notes: Values must be in range of slices and subslices in the configuration.
	26:24	**DUAL_SUBSLICEID** Access: R/W Value determines the dual-subslice ID (or l3_bank) driven to MCR. Value / Name: 000b: **[Default]** [001b,111b]: Programming Notes: Values must be in range of slices and subslices in the configuration.

MCRPKT_CTRL - MCR Packet Control		
	23:0	**Reserved** Access: RO Format: MBZ

MCR Packet Control 0

MCRPKT_CTRL_MCFG - MCR Packet Control 0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00FD0h-00FD3h	
DWord	**Bit**	**Description**
0	31	**MULTICAST MCFG** Default Value: 1b Access: R/W Value determines the multicast value driven to MCR for accesses to MCFG. 0 - not multicast 1 - multicast The usage model is that the value is returned to the default (1), after completion of the unicast request.
	30:27	**SLICEID MCFG** Default Value: 0000b Access: R/W Value driven on packet field Slice ID targeting MCFG.
	26:24	**DUAL_SUBSLICEID MCFG** Default Value: 000b Access: R/W Value determines the SQIDI ID driven on packet field Dual-Subslice ID targeting MCFG.
	23:0	**Reserved** Access: RO Format: MBZ

MCRPacketControl1

MCRPKT_CTRL_MDRB - MCRPacketControl1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00FD4h-00FD7h	
DWord	**Bit**	**Description**
0	31	**MULTICAST MDRB** Default Value: 1b Access: R/W Value determines the multicast value driven to MCR for accesses to MDRB. 0 - not multicast 1 - multicast The usage model is that the value is returned to the default (1), after completion of the unicast request.
	30:27	**SLICEID MDRB** Default Value: 0000b Access: R/W Value driven on packet field Slice ID targeting MDRB.
	26:24	**DUAL_SUBSLICEID MDRB** Default Value: 000b Access: R/W Value determines the SQIDI ID driven on packet field Dual-Subslice ID targeting MDRB.
	23:0	**Reserved** Access: RO Format: MBZ

MCR Packet Control 2

MCRPKT_CTRL_SF - MCR Packet Control 2		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00FD8h-00FDBh	
DWord	**Bit**	**Description**
0	31	**MULTICAST SF** Default Value: 1b Access: R/W Value determines the multicast value driven to MCR for accesses to SF. 0 - not multicast 1 - multicast The usage model is that the value is returned to the default (1), after completion of the unicast request.
	30:27	**SLICEID SF** Default Value: 0000b Access: R/W Value driven on packet field Slice ID targeting SF.
	26:24	**DUAL_SUBSLICEID SF** Default Value: 000b Access: R/W Value determines the SQIDI ID driven on packet field Dual-Subslice ID targeting SF.
	23:0	**Reserved** Access: RO Format: MBZ

MDRB Context Base 1

MDRB_CTXBASE1 - MDRB Context Base 1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	00DC8h	
RC6 Save Location NOTE: This register is not used since MDRB unit no longer exists. This will be removed in a future project		
DWord	**Bit**	**Description**
0	31:6	**MDRB Memory Base Low** Access: R/W Lock _Custom_GTIReset: BUS This field is used to set the base of memory where the RC6 power context will be saved This value MUST be above the base and below the top of stolen memory This register is locked (becomes read-only) when RC6MEMLOCK is 1
	5:1	**Reserved** Access: RO Format: MBZ
	0	**Ctx Base is Enabled** Access: R/W Lock _Custom_GTIReset: BUS 1'b0 : The MDRB base has not been enabled, so don't do the MDRB context save (This is default value and BIOS has to program it to enable context save) 1'b1 : The MDRB base has been enabled, so go ahead with the context save

MDRB Context Base 2

MDRB_CTXBASE2 - MDRB Context Base 2		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	00DCCh	
RC6 Base Location NOTE: This register is not used since MDRB unit is no more there.		
DWord	**Bit**	**Description**
0	31:0	**MDRB Memory Base High** Access: R/W Lock _Custom_GTIReset: BUS This field is used to set the base of memory where the RC6 power context will be saved This value MUST be above the base and below the top of stolen memory This register is locked (becomes read-only) when RC6MEMLOCK is 1

MED0HCP0PowerGoodDelay

MED0HCP0_PG_DLY - MED0HCP0PowerGoodDelay		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
IA/Context savable by MGSR/punit		
DWord	**Bit**	**Description**
0	31:16	**Power Good Delay** Default Value: 00C8h PWRGOOD_DLY Value
	15:0	**RAMP Delay** Default Value: 00C8h RAMP_DLY Value

MED0MFX0PowerGoodDelay

MED0MFX0_PG_DLY - MED0MFX0PowerGoodDelay		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
IA/Context savable by MGSR/punit		
DWord	**Bit**	**Description**
0	31:16	**Power Good Delay** Default Value: 00C8h PWRGOOD_DLY Value
	15:0	**RAMP Delay** Default Value: 00C8h RAMP_DLY Value

MED0PowerGoodDelay

MED0_PG_DLY - MED0PowerGoodDelay		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
IA/Context savable by MGSR/punit		
DWord	**Bit**	**Description**
0	31:16	**Power Good Delay** Default Value: 00C8h PWRGOOD_DLY Value
	15:0	**RAMP Delay** Default Value: 00C8h RAMP_DLY Value

MED1HCP0PowerGoodDelay

MED1HCP0_PG_DLY - MED1HCP0PowerGoodDelay		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
IA/Context savable by MGSR/punit		
DWord	**Bit**	**Description**
0	31:16	**Power Good Delay** Default Value: 00C8h PWRGOOD_DLY Value
	15:0	**RAMP Delay** Default Value: 00C8h RAMP_DLY Value

MED1MFX0PowerGoodDelay

MED1MFX0_PG_DLY - MED1MFX0PowerGoodDelay		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
IA/Context savable by MGSR/punit		
DWord	**Bit**	**Description**
0	31:16	**Power Good Delay** Default Value: 00C8h PWRGOOD_DLY Value
	15:0	**RAMP Delay** Default Value: 00C8h RAMP_DLY Value

MED1PowerGoodDelay

MED1_PG_DLY - MED1PowerGoodDelay		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
IA/Context savable by MGSR/punit		
DWord	**Bit**	**Description**
0	31:16	**Power Good Delay** Default Value: 00C8h PWRGOOD_DLY Value
	15:0	**RAMP Delay** Default Value: 00C8h RAMP_DLY Value

Media Die Recovery

MED_DIE_RECOVERY - Media Die Recovery	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	12298h
Name:	VCS Media Die Recovery
ShortName:	VCS_MED_DIE_RECOVERY

This register is stored in the VCS but is used in the HWM unit. This register programs the die recovery override and engine ID's.

DWord	Bit	Description
0	31:12	**Reserved** Access: RO Format: MBZ
	11:9	**Forced Next Engine ID** Access: R/W This field is the next engine ID.
	8	**Force Next Engine ID** Access: R/W The bit forces the next engine ID.
	7:4	**Reserved** Access: RO Format: MBZ
	3:1	**Forced Previous Engine ID** Access: R/W This field is the previous engine ID.
	0	**Force Previous Engine ID** Access: R/W The bit forces the previous engine ID.

Media FIFO Messaging Register for Shadow Register Unit

MSG_FIFO_MGSR_MEDIA - Media FIFO Messaging Register for Shadow Register Unit

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	080E4h
Name:	Media FIFO Messaging Register for Shadow Register Unit
ShortName:	MSG_FIFO_MGSR_MEDIA

Register that has the ACK information, back from MGSR as to whether a specific VD/VE Box has been blocked/unblocked
0 -- Box has been blocked
1 -- Box has been unblocked
Message registers have bit-wise masking applied for writes. The register consists of 16 bits of data in [15:0], and 16 bits of corresponding masks in [31:16].
To set bit0, for example, the data would be 0x0001_0001.
To clear bit0, for example, the data would be 0x0001_0000.
Note that mask bit is the data bit offset + 16.
Message registers are protected from non-GT writes via the Message Channel.

DWord	Bit	Description
0	31:12	**Reserved** Access: RO Format: MBZ
	11	**Acknowledge that Media FIFO has been Blocked for VEBOX3** Access: R/W _Custom_GTIReset: BUS Acknowledge that MEDIA FIFO has been Blocked for VEBOX3 1'b0 : Media FIFO Block Ack for VEBOX3(default) 1'b1 : Media FIFO Unblock Ack VEBOX3 Note that the block request goes to MGSR, and MGSR responds to GPM with the blocked ack.
	10	**Acknowledge that Media FIFO has been Blocked for VEBOX2** Access: R/W _Custom_GTIReset: BUS Acknowledge that MEDIA FIFO has been Blocked for VEBOX2 1'b0 : Media FIFO Block Ack for VEBOX2(default) 1'b1 : Media FIFO Unblock Ack VEBOX2 Note that the block request goes to MGSR, and MGSR responds to GPM with the blocked ack.

MSG_FIFO_MGSR_MEDIA - Media FIFO Messaging Register for Shadow Register Unit

	9	**Acknowledge that Media FIFO has been Blocked for VEBOX1** Access: R/W _Custom_GTIReset: BUS Acknowledge that MEDIA FIFO has been Blocked for VEBOX1 1'b0 : Media FIFO Block Ack for VEBOX1(default) 1'b1 : Media FIFO Unblock Ack VEBOX1 Note that the block request goes to MGSR, and MGSR responds to GPM with the blocked ack.
	8	**Acknowledge that Media FIFO has been Blocked for VEBOX0** Access: R/W _Custom_GTIReset: BUS Acknowledge that MEDIA FIFO has been Blocked for VEBOX0 1'b0 : Media FIFO Block Ack for VEBOX0(default) 1'b1 : Media FIFO Unblock Ack VEBOX0 Note that the block request goes to MGSR, and MGSR responds to GPM with the blocked ack.
	7	**Acknowledge that Media FIFO has been Blocked for VDBOX7** Access: R/W _Custom_GTIReset: BUS Acknowledge that MEDIA FIFO has been Blocked for VDBOX7 1'b0 : Media FIFO Block Ack for VDBOX7(default) 1'b1 : Media FIFO Unblock Ack VDBOX7 Note that the block request goes to MGSR, and MGSR responds to GPM with the blocked ack.
	6	**Acknowledge that Media FIFO has been Blocked for VDBOX6** Access: R/W _Custom_GTIReset: BUS Acknowledge that MEDIA FIFO has been Blocked for VDBOX6 1'b0 : Media FIFO Block Ack for VDBOX6(default) 1'b1 : Media FIFO Unblock Ack VDBOX6 Note that the block request goes to MGSR, and MGSR responds to GPM with the blocked ack.
	5	**Acknowledge that Media FIFO has been Blocked for VDBOX5** Access: R/W _Custom_GTIReset: BUS Acknowledge that MEDIA FIFO has been Blocked for VDBOX5 1'b0 : Media FIFO Block Ack for VDBOX5(default)

		1'b1 : Media FIFO Unblock Ack VDBOX5 Note that the block request goes to MGSR, and MGSR responds to GPM with the blocked ack.
	4	**Acknowledge that Media FIFO has been Blocked for VDBOX4** Access: R/W _Custom_GTIReset: BUS Acknowledge that MEDIA FIFO has been Blocked for VDBOX4 1'b0 : Media FIFO Block Ack for VDBOX4(default) 1'b1 : Media FIFO Unblock Ack VDBOX4 Note that the block request goes to MGSR, and MGSR responds to GPM with the blocked ack.
	3	**Acknowledge that Media FIFO has been Blocked for VDBOX3** Access: R/W _Custom_GTIReset: BUS Acknowledge that MEDIA FIFO has been Blocked for VDBOX3 1'b0 : Media FIFO Block Ack for VDBOX3(default) 1'b1 : Media FIFO Unblock Ack VDBOX3 Note that the block request goes to MGSR, and MGSR responds to GPM with the blocked ack.
	2	**Acknowledge that Media FIFO has been Blocked for VDBOX2** Access: R/W _Custom_GTIReset: BUS Acknowledge that MEDIA FIFO has been Blocked for VDBOX2 1'b0 : Media FIFO Block Ack for VDBOX2(default) 1'b1 : Media FIFO Unblock Ack VDBOX2 Note that the block request goes to MGSR, and MGSR responds to GPM with the blocked ack.
	1	**Acknowledge that Media FIFO has been Blocked for VDBOX1** Access: R/W _Custom_GTIReset: BUS Acknowledge that MEDIA FIFO has been Blocked for VDBOX1 1'b0 : Media FIFO Block Ack for VDBOX1(default) 1'b1 : Media FIFO Unblock Ack VDBOX1 Note that the block request goes to MGSR, and MGSR responds to GPM with the blocked ack.

MSG_FIFO_MGSR_MEDIA - Media FIFO Messaging Register for Shadow Register Unit

	0	**Acknowledge that Media FIFO has been Blocked for VDBOX0** Access: R/W _Custom_GTIReset: BUS Acknowledge that MEDIA FIFO has been Blocked for VDBOX0 1'b0 : Media FIFO Block Ack for VDBOX0(default) 1'b1 : Media FIFO Unblock Ack VDBOX0 Note that the block request goes to MGSR, and MGSR responds to GPM with the blocked ack.

Media unit Level Clock Gating override during rstflow 94B0

MEDMISCCP94B0 - Media unit Level Clock Gating override during rstflow 94B0

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	094B0h

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**misccp Clock Gating Disable during rstflow** Access: R/W _Custom_GTIReset: BUS misccp Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always) w/a for A-step would be to make BIOS to write value 1 to this bit for power-on. Default value = 1

MEMRR_BASE_LSB

MEMRR_BASE_LSB - MEMRR_BASE_LSB

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	108340h

The EMRR range is used to protect Xucode memory from unauthorized reads and writes. Any IO access to this range is aborted. This register controls the location of the EMRR range by indicating its starting address. It functions in tandem with the EMRR mask register.

DWord	Bit	Description
0	31:12	**RANGE_BASE** Default Value: 0000000h Access: RO _Custom_GTIReset: BUS This field corresponds to bits 38:12 of the base address memory range which is allocated to EMRR memory.
	11:4	**Reserved** Access: RO Format: MBZ
	3	**CONFIGURED** Default Value: 0h Access: RO _Custom_GTIReset: BUS This bitfield is required to enable the PRMRR range
	2:0	**Reserved** Access: RO Format: MBZ

MEMRR_BASE_MSB

MEMRR_BASE_MSB - MEMRR_BASE_MSB		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
SOC_Consumer:	BIOS	
Address:	108344h	
The EMRR range is used to protect Xucode memory from unauthorized reads and writes. Any IO access to this range is aborted. This register controls the location of the EMRR range by indicating its starting address. It functions in tandem with the EMRR mask register.		
DWord	**Bit**	**Description**
0	31:0	**RANGE_BASE** Default Value: 000h Access: RO _Custom_GTIReset: BUS This field corresponds to bits 63:12 of the base address memory range which is allocated to EMRR memory.

MEMRR_MASK_LSB

MEMRR_MASK_LSB - MEMRR_MASK_LSB

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	108380h

This register controls the size of the EMRR range by indicating which address bits must match the EMRR base register value.

DWord	Bit	Description
0	31:12	**RANGE_BASE** Default Value: 0000000h Access: RO _Custom_GTIReset: BUS This field indicates which address bits must match EMRR base in order to qualify as an EMRR access.
	11	**RANGE_EN** Default Value: 0b Access: RO _Custom_GTIReset: BUS Indicates whether the EMRR range is enabled and valid.
	10	**LOCK** Default Value: 0b Access: RO _Custom_GTIReset: BUS Setting this bit locks all writeable settings in this register, including itself.
	9:0	**Reserved** Access: RO Format: MBZ

MEMRR_MASK_MSB

MEMRR_MASK_MSB - MEMRR_MASK_MSB		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
SOC_Consumer:	BIOS	
Address:	108384h	
This register controls the size of the EMRR range by indicating which address bits must match the EMRR base register value.		
DWord	**Bit**	**Description**
0	31:0	**RANGE_MASK** Default Value: 000h Access: RO _Custom_GTIReset: BUS This field indicates which address bits must match EMRR base in order to qualify as an EMRR access.

Message Address

MA_0_2_0_PCI - Message Address

Register Space:	PCI: 0/2/0
Size (in bits):	32
Address:	000B0h

This register contains the Message Address for MSIs sent by the device.

DWord	Bit	Description
0	31:2	**Message Address Field** Default Value: 00000000000000000000000000000000b Access: R/W _Custom_GTIReset: BUS Used by system software to assign an MSI address to the device.The device handles an MSI by writing the padded contents of the MD register to this address.
	1:0	**Force Dword Align** Default Value: 00b Access: RO _Custom_GTIReset: BUS Hardwired to 0 so that addresses assigned by system software are always aligned on a DWORD address boundary.

Message Control

MC_0_2_0_PCI - Message Control

Register Space: PCI: 0/2/0
Size (in bits): 16

Address: 000AEh

Message Signaled Interrupt control register. System software can modify bits in this register, but the device is prohibited from doing so. If the device writes the same message multiple times, only one of those messages is guaranteed to be serviced. If all of them must be serviced, the device must not generate the same message again until the driver services the earlier one.

DWord	Bit	Description
0	15:9	**Reserved** Access: RO Format: MBZ
	8	**Per Vector Mask Capable** Default Value: 1b Access: RO _Custom_GTIReset: BUS SR-IOV requires this capability.
	7	**64 Bit Capable** Access: RO _Custom_GTIReset: BUS **Description**: Hardwired to 0 to indicate that the function does not implement the upper 32 bits of the Message address register and is incapable of generating a 64-bit memory address. **Value** 0b: **[Default]**
	6:4	**Multiple Message Enable** Default Value: 000b Access: R/W _Custom_GTIReset: BUS System software programs this field to indicate the actual number of messages allocated to this device. This number will be equal to or less than the number actually requested. Value: Number of requests000: 1001: 2010: 4011: 8100: 16101: 32110: Reserved111: Reserved
	3:1	**Multiple Message Capable** Default Value: 000b Access: RO

MC_0_2_0_PCI - Message Control

		_Custom_GTIReset: BUS System Software reads this field to determine the number of messages being requested by this device. Hardwired to 000b to indicate number of requests is 1.
	0	**MSI Enable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS Controls the ability of this device to generate MSIs.

Message Data

MD_0_2_0_PCI - Message Data		
Register Space:	PCI: 0/2/0	
Size (in bits):	16	
Address:	000B4h	
This register contains the Message Data for MSIs sent by the device.		
DWord	**Bit**	**Description**
0	15:0	**MESSDATA** Default Value: 0000000000000000b Access: R/W _Custom_GTIReset: BUS Base message data pattern assigned by system software and used to handle an MSI from the device. When the device must generate an interrupt request, it writes a 32-bit value to the memory address specified in the MA register. The upper 16 bits are always set to 0. The lower 16 bits are supplied by this register.

Message Signaled Interrupts Capability ID

MSI_CAPID_0_2_0_PCI - Message Signaled Interrupts Capability ID

Register Space:	PCI: 0/2/0
Size (in bits):	16
Address:	000ACh

When a device supports MSI it can generate an interrupt request to the processor by writing a predefined data item (a message) to a predefined memory address.

DWord	Bit	Description
0	15:8	**Pointer to Next Capability** Default Value: 11010000b Access: RO _Custom_GTIReset: BUS This is a hardwired pointer to the next item in the capabilities list.
	7:0	**Capability ID** Default Value: 00000101b Access: RO _Custom_GTIReset: BUS This field is hardwired to the value 05h to identify the CAP_ID as being for MSI registers.

MFC_AVC_CABAC_INSERTION_COUNT

AVC_CABAC_INSERTION_COUNT - MFC_AVC_CABAC_INSERTION_COUNT	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	128ACh

This register stores the count in bytes of **CABAC ZERO_WORD insertion. It is primarily provided for statistical data gathering.**

DWord	Bit	Description
0	31:0	**MFC AVC Cabac Insertion Count** Total number of bytes in the bitstream output before for the CABAC zero word insertion. This count is updated each time when the insertion count is incremented.

MFC_AVC Bitstream Decoding Front-End Parsing Logic Error Counter

MFC_VIN_AVD_ERROR_CNTR - MFC_AVC Bitstream Decoding Front-End Parsing Logic Error Counter

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	12804h

DWord	Bit	Description
0 avd_error_flagsR[31:0]	31:0	**Reserved** Access: RO Format: MBZ

MFC Image Status Control

MFC_IMAGE_STATUS_CONTROL - MFC Image Status Control		
Register Space:	MMIO: 0/2/0	
Access:	RO	
Size (in bits):	32	
Address:	128B8h	
This register stores the suggested data for next frame in multi-pass.		
DWord	**Bit**	**Description**
0	31:24	**Cumulative slice delta QP**
	23:16	**QP Value** suggested slice QP delta value for frame level Rate control. This value can be +ve or -ve
	15	**QP-Polarity Change** Cumulative slice delta QP polarity change.
	14:13	**Num-Pass Polarity Change** Number of passes after cumulative slice delta QP polarity changes.
	12	**VDENC Slice Overflow Error Occurred** Format: U1 True when slice size exceeds slice max size on final pass when VDENC is using attempting to use slice overflow prevention.
	11:8	**Total Num-Pass**
	7:4	**Reserved** Access: RO Format: MBZ
	3	**Missing Huffman Code** Jpeg HW encoder reports if Huffman table entry is missing.
	2	**Panic** Panic triggered to avoid too big packed file.
	1	**Frame Bit Count** Frame Bit count over-run/under-run flag
	0	**Max Conformance Flag** Max Macroblock conformance flag or Frame Bit count over-run/under-run

MFC Image Status Mask

MFC_IMAGE_STATUS_MASK - MFC Image Status Mask		
Register Space:	MMIO: 0/2/0	
Access:	RO	
Size (in bits):	32	
Address:	128B4h	
This register stores the image status(flags).		
DWord	**Bit**	**Description**
0	31:0	**Control Mask** Control Mask for dynamic frame repeat.

MFC QP Status Count

MFC_QUP_CT - MFC QP Status Count		
Register Space:	MMIO: 0/2/0	
Access:	RO	
Size (in bits):	32	
Address:	128BCh	
This register stores the suggested QP COUNTS in multi-pass.		
DWord	**Bit**	**Description**
0	31:24	**Cumulative QP Adjust** Format: U8 Cumulative QP adjustment after multiple passes. If there is no need to multi-pass, this value would be zero. (This is in sign magnitude form).
	23:0	**Cumulative QP** Format: U24 Cumulative QP for all MB of a Frame (Can be used for computing average QP).

MFD Error Status

MFD_ERROR_STATUS - MFD Error Status	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1C0800h
ShortName:	MFD_ERROR_STATUS_VCS0
Description:	For VDBox0
Address:	1C4800h
ShortName:	MFD_ERROR_STATUS_VCS1
Description:	For VDBox1
Address:	1D0800h
ShortName:	MFD_ERROR_STATUS_VCS2
Description:	For VDBox2
Address:	1D4800h
ShortName:	MFD_ERROR_STATUS_VCS3
Description:	For VDBox3
Address:	1E0800h
ShortName:	MFD_ERROR_STATUS_VCS4
Description:	For VDBox4
Address:	1E4800h
ShortName:	MFD_ERROR_STATUS_VCS5
Description:	For VDBox5
Address:	1F0800h
ShortName:	MFD_ERROR_STATUS_VCS6
Description:	For VDBox6
Address:	1F4800h
ShortName:	MFD_ERROR_STATUS_VCS7
Description:	For VDBox7

This register stores the error status flags and count reports by the bit-stream decoder. This register is not part of hardware context save and restore. Driver should read the content prior to starting a new batch/frame.

DWord	Bit	Description
0	31:20	**Reserved** Access: RO Format: MBZ

MFD_ERROR_STATUS - MFD Error Status		
	19:16	**AVC Short Format Error Flags** Exists If: // AVC Short Format == Ture Bit-stream error detected by VLD short format bit-stream decoder**.** These flags are reset at the beginning of a frame and updated until starting of another frame. [19] Slice Type SE Error Flag Invalid Slice Type SE [18] MMCO SE Error Flag Invalid memory management control operation SE. MMCO Loop does not end (mmco control != 0) even after all MMCO SEs are decoded OR MMCO SEs are still being decoded and MMCO SE loop end (mmco control == 0) is hit [17] Reordering IDC Error Flag Syntax Element modification_of_pic_nums_idc >= 6 OR modification_of_pic_nums_idc != 3 (end of reordering loop) but reordering count has already hit maximum value [16] Premature bitstream end is hit before finishing slice header decode
	15:0	**Bit-stream Error flags** Exists If: // AVC CAVLC, AVC CABAC, VC1 and MPEG2 == True Bitstream error detected by the VLD bit-steram decoder. These flags are reset at the beginning of a frame and updated until starting of another frame. AVC CAVLC: Please refer to AVC CAVLC table for each bit field AVC CABAC: Please refer to AVC CABAC table for each bit field VC1: Please refer to VC1 table for each bit field MPEG2: Please refer to MPEG2 table for each bit field

MFD Picture Parameter

MFD_PICTURE_PARAM - MFD Picture Parameter	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	1C0820h
ShortName:	MFD_PICTURE_PARAM_VCS0
Description:	For VDbox0
Address:	1C4820h
ShortName:	MFD_PICTURE_PARAM_VCS1
Description:	For VDbox1
Address:	1D0820h
ShortName:	MFD_PICTURE_PARAM_VCS2
Description:	For VDbox2
Address:	1D4820h
ShortName:	MFD_PICTURE_PARAM_VCS3
Description:	For VDbox3
Address:	1E0820h
ShortName:	MFD_PICTURE_PARAM_VCS4
Description:	For VDbox4
Address:	1E4820h
ShortName:	MFD_PICTURE_PARAM_VCS5
Description:	For VDbox5
Address:	1F0820h
ShortName:	MFD_PICTURE_PARAM_VCS6
Description:	For VDbox6
Address:	1F4820h
ShortName:	MFD_PICTURE_PARAM_VCS7
Description:	For VDbox7

DWord	Bit	Description
0	31:0	**Reserved** Access: RO Format: MBZ

MFX Frame BitStream SE/BIN Count

MFX_SE_BIN_CT - MFX Frame BitStream SE/BIN Count	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1C086Ch
ShortName:	MFX_SE_BIN_CT_VCS0
Description:	For VDBox0
Address:	1C486Ch
ShortName:	MFX_SE_BIN_CT_VCS1
Description:	For VDBox1
Address:	1D086Ch
ShortName:	MFX_SE_BIN_CT_VCS2
Description:	For VDBox2
Address:	1D486Ch
ShortName:	MFX_SE_BIN_CT_VCS3
Description:	For VDBox3
Address:	1E086Ch
ShortName:	MFX_SE_BIN_CT_VCS4
Description:	For VDBox4
Address:	1E486Ch
ShortName:	MFX_SE_BIN_CT_VCS5
Description:	For VDBox5
Address:	1F086Ch
ShortName:	MFX_SE_BIN_CT_VCS6
Description:	For VDBox6
Address:	1F486Ch
ShortName:	MFX_SE_BIN_CT_VCS7
Description:	For VDBox7

This register stores the number of BINs (AVC CABAC) and SEs (CAVLD, VLD) decoded in a frame. This register is not part of hardware context save and restore.

DWord	Bit	Description
0	31:0	**MFX Frame Bit-stream SE/BIN Count** Total number of BINs/SEs decoded in current frame. This number is used with frame performance count to derive Bin/clk or SE/clk.

MFX Frame Macroblock Count

MFX_MB_COUNT - MFX Frame Macroblock Count	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1C0868h
ShortName:	MFX_MB_COUNT_VCS0
Description:	For VDBox0
Address:	1C4868h
ShortName:	MFX_MB_COUNT_VCS1
Description:	For VDBox1
Address:	1D0868h
ShortName:	MFX_MB_COUNT_VCS2
Description:	For VDBox2
Address:	1D4868h
ShortName:	MFX_MB_COUNT_VCS3
Description:	For VDBox3
Address:	1E0868h
ShortName:	MFX_MB_COUNT_VCS4
Description:	For VDBox4
Address:	1E4868h
ShortName:	MFX_MB_COUNT_VCS5
Description:	For VDBox5
Address:	1F0868h
ShortName:	MFX_MB_COUNT_VCS6
Description:	For VDBox6
Address:	1F4868h
ShortName:	MFX_MB_COUNT_VCS7
Description:	For VDBox7

This register stores the number of Macro-blocks decoded/encoded in current frame. This register is not part of hardware context save and restore.

DWord	Bit	Description
0	31:20	**MBZ** Exists If: // JPEG == True Format: MBZ This field is currently reserved

MFX_MB_COUNT - MFX Frame Macroblock Count

	31:16	**Intra MB Count** Exists If: // AVC CAVLC, AVC CABAC, VC1 and MPEG2 == True Format: U16
	19:0	**JPEG Block Count** Exists If: // JPEG == True Format: U20 This 20-bit field indicates the number of 8x8 blocks within the JPEG frame. This field is clear at the start of decoding a new frame.
	15:0	**Number of MB Concealment** Exists If: // AVC CAVLC, AVC CABAC, VC1 and MPEG2 == True This 16-bit field indicates the number of MB is concealed by hardware. This field is clear at the start of decoding a new frame.

MFX Frame Row-Stored/BitStream Read Count

MFX_ROW_PER_BS_COUNT - MFX Frame Row-Stored/BitStream Read Count

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	12880h

This register stores the total number of row-stored/bit-stream read requests made by the pre-fetch engine per frame.This register is not part of hardware context save and restore.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**MFX row-stored/bit-stream read request Count** Total number of row-stored/bit-stream read requests sent by the memory pre-fetch engine per frame.

MFX PAK MPEG TS STATUS

MFX_PAK_MPEG_TS_STATUS - MFX PAK MPEG TS STATUS

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	12950h

This register stores MPEGTS packet status information

DWord	Bit	Description
0	31:28	**Next Continuity Center** Format: U4 HW will update the continuity counter the next MPEGTS packet stream for this stream ID needs to place in the bitstream.
	27:16	**Reserved** Access: RO Format: MBZ
	15:0	**MPEGTS Packet Count** Format: U16 This field counts the total number of written MPEGTS video packets by PAK HW. The PES header (which contains the PCR and PTS value) is included in this count as well.

MFX Pipeline Status Flags

MFX_STATUS_FLAGS - MFX Pipeline Status Flags	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1C0838h
ShortName:	MFX_STATUS_FLAGS_VCS0
Description:	For VDBox0
Address:	1C4838h
ShortName:	MFX_STATUS_FLAGS_VCS1
Description:	For VDBox1
Address:	1D0838h
ShortName:	MFX_STATUS_FLAGS_VCS2
Description:	For VDBox2
Address:	1D4838h
ShortName:	MFX_STATUS_FLAGS_VCS3
Description:	For VDBox3
Address:	1E0838h
ShortName:	MFX_STATUS_FLAGS_VCS4
Description:	For VDBox4
Address:	1E4838h
ShortName:	MFX_STATUS_FLAGS_VCS5
Description:	For VDBox5
Address:	1F0838h
ShortName:	MFX_STATUS_FLAGS_VCS6
Description:	For VDBox6
Address:	1F4838h
ShortName:	MFX_STATUS_FLAGS_VCS7
Description:	For VDBox7
This register stores the various pipeline status flags. This register is not part of hardware context save and restore.	

DWord	Bit	Description
0	31:17	**Reserved** Access: RO Format: MBZ

MFX_STATUS_FLAGS - MFX Pipeline Status Flags

	16	**MFX Active** Frame decoding/encoding is in progress.Set on frame_start; clear on frame_end.
	15:10	**Reserved** Access: RO Format: MBZ
	9	**Streamout Enable**
	8	**Reserved**
	7	**Post Deblocking Mode Enable**
	6	**Pre Deblocking Mode Enable**
	5	**Decoder Mode Select** 0 – Configure the MFD Engine for VLD Mode 1 – Configure the MFD Engine for IT Mode
	4	**Codec Select** 0 – Decode 1 – Encode
	3:2	**Video Mode** 00b – MPEG2 01b – VC1 10b – AVC 11b – JPEG
	1	**Decoder Short Format Mode** 0 – AVC/VC1 Short Format Mode is in use 1 – AVC/VC1 Long Format Mode is in use
	0	**Stitch Mode** 0b – Not in Stitch Mode 1b – In the Special Stitch Mode

MFX SFC LOCK Request

MFX_SFC_LOCK_REQUEST - MFX SFC LOCK Request

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	1C088Ch
ShortName:	MFX_SFC_LOCK_REQUEST_VCS0
Description:	For VDBox0
Address:	1D088Ch
ShortName:	MFX_SFC_LOCK_REQUEST_VCS2
Description:	For VDBox2
Address:	1E088Ch
ShortName:	MFX_SFC_LOCK_REQUEST_VCS4
Description:	For VDBox4
Address:	1F088Ch
ShortName:	MFX_SFC_LOCK_REQUEST_VCS6
Description:	For VDBox6

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**MFX_SFC_Forced_Lock** Format: U1 This bit can only be set by driver and it has to be clear by driver as well. Driver should set this bit before issuing the software (watchdog timer) reset. It tells MFX that a software reset is going to happen. MFX then issues a forced lock to SFC. If SFC is currently locked to MFX, SFC should not unlock itself from MFX. If SFC is NOT currently locked to MFX, SFC should not accept the lock request from MFX. Driver needs to clear this bit after the software reset sequence is complete.

MFX SFC LOCK Status

MFX_SFC_LOCK_STATUS - MFX SFC LOCK Status

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32

Address:	1C0890h
ShortName:	MFX_SFC_LOCK_STATUS_VCS0
Description:	For VDbox0

Address:	1D0890h
ShortName:	MFX_SFC_LOCK_STATUS_VCS2
Description:	For VDBox2

Address:	1E0890h
ShortName:	MFX_SFC_LOCK_STATUS_VCS4
Description:	For VDBox4

Address:	1F0890h
ShortName:	MFX_SFC_LOCK_STATUS_VCS6
Description:	For VDBox6

DWord	Bit	Description
0	31:2	**Reserved** Access: RO Format: MBZ
	1	**MFX_SFC_Forced_Act** Format: U1 This bit can only be set by hardware and it has to be clear by hardware as well. This bit is going to be polled by driver. This bit indicates that MFX has received MFX_SFC_Forced_Lock from driver and it has sent that signal to SFC. Once this bit is set, it indicates SFC status (lock or unlock) will not be changed anymore. Driver will be safe to start the reset process after this bit is set. Hardware has to de-assert this bit after driver de-assert MFX_SFC_Forced_Lock as well.
	0	**MFX_SFC_Usage** Format: U1 This bit can only be set by hardware and it has to be clear by hardware as well. This bit indicates SFC is currently locked to MFX. This bit should be set after SFC accepts the lock request from MFX. This bit should be clear once SFC finishes the workload and unlocked from MFX. In case a reset happens on MFX, this bit must be reset once a new workload is received

MGCMD

MGCMD - MGCMD

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	108300h

Mirror GT hardware uses for Vtd state.

DWord	Bit	Description
0	31	**TE** Access: RO _Custom_GTIReset: BUS 0: Disable Vtd DMA remapping 1: Enable Vtd DMA remapping
	30:26	**Reserved** Access: RO Format: MBZ
	25	**IRE** Access: RO _Custom_GTIReset: BUS No HW usage model. 0: Disable interrupt-remapping hardware 1: Enable interrupt-remapping hardware
	24:0	**Reserved** Access: RO Format: MBZ

MGSR GTI PD Control

GTIPD_CTRL - MGSR GTI PD Control

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	BUS
CrashLogSaved:	true
CrashLogVisibility:	cspec
Address:	00E04h-00E07h

DWord	Bit	Description
0	31:16	**Mask Bits** Access: RO Mask bits apply to [15:0] of same register. If mask is set to 1, corresponding bit in [15:0] is written. If mask is set to 0, corresponding bit in [15:0] is unaffected.
	15:6	**Reserved** Access: RO Format: MBZ
	5:4	**Reserved** Access: RO Format: MBZ
	3	**Reserved** Access: RO Format: MBZ
	2	**Render Unblock** Default Value: 0b Block Access: R/WC Render unblock (1) or block (0)
	1	**GT Block Mode** Default Value: 1b Standby mode Access: R/WC Set GT C6 Standby mode (1) or CPD (0)
	0	**GT Unblock** Default Value: 0b Block Access: R/WC GT unblock (1) or block (0)

MGSR Media PD Control

MEDIAPD_CTRL - MGSR Media PD Control

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	BUS
CrashLogSaved:	true
CrashLogVisibility:	cspec
Address:	00E00h-00E03h

DWord	Bit	Description
0	31:16	**Mask Bits** Access: RO Mask bits applied to [15:0] of same register. If mask is set to 1, corresponding bit in [15:0] is written. If mask is set to 0, corresponding bit in [15:0] is unaffected.
	15:12	**Reserved** Access: RO Format: MBZ
	11:8	**VEBOX Unblock** Default Value: 0b Access: R/WC VEBOX unblock indications for VEBOX[3:0] (1) or block (0)
	7:0	**VDBOX Unblock** Default Value: 0b Access: R/WC VDBOX unblock indications for VDBOX[7:0] (1) or block (0) VDBOX[1:0] reside in Media slice0, VDBOX[3:2] in slice1, VDBOX[5:4] in slice2, VDBOX[7:6] in slice3

MGSR Programmable Shadow 0

PROGSHADOW_0 - MGSR Programmable Shadow 0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00EE0h-00EE3h	
DWord	**Bit**	**Description**
0	31:26	**Reserved** Access: RO Format: MBZ
	25:0	**Shadow Address** Default Value: 00000h Access: R/W Programmable shadow register address.

MGSR Programmable Shadow 1

PROGSHADOW_1 - MGSR Programmable Shadow 1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00EE4h-00EE7h	
DWord	**Bit**	**Description**
0	31:26	**Reserved** Access: RO Format: MBZ
	25:0	**Shadow Address** Default Value: 00000h Access: R/W Programmable shadow register address.

Minimum Grant

MINGNT_0_2_0_PCI - Minimum Grant		
Register Space: PCI: 0/2/0 Size (in bits): 8		
Address: 0003Eh		
The Integrated Graphics Device has no requirement for the settings of Latency Timers.		
DWord	**Bit**	**Description**
0	7:0	**Minimum Grant Value** Default Value: 00000000b Access: RO _Custom_GTIReset: BUS

Minute IA Force Fence

MIA_FORCE_FENCE - Minute IA Force Fence

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

Programming Notes

Write by MinIAto issue a fence. Written value is not stored.

Write to this register by the core will trigger GUC Hardware to do following sequencing
Stall the core interface.
Check for all prior writes generated by the core are completed and to global observable point.
Un-Stall the core interface.

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**Reserved** Access: RO Format: MBZ

Minute IA Force TLB Invalidate

MIA_INV_TLB - Minute IA Force TLB Invalidate

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

Programming Notes

Write to issue a TLB invalidate or Fence write. Written value is not stored.

Write to this register by the core will trigger GUC Hardware to do following sequencing
Stall the core interface.
Check for all prior writes generated by the core are completed and to global observable point.
TLB invalidation generation and completion to memory interface.
Un-Stall the core interface.

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**Reserved** Access: RO Format: MBZ

Mirror FUSE A

MIRROR_FUSE_A - Mirror FUSE A		
Register Space:	MMIO: 0/2/0	
Access:	RO	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Mirror Register for Fuses in INF domain.		
DWord	**Bit**	**Description**
0	31:1	**Reserved** Reserved
	0	**Reserved** Access: RO Format: MBZ

Mirror GT C6 entry Count

MIRROR_GT_C6_ENTRY_COUNT - Mirror GT C6 entry Count	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00DACh
Name:	Mirror GT C6 entry Count
ShortName:	MIRROR_GT_C6_ENTRY_COUNT

This register holds the mirrored value, from OA, of the number of GT C6 entries that have happened

DWord	Bit	Description
0	31:0	**Count for number of C6 entries** Access: R/W _Custom_GTIReset: BUS Mirror the number of times that GT has entered C6

Mirror of Base Data of Stolen Memory

BDSM_0_2_0_PCI - Mirror of Base Data of Stolen Memory

Register Space:	PCI: 0/2/0
Size (in bits):	64
Address:	000C0h

Mirror of BSDM from GTTMMADR space. This register contains the base address of graphics data stolen DRAM memory.

DWord	Bit	Description
0..1	63:32	**Graphics Base of Stolen Memory MSB** Default Value: 000000000000b Access: RO Variant _Custom_GTIReset: BUS This register contains bits 63 to 20 of the base address of stolen DRAM memory. BIOS is now able to allocate GDSM above 4GB.
	31:20	**Graphics Base of Stolen Memory LSB** Default Value: 000000000000b Access: RO Variant _Custom_GTIReset: BUS This register contains bits 63 to 20 of the base address of stolen DRAM memory. BIOS is now able to allocate GDSM above 4GB.
	19:1	**Reserved** Access: RO Format: MBZ
	0	**Spare** Default Value: 0b Access: RO Variant _Custom_GTIReset: BUS This was a lock bit prior.

Mirror of Device Enable

DEVEN0_0_2_0_PCI - Mirror of Device Enable

Register Space:	PCI: 0/2/0
Size (in bits):	16
Address:	00054h

Mirror of DEVEN_0_0_0_PCI.

DWord	Bit	Description
0	15	**Reserved** Access: RO Format: MBZ
	14	**Chap Enable** Default Value: 0b Access: RO Variant _Custom_GTIReset: BUS
	13	**Device 6 Enable** Default Value: 0b Access: RO _Custom_GTIReset: BUS
	12:11	**Reserved** Access: RO Format: MBZ
	10	**Device 5 Enable** Default Value: 0b Access: RO _Custom_GTIReset: BUS
	9:8	**Reserved** Access: RO Format: MBZ
	7	**Device 4 Enable** Default Value: 1b Access: RO Variant _Custom_GTIReset: BUS
	6	**Reserved** Access: RO Format: MBZ

DEVEN0_0_2_0_PCI - Mirror of Device Enable

5	**Device 3 enable for Display HD Audio**
	Default Value: 1b Access: RO Variant _Custom_GTIReset: BUS
4	**Internal Graphics Engine**
	Default Value: 1b Access: RO Variant _Custom_GTIReset: BUS 0: Bus 0 Device 2 is disabled and hidden1: Bus 0 Device 2 is enabled and visibleThis bit will be set to 0b and remain 0b if Device 2 capability is disabled.
3	**PEG10 Enable**
	Default Value: 1b Access: RO Variant _Custom_GTIReset: BUS
2	**PEG11 Enable**
	Default Value: 1b Access: RO Variant _Custom_GTIReset: BUS
1	**PEG12 Enable**
	Default Value: 1b Access: RO Variant _Custom_GTIReset: BUS
0	**Host Bridge**
	Default Value: 1b Access: RO _Custom_GTIReset: BUS

Mirror of DPRB LSB

DPRB_LSB - Mirror of DPRB LSB		
Register Space: MMIO: 0/2/0 Size (in bits): 32		
Address: 090E8h		
DWord	**Bit**	**Description**
0	31:20	**DPRBASE** Access: RO _Custom_GTIReset: BUS 1MB aligned base of DMA Range
	19:0	**Spares** Access: RO _Custom_GTIReset: BUS

Mirror of DPRB MSB

DPRB_MSB - Mirror of DPRB MSB		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	090ECh	
DWord	**Bit**	**Description**
0	31:0	**DPRBASE** Access: RO _Custom_GTIReset: BUS 1MB aligned base of DMA Range. Bits 63:32 of the DPRbase.

Mirror of DSMBASE LSB

DSMB_LSB - Mirror of DSMBASE LSB		
Register Space: MMIO: 0/2/0 Size (in bits): 32		
Address: 090D0h		
DSM Base		
DWord	**Bit**	**Description**
0	31:20	**DSM Base Register LSB** Access: RO _Custom_GTIReset: BUS This register contains the base address of stolen DRAM memory.
	19:0	**Spares** Access: RO _Custom_GTIReset: BUS

Mirror of DSMBASE MSB

DSMB_MSB - Mirror of DSMBASE MSB		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	090D4h	
DSM Base		
DWord	**Bit**	**Description**
0	31:0	**DSM Base Register** Access: RO _Custom_GTIReset: BUS This register contains the base address of stolen DRAM memory.

Mirror of EMRR Base LSB

EMRRBASE_LSB - Mirror of EMRR Base LSB		
Register Space: MMIO: 0/2/0 Size (in bits): 32		
Address: 09200h		
Mirror of EMRR Base		
DWord	**Bit**	**Description**
0	31:12	**EMRR Base LSB** Access: RO _Custom_GTIReset: BUS EMRR Base Value.
	11:0	**Spares** Access: RO _Custom_GTIReset: BUS

Mirror of EMRR Base MSB

EMRRBASE_MSB - Mirror of EMRR Base MSB		
Register Space: MMIO: 0/2/0 Size (in bits): 32		
Address: 09204h		
Mirror of EMRR Base		
DWord	**Bit**	**Description**
0	31:0	**EMRR Base MSB** Access: RO _Custom_GTIReset: BUS EMRR Base Value.

Mirror of EU Disable Fuses - Register0

MIRROR_EU_DISABLE0 - Mirror of EU Disable Fuses - Register0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	09134h	
DWord	**Bit**	**Description**
0	31:8	**Reserved** Access: RO Format: MBZ
	7:0	**EU Disable Fuses** Access: RO Variant _Custom_GTIReset: BUS One Bit per EU In a sub-slice. Enable/Disable the same EU# in all the Sub-Slices. **Programming Notes** Bit values of 0 indicate enabled; Bit values of 1 indicate disabled.

Mirror of FUSE1 Control DW

FUSE1 - Mirror of FUSE1 Control DW		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	0911Ch	
DWord	**Bit**	**Description**
0	31:27	**Spares** Access: RO _Custom_GTIReset: BUS
	26	**VA DISABLE** Access: RO _Custom_GTIReset: BUS
	25:24	**Reserved** Access: RO Format: MBZ
	23	**Reserved**
	22	**Reserved**
	21	**Reserved**
	20	**HUC AUTH BYPASS** Access: RO _Custom_GTIReset: BUS This fuse works for both Micro Controllers present in Media pipeline.
	19	**HUC DISABLE** Access: RO _Custom_GTIReset: BUS This fuse works for both Micro Controllers present in Media pipeline.
	18	**Reserved**
	17:16	**Spares1** Access: RO _Custom_GTIReset: BUS
	15	**Authentication Bypass** Access: RO _Custom_GTIReset: BUS
	14	**Reserved**

FUSE1 - Mirror of FUSE1 Control DW

13	**Spares2**	
	Access:	RO
	_Custom_GTIReset:	BUS
12	**Reserved**	
11	**Render Disable**	
	Access:	RO
	_Custom_GTIReset:	BUS
10:9	**Spares3**	
	Access:	RO
	_Custom_GTIReset:	BUS
8	**VME IME Enable**	
	Access:	RO
	_Custom_GTIReset:	BUS
7	**VME CRE Enable**	
	Access:	RO
	_Custom_GTIReset:	BUS
6:5	**Media Decode**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Applicable to Media - Fuse to disable VIN from processing media_objs or turn off the entire crclk tree trunk.	
4	**Disable GT3 Slice Shutdown**	
	Access:	RO
	_Custom_GTIReset:	BUS
	N/A -- Not used by GT hardware: This fuse is actually enforced by the PCU;it is reflected here for driver information only.	
3	**Reserved**	
2	**Spares4**	
	Access:	RO
	_Custom_GTIReset:	BUS
1:0	**Media Encode**	
	Access:	RO
	_Custom_GTIReset:	BUS
	Applicable to Media - One fuse to disable VIN from processing Pak_obj. Second fuse to disable VME.	

Mirror of Fuse 3 control DW

FUSE3 - Mirror of Fuse 3 control DW		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	09118h	
FUSE MIRROR 3		

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28	**GT L3 HASH MODE FUSE** Access: RO _Custom_GTIReset: BUS L3 Bank Hash Mode (used in certain dynamic slice configurations) 0b = Use safe mode, hashing to fewer banks in specific cases 1b = Use all banks when possible
	27:26	**Reserved** Access: RO Format: MBZ
	25:24	**GT WGBox Configuration Fuse** Access: RO _Custom_GTIReset: BUS WGBox configuration 00b = Both WGBOXes enabled 01b = WGBOX1 enabled 10b = WGBOX0 enabled
	23:16	**Reserved** Access: RO Format: MBZ
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**GT L3 MODE FUSE** Access: RO _Custom_GTIReset: BUS L3 Bank Disable Select (only used in certain dynamic slice configurations and sub-single slice

FUSE3 - Mirror of Fuse 3 control DW		
		SKUs) 1 - Disabled; 0 - Enabled Bit0 - Bank0 Bit1 - Bank1 Likewise, Bit7 - Bank7

Mirror of Global Command Register

GCMD - Mirror of Global Command Register		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	090CCh	
DWord	**Bit**	**Description**
0	31	**Translation Enable** Access: RO _Custom_GTIReset: BUS Software writes to this field to request hardware to enable/disable DMA-remapping hardware. 0: Disable DMA-remapping hardware. 1: Enable DMA-remapping hardware. Hardware reports the status of the translation enable operation through the TES field in the Global Status register. Before enabling (or re-enabling) DMA-remapping hardware through this field, software must: • Setup the DMA-remapping structures in memory. • Flush the write buffers (through WBF field), if write buffer flushing is reported as required. • Set the root-entry table pointer in hardware (through SRTP field). • Perform global invalidation of the context-cache and global invalidation of IOTLB • If advanced fault logging supported, setup fault log pointer (through SFL field) and enable advanced fault logging (through EAFL field). Refer to Section 9 for detailed software requirements. There may be active DMA requests in the platform when software updates this field. Hardware must enable or disable remapping logic only at deterministic transaction boundaries, so that any in-flight transaction is either subject to remapping or not at all. Hardware implementations supporting DMA draining must drain any in-flight translated DMA read/write requests queued within the root complex before completing the translation enable command and reflecting the status of the command through the TES field in the GSTS_REG. Value returned on read of this field is undefined.
	30	**Set Root Table Pointer** Access: RO _Custom_GTIReset: BUS Software sets this field to set/update the root-entry table pointer used by hardware. The root-entry table pointer is specified through the Root-entry Table Address register. Hardware reports the status of the "root table pointer set" operation through the RTPS field in the Global Status register. The root table pointer set operation must be performed before enabling or re-enabling(after disabling) DMA remapping through the TE field. After a "root table pointer set" operation, software must globally invalidate the context cache and then globally invalidate the IOTLB. This is required to ensure hardware uses only there mapping structures referenced by the new root table pointer, and not any stale cached entries. While DMA remapping is active, software may update the root table pointer through this field.

GCMD - Mirror of Global Command Register

		However, to ensure valid in-flight DMA requests are deterministically remapped, software must ensure that the structures referenced by the new root table pointer are programmed to provide the same remapping results as the structures referenced by the previous root-table pointer. Clearing this bit has no effect. The value returned on a read of this field is undefined.
	29	**Set Fault Log** Access: RO _Custom_GTIReset: BUS This field is valid only for implementations supporting advanced fault logging. Software sets this field to request hardware to set/update the fault-log pointer used by hardware. The fault-log pointer is specified through Advanced Fault Log register. Hardware reports the status of the fault log set operation through the FLS field in the Global Status register. The fault log pointer must be set before enabling advanced fault logging (through EAFL field).Once advanced fault logging is enabled, the fault log pointer may be updated through this field while DMA remapping is active. Clearing this bit has no effect. The value returned on read of this field is undefined.
	28	**Enable Fault Logging** Access: RO _Custom_GTIReset: BUS This field is valid only for implementations supporting advanced fault logging. Software writes to this field to request hardware to enable or disable advanced fault logging. 0: Disable advanced fault logging. In this case, translation faults are reported through the Fault Recording registers. 1: Enable use of memory-resident fault log. When enabled, translation faults are recorded in the memory-resident log. The fault log pointer must be set in hardware (through SFL field) before enabling advanced fault logging. Hardware reports the status of the advanced fault logging enable operation through the AFLS field in the Global Status register. Value returned on read of this field is undefined.
	27	**Write Buffer Flush** Access: RO _Custom_GTIReset: BUS This bit is valid only for implementations requiring write buffer flushing. Software sets this field to request hardware to flush the root-complex internal write buffers. This is done to ensure any updates to the memory-resident remapping structures are not held in any internal write posting buffers. Refer to Section 11.1 for details on write-buffer flushing requirements. Hardware reports the status of the write buffer flushing operation through the WBFS field in the Global Status register. Clearing this bit has no effect. Value returned on read of this field is undefined.

GCMD - Mirror of Global Command Register

26 Queued Invalidation Enable

Access:	RO
_Custom_GTIReset:	BUS

This field is valid only for implementations supporting queued invalidations.
Software writes to this field to enable or disable queued invalidations.
0: Disable queued invalidations.
1: Enable use of queued invalidations.
Hardware reports the status of queued invalidation enable operation through QIES field in the Global Status register.
Refer to Section 6.2.2 for software requirements for enabling/disabling queued invalidations.
The value returned on a read of this field is undefined.

25 Interrupt Remapping Enable

Access:	RO
_Custom_GTIReset:	BUS

This field is valid only for implementations supporting interrupt remapping.
0: Disable interrupt-remapping hardware
1: Enable interrupt-remapping hardware
Hardware reports the status of the interrupt remapping enable operation through the IRES field in the Global Status register.
There may be active interrupt requests in the platform when software updates this field. Hardware must enable or disable interrupt-remapping logic only at deterministic transaction boundaries, so that any in-flight interrupts are either subject to remapping or not at all.
Hardware implementations must drain any in-flight interrupts requests queued in the Root-Complex before completing the interrupt-remapping enable command and reflecting the status of the command through the IRES field in the Global Status register.
The value returned on a read of this field is undefined.

24 Set Interrupt Remap Table Pointer

Access:	RO
_Custom_GTIReset:	BUS

This field is valid only for implementations supporting interrupt-remapping.
Software sets this field to set/update the interrupt remapping table pointer used by hardware. The interrupt remapping table pointer is specified through the Interrupt Remapping Table Address register.
Hardware reports the status of the interrupt remapping table pointer set operation through the IRTPS field in the Global Status register.
The interrupt remap table pointer set operation must be performed before enabling or re-enabling(after disabling) interrupt-remapping hardware through the IRE field.
After an interrupt remap table pointer set operation, software must globally invalidate the interrupt entry cache. This is required to ensure hardware uses only the interrupt-remapping entries referenced by the new interrupt remap table pointer, and not any stale cached entries.
While interrupt remapping is active, software may update the interrupt remapping table pointer through this field. However, to ensure valid in-flight interrupt requests are deterministically remapped, software must ensure that the structures referenced by the new interrupt remap table pointer are programmed to provide the same remapping results as the structures referenced by

GCMD - Mirror of Global Command Register

		the previous interrupt remap table pointer. Clearing this bit has no effect. The value returned on a read of this field is undefined.
	23	**Compatibility Format Interrupt** Access: RO _Custom_GTIReset: BUS This field is valid only for Intel(R)64 implementations supporting interrupt-remapping. Software writes to this field to enable or disable Compatibility Format interrupts on Intel(R)64 platforms. The value in this field is effective only when interrupt-remapping is enabled and Legacy Interrupt Mode is active. 0: Block Compatibility format interrupts. 1: Process Compatibility format interrupts as pass-through (bypass interrupt remapping). Hardware reports the status of updating this field through the CFIS field in the Global Status register. Refer to Section 5.4.1 for details on Compatibility Format interrupt requests. The value returned on a read of this field is undefined. This field is not implemented on Itanium(TM) implementations.
	22:0	**Spares** Access: RO _Custom_GTIReset: BUS

Mirror of GMCH Graphics Control Register

MGGC - Mirror of GMCH Graphics Control Register

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	09094h

Mirror of GMCH Graphics Control Register.

DWord	Bit	Description
0	31:16	**Spares** Access: RO _Custom_GTIReset: BUS
	15:8	**Graphics Mode Select** Default Value: 3h Access: RO _Custom_GTIReset: BUS This field selects the amount of Main Memory that is pre-allocated to support the Internal Graphics device in VGA (non-linear) and Native (linear) modes. The BIO Sensures that memory is pre-allocated only when Internal graphics is enabled. 0h: No memory pre-allocated. Device 2 (IGD) does not claim VGA cycles (Mem and IO),and the Sub-Class Code field within Device 2 function 0 Class Code register is 80. 1h-4h: Reserved. 5h-Dh: DVMT (UMA) mode, memory pre-allocated for frame buffer, in quantities as shown in the Encoding table. Eh-Fh: Reserved. NOTE: This register is locked and becomes Read Only when the D_LCK bit in the SMRAMC register is set. This register is also LT lockable. Hardware does not clear or set any of these bits automatically based on IGD being disabled/enabled. BIOS Requirement: BIOS must not set this field to 0h if IVD (bit 1 of this register) is 0.
	7:6	**GTT Graphics Memory Size** Access: RO _Custom_GTIReset: BUS This field selects the amount of Main Memory that is pre-allocated to support the Internal Graphics Translation Table. The BIOS ensures that memory is pre-allocated only when Internal graphics is enabled. GSM is assumed to be a contiguous physical DRAM space with DSM, and BIOS needs to allocate a contiguous memory chunk. Hardware derives the base of GSM from DSM only using the GSM size programmed in the register. 0h: No memory pre-allocated. GTT cycles (Mem and IO) are not claimed. 1h: 2 MB of memory pre-allocated for GTT. 2h: 4 MB of memory pre-allocated for GTT.

		3h: 8 MB of memory pre-allocated for GTT. Hardware functionality in case of programming this value to Reserved is not guaranteed. This register is locked and becomes Read Only when the D_LCK bit in the SMRAMC register is set.
	5:3	**Spares2** Access: RO _Custom_GTIReset: BUS
	2	**Versatile Acceleration Mode Enable** Access: RO _Custom_GTIReset: BUS Enables the use of the iGFX engines for Versatile Acceleration. 1 - iGFX engines are in Versatile Acceleration Mode. Device 2 Class Code is 048000h. 0 - iGFX engines are in iGFX Mode. Device 2 Class Code is 030000h.
	1	**IGD VGA Disable** Access: RO _Custom_GTIReset: BUS 0: Enable. Device 2 (IGD) claims VGA memory and IO cycles, the Sub-Class Code within Device 2 Class Code register is 00. 1: Disable. Device 2 (IGD) does not claim VGA cycles (Mem and IO), and the Sub-Class Code field within Device 2 function 0 Class Code register is 80. BIOS Requirement: BIOS must not set this bit to 0 if the GMS field (bits 6:4 of this register)pre-allocates no memory. This bit MUST be set to 1 if Device 2 is disabled either via a fuse or fuse override(CAPID0[46] = 1) or via a register (DEVEN[3] = 0). This register is locked by LT lock.
	0	**Spares3** Access: RO _Custom_GTIReset: BUS

Mirror of Graphics Translation Table and Memory Mapped Range Address (31:0)

GTTMMADR_LSB - Mirror of Graphics Translation Table and Memory Mapped Range Address (31:0)

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	09124h

This register requests allocation for the combined Graphics Translation Table Modification Range and Memory Mapped Range. The range requires 4 MB combined for MMIO and Global GTT aperture, with 2MBof that used by MMIO and 2MB used by GTT. GTTADR begins at (GTTMMADR + 2 MB) while the MMIO base address is the same as GTTMMADR.
For the Global GTT, this range is defined as a memory BAR in graphics device config space. It is an alias into which software is required to write Page Table Entry values (PTEs). Software may read PTE values from the global Graphics Translation Table (GTT). PTEs cannot be written directly into the global GTT memory area.
The device snoops writes to this region in order to invalidate any cached translations within the various TLBs implemented on-chip. There are some exceptions to this - see GTT-TLB in the Programming Interface chapter.
The allocation is for 4MB and the base address is defined by bits [38:22].

DWord	Bit	Description
0	31:22	**Memory Base Address (LSB - 31:22 of 38:22)** Access: RO _Custom_GTIReset: BUS Set by the OS, these bits correspond to address signals [38:22]. 4MB combined for MMIO and Global GTT table aperture (2MB for MMIO and 2 MB for GTT).
	21:4	**Spares** Access: RO _Custom_GTIReset: BUS
	3	**Prefetchable Memory** Access: RO _Custom_GTIReset: BUS Hardwired to 0 to prevent prefetching.
	2:1	**Memory Type** Access: RO _Custom_GTIReset: BUS 00b: To indicate 32 bit base address. 01b: Reserved. 10b: To indicate 64 bit base address. 11b: Reserved.

GTTMMADR_LSB - Mirror of Graphics Translation Table and Memory Mapped Range Address (31:0)		
	0	**Memory I/O Space** Access: RO _Custom_GTIReset: BUS Hardwired to 0 to indicate memory space.

Mirror of Graphics Translation Table and Memory Mapped Range Address UDW

GTTMMADR_MSB - Mirror of Graphics Translation Table and Memory Mapped Range Address UDW

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	09128h

This register requests allocation for the combined Graphics Translation Table Modification Range and Memory Mapped Range. The range requires 4 MB combined for MMIO and Global GTT aperture, with2MB of that used by MMIO and 2MB used by GTT. GTTADR begins at (GTTMMADR + 2 MB) while the MMIO base address is the same as GTTMMADR.
For the Global GTT, this range is defined as a memory BAR in graphics device config space. It is an alias into which software is required to write Page Table Entry values (PTEs). Software may read PTE values from the global Graphics Translation Table (GTT). PTEs cannot be written directly into the global GTT memory area.
The device snoops writes to this region in order to invalidate any cached translations within the various TLBs implemented on-chip. There are some exceptions to this - see GTT-TLB in the Programming Interface chapter.
The allocation is for 4MB and the base address is defined by bits [38:22].

DWord	Bit	Description
0	31:7	**Spares** Access: RO _Custom_GTIReset: BUS
	6:0	**Memory Base Address (MSB - 38:32 of 38:22)** Access: RO _Custom_GTIReset: BUS Set by the OS, these bits correspond to address signals [38:22]. 4MB combined for MMIO and Global GTT table aperture (2MB for MMIO and 2 MB for GTT).

Mirror of GSMBASE LSB

GSMB_LSB - Mirror of GSMBASE LSB		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	090D8h	
This register contains the base address of stolen DRAM memory for the GTT.		
DWord	**Bit**	**Description**
0	31:20	**GSM Base** Access: RO _Custom_GTIReset: BUS This register contains the base address of stolen DRAM memory for the GTT.
	19:0	**Spares** Access: RO _Custom_GTIReset: BUS

Mirror of GSMBASE MSB

GSMB - Mirror of GSMBASE MSB		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	090DCh	
This register contains the base address of stolen DRAM memory for the GTT.		
DWord	**Bit**	**Description**
0	31:0	**GSM Base** Access: RO _Custom_GTIReset: BUS This register contains the base address of stolen DRAM memory for the GTT.

Mirror of GT DSS Enable Fuses

MIRROR_GT_DSS_ENABLE - Mirror of GT DSS Enable Fuses

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	0913Ch

DWord	Bit	Description
0	31:0	**GT DSS Enable Fuses**

Default Value:	0000003fh Default Value
Access:	RO
_Custom_GTIReset:	BUS

Description
DSS Enable Fuses Encoding:- 32'h0000 = All Sub-Slices Disabled Note: Encoding starts with bit 0 mapped to DSS0 of slice0 and goes on up to bit 31 as per the Project's Number of DSS per Slice. See below for elaborated mapping note for example where there are 6 DSS per Slice- bit0 - DSS 0 of Slice 0 bit1 - DSS 1 of Slice 0 Likewise - bit5- DSS 5of Slice 0 bit 6- DSS0 of Slice 1

Programming Notes
Bit values of 1 indicate DSS Enabled. Bit values of 0 indicate DSS Disabled.

Mirror of GT Slice Enable Fuses

MIRROR_GT_SLICE_EN - Mirror of GT Slice Enable Fuses		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	09138h	
DWord	**Bit**	**Description**
0	31:8	**Reserved** Access: RO Format: MBZ
	7:0	**GT slice Enable Fuses** Access: RO _Custom_GTIReset: BUS Slice Enable Fuses Bit0 - Slice 0 Enabled Bit1 - Slice;1 Enabled Likewise; Bit7 - Slice;7 Enabled **Programming Notes** Bit values of;1 indicate enabled. Bit values of;0 indicate disabled.

Mirror of GT VEBOX and VDBOX Disable

MIRROR_GT_VEBOX_VDBOX_DISABLE - Mirror of GT VEBOX and VDBOX Disable

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	09140h

Mirror of GT VEBOX and VDBOX Disable

DWord	Bit	Description
0	31:20	**Reserved** Access: RO Format: MBZ
	19:16	**GT VEBOX DISABLE** _Custom_GTIReset: BUS VEBOX config fuses encoding:- bit 0 = VEBOX 0 disabled bit 1 = VEBOX 1 disabled bit 2 = VEBOX 2 disabled bit 3 = VEBOX 3 disabled **Programming Notes** Bit values of 0 indicate enabled. Bit values of 1 indicate disabled.
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**GT VDBOX DISABLE** Access: RO _Custom_GTIReset: BUS VDBOX config fuses encoding:- bit 0 = VDBOX 0 disabled bit 1 = VDBOX 1 disabled Likewise, bit 7 = VDBOX 7 disabled **Programming Notes** Bit values of 0 indicate enabled. Bit values of 1 indicate disabled.

Mirror of MBC IDIMSG Reg

GAM_MBC_MIRROR - Mirror of MBC IDIMSG Reg

Register Space: MMIO: 0/2/0

Size (in bits): 32

MBC IDI Message Register Mirror in GAM

DWord	Bit	Description
0	31:16	**Mask Bits** Access: RO _Custom_GTIReset: BUS Reserved.
	15:9	**Reserved** Access: RO Format: MBZ
	8	**Reserved**
	7:0	**Reserved** Access: RO Format: MBZ

Mirror of PCICMD MAE/BME

PCICMD - Mirror of PCICMD MAE/BME		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	0912Ch	
DWord	**Bit**	**Description**
0	31:11	**Spare** Access: R/W _Custom_GTIReset: BUS
	10	**Interrupt Disable** Access: R/W _Custom_GTIReset: BUS This bit disables the device from asserting INTx#. 0: Enable the assertion of this device's INTx# signal. 1: Disable the assertion of this device's INTx# signal. DO_INTx messages are not sent to DMI. GSA Implementation: When 1, blocks the sending of an MSI interrupt and blocks the sending of a Line interrupt.(The interrupt status is not blocked from being reflected in the INTSTS bit.) When 0, permits the sending of an MSI interrupt or Line interrupt.
	9	**Fast Back to Back** Access: R/W _Custom_GTIReset: BUS Not Implemented. Hardwired to 0.
	8	**SERR Enable** Access: R/W _Custom_GTIReset: BUS Not Implemented. Hardwired to 0.
	7	**Address/Data Stepping Enable** Access: R/W _Custom_GTIReset: BUS Not Implemented. Hardwired to 0.
	6	**Parity Error Enable** Access: R/W _Custom_GTIReset: BUS Not Implemented. Hardwired to 0. Since the IGD belongs to the category of devices that does not corrupt programs or data in system memory or hard drives, the IGD ignores any parity error that it detects and continues with normal operation.

PCICMD - Mirror of PCICMD MAE/BME

	5	**Video Pallete Snooping** Access: R/W _Custom_GTIReset: BUS This bit is hardwired to 0 to disable snooping.
	4	**Memory Write and Invalidate Enable** Access: R/W _Custom_GTIReset: BUS Hardwired to 0. The IGD does not support memory write and invalidate commands.
	3	**Special Cycle Enable** Access: R/W _Custom_GTIReset: BUS This bit is hardwired to 0. The IGD ignores Special cycles.
	2	**Bus Primary Enable** Access: R/W _Custom_GTIReset: BUS 0: Disable IGD bus primaring. 1: Enable the IGD to function as a PCI compliant primary. GSA Implementation: When 0, blocks the sending of MSI interrupts. When 1, permits the sending of above. (Note: See descriptions of the INTDIS, MSE, and INTSTS bits.)
	1	**Memory Access Enable** Access: R/W _Custom_GTIReset: BUS This bit controls the IGD's response to memory space accesses. 0: Disable. 1: Enable.
	0	**I/O Access Enable** Access: R/W _Custom_GTIReset: BUS This bit controls the IGD's response to I/O space accesses. 0: Disable. 1: Enable.

Mirror of PMR HIGH LIMIT (31-0)

PMRHLIMIT_LSB - Mirror of PMR HIGH LIMIT (31-0)

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	090C0h

Register to set up the limit address of DMA-protected high-memory region.
When the LT CMD.LOCK.PMRC command is invoked, this register is locked (treated as RO). When the LT CMD.UNLOCK.PMRCcommand is invoked, this register is unlocked (treated as RW).
This register is always treated as RO for implementations not supporting protected high memory region (PHMR field reported as 0 in the Capability register).
The alignment of the protected high memory region limit depends on the number of reserved bits (N:0) of this register. Software may determine N by writing all 1's to this register, and finding most significant zero bit position below host address width (HAW) in the value read back from the register. Bits N:0 of the limit register are decoded by hardware as all 1s.
The protected high-memory base & limit registers function as follows.
Programming the protected low-memory base and limit registers with the same value in bits HAW:(N+1) specifies a protected low-memory region of size 2^^(N+1) bytes.
Programming the protected high-memory limit register with a value less than the protected high-memory base register disables the protected high-memory region.

DWord	Bit	Description
0	31:20	**PMR HIGH LIMIT (LSB - 31:20 of 38:20)** Access: RO _Custom_GTIReset: BUS This register specifies the last host physical address of the DMA-protected high-memory region in system memory. Hardware ignores and does not implement bits 63:HAW, where HAW is the host address width.0
	19:0	**Spares** Access: RO _Custom_GTIReset: BUS

Mirror of PMR HIGH LIMIT 63-32

PMRHLIMIT_MSB - Mirror of PMR HIGH LIMIT 63-32

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	090C4h

Register to set up the limit address of DMA-protected high-memory region.
When the LT CMD.LOCK.PMRC command is invoked, this register is locked (treated as RO). When the LT CMD.UNLOCK.PMRCcommand is invoked, this register is unlocked (treated as RW). Refer to Chapter 12 for security considerations.
This register is always treated as RO for implementations not supporting protected high memory region (PHMR field reported as 0 in the Capability register).
The alignment of the protected high memory region limit depends on the number of reserved bits (N:0) of this register. Software may determine N by writing all 1s to this register, and finding most significant zero bit position below host address width (HAW) in the value read back from the register. Bits N:0 of the limit register are decoded by hardware as all 1s.
The protected high-memory base & limit registers function as follows.
Programming the protected low-memory base and limit registers with the same value in bits HAW:(N+1) specifies a protected low-memory region of size 2^^(N+1) bytes.
Programming the protected high-memory limit register with a value less than the protected high-memory base register disables the protected high-memory region.

DWord	Bit	Description
0	31:7	**Spares** Access: RO _Custom_GTIReset: BUS
	6:0	**PMR HIGH LIMIT (MSB - 38:32 of 38:20)** Access: RO _Custom_GTIReset: BUS This register specifies the last host physical address of the DMA-protected high-memory region in system memory. Hardware ignores and does not implement bits 63:HAW, where HAW is the host address width.

Mirror of RC0 Hysterisis free

MIRROR_RC0_HYSTERISIS_FREE - Mirror of RC0 Hysterisis free		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	00D94h	
Name:	Mirror of RC0 Hysterisis free	
ShortName:	MIRROR_RC0_HYSTERISIS_FREE	
Mirror of RC0 Hysterisis free from OA register DB84h		
DWord	**Bit**	**Description**
0	31:0	**Count RC0 Hysterisis free** Access: R/W _Custom_GTIReset: BUS

Mirror RC0 Any Engine Busy Free

MIRROR_RC0_ANY_ENGINE_BUSY_FREE - Mirror RC0 Any Engine Busy Free	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00D90h
Name:	Mirror RC0 Any Engine Busy Free
ShortName:	MIRROR_RC0_ANY_ENGINE_BUSY_FREE

This register mirrors the count from OA register DB80h

DWord	Bit	Description
0	31:0	**Count for RC0 Any Engine Busy Free** Access: R/W _Custom_GTIReset: BUS Mirror RC0 Any Engine Busy Free by OA

Mirror RC0 wake count free

MIRROR_RC0_WAKE_COUNT_FREE - Mirror RC0 wake count free	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00D98h
Name:	Mirror RC0 wake count free
ShortName:	MIRROR_RC0_WAKE_COUNT_FREE
Access:	R/W
This mirrors the OA register DB88h	

DWord	Bit	Description
0	31:0	**Count for RC0 wake count free** Access: R/W _Custom_GTIReset: BUS

Mirror Rc6 post hysterisis overhead free

MIRROR_RC6_POST_HYSTERISIS_OVERHEAD_FREE - Mirror Rc6 post hysterisis overhead free	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00DA0h
Name:	Mirror Rc6 post hysterisis overhead free
ShortName:	MIRROR_RC6_POST_HYSTERISIS_OVERHEAD_FREE
This is the mirror from OA for register DB90h	

DWord	Bit	Description
0	31:0	**Count for Mirror Rc6 post hysterisis overhead free** Access: R/W _Custom_GTIReset: BUS

Mirror rc6 pre Engine wake overhead free

MIRROR_RC6_PRE_ENGINE_WAKE_OVERHEAD_FREE - Mirror rc6 pre Engine wake overhead free	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00D9Ch
Name:	Mirror rc6 pre Engine wake overhead free
ShortName:	MIRROR_RC6_PRE_ENGINE_WAKE_OVERHEAD_FREE

Mirrors the OA register DB8Ch

DWord	Bit	Description
0	31:0	**Count for rc6 pre Engine wake overhead free** Access: R/W _Custom_GTIReset: BUS

Misc Clocking Reset Control Registers

MISCCPCTL - Misc Clocking Reset Control Registers

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 09424h

Miscellaneous Clocking / Reset Control Registers

DWord	Bit	Description
0	31	**clock gate control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of MISCCPCTL register are R/W 1 = All bits of MISCCPCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR. Value / Name / Description: 0 / Bits of MISCCPCTL register are R/W 1 / All bits of MISCCPCTL register are RO (including this lock bit)
	30:24	**Bonus ECO bits** Access: R/W _Custom_GTIReset: BUS Bonus ECO bits
	23:20	**DOP clock gating enable for VEbox clks** Access: R/W _Custom_GTIReset: BUS Controls the Enabling of the DOP-level Vebox (cv0clk) Clock Gating via PM event messages 1 - Clock gating is enabled 0 - Clock gating is disabled
	19:16	**DOP clock gating enable for SFC media clks** Access: R/W _Custom_GTIReset: BUS Controls the Enabling of the DOP-level SFC (csfcclk) Clock Gating in media1 via PM event messages 1 - Clock gating is enabled 0 - Clock gating is disabled
	15:8	**DOP clock gate enable for Media Clocks** Access: R/W _Custom_GTIReset: BUS Controls the Enabling of the DOP-level Render (cmclk) Clock Gating via PM event messages

		1 - Clock gating is enabled 0 - Clock gating is disabled
	7	**DOP clock gating enable for posh clks** Access: R/W _Custom_GTIReset: BUS Controls the Enabling of the DOP-level posh (cposclk) Clock Gating via PM event messages 1 - Clock gating is enabled 0 - Clock gating is disabled
	6	**DOP clock gating enable for Media ampler clks** Access: R/W _Custom_GTIReset: BUS Controls the Enabling of the DOP-level Media sampler (scmsclk) Clock Gating via PM event messages 1 - Clock gating is enabled 0 - Clock gating is disabled
	5	**WIDI1 DOP clock gating enable** Access: R/W _Custom_GTIReset: BUS Controls the Enabling of the DOP-level Media sampler (cw1clk) Clock Gating via PM event messages 1 - Clock gating is enabled 0 - Clock gating is disabled
	4	**DOP CLock gating Enable for Widi 0 clocks** Access: R/W _Custom_GTIReset: BUS Controls the Enabling of the DOP-level Render (cwclk) Clock Gating via PM event messages 1 - Clock gating is enabled 0 - Clock gating is disabled
	3	**DOP Clcok gating enable for GUC clocks** Access: R/W _Custom_GTIReset: BUS Controls the Enabling of the DOP-level Render (cpclk) Clock Gating via PM event messages 1 - Clock gating is enabled 0 - Clock gating is disabled
	2	**DOP clock gating Enable for Fix clocks (cfclk)** Access: R/W _Custom_GTIReset: BUS Controls the Enabling of the DOP-level Render (cfclk/cf2xclk) Clock Gating via PM event messages 1 - Clock gating is enabled 0 - Clock gating is disabled

MISCCPCTL - Misc Clocking Reset Control Registers

	1	**DOP Clock Gating Enable for Render Clocks** Access: R/W _Custom_GTIReset: BUS Controls the Enabling of the DOP-level Render (crclk/cr2xclk) Clock Gating via PM event messages 1 - Clock gating is enabled 0 - Clock gating is disabled
	0	**Reserved** Access: RO Format: MBZ

MISC CTX control register

MISCCTXCTL - MISC CTX control register		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	0942Ch	
DWord	**Bit**	**Description**
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**Context Restore ACk indication from Csunit** Access: R/W Set _Custom_GTIReset: BUS Context Restore ACk indication from Csunit 1'b1 : Csunit has completed restoring CPunits adress space Once set, CPunit hardware clears this bit after sending the ctx save ack done message to CS 1'b0 : Csunit has NOT completed restoring CPunits adress space

Miscellaneous Message Register 0 for GTI Doorbell Unit

DRBMISC0 - Miscellaneous Message Register 0 for GTI Doorbell Unit

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	01980h

Message registers have bit-wise masking applied for writes. The register consists of 16 bits of data in [15:0], and 16 bits of corresponding masks in [31:16].
In order to set bit0, for example, the data would be 0x0001_0001. In order to clear bit0, for example, the data would be 0x0001_0000.
Note that mask bit is the data bit offset + 16. Message registers are protected from non-GT writes via the Message Channel.

DWord	Bit	Description
0	31:16	**Context Save Mask** Default Value: 001bh Access: WO _Custom_GTIReset: BUS When context save (a read) is in progress, mask is forced to particular value to save off messages that need to be retained across an RC6 event.
	15:7	**Reserved** Access: RO Format: MBZ
	6	**U2CDATPERR** Default Value: 0 Access: RO _Custom_GTIReset: BUS A one indicates that Data Parity Error was detected.
	5	**U2CADDPERR** Default Value: 0 Access: RO _Custom_GTIReset: BUS A one indicates that Address Parity Error was detected.
	4	**CV History Bit** Access: R/W Hardware Clear _Custom_GTIReset: BUS

DRBMISC0 - Miscellaneous Message Register 0 for GTI Doorbell Unit

	CV History Bit Indicates that doorbell unit should mark all active doorbells as RFO required. Basically it means there may have been some snoops that were missed. RPM writes bit every time CV changes.
3	**Reserved**
2	**Reserved** Access: RO Format: MBZ
1	**RFO Enable indication from RPM** Access: R/W Hardware Clear _Custom_GTIReset: BUS RFO Enable bit from RPM Requires MDRB to handshake with ack 1 – Allow RFOs (RPM Enable) 0 – Block RFOs (RPM Disable) **[Default]**
0	**RFO Enable indication from U2C** Access: R/W Hardware Clear _Custom_GTIReset: BUS RFO Enable bit from U2C Requires MDRB to handshake with ack 1 – Allow RFOs (Enable) 0 – Block RFOs (Disable) **[Default]**

Misc Reset Control Register

RSTCTL - Misc Reset Control Register

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	09420h

Miscellaneous reset control registers.

DWord	Bit	Description
0	31:4	**Reserved** Access: RO Format: MBZ
	3:2	**Reset Staggering Period Control** Access: R/W _Custom_GTIReset: BUS Reset assertion staggering period between reset domains during FLR and soft-resets: 00: 6 csclk staggering reset assertion staggering 01: 12 cs clocks 10: 18 cs clocks 11: 24 cs clocks
	1:0	**Reset Residency Control** Access: R/W _Custom_GTIReset: BUS Reset assertion residency period for FLR and soft-resets. "00" : 8 cs clocks "01" : 16 cs clocks "10" : 32 cs clocks "11" : 64 cs clocks

MMIO_INDEX

MMIO_INDEX - MMIO_INDEX

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	00000h

Contains address and target.
Punit cannot access IO space from message channel.
A 32 bit IO write to this port loads the offset of the MMIO register or offset into the GTT that needs to be accessed.
An IO Read returns the current value of this register. An 8/16 bit IO write to this register is completed but does not update this register.
This mechanism to access internal graphics MMIO registers must not be used to access VGA IO registers which are mapped through the MMIO space. VGA registers must be accessed directly through the dedicated VGA IO ports. This is used by SBIOS. It is not used by graphics driver.
This register is only accessible through the IOSF Primary bus. The base register is defined by IOBAR.

DWord	Bit	Description
0	31:2	**Register_offset** Default Value: 00000000h Access: R/W _Custom_GTIReset: BUS This field selects GTT entry or any one of the Dword registers within the MMIO register space of this device.
	1:0	**Target** Default Value: 00b Access: R/W _Custom_GTIReset: BUS 00 = MMIO Registers, 01 = GTT, 1X = Reserved

Mode Register for GAB

GAB_MODE - Mode Register for GAB

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV

The GAB_MODE register contains information that controls configurations in the GAB.

DWord	Bit	Description
0	31:16	**Mask** Access: WO Format: Mask _Custom_GTIReset: DEV
	15:10	**Reserved** Access: RO Format: MBZ
	9	**Reserved**
	8:7	**Reserved** Format: PBC
	6	**Disable Hashing on BCS** _Custom_GTIReset: DEV Two command ports (P0 and P1) from BLT to GAM are address hashed. BLB to GAB has two ports and address hashing is done internal to BLB. BCS to GAB is a single command port, and GAB address hashes the BCS requests into P0 and P1. **Value** / **Name** / **Description** 0b / **[Default]** / BCS traffic is hashed onto P0 and P1 1b / / BCS traffic is sent only on P0 (no hashing) **Programming Notes** When PendQ hashing control is disabled in GAM BLT Module, BCS hashing in GAB must also be disabled by setting this bit to '1. If this is not followed, there can be memory ordering violations on BCS transactions.
	5:3	**BLB Arbitration Priority** Format: U3 _Custom_GTIReset: DEV
	2:0	**BCS Arbitration Priority** Format: U3 _Custom_GTIReset: DEV

Mode Register for GAC

GAC_MODE - Mode Register for GAC	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	1C00A0h-1C00A3h
Name:	Mode Register for GAC
ShortName:	GAC_MODE_VCSUNIT_BE0
Address:	1C40A0h-1C40A3h
Name:	Mode Register for GAC
ShortName:	GAC_MODE_VCSUNIT_BE1
Address:	1D00A0h-1D00A3h
Name:	Mode Register for GAC
ShortName:	GAC_MODE_VCSUNIT_BE2
Address:	1D40A0h-1D40A3h
Name:	Mode Register for GAC
ShortName:	GAC_MODE_VCSUNIT_BE3
Address:	1E00A0h-1E00A3h
Name:	Mode Register for GAC
ShortName:	GAC_MODE_VCSUNIT_BE4
Address:	1E40A0h-1E40A3h
Name:	Mode Register for GAC
ShortName:	GAC_MODE_VCSUNIT_BE5
Address:	1F00A0h-1F00A3h
Name:	Mode Register for GAC
ShortName:	GAC_MODE_VCSUNIT_BE6
Address:	1F40A0h-1F40A3h
Name:	Mode Register for GAC
ShortName:	GAC_MODE_VCSUNIT_BE7

The GAC_MODE register contains information that controls configurations in the GAC.

DWord	Bit	Description
0	31:16	**Mask** Access: WO Format: Mask _Custom_GTIReset: BUS

GAC_MODE - Mode Register for GAC

	15:1	**Reserved** Access: RO Format: MBZ
	0	**GACunit VCS Fence Performance fix Override** Format: Disable _Custom_GTIReset: BUS

Value	Name	Description
0	**[Default]**	Performance fix **enabled** to block client credits until VCS fence advances.
1		Performance fix to block credits until VCS fence advances, **disabled.** Fence will not block input traffic from clients and will advance only after ingress FIFOs are empty (Legacy behavior.)

Mode Register for GAFS

GAFS_MODE - Mode Register for GAFS

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0212Ch

DWord	Bit	Description
0	31:16	**Mask Bits** Access: WO Format: Mask Masks: These bits serve as write enables for bits 15:0. If this register is written with any of these bits clear the corresponding bit in the field 15:0 will not be modified. Reading these bits always returns 0s.
	15:13	**Reserved** Format: PBC
	12:11	**CLR0 clients ROB low priority threshold Control** This bit field is used to control the threshold at which CLR0 clients will move to the low priority group in the GAFS ROB arbiter. 00b **[Default]** Low priority threshold value = 'd12 01b Low priority threshold value = 'd24 10b Low priority threshold value = 'd36 11b Low priority threshold value = 'd48
	10	**Reserved** Format: PBC
	9	**Min Alloc Configuration** This field can be used to adjust min alloc settings in the read ROB structure in GAFS. This is used in conjunction with bit [2] of this register. {Bit[9], Bit[2]} = 2'b00 : Original values {Bit[9], Bit[2]} = 2'b01 : Override original values to gain 12 ROB locations for generic use {Bit[9], Bit[2]} = 2'b10 :Override original values to gain 22 ROB locations for generic use {Bit[9], Bit[2]} = 2'b11 : Original values **Programming Notes** This bit must be programmed to 1 for achieving performance targets.
	8:3	**Reserved** Format: PBC

GAFS_MODE - Mode Register for GAFS

	2	**Min Alloc Configuration control0** This field can be used to adjust min alloc settings in the read ROB structure in GAFS. This is used in conjunction with bit [9] of this register. {Bit[9], Bit[2]} = 2'b00 : Original values {Bit[9], Bit[2]} = 2'b01 : Override original values to gain 12 ROB locations for generic use {Bit[9], Bit[2]} = 2'b10 :Override original values to gain 22 ROB locations for generic use {Bit[9], Bit[2]} = 2'b11 : Original values **Programming Notes** This bit must be programmed to 0 for achieving performance targets.
	1:0	**Reserved** Format: PBC

Mode Register for Software Interface

MI_MODE - Mode Register for Software Interface	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0209Ch-0209Fh
Name:	Mode Register for Software Interface
ShortName:	MI_MODE_RCSUNIT
Address:	1809Ch-1809Fh
Name:	Mode Register for Software Interface
ShortName:	MI_MODE_POCSUNIT
Address:	2209Ch-2209Fh
Name:	Mode Register for Software Interface
ShortName:	MI_MODE_BCSUNIT
Address:	1C009Ch-1C009Fh
Name:	Mode Register for Software Interface
ShortName:	MI_MODE_VCSUNIT0
Address:	1C409Ch-1C409Fh
Name:	Mode Register for Software Interface
ShortName:	MI_MODE_VCSUNIT1
Address:	1C809Ch-1C809Fh
Name:	Mode Register for Software Interface
ShortName:	MI_MODE_VECSUNIT0
Address:	1D009Ch-1D009Fh
Name:	Mode Register for Software Interface
ShortName:	MI_MODE_VCSUNIT2
Address:	1D409Ch-1D409Fh
Name:	Mode Register for Software Interface
ShortName:	MI_MODE_VCSUNIT3
Address:	1D809Ch-1D809Fh
Name:	Mode Register for Software Interface
ShortName:	MI_MODE_VECSUNIT1
Address:	1E009Ch-1E009Fh
Name:	Mode Register for Software Interface
ShortName:	MI_MODE_VCSUNIT4

MI_MODE - Mode Register for Software Interface

Address:	1E409Ch-1E409Fh
Name:	Mode Register for Software Interface
ShortName:	MI_MODE_VCSUNIT5

Address:	1E809Ch-1E809Fh
Name:	Mode Register for Software Interface
ShortName:	MI_MODE_VECSUNIT2

Address:	1F009Ch-1F009Fh
Name:	Mode Register for Software Interface
ShortName:	MI_MODE_VCSUNIT6

Address:	1F409Ch-1F409Fh
Name:	Mode Register for Software Interface
ShortName:	MI_MODE_VCSUNIT7

Address:	1F809Ch-1F809Fh
Name:	Mode Register for Software Interface
ShortName:	MI_MODE_VECSUNIT3

Address:	1A09Ch-1A09Fh
Name:	Mode Register for Software Interface
ShortName:	MI_MODE_CCSUNIT0

The MI_MODE register contains information that controls software interface aspects of the Memory Interface function.

DWord	Bit	Description
0	31:16	**Mask** Access: WO Format: Mask A 1 in a bit in this field allows the modification of the corresponding bit in Bits 15:0
	15	**Suspend Flush** Format: U1

Value	Name	Description
0h	No Delay **[Default]**	HW will not delay flush, this bit will get cleared by MI_SUSPEND_FLUSH as well
1h	Delay Flush	Suspend flush is active

Programming Notes
This should only be written to from the ring using MI_SUSPEND_FLUSH. It is considered undefined if written by software through MMIO

MI_MODE - Mode Register for Software Interface

14 **RCS POSH LRCA Disable**

Source:	RenderCS
Exists If:	//RCS

This bit controls the context save of the MI_LOAD_REGISTER_IMM command corresponding to POSH_LRCA as part of the RCS context image. This primary purpose of this bit is for backward compatibility for the render context image.

Value	Name	Description
1		RCS saves the MI_LOAD_REGISTER_IMM command corresponding to POSH_LRCA in context image as NOOPs.
0	**[Default]**	RCS saves the MI_LOAD_REGISTER_IMM command corresponding to POSH_LRCA as part of the context image.

13 **Disable MI_SET_CONTEXT for Execution List**

Format:	U1

Value	Name	Description
0h	Allow **[Default]**	Allow MI_SET_CONTEXT in Execlist mode
1h	Disable	Disable MI_SET_CONTEXT in Execlist Mode

12 **Nested Batch Buffer Enable**

This bit is used to enable or disable third level batch buffer and associated functionality.

Value	Name	Description
0	**[Default]**	HW will only support first level, second level, chained first level and chained second level batch buffer. Nested Level Batch Buffer field must be set in MI_BATCH_BUFFER_START command programmed in a second level batch buffer to inferas a chained second level batch buffer.
1		HW will support first level, second level, third level, chained first level, chained second level and chained third level batch buffer. Nested Level Batch Buffer field must not be set in MI_BATCH_BUFFER_START command programmed in a second level batch buffer to infer a chained second level batch buffer and similarly for chained third level batch buffer.

Programming Notes
This bit must be only programmed from a ring buffer and must not be programmed from a batch buffer. This bit must be static for a given context and remain same through out the life time of a given context.

11 **Invalidate UHPTR enable**

Source:	RenderCS, VideoCS, VideoCS2, VideoEnhancementCS
Exists If:	//RCS, VCS, VECS, BCS
Format:	Enable

If bit set H/W clears the valid bit of UHPTR (2134h, bit 0) when current active head pointer is equal to UHPTR.

MI_MODE - Mode Register for Software Interface

	10	**Atomic Read Return for MI_COPY_MEM_MEM** Format: U1 **Value** / **Name** / **Description** 0h / Disable **[Default]** / Hardware does a regular memory fence write to complete the write to the destination address before moving to the next instruction. 1h / Enable / Hardware does Atomic Move with Read Return to complete the write to the destination address before moving to the next instruction.
	9	**Rings Idle** Format: U1 Read Only Status bit **Value** / **Name** / **Description** 0h / Not Idle **[Default]** / Parser not Idle or Ring Arbiter not Idle. 1h / Idle / Parser Idle and Ring Arbiter Idle. **Programming Notes** Writes to this bit are not allowed.
	8	**Stop Rings** Format: U1 **Value** / **Name** / **Description** 0h / **[Default]** / Normal Operation. 1h / / Parser is turned off and Ring arbitration is turned off. **Programming Notes** Software must set this bit to force the Rings and Command Parser to Idle. Software must read a 1 in the Ring Idle bit after setting this bit to ensure that the hardware is idle. Software must clear this bit for Rings to resume normal operation.
	7:5	**Reserved** Format: PBC
	4:2	**Reserved** Source: BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS Format: PBC
	4:1	**Predicate Enable** Source: RenderCS, PositionCS This field gets set when "MI_SET_PREDICATE" command is parsed by render command streamer. Predicate Disable is the default mode of operation.

MI_MODE - Mode Register for Software Interface

Value	Name	Description
0h	Predicate Disable	Predication is Disabled and RCS will process commands as usual.
1h	Predicate on Result2 clear	Following Commands will be NOOPED by RCS only if the MI_PREDICATE_RESULT_2 is clear.
2h	Predicate on Result2 set	Following Commands will be NOOPED by RCS only if the MI_PREDICATE_RESULT_2 is set.
3h	Predicate on Result clear	Following Commands will be NOOPED by RCS only if the MI_PREDICATE_RESULT is clear.
4h	Predicate on Result set	Following Commands will be NOOPED by RCS only if the MI_PREDICATE_RESULT is set.
5h	Predicate when two or more slices enabled	Following Commands will be NOOPED by RCS only when one slice is enabled, NOOPED when more than one slice is enabled.
6h	Predicate when one or three slices enabled	Following Commands will be Executed by RCS only when two slices are enabled, NOOPED when one or three slices are enabled.
7h	Predicate when one or two slices enabled	Following Commands will be Executed by RCS only when all the three slices are enabled, NOOPED when less than three slices are enabled.
Bh	NOOP in RenderCS	When RenderCS parses MI_SET_PREDICATE command with Predicate Enable set to NOOP in RenderCS, RenderCS NOOPs all the subsequent commands parsed unconditionally until the predication is disabled/modified using MI_SET_PREDICATE command. Other command streamers (non RenderCS) on parsing MI_SET_PREDICATE command with Predicate Enable set to NOOP in RenderCS dont take any action and is equivalent to parsing MI_NOOP command.
Ch	NOOP in PositionCS	When PositionCS parses MI_SET_PREDICATE command with Predicate Enable set to NOOP in PositionCS, PositionCS NOOPs all the subsequent commands parsed unconditionally until the predication is disabled/modified using MI_SET_PREDICATE command. Other command streamers (non PositionCS) on parsing MI_SET_PREDICATE command with Predicate Enable set to NOOP in PositionCS dont take any action and is equivalent to parsing MI_NOOP command.
8h, 9h, Ah	Reserved	
Dh, Eh	Reserved	
Fh	Predicate Always	Following Commands will be NOOPED by RCS unconditionally.

MI_MODE - Mode Register for Software Interface

		Programming Notes SW must use MI_SET_PREDICATE instead of MMIO access.
	1	**Reserved** Source: VideoCS, VideoCS2, VideoEnhancementCS Format: PBC
	1	**Reserved** Source: BlitterCS Format: PBC
	0	**Reserved** Source: CommandStreamer Format: PBC

MSI Mask Bits

MSI_MASK_0_2_0_PCI - MSI Mask Bits		
Register Space:	PCI: 0/2/0	
Size (in bits):	32	
Address:	000B8h	
This register contains the MSI Mask Bits		
DWord	**Bit**	**Description**
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**Mask Bit for Vector 0** Default Value: 0b Access: R/W _Custom_GTIReset: BUS For each Mask bit that is set, the function is prohibited from sending the associated message.

MSI Pending Bits

MSI_PEND_0_2_0_PCI - MSI Pending Bits

Register Space:	PCI: 0/2/0
Size (in bits):	32
Address:	000BCh

This register contains the MSI Pending Bits

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**Pending Bit for Vector 0** Default Value: 0b Access: RO Variant _Custom_GTIReset: BUS For each Pending bit that is set, the function has a pending associated message. If this bit is set when the corresponding vector's Mask bit is cleared, the function will send an MSI and then clear the Pending bit.

Multi Context Control Register

MLTICTXCTL - Multi Context Control Register

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B170h

Masked register for Multi Context

DWord	Bit	Description
0	31:16	**Mask** Access: WO Format: Mask Mask bits for each corresponding bit in [15:0]
	15	**AID for RCS** AID0: The AID value for QID = 0
	14	**AID for CCS** AID1: The AID value for QID = 1
	13:4	**Reserved** Access: RO Format: MBZ
	3	**Reserved** Access: RO Format: MBZ
	2	**3D on Render** 3D_ON_RENDER: When set, the L3 will use Render config values, else compute config values.
	1:0	**Multi Context Enable** This Mode enables duel queues to execute on the machine: 1 RCS + 1 CCS If set, then L3 will carve out 4KB of URB per bank for CCS usage. Reserved: 10b, 11b Value / Name: 0 – RCS only enabled **[Default]** 1 – RCS and CCS enabled

Multi Size Aperture Control

MSAC_0_2_0_PCI - Multi Size Aperture Control

Register Space:	PCI: 0/2/0
Size (in bits):	32
Address:	00060h

Description

This register contains MSAC register which determines the size of the graphics memory aperture (GMADR) in function 0 and in the trusted space, and affects certain bits of the GMADR register.
Bits [20:16] 00000b: 128MB, GMADR[26:4] is hardwired to all 0
Bits [20:16] 00001b: 256MB, GMADR[27:4] overridden to all 0
Bits [20:16] 00010b: illegal (hardware will treat this as 00011b)
Bits [20:16] 00011b: 512MB, GMADR[28:27] overridden to all 0
Bits [20:16] 00100-00110b: illegal (hardware will treat this as 00111b)
Bits [20:16] 00111b: 1024MB, GMADR[29:27] overridden to all 0
Bits [20:16] 01000-01110b: illegal (hardware will treat this as 01111b)
Bits [20:16] 01111b: 2048MB, GMADR[30:27] overridden to all 0
Bits [20:16] 10000-11110b: illegal (hardware will treat this as 11111b)
Bits [20:16] 11111b: 4096MB, GMADR[31:27] overridden to all 0

DWord	Bit	Description
0	31:21	**Reserved R/W** Default Value: 000h Access: R/W _Custom_GTIReset: BUS Scratch Bits
	20	**Untrusted Aperture Size Bit 4** Default Value: 0b Access: R/W Key _Custom_GTIReset: BUS
	19	**Untrusted Aperture Size Bit 3** Default Value: 0b Access: R/W Key _Custom_GTIReset: BUS
	18	**Untrusted Aperture Size Bit 2** Default Value: 0b Access: R/W Key _Custom_GTIReset: BUS

MSAC_0_2_0_PCI - Multi Size Aperture Control

	17	**Untrusted Aperture Size Bit 1** Default Value: 0b Access: R/W Key _Custom_GTIReset: BUS
	16	**Untrusted Aperture Size Bit 0** Access: R/W Key _Custom_GTIReset: BUS Value / Name / Description 0b / **[Default]** /
	15:0	**Reserved** Access: RO Format: MBZ

NDE_RSTWRN_OPT

NDE_RSTWRN_OPT

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	46408h-4640Bh
Name:	North Display Reset Warn Options
ShortName:	NDE_RSTWRN_OPT
Reset:	global

This register is used to control the display behavior on receiving a Reset Warning or FLR.

DWord	Bit	Description
0	31:7	**Reserved** Access: RO Format: MBZ
	6	**Reserved** Access: RO Format: MBZ
	5	**Reserved**
	4	**RST PCH Handshake En** This field enables the handshake with south display when processing the reset. This applies to all types of DE resets. By default it is disabled and the north display will not wait for south display to acknowledge the reset. Value: 0b – Disable; 1b – Enable Programming Notes: This must be programmed as part of the display initialization sequence.
	3:0	**Reserved** Access: RO Format: MBZ

NONPIPE_ISOCREQ

NONPIPE_ISOCREQ

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
Address:	45230h-45237h
Name:	Non Pipe Isoch Request
ShortName:	NONPIPE_ISOCREQ
Reset:	soft

This register is used to trigger an IsocReq that is not associated with a pipe. When enabled, the write to DWord 1 (higher address DWord) of this register triggers an IsocReq to be sent with the last written values from both DWords.

DWord	Bit	Description
0	31:16	**LTR** Access: R/W This field specifies the latency tolerance (LTR) in microseconds.
	15:0	**Bandwidth** Access: R/W This field specifies the bandwidth requirement in multiples of 100 MB/s.
1	31	**Enable** Access: R/W This field enables IsocReq to be sent when DWord 1 of this register is written. Value: 1b — Name: Enable Value: 0b — Name: Disable
	30:8	**Reserved** Access: RO Format: MBZ
	7:0	**Delay** Access: R/W This field specifies the downwards transition delay in milliseconds.

NOP Identification Register

NOPID - NOP Identification Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02094h-02097h
Name:	NOP Identification Register
ShortName:	NOPID_RCSUNIT
Address:	18094h-18097h
Name:	NOP Identification Register
ShortName:	NOPID_POCSUNIT
Address:	22094h-22097h
Name:	NOP Identification Register
ShortName:	NOPID_BCSUNIT
Address:	1C0094h-1C0097h
Name:	NOP Identification Register
ShortName:	NOPID_VCSUNIT0
Address:	1C4094h-1C4097h
Name:	NOP Identification Register
ShortName:	NOPID_VCSUNIT1
Address:	1C8094h-1C8097h
Name:	NOP Identification Register
ShortName:	NOPID_VECSUNIT0
Address:	1D0094h-1D0097h
Name:	NOP Identification Register
ShortName:	NOPID_VCSUNIT2
Address:	1D4094h-1D4097h
Name:	NOP Identification Register
ShortName:	NOPID_VCSUNIT3
Address:	1D8094h-1D8097h
Name:	NOP Identification Register
ShortName:	NOPID_VECSUNIT1
Address:	1E0094h-1E0097h
Name:	NOP Identification Register

NOPID - NOP Identification Register

ShortName:	NOPID_VCSUNIT4
Address:	1E4094h-1E4097h
Name:	NOP Identification Register
ShortName:	NOPID_VCSUNIT5
Address:	1E8094h-1E8097h
Name:	NOP Identification Register
ShortName:	NOPID_VECSUNIT2
Address:	1F0094h-1F0097h
Name:	NOP Identification Register
ShortName:	NOPID_VCSUNIT6
Address:	1F4094h-1F4097h
Name:	NOP Identification Register
ShortName:	NOPID_VCSUNIT7
Address:	1F8094h-1F8097h
Name:	NOP Identification Register
ShortName:	NOPID_VECSUNIT3
Address:	1A094h-1A097h
Name:	NOP Identification Register
ShortName:	NOPID_CCSUNIT0

The NOPID register contains the Noop Identification value specified by the last MI_NOOP instruction that enabled this register to be updated.

DWord	Bit	Description
0	31:22	**Reserved** Access: RO Format: MBZ
	21:0	**Reserved**

Number Of VFs

SRIOV_NUMOFVFS_0_2_0_PCI - Number Of VFs		
Register Space: PCI: 0/2/0 Size (in bits): 32		
Address: 00330h		
Number of VFs enabled by the VMM.		
DWord	**Bit**	**Description**
0	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**Function Dependency Link** Default Value: 00000000b Access: RO _Custom_GTIReset: BUS Same value as the Physical function number indicating no Dependency
	15:0	**Number of Virtual Functions** Default Value: 0000000000000000b Access: R/W _Custom_GTIReset: BUS System SW shall set this field to control the number of VFs that are visible. This field must be programmed before setting VF Enable. Changing this field when VF Enable is set will produced undefined behavior as per the SR-IOV specification. HW will ignore the new value programmed.

OAC Aggregate Perf Counter A0

OAC_OAPERF_A0 - OAC Aggregate Perf Counter A0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register reflects the count value of the OA Performance counter A0.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads. Value: 0x00000000 **[Default]** Value: [0x00000001-0xFFFFFFFF]

OAC Aggregate Perf Counter A0 Upper DWord

OAC_OAPERF_A0_UPPER - OAC Aggregate Perf Counter A0 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A0 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A4

OAC_OAPERF_A4 - OAC Aggregate Perf Counter A4		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A4. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read.Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO,so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A4 Upper DWord

OAC_OAPERF_A4_UPPER - OAC Aggregate Perf Counter A4 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A4 to be read. .Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32of the live performance counter value when read..Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO,so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A7

OAC_OAPERF_A7 - OAC Aggregate Perf Counter A7		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A7. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A7 Upper DWord

OAC_OAPERF_A7_UPPER - OAC Aggregate Perf Counter A7 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A7 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A8

OAC_OAPERF_A8 - OAC Aggregate Perf Counter A8		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A8. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A8 Upper DWord

OAC_OAPERF_A8_UPPER - OAC Aggregate Perf Counter A8 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A8 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A9

OAC_OAPERF_A9 - OAC Aggregate Perf Counter A9		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A9. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A9 Upper DWord

OAC_OAPERF_A9_UPPER - OAC Aggregate Perf Counter A9 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A9 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A10

OAC_OAPERF_A10 - OAC Aggregate Perf Counter A10		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A10. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A10 Upper DWord

OAC_OAPERF_A10_UPPER - OAC Aggregate Perf Counter A10 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A10 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A11

OAC_OAPERF_A11 - OAC Aggregate Perf Counter A11		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A11. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A11 Upper DWord

OAC_OAPERF_A11_UPPER - OAC Aggregate Perf Counter A11 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A11 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A12

OAC_OAPERF_A12 - OAC Aggregate Perf Counter A12		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A12. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A12 Upper DWord

OAC_OAPERF_A12_UPPER - OAC Aggregate Perf Counter A12 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A12 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A13

OAC_OAPERF_A13 - OAC Aggregate Perf Counter A13		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A13. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A13 Upper DWord

OAC_OAPERF_A13_UPPER - OAC Aggregate Perf Counter A13 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A13 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A14

OAC_OAPERF_A14 - OAC Aggregate Perf Counter A14		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A14. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A14 Upper DWord

OAC_OAPERF_A14_UPPER - OAC Aggregate Perf Counter A14 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A14 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A15

OAC_OAPERF_A15 - OAC Aggregate Perf Counter A15		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A15. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A15 Upper DWord

OAC_OAPERF_A15_UPPER - OAC Aggregate Perf Counter A15 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A15 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A16

OAC_OAPERF_A16 - OAC Aggregate Perf Counter A16		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A16. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A16 Upper DWord

OAC_OAPERF_A16_UPPER - OAC Aggregate Perf Counter A16 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A16 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A17

OAC_OAPERF_A17 - OAC Aggregate Perf Counter A17		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A17. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A17 Upper DWord

OAC_OAPERF_A17_UPPER - OAC Aggregate Perf Counter A17 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A17 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A18

OAC_OAPERF_A18 - OAC Aggregate Perf Counter A18		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A9. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A18 Upper DWord

OAC_OAPERF_A18_UPPER - OAC Aggregate Perf Counter A18 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A18 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A19

OAC_OAPERF_A19 - OAC Aggregate Perf Counter A19		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A19. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read.Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO,so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A19 Upper DWord

OAC_OAPERF_A19_UPPER - OAC Aggregate Perf Counter A19 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A19 to be read.Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO,so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A20

OAC_OAPERF_A20 - OAC Aggregate Perf Counter A20		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A20. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO,so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A20 Upper DWord

OAC_OAPERF_A20_UPPER - OAC Aggregate Perf Counter A20 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A20 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO,so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A28

OAC_OAPERF_A28 - OAC Aggregate Perf Counter A28		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A28. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A28 Upper DWord

OAC_OAPERF_A28_UPPER - OAC Aggregate Perf Counter A28 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A28 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A29

OAC_OAPERF_A29 - OAC Aggregate Perf Counter A29		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A29. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A29 Upper DWord

OAC_OAPERF_A29_UPPER - OAC Aggregate Perf Counter A29 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A29 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A30

OAC_OAPERF_A30 - OAC Aggregate Perf Counter A30		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A30. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A30 Upper DWord

OAC_OAPERF_A30_UPPER - OAC Aggregate Perf Counter A30 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A30 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A31

OAC_OAPERF_A31 - OAC Aggregate Perf Counter A31		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A31		
DWord	**Bit**	**Description**
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A31 Upper DWord

OAC_OAPERF_A31_UPPER - OAC Aggregate Perf Counter A31 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A31 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A32

OAC_OAPERF_A32 - OAC Aggregate Perf Counter A32		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A32		
DWord	**Bit**	**Description**
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OACAggregatePerfCounterA32UpperDWord

OAC_OAPERF_A32_UPPER - OACAggregatePerfCounterA32UpperDWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A32 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A33

OAC_OAPERF_A33 - OAC Aggregate Perf Counter A33		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A33		
DWord	**Bit**	**Description**
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OACAggregatePerfCounterA33UpperDWord

OAC_OAPERF_A33_UPPER - OACAggregatePerfCounterA33UpperDWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A33 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A34

OAC_OAPERF_A34 - OAC Aggregate Perf Counter A34		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A34		
DWord	**Bit**	**Description**
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OACAggregatePerfCounterA34UpperDWord

OAC_OAPERF_A34_UPPER - OACAggregatePerfCounterA34UpperDWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A34 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this upper register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Aggregate Perf Counter A35

OAC_OAPERF_A35 - OAC Aggregate Perf Counter A35		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
This register reflects the count value of the OA Performance counter A35		
DWord	**Bit**	**Description**
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OACAggregatePerfCounterA35UpperDWord

OAC_OAPERF_A35_UPPER - OACAggregatePerfCounterA35UpperDWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter A35 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this upper register.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Boolean_Counter_B0

OAC_OAPERF_B0 - OAC Boolean_Counter_B0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter B0 to be read. Since what conditions cause B0 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Boolean_Counter_B1

OAC_OAPERF_B1 - OAC Boolean_Counter_B1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter B1 to be read. Since what conditions cause B1 to increment are defined by the programming of CEC1-0/CEC1-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Boolean_Counter_B2

OAC_OAPERF_B2 - OAC Boolean_Counter_B2

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter B2 to be read. Since what conditions cause B2 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Boolean_Counter_B3

OAC_OAPERF_B3 - OAC Boolean_Counter_B3

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter B3 to be read. Since what conditions cause B3 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Boolean_Counter_B4

OAC_OAPERF_B4 - OAC Boolean_Counter_B4

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter B4 to be read. Since what conditions cause B4 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Boolean_Counter_B5

OAC_OAPERF_B5 - OAC Boolean_Counter_B5

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter B5 to be read. Since what conditions cause B5 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Boolean_Counter_B6

OAC_OAPERF_B6 - OAC Boolean_Counter_B6

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter B6 to be read. Since what conditions cause B6 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Boolean_Counter_B7

OAC_OAPERF_B7 - OAC Boolean_Counter_B7

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register enables the current live value of performance counter B7 to be read. Since what conditions cause B7 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC GPU Ticks Counter

OAC_GPU_TICKS - OAC GPU Ticks Counter

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

Reading this register returns the live value of the GPU ticks counter that is sampled and included in the performance counter report header. It increments every GFX clock and hence the number of increments per second changes with GFX frequency.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the GPU tick counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC GPU Ticks Counter Upper DWord

OAC_GPU_TICKS_UPPER - OAC GPU Ticks Counter Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

Reading this register returns the live value of the Upper DWord of GPU ticks counter that is sampled and included in the performance counter report header. It increments every GFX clock and hence the number of increments per second changes with GFX frequency.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32 of the GPU tick counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAC Observation Architecture Control

OAC_OACONTROL - OAC Observation Architecture Control

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register controls OAC functionality, report format, and context filtering. If OAC is enabled then it should be enabled in CS as well

DWord	Bit	Description
0	31:4	**Reserved** Format: PBC _Custom_GTIReset: DEV
	3:1	**Counter Select** _Custom_GTIReset: DEV This field selects which performance counter report format to use, please refer to Performance Counter Report Formats section for more details on the structure of the format.
	0	**Performance Counter Enable** Format: Enable _Custom_GTIReset: DEV Global performance counter enable. If clear, no counting will occur. MI_REPORT_PERF_COUNT is undefined when clear. **Programming Notes** "OAC Context Enable" mode bit in RenderEngne CTX_SR_CTL register must be set when "Programmer Counter Enable" is set.

OAC Observation Architecture Status Register

OAC_OASTATUS - OAC Observation Architecture Status Register

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

This register provides status of report buffer and overflow conditions as well as status of some flags which hardware needs for internal purpose. Software should not program these status bits. Status bits are active high flags (0=false, 1=true)

DWord	Bit	Description
0	31:22	**Reserved** Format: PBC _Custom_GTIReset: DEV
	21	**Start Trigger Flag 1** _Custom_GTIReset: DEV Value / Name: 0 **[Default]**; 1 **Programming Notes**: This bit is for HW internal use to context save /restore Start Trigger 1 occurrence On RC6 entry. Software should not program this bit.
	20	**Start Trigger Flag 2** _Custom_GTIReset: DEV Value / Name: 0 **[Default]**; 1 **Programming Notes**: This bit is for HW internal use to context save /restore Start Trigger 2 occurrence On RC6 entry. Software should not program this bit.
	19	**Report Trigger Flag 1** _Custom_GTIReset: DEV Value / Name: 0 **[Default]**; 1

OAC_OASTATUS - OAC Observation Architecture Status Register

		Programming Notes This bit is for HW internal use to context save /restore Report Trigger 1 occurrence On RC6 entry. Software should not program this bit.
	18	**Report Trigger Flag 2** _Custom_GTIReset: DEV **Value** / **Name** 0 **[Default]** 1 **Programming Notes** This bit is for HW internal use to context save /restore Report Trigger 2 occurrence On RC6 entry. Software should not program this bit.
	17:6	**Reserved** Format: PBC _Custom_GTIReset: DEV
	5:2	**Reserved** Format: PBC _Custom_GTIReset: DEV
	1	**Accumulator Overflow** _Custom_GTIReset: DEV This field indicates that the one or more event accumulator inside the slice-OAunit has overflowed. Once set, this bit will remain set, until SW resets it by either soft reset or writing a 0 to it. **Value** / **Name** / **Description** 0 **[Default]** No overflow has occurred. 1 Overflow has occurred.
	0	**Counter Overflow** _Custom_GTIReset: DEV This bit is set if any of the counters overflows. This bit can be reset by SW by either soft reset or writing a 1 to it. **Value** / **Name** / **Description** 0 **[Default]** Counter Overflow not occurred 1 Counter Overflow occurred

OAG Aggregate Perf Counter A0

OAG_OAPERF_A0 - OAG Aggregate Perf Counter A0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D980h

This register reflects the count value of the OA Performance counter A0.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads. Value: 0x00000000 **[Default]** Value: [0x00000001-0xFFFFFFFF]

OAG Aggregate Perf Counter A0 Upper DWord

OAG_OAPERF_A0_UPPER - OAG Aggregate Perf Counter A0 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D984h

This register enables the current live value of performance counter A0 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A1

OAG_OAPERF_A1 - OAG Aggregate Perf Counter A1		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0D988h	
This register reflects the count value of the OA Performance counter A1. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A1 Upper DWord

OAG_OAPERF_A1_UPPER - OAG Aggregate Perf Counter A1 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D98Ch

This register enables the current live value of performance counter A1 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A2

OAG_OAPERF_A2 - OAG Aggregate Perf Counter A2		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0D990h	
This register reflects the count value of the OA Performance counter A2. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A2 Upper DWord

OAG_OAPERF_A2_UPPER - OAG Aggregate Perf Counter A2 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D994h

This register enables the current live value of performance counter A2 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A3

OAG_OAPERF_A3 - OAG Aggregate Perf Counter A3		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0D998h	
This register reflects the count value of the OA Performance counter A3. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A3 Upper DWord

OAG_OAPERF_A3_UPPER - OAG Aggregate Perf Counter A3 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D99Ch

This register enables the current live value of performance counter A3 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A4

OAG_OAPERF_A4 - OAG Aggregate Perf Counter A4		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0D9A0h	
This register reflects the count value of the OA Performance counter A4. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A4 Lower DWord Free

OAG_OAPERF_A4_LOWER_FREE - OAG Aggregate Perf Counter A4 Lower DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DAD4h

This register counts the same event as counter A4 however is not affected by context ID or other conditions that prevent A4 from incrementing. This counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted used this register.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A4 Upper DWord

OAG_OAPERF_A4_UPPER - OAG Aggregate Perf Counter A4 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D9A4h

This register enables the current live value of performance counter A4 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A4 Upper DWord Free

OAG_OAPERF_A4_UPPER_FREE - OAG Aggregate Perf Counter A4 Upper DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DAD8h

This register counts the same event as counter A4 however is not affected by context ID or other conditions that prevent A4 from incrementing. This counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted using this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A5

OAG_OAPERF_A5 - OAG Aggregate Perf Counter A5		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0D9A8h	
This register reflects the count value of the OA Performance counter A5. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A5 Upper DWord

OAG_OAPERF_A5_UPPER - OAG Aggregate Perf Counter A5 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D9ACh

This register enables the current live value of performance counter A5 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A6

OAG_OAPERF_A6 - OAG Aggregate Perf Counter A6		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0D9B0h	
This register reflects the count value of the OA Performance counter A6. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A6 Lower DWord Free

OAG_OAPERF_A6_LOWER_FREE - OAG Aggregate Perf Counter A6 Lower DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DADCh

This register counts the same event as counter A6 however is not affected by context ID or other conditions that prevent A6 from incrementing. his counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted used this register.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A6 Upper DWord

OAG_OAPERF_A6_UPPER - OAG Aggregate Perf Counter A6 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D9B4h

This register enables the current live value of performance counter A6 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A6 Upper DWord Free

OAG_OAPERF_A6_UPPER_FREE - OAG Aggregate Perf Counter A6 Upper DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DAE0h

This register counts the same event as counter A6 however is not affected by context ID or other conditions that prevent A6 from incrementing. his counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted using this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A7

OAG_OAPERF_A7 - OAG Aggregate Perf Counter A7		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0D9B8h	
This register reflects the count value of the OA Performance counter A7. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A7 Upper DWord

OAG_OAPERF_A7_UPPER - OAG Aggregate Perf Counter A7 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D9BCh

This register enables the current live value of performance counter A7 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A8

OAG_OAPERF_A8 - OAG Aggregate Perf Counter A8		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0D9C0h	
This register reflects the count value of the OA Performance counter A8. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A8 Upper DWord

OAG_OAPERF_A8_UPPER - OAG Aggregate Perf Counter A8 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D9C4h

This register enables the current live value of performance counter A8 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A9

OAG_OAPERF_A9 - OAG Aggregate Perf Counter A9		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0D9C8h	
This register reflects the count value of the OA Performance counter A9. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A9 Upper DWord

OAG_OAPERF_A9_UPPER - OAG Aggregate Perf Counter A9 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D9CCh

This register enables the current live value of performance counter A9 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A10

OAG_OAPERF_A10 - OAG Aggregate Perf Counter A10		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0D9D0h	
This register reflects the count value of the OA Performance counter A10. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A10 Upper DWord

OAG_OAPERF_A10_UPPER - OAG Aggregate Perf Counter A10 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D9D4h

This register enables the current live value of performance counter A10 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A11

OAG_OAPERF_A11 - OAG Aggregate Perf Counter A11		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0D9D8h	
This register reflects the count value of the OA Performance counter A11. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A11 Upper DWord

OAG_OAPERF_A11_UPPER - OAG Aggregate Perf Counter A11 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D9DCh

This register enables the current live value of performance counter A11 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A12

OAG_OAPERF_A12 - OAG Aggregate Perf Counter A12		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0D9E0h	
This register reflects the count value of the OA Performance counter A12. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A12 Upper DWord

OAG_OAPERF_A12_UPPER - OAG Aggregate Perf Counter A12 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D9E4h

This register enables the current live value of performance counter A12 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A13

OAG_OAPERF_A13 - OAG Aggregate Perf Counter A13		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0D9E8h	
This register reflects the count value of the OA Performance counter A13. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A13 Upper DWord

OAG_OAPERF_A13_UPPER - OAG Aggregate Perf Counter A13 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D9ECh

This register enables the current live value of performance counter A13 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A14

OAG_OAPERF_A14 - OAG Aggregate Perf Counter A14		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0D9F0h	
This register reflects the count value of the OA Performance counter A14. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A14 Upper DWord

OAG_OAPERF_A14_UPPER - OAG Aggregate Perf Counter A14 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D9F4h

This register enables the current live value of performance counter A14 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A15

OAG_OAPERF_A15 - OAG Aggregate Perf Counter A15		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0D9F8h	
This register reflects the count value of the OA Performance counter A15. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A15 Upper DWord

OAG_OAPERF_A15_UPPER - OAG Aggregate Perf Counter A15 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D9FCh

This register enables the current live value of performance counter A15 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A16

OAG_OAPERF_A16 - OAG Aggregate Perf Counter A16		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DA00h	
This register reflects the count value of the OA Performance counter A16. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A16 Upper DWord

OAG_OAPERF_A16_UPPER - OAG Aggregate Perf Counter A16 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DA04h

This register enables the current live value of performance counter A16 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A17

OAG_OAPERF_A17 - OAG Aggregate Perf Counter A17		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DA08h	
This register reflects the count value of the OA Performance counter A17. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A17 Upper DWord

OAG_OAPERF_A17_UPPER - OAG Aggregate Perf Counter A17 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DA0Ch

This register enables the current live value of performance counter A17 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A18

OAG_OAPERF_A18 - OAG Aggregate Perf Counter A18		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DA10h	
This register reflects the count value of the OA Performance counter A9. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A18 Upper DWord

OAG_OAPERF_A18_UPPER - OAG Aggregate Perf Counter A18 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DA14h

This register enables the current live value of performance counter A18 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A19

OAG_OAPERF_A19 - OAG Aggregate Perf Counter A19		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DA18h	
This register reflects the count value of the OA Performance counter A19. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A19 Lower DWord Free

OAG_OAPERF_A19_LOWER_FREE - OAG Aggregate Perf Counter A19 Lower DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DAE4h

This register counts the same event as counter A19 however is not affected by context ID or other conditions that prevent A19 from incrementing. This counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted used this register.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A19 Upper DWord

OAG_OAPERF_A19_UPPER - OAG Aggregate Perf Counter A19 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DA1Ch

This register enables the current live value of performance counter A19 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A19 Upper DWord Free

OAG_OAPERF_A19_UPPER_FREE - OAG Aggregate Perf Counter A19 Upper DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DAE8h

This register counts the same event as counter A19 however is not affected by context ID or other conditions that prevent A19 from incrementing. This counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted using this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A20

OAG_OAPERF_A20 - OAG Aggregate Perf Counter A20		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DA20h	
This register reflects the count value of the OA Performance counter A20. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A20 Lower DWord Free

OAG_OAPERF_A20_LOWER_FREE - OAG Aggregate Perf Counter A20 Lower DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DAECh

This register counts the same event as counter A20 however is not affected by context ID or other conditions that prevent A20 from incrementing. This counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted used this register.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A20 Upper DWord

OAG_OAPERF_A20_UPPER - OAG Aggregate Perf Counter A20 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DA24h

This register enables the current live value of performance counter A20 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A20 Upper DWord Free

OAG_OAPERF_A20_UPPER_FREE - OAG Aggregate Perf Counter A20 Upper DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DAF0h

This register counts the same event as counter A20 however is not affected by context ID or other conditions that prevent A20 from incrementing. his counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted using this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A21

OAG_OAPERF_A21 - OAG Aggregate Perf Counter A21		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DA28h	
This register reflects the count value of the OA Performance counter A21. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A21 Upper DWord

OAG_OAPERF_A21_UPPER - OAG Aggregate Perf Counter A21 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DA2Ch

This register enables the current live value of performance counter A21 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A22

OAG_OAPERF_A22 - OAG Aggregate Perf Counter A22		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DA30h	
This register reflects the count value of the OA Performance counter A22. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A22 Upper DWord

OAG_OAPERF_A22_UPPER - OAG Aggregate Perf Counter A22 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DA34h

This register enables the current live value of performance counter A22 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A23

OAG_OAPERF_A23 - OAG Aggregate Perf Counter A23		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DA38h	
This register reflects the count value of the OA Performance counter A23. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A23 Upper DWord

OAG_OAPERF_A23_UPPER - OAG Aggregate Perf Counter A23 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DA3Ch

This register enables the current live value of performance counter A23 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A24

OAG_OAPERF_A24 - OAG Aggregate Perf Counter A24		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DA40h	
This register reflects the count value of the OA Performance counter A24. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A24 Upper DWord

OAG_OAPERF_A24_UPPER - OAG Aggregate Perf Counter A24 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DA44h

This register enables the current live value of performance counter A24 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A25

OAG_OAPERF_A25 - OAG Aggregate Perf Counter A25		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DA48h	
This register reflects the count value of the OA Performance counter A25. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A25 Upper DWord

OAG_OAPERF_A25_UPPER - OAG Aggregate Perf Counter A25 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DA4Ch

This register enables the current live value of performance counter A25 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A26

OAG_OAPERF_A26 - OAG Aggregate Perf Counter A26		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DA50h	
This register reflects the count value of the OA Performance counter A26. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A26 Upper DWord

OAG_OAPERF_A26_UPPER - OAG Aggregate Perf Counter A26 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DA54h

This register enables the current live value of performance counter A26 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A27

OAG_OAPERF_A27 - OAG Aggregate Perf Counter A27		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DA58h	
This register reflects the count value of the OA Performance counter A27. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A27 Upper DWord

OAG_OAPERF_A27_UPPER - OAG Aggregate Perf Counter A27 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DA5Ch

This register enables the current live value of performance counter A27 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A28

OAG_OAPERF_A28 - OAG Aggregate Perf Counter A28		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DA60h	
This register reflects the count value of the OA Performance counter A28. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A28 Upper DWord

OAG_OAPERF_A28_UPPER - OAG Aggregate Perf Counter A28 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DA64h

This register enables the current live value of performance counter A28 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A29

OAG_OAPERF_A29 - OAG Aggregate Perf Counter A29		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DA68h	
This register reflects the count value of the OA Performance counter A29. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A29 Upper DWord

OAG_OAPERF_A29_UPPER - OAG Aggregate Perf Counter A29 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DA6Ch

This register enables the current live value of performance counter A29 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A30

OAG_OAPERF_A30 - OAG Aggregate Perf Counter A30		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DA70h	
This register reflects the count value of the OA Performance counter A30. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A30 Upper DWord

OAG_OAPERF_A30_UPPER - OAG Aggregate Perf Counter A30 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DA74h

This register enables the current live value of performance counter A30 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A31

OAG_OAPERF_A31 - OAG Aggregate Perf Counter A31		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DA78h	
This register reflects the count value of the OA Performance counter A31		
DWord	**Bit**	**Description**
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A31 Upper DWord

OAG_OAPERF_A31_UPPER - OAG Aggregate Perf Counter A31 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DA7Ch

This register enables the current live value of performance counter A31 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A32

OAG_OAPERF_A32 - OAG Aggregate Perf Counter A32		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DA80h	
This register reflects the count value of the OA Performance counter A32		
DWord	**Bit**	**Description**
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A33

OAG_OAPERF_A33 - OAG Aggregate Perf Counter A33		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DA84h	
This register reflects the count value of the OA Performance counter A33		
DWord	**Bit**	**Description**
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A34

OAG_OAPERF_A34 - OAG Aggregate Perf Counter A34		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DA88h	
This register reflects the count value of the OA Performance counter A34		
DWord	**Bit**	**Description**
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter A35

OAG_OAPERF_A35 - OAG Aggregate Perf Counter A35		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DA8Ch	
This register reflects the count value of the OA Performance counter A35		
DWord	**Bit**	**Description**
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter SPM0 Lower DWord Free

OAG_OAPERF_SPM0_LOWER_FREE - OAG Aggregate Perf Counter SPM0 Lower DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB30h

This register counts the same event as counter SPM0 however is not affected by context ID or other conditions that prevent SPM0 from incrementing. This counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted used this register.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter SPM0 Upper DWord Free

OAG_OAPERF_SPM0_UPPER_FREE - OAG Aggregate Perf Counter SPM0 Upper DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB34h

This register counts the same event as counter SPM0 however is not affected by context ID or other conditions that prevent SPM0 from incrementing. This counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted using this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter SPM1 Lower DWord Free

OAG_OAPERF_SPM1_LOWER_FREE - OAG Aggregate Perf Counter SPM1 Lower DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB38h

This register counts the same event as counter SPM1 however is not affected by context ID or other conditions that prevent SPM1 from incrementing. This counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted used this register.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter SPM1 Upper DWord Free

OAG_OAPERF_SPM1_UPPER_FREE - OAG Aggregate Perf Counter SPM1 Upper DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB3Ch

This register counts the same event as counter SPM1 however is not affected by context ID or other conditions that prevent SPM1 from incrementing. This counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted using this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter SPM2 Lower DWord Free

OAG_OAPERF_SPM2_LOWER_FREE - OAG Aggregate Perf Counter SPM2 Lower DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB40h

This register counts the same event as counter SPM2 however is not affected by context ID or other conditions that prevent SPM2 from incrementing. This counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted used this register.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter SPM2 Upper DWord Free

OAG_OAPERF_SPM2_UPPER_FREE - OAG Aggregate Perf Counter SPM2 Upper DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB44h

This register counts the same event as counter SPM2 however is not affected by context ID or other conditions that prevent SPM2 from incrementing. This counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted using this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter SPM3 Lower DWord Free

OAG_OAPERF_SPM3_LOWER_FREE - OAG Aggregate Perf Counter SPM3 Lower DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB48h

This register counts the same event as counter SPM3 however is not affected by context ID or other conditions that prevent SPM3 from incrementing. This counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted used this register.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter SPM3 Upper DWord Free

OAG_OAPERF_SPM3_UPPER_FREE - OAG Aggregate Perf Counter SPM3 Upper DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB4Ch

This register counts the same event as counter SPM3 however is not affected by context ID or other conditions that prevent SPM3 from incrementing. This counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted using this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter SPM4 Lower DWord Free

OAG_OAPERF_SPM4_LOWER_FREE - OAG Aggregate Perf Counter SPM4 Lower DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB50h

This register counts the same event as counter SPM4 however is not affected by context ID or other conditions that prevent SPM4 from incrementing. This counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted used this register.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter SPM4 Upper DWord Free

OAG_OAPERF_SPM4_UPPER_FREE - OAG Aggregate Perf Counter SPM4 Upper DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB54h

This register counts the same event as counter SPM4 however is not affected by context ID or other conditions that prevent SPM4 from incrementing. This counter is free running, always enabled and counting irrespective of OA enabled or disabled. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted using this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter SPM5 Lower DWord Free

OAG_OAPERF_SPM5_LOWER_FREE - OAG Aggregate Perf Counter SPM5 Lower DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB58h

Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted used this register. This counter is free running, always enabled and counts irrespective of OA enable configuration.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Aggregate Perf Counter SPM5 Upper DWord Free

OAG_OAPERF_SPM5_UPPER_FREE - OAG Aggregate Perf Counter SPM5 Upper DWord Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB5Ch

Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is counted using this register. This counter is free running, always enabled and counts irrespective of OA enable configuration.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Any Media Fixed Function 0 Wake Count Free

OAG_ANY_MEDIA_FF0_WAKE_COUNT_FREE - OAG Any Media Fixed Function 0 Wake Count Free	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DBA8h

This register counts the number of media fixed function wake events for Media FF0. This counter is free running, always enabled and counts irrespective of OA being enabled or disabled.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Any Media Fixed Function 1 Wake Count Free

OAG_ANY_MEDIA_FF1_WAKE_COUNT_FREE - OAG Any Media Fixed Function 1 Wake Count Free	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DBB8h
This register counts the number of media fixed function wake events. This counter is free running, always enabled and counting irrespective of OA enabled or disabled.	

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Any Media Fixed Function Busy Free

OAG_ANY_MEDIA_FF_BUSY_FREE - OAG Any Media Fixed Function Busy Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DBA0h

This register counts the time that any media fixed function is busy. This counter is free running, always enabled and counts irrespective of OA being enabled or disabled.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG BLT Busy Free

OAG_BLT_BUSY_FREE - OAG BLT Busy Free		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DBBCh	
This register counts the time that BLT engine is busy. This counter is free running, always enabled and counting irrespective of OA enabled or disabled.		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG BLT Wake Count Free

OAG_BLT_WAKE_COUNT_FREE - OAG BLT Wake Count Free		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DBC0h	
This register counts the number of times that BLT wakes up. This counter is free running, always enabled and counting irrespective of OA enabled or disabled.		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Boolean Counter B0

OAG_OAPERF_B0 - OAG Boolean Counter B0	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DA94h

This register enables the current live value of performance counter B0 to be read. Since what conditions cause B0 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Boolean Counter B1

OAG_OAPERF_B1 - OAG Boolean Counter B1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DA98h

This register enables the current live value of performance counter B1 to be read. Since what conditions cause B1 to increment are defined by the programming of CEC1-0/CEC1-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Boolean Counter B2

OAG_OAPERF_B2 - OAG Boolean Counter B2	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DA9Ch

This register enables the current live value of performance counter B2 to be read. Since what conditions cause B2 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Boolean Counter B3

OAG_OAPERF_B3 - OAG Boolean Counter B3	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DAA0h

This register enables the current live value of performance counter B3 to be read. Since what conditions cause B3 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Boolean Counter B4

OAG_OAPERF_B4 - OAG Boolean Counter B4	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DAA4h

This register enables the current live value of performance counter B4 to be read. Since what conditions cause B4 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Boolean Counter B5

OAG_OAPERF_B5 - OAG Boolean Counter B5	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DAA8h

This register enables the current live value of performance counter B5 to be read. Since what conditions cause B5 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Boolean Counter B6

OAG_OAPERF_B6 - OAG Boolean Counter B6	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DAACh

This register enables the current live value of performance counter B6 to be read. Since what conditions cause B6 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Boolean Counter B7

OAG_OAPERF_B7 - OAG Boolean Counter B7	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DAB0h

This register enables the current live value of performance counter B7 to be read. Since what conditions cause B7 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Context Control

OAG_OAGLBCTXCTRL - OAG Context Control

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02B28h

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: PBC _Custom_GTIReset: DEV
	7:2	**Timer Period** Default Value: 00b Access: R/W Specifies the period of the timer strobe as a function of the minimum TIME_STAMP resolution. The period is determined by selecting a specified bit from the TIME_STAMP register as follows: StrobePeriod = MinimumTimeStampPeriod * 2 **(TimerPeriod + 1)** The exponent is defined by this field. **Programming Notes** **Note:** The TIME_STAMP is not reset at start time so the phase of the strobe is not synchronized with the enable of the OA unit. This could result in approximately a full StrobePeriod elapsing prior to the first trigger. Usage for this mechanism should be time based periodic triggering, typically.
	1	**Timer Enable** Access: R/W This field enables the timer logic to output a periodic strobe, as defined by the Timer Period. When disabled the timer output is not asserted. Users should be aware that the timer-based report logic and the Boolean report trigger logic are logically OR'd together with no buffering of report triggers. This implies that only one performance counter report will be generated for clocks where both the timer-based report logic and the Boolean report trigger logic trigger a performance counter report. **Value** / **Name** / **Description** 0 / Disable **[Default]** / Counter does not get written out on regular interval 1 / Enable / Counter gets written out on regular intervals, defined by the Timer Period **Programming Notes** When "Specific Context Enable" bit is set to '1' in OACONTROL register, Timer based report trigger function gets enabled only for the selected context.

OAG_OAGLBCTXCTRL - OAG Context Control

		When "Specific Context Enable" bit is set to '0' in OACONTROL register, Timer based report trigger function gets enabled for all contexts.
	0	**Counter Stop-Resume Mechanism** Access: R/W Counter stop-resume mechanism

Value	Name	Description
1	Resume	Resume counting for all counters

OAG Customizable Event Creation 0-0

OAG_CEC00 - OAG Customizable Event Creation 0-0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D940h

Description
This register is used to define custom counter event 0, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.
This register can be used to customize counters for events from both unslice and slice units.

DWord	Bit	Description
0	31:21	**Negate** Format: U11 _Custom_GTIReset: DEV The 11-bit array defined in this field allows input bus bits [10:0] to be individually negated in order to facilitate more complicated custom event creation (e.g. A & (!B | !C)). Note that LSB of this field affects bit 0 of the selected input bus. 0b – Pass-through – Input bit is passed through to comparator as is 1b – Negated – Input bit is negated before passing to comparator
	20:19	**Source Select** Format: U2 _Custom_GTIReset: DEV Selects the input signals to the Boolean event definition logic (see block diagram in the Custom Event Counters section). 01b – Prev Event – Selects the conditioned/flopped input from the last CEC block as the input bus to CEC0 block 11b – Reserved –
	18:3	**Compare Value** Format: U16 _Custom_GTIReset: DEV The value in this field is compared the 16-bit conditioned input bus that are fed into the comparator(see block diagram in the Custom Event Counters section). The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the custom event is asserted. This signal in turn can be counted by the B0 performance counter or fed into other CEC blocks.

OAG_CEC00 - OAG Customizable Event Creation 0-0

2:0 **Compare Function**

Format:	U3
_Custom_GTIReset:	DEV

This field selects the function used by the CEC0 comparator when comparing the compare value to the value active on the CEC0 conditioned input bus (see block diagram in the Custom Event Counters section).

Value	Name	Description
000b	Any Are Equal	Compare and assert output if any are equal (Can be used as OR function)
001b	Greater Than	Compare and assert output if greater than
010b	Equal	Compare and assert output if equal to (Can also be used as AND function)
011b	Greater Than or Equal	Compare and assert output if greater than or equal
100b	Less Than	Compare and assert output if less than
101b	Not Equal	Compare and assert output if not equal
110b	Less Than or Equal	Compare and assert output if less than or equal
111b	Reserved	

OAG Customizable Event Creation 1-0

OAG_CEC10 - OAG Customizable Event Creation 1-0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D948h

Description
This register is used to define custom counter event 1, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.
This register can be used to customize counters for events from both unslice and slice units.

DWord	Bit	Description
0	31:21	**Negate** Format: U11 _Custom_GTIReset: DEV The 11-bit array defined in this field allows input bus bits [10:0] to be individually negated in order to facilitate more complicated custom event creation (e.g. A & (!B | !C)). Note that LSB of this field affects bit 0 of the selected input bus. 0b – Pass-through – Input bit is passed through to comparator as is 1b – Negated – Input bit is negated before passing to comparator
	20:19	**Source Select** Format: U2 _Custom_GTIReset: DEV Selects the input signals to the Boolean event definition logic (see block diagram in the Custom Event Counters section). 01b – Prev Event – Selects the conditioned/flopped input from the previous CEC block as the input bus to this CEC block 11b – Reserved –
	18:3	**Compare Value** Format: U16 _Custom_GTIReset: DEV The value in this field is compared the 16-bit conditioned input bus that are fed into the comparator(see block diagram in the Custom Event Counters section). The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the custom event is asserted. This signal in turn can be counted by the B0 performance counter or fed into other CEC blocks.

OAG_CEC10 - OAG Customizable Event Creation 1-0

2:0 **Compare Function**

Format:	U3
_Custom_GTIReset:	DEV

This field selects the function used by the CEC comparator when comparing the compare value to the value active on the CEC conditioned input bus (see block diagram in the Custom Event Counters section).

Value	Name	Description
000b	Any Are Equal	Compare and assert output if any are equal (Can be used as OR function)
001b	Greater Than	Compare and assert output if greater than
010b	Equal	Compare and assert output if equal to (Can also be used as AND function)
011b	Greater Than or Equal	Compare and assert output if greater than or equal
100b	Less Than	Compare and assert output if less than
101b	Not Equal	Compare and assert output if not equal
110b	Less Than or Equal	Compare and assert output if less than or equal
111b	Reserved	

OAG Customizable Event Creation 1-1

OAG_CEC11 - OAG Customizable Event Creation 1-1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D94Ch

Description
This register configures the input conditioning portion of CEC (custom event creation) block 1, bit definitions in this register refer to the CEC block diagram.
This register can be used to customize counters for events from both unslice and slice units.

DWord	Bit	Description
0	31:16	**Considerations** _Custom_GTIReset: DEV This 16-bit field allows individual bits of the bus selected as the input to CEC block to be delayed by 1 clock relative to the non-delayed bits in the bus (see block diagram in the Custom Event Counters section). See CEC0-1 register definition for an example use case. Value / Name / Description 0b / Live / Input bit is not delayed by 1 clock before event calculation 1b / Delayed / Input bit is delayed by 1 clock before event calculation
	15:0	**Mask** _Custom_GTIReset: DEV This 16-bit field allows individual input bits to be ignored in custom event calculation. See block diagram in the Custom Event Counters section for more details on where this field is used. Value / Name / Description 0b / Unmasked / Input bit is considered in event calculation 1b / Masked / Input bit is ignored in event calculation

OAG Customizable Event Creation 2-0

OAG_CEC20 - OAG Customizable Event Creation 2-0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D950h

Description
This register is used to define custom counter event 2, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.
This register can be used to customize counters for events from both unslice and slice units.

DWord	Bit	Description
0	31:21	**Negate** Format: U11 _Custom_GTIReset: DEV The 11-bit array defined in this field allows input bus bits [10:0] to be individually negated in order to facilitate more complicated custom event creation (e.g. A & (!B | !C)). Note that LSB of this field affects bit 0 of the selected input bus. 0b – Pass-through – Input bit is passed through to comparator as is 1b – Negated – Input bit is negated before passing to comparator
	20:19	**Source Select** Format: U2 _Custom_GTIReset: DEV Selects the input signals to the Boolean event definition logic (see block diagram in the Custom Event Counters section). 01b – Prev Event – Selects the conditioned/flopped input from the previous CEC block as the input bus to this CEC block 11b – Reserved –
	18:3	**Compare Value** Format: U16 _Custom_GTIReset: DEV The value in this field is compared the 16-bit conditioned input bus that are fed into the comparator(see block diagram in the Custom Event Counters section). The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the custom event is asserted. This signal in turn can be counted by the B0 performance counter or fed into other CEC blocks.

OAG_CEC20 - OAG Customizable Event Creation 2-0

	2:0	**Compare Function** Format: U3 _Custom_GTIReset: DEV This field selects the function used by the CEC comparator when comparing the compare value to the value active on the CEC conditioned input bus (see block diagram in the Custom Event Counters section).

Value	Name	Description
000b	Any Are Equal	Compare and assert output if any are equal (Can be used as OR function)
001b	Greater Than	Compare and assert output if greater than
010b	Equal	Compare and assert output if equal to (Can also be used as AND function)
011b	Greater Than or Equal	Compare and assert output if greater than or equal
100b	Less Than	Compare and assert output if less than
101b	Not Equal	Compare and assert output if not equal
110b	Less Than or Equal	Compare and assert output if less than or equal
111b	Reserved	

OAG Customizable Event Creation 2-1

OAG_CEC21 - OAG Customizable Event Creation 2-1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D954h

Description
This register configures the input conditioning portion of CEC (custom event creation) block 2, bit definitions in this register refer to the CEC block diagram.
This register can be used to customize counters for events from both unslice and slice units.

DWord	Bit	Description
0	31:16	**Considerations** _Custom_GTIReset: DEV This 16-bit field allows individual bits of the bus selected as the input to CEC block to be delayed by 1 clock relative to the non-delayed bits in the bus (see block diagram in the Custom Event Counters section). See CEC0-1 register definition for an example use case. Value / Name / Description 0b / Live / Input bit is not delayed by 1 clock before event calculation 1b / Delayed / Input bit is delayed by 1 clock before event calculation
	15:0	**Mask** _Custom_GTIReset: DEV This 16-bit field allows individual input bits to be ignored in custom event calculation. See block diagram in the Custom Event Counters section for more details on where this field is used. Value / Name / Description 0b / Unmasked / Input bit is considered in event calculation 1b / Masked / Input bit is ignored in event calculation

OAG Customizable Event Creation 3-0

OAG_CEC30 - OAG Customizable Event Creation 3-0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D958h

Description
This register is used to define custom counter event 3, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.
This register can be used to customize counters for events from both unslice and slice units.

DWord	Bit	Description
0	31:21	**Negate** Format: U11 _Custom_GTIReset: DEV The 11-bit array defined in this field allows input bus bits [10:0] to be individually negated in order to facilitate more complicated custom event creation (e.g. A & (!B | !C)). Note that LSB of this field affects bit 0 of the selected input bus. 0b – Pass-through – Input bit is passed through to comparator as is 1b – Negated – Input bit is negated before passing to comparator
	20:19	**Source Select** Format: U2 _Custom_GTIReset: DEV Selects the input signals to the Boolean event definition logic (see block diagram in the Custom Event Counters section). 01b – Prev Event – Selects the conditioned/flopped input from the previous CEC block as the input bus to this CEC block 11b – Reserved –
	18:3	**Compare Value** Format: U16 _Custom_GTIReset: DEV The value in this field is compared the 16-bit conditioned input bus that are fed into the comparator(see block diagram in the Custom Event Counters section). The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the custom event is asserted. This signal in turn can be counted by the B0 performance counter or fed into other CEC blocks.

OAG_CEC30 - OAG Customizable Event Creation 3-0

2:0 **Compare Function**

Format:	U3
_Custom_GTIReset:	DEV

This field selects the function used by the CEC comparator when comparing the compare value to the value active on the CEC conditioned input bus (see block diagram in the Custom Event Counters section).

Value	Name	Description
000b	Any Are Equal	Compare and assert output if any are equal (Can be used as OR function)
001b	Greater Than	Compare and assert output if greater than
010b	Equal	Compare and assert output if equal to (Can also be used as AND function)
011b	Greater Than or Equal	Compare and assert output if greater than or equal
100b	Less Than	Compare and assert output if less than
101b	Not Equal	Compare and assert output if not equal
110b	Less Than or Equal	Compare and assert output if less than or equal
111b	Reserved	

OAG Customizable Event Creation 3-1

OAG_CEC31 - OAG Customizable Event Creation 3-1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D95Ch

Description
This register configures the input conditioning portion of CEC (custom event creation) block 3, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.
This register can be used to customize counters for events from both unslice and slice units.

DWord	Bit	Description
0	31:16	**Considerations** _Custom_GTIReset: DEV This 16-bit field allows individual bits of the bus selected as the input to CEC block to be delayed by 1 clock relative to the non-delayed bits in the bus (see block diagram in the Custom Event Counters section). See CEC0-1 register definition for an example use case. Value / Name / Description 0b / Live / Input bit is not delayed by 1 clock before event calculation 1b / Delayed / Input bit is delayed by 1 clock before event calculation
	15:0	**Mask** _Custom_GTIReset: DEV This 16-bit field allows individual input bits to be ignored in custom event calculation. See block diagram in the Custom Event Counters section for more details on where this field is used. Value / Name / Description 0b / Unmasked / Input bit is considered in event calculation 1b / Masked / Input bit is ignored in event calculation

OAG Customizable Event Creation 4-0

OAG_CEC40 - OAG Customizable Event Creation 4-0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D960h

Description
This register is used to define custom counter event 4, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.
This register can be used to customize counters for events from both unslice and slice units.

DWord	Bit	Description
0	31:21	**Negate** Format: U11 _Custom_GTIReset: DEV The 11-bit array defined in this field allows input bus bits [10:0] to be individually negated in order to facilitate more complicated custom event creation (e.g. A & (!B | !C)). Note that LSB of this field affects bit 0 of the selected input bus. 0b – Pass-through – Input bit is passed through to comparator as is 1b – Negated – Input bit is negated before passing to comparator
	20:19	**Source Select** Format: U2 _Custom_GTIReset: DEV Selects the input signals to the Boolean event definition logic (see block diagram in the Custom Event Counters section). 01b – Prev Event – Selects the conditioned/flopped input from the last CEC block as the input bus to CEC0 block 11b – Reserved –
	18:3	**Compare Value** Format: U16 _Custom_GTIReset: DEV The value in this field is compared the 16-bit conditioned input bus that are fed into the comparator(see block diagram in the Custom Event Counters section). The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the custom event is asserted. This signal in turn can be counted by the B0 performance counter or fed into other CEC blocks.

OAG_CEC40 - OAG Customizable Event Creation 4-0

2:0 **Compare Function**

Format:	U3
_Custom_GTIReset:	DEV

This field selects the function used by the CEC comparator when comparing the compare value to the value active on the CEC conditioned input bus (see block diagram in the Custom Event Counters section).

Value	Name	Description
000b	Any Are Equal	Compare and assert output if any are equal (Can be used as OR function)
001b	Greater Than	Compare and assert output if greater than
010b	Equal	Compare and assert output if equal to (Can also be used as AND function)
011b	Greater Than or Equal	Compare and assert output if greater than or equal
100b	Less Than	Compare and assert output if less than
101b	Not Equal	Compare and assert output if not equal
110b	Less Than or Equal	Compare and assert output if less than or equal
111b	Reserved	

OAG Customizable Event Creation 5-0

OAG_CEC50 - OAG Customizable Event Creation 5-0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D968h

Description
This register is used to define custom counter event 5, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.
This register can be used to customize counters for events from both unslice and slice units.

DWord	Bit	Description
0	31:21	**Negate** Format: U11 _Custom_GTIReset: DEV The 11-bit array defined in this field allows input bus bits [10:0] to be individually negated in order to facilitate more complicated custom event creation (e.g. A & (!B | !C)). Note that LSB of this field affects bit 0 of the selected input bus. 0b – Pass-through – Input bit is passed through to comparator as is 1b – Negated – Input bit is negated before passing to comparator
	20:19	**Source Select** Format: U2 _Custom_GTIReset: DEV Selects the input signals to the Boolean event definition logic (see block diagram in the Custom Event Counters section). 01b – Prev Event – Selects the conditioned/flopped input from the previous CEC block as the input bus to this CEC block 11b – Reserved –
	18:3	**Compare Value** Format: U16 _Custom_GTIReset: DEV The value in this field is compared the 16-bit conditioned input bus that are fed into the comparator(see block diagram in the Custom Event Counters section). The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the custom event is asserted. This signal in turn can be counted by the B0 performance counter or fed into other CEC blocks.

OAG_CEC50 - OAG Customizable Event Creation 5-0

2:0 **Compare Function**

Format:	U3
_Custom_GTIReset:	DEV

This field selects the function used by the CEC comparator when comparing the compare value to the value active on the CEC conditioned input bus (see block diagram in the Custom Event Counters section).

Value	Name	Description
000b	Any Are Equal	Compare and assert output if any are equal (Can be used as OR function)
001b	Greater Than	Compare and assert output if greater than
010b	Equal	Compare and assert output if equal to (Can also be used as AND function)
011b	Greater Than or Equal	Compare and assert output if greater than or equal
100b	Less Than	Compare and assert output if less than
101b	Not Equal	Compare and assert output if not equal
110b	Less Than or Equal	Compare and assert output if less than or equal
111b	Reserved	

OAG Customizable Event Creation 5-1

OAG_CEC51 - OAG Customizable Event Creation 5-1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D96Ch

Description
This register configures the input conditioning portion of CEC (custom event creation) block 5, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.
This register can be used to customize counters for events from both unslice and slice units.

DWord	Bit	Description
0	31:16	**Considerations** _Custom_GTIReset: DEV This 16-bit field allows individual bits of the bus selected as the input to CEC block to be delayed by 1 clock relative to the non-delayed bits in the bus (see block diagram in the Custom Event Counters section). See CEC0-1 register definition for an example use case. Value / Name / Description 0b / Live / Input bit is not delayed by 1 clock before event calculation 1b / Delayed / Input bit is delayed by 1 clock before event calculation
	15:0	**Mask** _Custom_GTIReset: DEV This 16-bit field allows individual input bits to be ignored in custom event calculation. See block diagram in the Custom Event Counters section for more details on where this field is used. Value / Name / Description 0b / Unmasked / Input bit is considered in event calculation 1b / Masked / Input bit is ignored in event calculation

OAG Customizable Event Creation 6-0

OAG_CEC60 - OAG Customizable Event Creation 6-0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D970h

Description
This register is used to define custom counter event 6, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.
This register can be used to customize counters for events from both unslice and slice units.

DWord	Bit	Description
0	31:21	**Negate** Format: U11 _Custom_GTIReset: DEV The 11-bit array defined in this field allows input bus bits [10:0] to be individually negated in order to facilitate more complicated custom event creation (e.g. A & (!B | !C)). Note that LSB of this field affects bit 0 of the selected input bus. **Value** / **Name** / **Description** / **Programming Notes** 0b / Pass-through / Input bit is passed through to comparator as is / 1b / Negated / Input bit is negated before passing to comparator /
	20:19	**Source Select** Format: U2 _Custom_GTIReset: DEV Selects the input signals to the Boolean event definition logic (see block diagram in the Custom Event Counters section). **Value** / **Name** / **Description** 01b / Prev Event / Selects the conditioned/flopped input from the previous CEC block as the input bus to this CEC block 11b / Reserved /
	18:3	**Compare Value** Format: U16 _Custom_GTIReset: DEV The value in this field is compared the 16-bit conditioned input bus that are fed into the comparator(see block diagram in the Custom Event Counters section). The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the custom event is asserted. This signal in turn can be counted by the B0 performance counter or fed into other CEC blocks.

OAG_CEC60 - OAG Customizable Event Creation 6-0

2:0 **Compare Function**

Format:	U3
_Custom_GTIReset:	DEV

This field selects the function used by the CEC comparator when comparing the compare value to the value active on the CEC conditioned input bus (see block diagram in the Custom Event Counters section).

Value	Name	Description
000b	Any Are Equal	Compare and assert output if any are equal (Can be used as OR function)
001b	Greater Than	Compare and assert output if greater than
010b	Equal	Compare and assert output if equal to (Can also be used as AND function)
011b	Greater Than or Equal	Compare and assert output if greater than or equal
100b	Less Than	Compare and assert output if less than
101b	Not Equal	Compare and assert output if not equal
110b	Less Than or Equal	Compare and assert output if less than or equal
111b	Reserved	

OAG Customizable Event Creation 6-1

OAG_CEC61 - OAG Customizable Event Creation 6-1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D974h

Description
This register configures the input conditioning portion of CEC (custom event creation) block 6, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.
This register can be used to customize counters for events from both unslice and slice units.

DWord	Bit	Description
0	31:16	**Considerations** _Custom_GTIReset: DEV This 16-bit field allows individual bits of the bus selected as the input to CEC block to be delayed by 1 clock relative to the non-delayed bits in the bus (see block diagram in the Custom Event Counters section). See CEC0-1 register definition for an example use case. Value / Name / Description 0b / Live / Input bit is not delayed by 1 clock before event calculation 1b / Delayed / Input bit is delayed by 1 clock before event calculation
	15:0	**Mask** _Custom_GTIReset: DEV This 16-bit field allows individual input bits to be ignored in custom event calculation. See block diagram in the Custom Event Counters section for more details on where this field is used. Value / Name / Description 0b / Unmasked / Input bit is considered in event calculation 1b / Masked / Input bit is ignored in event calculation

OAG Customizable Event Creation 7-0

OAG_CEC70 - OAG Customizable Event Creation 7-0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D978h

Description
This register is used to define custom counter event 7, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.
This register can be used to customize counters for events from both unslice and slice units.

DWord	Bit	Description
0	31:21	**Negate** Format: U11 _Custom_GTIReset: DEV The 11-bit array defined in this field allows input bus bits [10:0] to be individually negated in order to facilitate more complicated custom event creation (e.g. A & (!B | !C)). Note that LSB of this field affects bit 0 of the selected input bus. 0b – Pass-through – Input bit is passed through to comparator as is 1b – Negated – Input bit is negated before passing to comparator
	20:19	**Source Select** Format: U2 _Custom_GTIReset: DEV Selects the input signals to the Boolean event definition logic (see block diagram in the Custom Event Counters section). 01b – Prev Event – Selects the conditioned/flopped input from the previous CEC block as the input bus to this CEC block 11b – Reserved –
	18:3	**Compare Value** Format: U16 _Custom_GTIReset: DEV The value in this field is compared the 16-bit conditioned input bus that are fed into the comparator(see block diagram in the Custom Event Counters section). The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the custom event is asserted. This signal in turn can be counted by the B0 performance counter or fed into other CEC blocks.

OAG_CEC70 - OAG Customizable Event Creation 7-0

2:0 **Compare Function**

Format:	U3
_Custom_GTIReset:	DEV

This field selects the function used by the CEC comparator when comparing the compare value to the value active on the CEC conditioned input bus (see block diagram in the Custom Event Counters section).

Value	Name	Description
000b	Any Are Equal	Compare and assert output if any are equal (Can be used as OR function)
001b	Greater Than	Compare and assert output if greater than
010b	Equal	Compare and assert output if equal to (Can also be used as AND function)
011b	Greater Than or Equal	Compare and assert output if greater than or equal
100b	Less Than	Compare and assert output if less than
101b	Not Equal	Compare and assert output if not equal
110b	Less Than or Equal	Compare and assert output if less than or equal
111b	Reserved	

OAG Customizable Event Creation 7-1

OAG_CEC71 - OAG Customizable Event Creation 7-1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D97Ch

Description
This register configures the input conditioning portion of CEC (custom event creation) block 3, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.
This register can be used to customize counters for events from both unslice and slice units.

DWord	Bit	Description
0	31:16	**Considerations** _Custom_GTIReset: DEV This 16-bit field allows individual bits of the bus selected as the input to CEC block to be delayed by 1 clock relative to the non-delayed bits in the bus (see block diagram in the Custom Event Counters section). See CEC0-1 register definition for an example use case. 0b – Live – Input bit is not delayed by 1 clock before event calculation 1b – Delayed – Input bit is delayed by 1 clock before event calculation
	15:0	**Mask** _Custom_GTIReset: DEV This 16-bit field allows individual input bits to be ignored in custom event calculation. See block diagram in the Custom Event Counters section for more details on where this field is used. 0b – Unmasked – Input bit is considered in event calculation 1b – Masked – Input bit is ignored in event calculation

OAG GPU Ticks Counter

OAG_GPU_TICKS - OAG GPU Ticks Counter		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DA90h	
Reading this register returns the live value of the GPU ticks counter that is sampled and included in the performance counter report header. It increments every GFX clock and hence the number of increments per second changes with GFX frequency.		
DWord	**Bit**	**Description**
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the GPU tick counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG GPU Ticks Counter Upper DWord

OAG_GPU_TICKS_UPPER - OAG GPU Ticks Counter Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

Reading this register returns the live value of the Upper DWord of GPU ticks counter that is sampled and included in the performance counter report header. It increments every GFX clock and hence the number of increments per second changes with GFX frequency.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32 of the GPU tick counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG GTI Busy Free

OAG_GTI_BUSY_FREE - OAG GTI Busy Free		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DBCCh	
This register counts the time that any engine residing in GTI(currently WG, BLT, or GuC) is busy. This counter is free running, always enabled and counting irrespective of OA enabled or disabled.		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG GTI Wake Count Free

OAG_GTI_WAKE_COUNT_FREE - OAG GTI Wake Count Free	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DBD0h

This register counts the number of times that GTI wakes up due to one of the engines resident in GTI (currently WG, BLT, and GuC) becoming busy. This counter is free running, always enabled and counting irrespective of OA enabled or disabled.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG GuC Busy Free

OAG_GUC_BUSY_FREE - OAG GuC Busy Free		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DBC4h	
This register counts the time that GuC is busy. This counter is free running, always enabled and counting irrespective of OA enabled or disabled.		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG GuC Wake Count Free

OAG_GUC_WAKE_COUNT_FREE - OAG GuC Wake Count Free	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DBC8h

This register counts the number of times that GuC wakes up. This counter is free running, always enabled and counting irrespective of OA enabled or disabled.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Interrupt Mask Register

OAG_OA_IMR - OAG Interrupt Mask Register

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB14h

The OAIMR register is used by software to control whether OA generates an interrupt or not.

DWord	Bit	Description
0	31:29	**Reserved** Default Value: 7h Format: PBC _Custom_GTIReset: DEV
	28	**Mask Bit** _Custom_GTIReset: DEV Value / Name / Description 0h / Not Masked / May generate an interrupt 1h / Masked **[Default]** / Will not generate an interrupt
	27:0	**Reserved** Default Value: FFFFFFFh Format: PBC _Custom_GTIReset: DEV

OAG MCPG0 Hysteresis Time Free

OAG_MCPG0_HYSTERESIS_TIME_FREE - OAG MCPG0 Hysteresis Time Free	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DBA4h
This register counts the time that MCPG hysteresis time is accumulating for media slice 0.This counter is free running, always enabled and counting irrespective of OA enabled or disabled.	

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Media Sampler Really Busy Free

OAG_MEDIA_SAMPLER_REALLY_BUSY_FREE - OAG Media Sampler Really Busy Free	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DBACh
This register counts the time that media sampler is really busy. This counter is free running, always enabled and counting irrespective of OA enabled or disabled.	

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Media Sampler Wake Count Free

OAG_MEDIA_SAMPLER_WAKE_COUNT_FREE - OAG Media Sampler Wake Count Free	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DBB0h

This register counts the number of times that media sampler wakes up. This counter is free running, always enabled and counting irrespective of OA enabled or disabled.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Observation Architecture Buffer

OAG_OABUFFER - OAG Observation Architecture Buffer

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB08h

This register is used to program the OA unit.

Programming Notes

This MMIO must be set before the OATAILPTR register and set after the OAHEADPTR register. This is to enable proper functionality of the overflow bit.

DWord	Bit	Description
0	31:6	**Report Buffer Offset** _Custom_GTIReset: DEV This field specifies 64B aligned GFX MEM address where the chap counter values are reported.
	5:3	**Inter Trigger Report Buffer Size** _Custom_GTIReset: DEV This field indicates the size of report buffer for time/event-based report trigger mechanisms.

Value	Name	Description
0h	128 KB or 1MB **[Default]**	All context considered.
1h	256 KB or 2MB	
2h	512 KB or 4MB	
3h	1 MB or 8MB	
4h	2 MB or 16MB	
5h	4 MB or 32MB	
6h	8 MB or 64MB	
7h	16 MB or 128MB	

	2	**OA Report Trigger Select** _Custom_GTIReset: DEV

Value	Name	Description
0		Level Report trigger
1		Edge Report trigger

	1	**Disable Overrun Mode** Format: Enable _Custom_GTIReset: DEV This field defines the mode of reporting for internal trigger/timer based reporting. When this bit is set, overrun does not lose reports but stops reporting. Based on the head and tail pointer, when

HW detects room for the report, it would resume reporting to the buffer. This mode would not set the over-run bit in the register. When this mode bit is reset, buffer overrun can happen and lose the reports while setting the buffer over-run bit.

Value	Name	Description
0h	Disable **[Default]**	Counter gets written out on regular intervals, defined by the Timer Period
1h	Enable	Counter does not get written out on regular interval

0 **Memory Select PPGTT/GGTT Access**

_Custom_GTIReset:	DEV

Value	Name
0h	PPGTT
1h	GGTT **[Default]**

Programming Notes
When each context has its own Per Process GTT, this field should be always set to GGTT.

OAG Observation Architecture Control

OAG_OACONTROL - OAG Observation Architecture Control

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DAF4h

This register controls global OA functionality, report format, interrupt steering and context filtering.

DWord	Bit	Description
0	31:18	**Reserved** Format: PBC _Custom_GTIReset: DEV
	17:7	**Reserved** Access: RO Format: MBZ
	6	**OA Context control select** _Custom_GTIReset: DEV **Description**: This field is obsolete Bit field to indicate which context control inputs are to be chosen in the OACS **Value 0**: OAG RCS Select **[Default]** — Select the OA context control from RCS (Default) **Value 1**: Global — Select the Global OA context control
	5	**Interrupt Steering Bit** _Custom_GTIReset: DEV When set, OACS unit sends interrupt messages to the SHIM of the GUC through message channel. When reset, OACS unit sends the interrupt message to Display Engine as config writes on GAM interface.
	4:2	**Counter Select** _Custom_GTIReset: DEV This field selects which performance counter report format to use, please refer to Performance Counter Report Formats section for more details on the structure of the format.
	1	**Reserved** Access: RO Format: MBZ

OAG_OACONTROL - OAG Observation Architecture Control

	0	**Performance Counter Enable** Format: Enable _Custom_GTIReset: DEV Global performance counter enable. If clear, no counting will occur. MI_REPORT_PERF_COUNT is undefined when clear. **Programming Notes** When this bit is set, OABUFFER, OAHEADPTR and OATAILPTR must be programmed correctly to ensure report triggers due to Context Switch and GO transition happen correctly.

OAG Observation Architecture Report Trigger 2

OAG_OAREPORTTRIG2 - OAG Observation Architecture Report Trigger 2

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D924h

This register controls some of the Boolean logic defining Boolean/threshold report trigger 0. The bit definitions in this register refer to the stages in the report trigger block diagram in the Performance Counter Reporting section.

Report triggers generated from OAREPORTTRIG 1-4 and OAREPORTTRIG 5-8 are ORed to form a new report trigger. Report trigger behavior can be derived by programming these two sets of OA REPORT registers with the same value. Users should be aware that while programming Timer based and Threshold Counter based triggers simultaneously for internal reporting, they should be programmed such way that they are not consecutively triggered. If programmed simultaneously, RTL pulse detection logic will have problem when these triggers occur in consecutive clock cycles.

DWord	Bit	Description
0	31	**Report Trigger Enable** Format: Enable _Custom_GTIReset: DEV Enable Boolean/threshold report trigger 0. Users should be aware that the timer-based report logic and the Boolean report trigger logic are logically OR'd together with no buffering of report triggers. This implies that only one performance counter report will be generated for clocks where both the timer-based report logic and the Boolean report trigger logic trigger a performance counter report. When "Specific Context Enable" bit set to '1' in OACONTROL register, Boolean/Threshold report trigger function gets enabled only for the selected context. When "Specific Context Enable" bit set to '0' in OACONTROL register, Boolean/Threshold report trigger function gets enabled for all contexts.
	30:24	**Reserved** Format: PBC _Custom_GTIReset: DEV
	23	**Threshold Enable** Format: Enable _Custom_GTIReset: DEV Enable the threshold compare logic within the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).

OAG_OAREPORTTRIG2 - OAG Observation Architecture Report Trigger 2

	22	**Invert D Enable 0** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the D stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).
	21	**Invert C Enable 1** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the C stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).
	20	**Invert C Enable 0** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the C stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).
	19	**Invert B Enable 3** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the B stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).
	18	**Invert B Enable 2** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the B stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).
	17	**Invert B Enable 1** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the B stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).
	16	**Invert B Enable 0** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the B stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).
	15	**Invert A Enable 15** Format: Enable _Custom_GTIReset: DEV

OAG_OAREPORTTRIG2 - OAG Observation Architecture Report Trigger 2

		Invert the specified signal at the A stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).
	14	**Invert A Enable 14** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).
	13	**Invert A Enable 13** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).
	12	**Invert A Enable 12** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).
	11	**Invert A Enable 11** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).
	10	**Invert A Enable 10** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).
	9	**Invert A Enable 9** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).
	8	**Invert A Enable 8** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).

OAG_OAREPORTTRIG2 - OAG Observation Architecture Report Trigger 2

	7	**Invert A Enable 7** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).
	6	**Invert A Enable 6** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).
	5	**Invert A Enable 5** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).
	4	**Invert A Enable 4** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).
	3	**Invert A Enable 3** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).
	2	**Invert A Enable 2** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).
	1	**Invert A Enable 1** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).

OAG_OAREPORTTRIG2 - OAG Observation Architecture Report Trigger 2

	0	**Invert A Enable 0** Format: Enable _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 0 logic (see block diagram in the Performance Counter Reporting section).

OAG Observation Architecture Report Trigger 6

OAG_OAREPORTTRIG6 - OAG Observation Architecture Report Trigger 6

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D934h

This register controls some of the Boolean logic defining Boolean/threshold report trigger 1. Note that Boolean report triggers 0 and 1 are logically OR'd together without buffering, this implies that only one performance counter report will be generated for clocks where both Boolean report triggers evaluate true. The bit definitions in this register refer to the stages in the report trigger block diagram in the Performance Counter Reporting section.

Report triggers generated from OAREPORTTRIG 1-4 and OAREPORTTRIG 5-8 are ORed to form a new report trigger. Report trigger behavior can be derived by programming these two sets of OA REPORT registers with the same value. Users should be aware that while programming Timer based and Threshold Counter based triggers simultaneously for internal reporting, they should be programmed such way that they are not consecutively triggered. If programmed simultaneously, RTL pulse detection logic will have problem when these triggers occur in consecutive clock cycles.

DWord	Bit	Description
0	31	**Report Trigger Enable** _Custom_GTIReset: DEV Enable Boolean/threshold report trigger 0. Users should be aware that the timer-based report logic and the Boolean report trigger logic are logically OR'd together with no buffering of report triggers. This implies that only one performance counter report will be generated for clocks where both the timer-based report logic and the Boolean report trigger logic trigger a performance counter report. When "Specific Context Enable" bit set to '1' in OACONTROL register, Boolean/Threshold report trigger function gets enabled only for the selected context. When "Specific Context Enable" bit set to '0' in OACONTROL register, Boolean/Threshold report trigger function gets enabled for all contexts.
	30:24	**Reserved** Format: PBC _Custom_GTIReset: DEV
	23	**Threshold Enable** _Custom_GTIReset: DEV Enable the threshold compare logic within the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).
	22	**Invert D Enable 0** _Custom_GTIReset: DEV

OAG_OAREPORTTRIG6 - OAG Observation Architecture Report Trigger 6

		Invert the specified signal at the D stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).
	21	**Invert C Enable 1** _Custom_GTIReset: DEV Invert the specified signal at the C stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).
	20	**Invert C Enable 0** _Custom_GTIReset: DEV Invert the specified signal at the C stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).
	19	**Invert B Enable 3** _Custom_GTIReset: DEV Invert the specified signal at the B stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).
	18	**Invert B Enable 2** _Custom_GTIReset: DEV Invert the specified signal at the B stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).
	17	**Invert B Enable 1** _Custom_GTIReset: DEV Invert the specified signal at the B stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).
	16	**Invert B Enable 0** _Custom_GTIReset: DEV Invert the specified signal at the B stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).
	15	**Invert A Enable 15** _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).
	14	**Invert A Enable 14** _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).
	13	**Invert A Enable 13** _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).

OAG_OAREPORTTRIG6 - OAG Observation Architecture Report Trigger 6

	12	**Invert A Enable 12** _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).
	11	**Invert A Enable 11** _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).
	10	**Invert A Enable 10** _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).
	9	**Invert A Enable 9** _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).
	8	**Invert A Enable 8** _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).
	7	**Invert A Enable 7** _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).
	6	**Invert A Enable 6** _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).
	5	**Invert A Enable 5** _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).
	4	**Invert A Enable 4** _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).
	3	**Invert A Enable 3** _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 1 logic (see

OAG_OAREPORTTRIG6 - OAG Observation Architecture Report Trigger 6

		block diagram in the Performance Counter Reporting section).
	2	**Invert A Enable 2** _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).
	1	**Invert A Enable 1** _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).
	0	**Invert A Enable 0** _Custom_GTIReset: DEV Invert the specified signal at the A stage of the Boolean/threshold report trigger 1 logic (see block diagram in the Performance Counter Reporting section).

OAG Observation Architecture Report Trigger Counter

OAG_OARPTTRIG_COUNTER - OAG Observation Architecture Report Trigger Counter

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB10h

This register provides status of report trigger threshold count 1 and 2. This register is for HW internal purpose and power context save/restored. This register must not be programmed by SW.

DWord	Bit	Description
0	31:16	**Report Trig Threshold Count 1 Status** _Custom_GTIReset: DEV **Programming Notes** This field is for HW internal use to context save/restore rpt trigger threshold count 1. It always indicates current value of HW's internal report trigger count. SW should not program these bits.
	15:0	**Report Trig Threshold count 2 status** _Custom_GTIReset: DEV **Programming Notes** This field is for HW internal use to context save/restore rpt trigger threshold count 2. It always indicates current value of HW's internal report trigger count. SW should not program these bits.

OAG Observation Architecture Start Trigger 5

OAG_OASTARTTRIG5 - OAG Observation Architecture Start Trigger 5

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0D910h

This register provides the threshold value optionally used to define the start trigger for B7-B4 counters. Note that the value in this register must match the value in OASTARTTRIG1 to have B7-B0 start at the same time. The bit definition in this register refers to the stages in the start trigger block diagram in the Performance Counter Reporting section.

DWord	Bit	Description
0	31:16	**Reserved** Format: PBC _Custom_GTIReset: DEV
	15:0	**Threshold Value** Format: U16 _Custom_GTIReset: DEV **Programming Notes** Threshold value for the compare logic within the start trigger logic for B7-B4 counters.

OAG Observation Architecture Start Trigger Counter

OAG_OASTARTTRIG_COUNTER - OAG Observation Architecture Start Trigger Counter

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB0Ch

This register provides status of start trigger threshold count 1 and 2. This register is for HW internal purpose.

DWord	Bit	Description
0	31:16	**Start Trig Threshold Count 1 Status** _Custom_GTIReset: DEV **Programming Notes** This field is for HW internal use to context save/restore start trigger threshold count 1. It always indicates current value of HW's internal start trigger count. SW should not program these bits.
	15:0	**Start Trig Threshold count 2 status** _Custom_GTIReset: DEV **Programming Notes** : This field is for HW internal use to context save/restore start trigger threshold count 2. It always indicates current value of HW's internal start trigger count. SW should not program these bits.

OAG Observation Architecture Status Register

OAG_OASTATUS - OAG Observation Architecture Status Register

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DAFCh

This register provides status of report buffer and overflow conditions as well as status of some flags which hardware needs for internal purpose. Software should not program these status bits. Status bits are active high flags (0=false, 1=true)

DWord	Bit	Description
0	31:22	**Reserved** Format: PBC _Custom_GTIReset: DEV
	21	**Start Trigger Flag 1** _Custom_GTIReset: DEV Value / Name: 0 **[Default]**; 1 Programming Notes: This bit is for HW internal use to context save /restore Start Trigger 1 occurrence On RC6 entry. Software should not program this bit.
	20	**Start Trigger Flag 2** _Custom_GTIReset: DEV Value / Name: 0 **[Default]**; 1 Programming Notes: This bit is for HW internal use to context save /restore Start Trigger 2 occurrence On RC6 entry. Software should not program this bit.
	19	**Report Trigger Flag 1** _Custom_GTIReset: DEV Value / Name: 0 **[Default]**

OAG_OASTATUS - OAG Observation Architecture Status Register

1	

Programming Notes

This bit is for HW internal use to context save /restore Report Trigger 1 occurrence On RC6 entry. Software should not program this bit.

18 **Report Trigger Flag 2**

_Custom_GTIReset:	DEV

Value	Name
0	**[Default]**
1	

Programming Notes

This bit is for HW internal use to context save /restore Report Trigger 2 occurrence On RC6 entry. Software should not program this bit.

17 **Tail Pointer Wrap Flag**

Format:	U1
_Custom_GTIReset:	DEV

Value	Name
0	
1	**[Default]**

Programming Notes

This bit is for HW internal use to context save /restore Tail Pointer Wrap Flag. SW should not program this bit. This bit gets programmed only when Tail Pointer Wrap Mask bit is set.

16 **Head Pointer Wrap Flag**

Format:	U1
_Custom_GTIReset:	DEV

Value	Name
0	
1	**[Default]**

Programming Notes

This bit is for HW internal use to context save /restore Head Pointer Wrap Flag. SW should not program this bit. This bit gets programmed only when Head Pointer Wrap Mask bit is set.

OAG_OASTATUS - OAG Observation Architecture Status Register

	15:7	**Reserved** Format: PBC _Custom_GTIReset: DEV
	6	**Reserved** Access: RO Format: MBZ
	5	**Reserved** Format: PBC _Custom_GTIReset: DEV
	4	**Accumulator Overflow** _Custom_GTIReset: DEV This field indicates that the one or more event accumulator inside the slice-OAunit has overflowed. Once set, this bit will remain set, until SW resets it by either soft reset or writing a 0 to it.

Value	Name	Description
0	**[Default]**	No overflow has occurred.
1		Overflow has occurred.

	3	**Overrun Status** _Custom_GTIReset: DEV This bit is read only and writing to this bit will have no effect. This bit will reflect the status of overrun irrespective of Overun Mode enabled or disabled.

Value	Name	Description
0	**[Default]**	No overrun has occurred
1		Overrun has occurred

	2	**Counter Overflow** _Custom_GTIReset: DEV This bit is set if any of the counters overflows. This bit can be reset by SW by either soft reset or writing a 0 to it.

Value	Name	Description
0	**[Default]**	Counter Overflow Not Occurred
1		Counter Overflow Occurred

	1	**Buffer Overflow** _Custom_GTIReset: DEV This bit is set when the Tail-pointer - Head pointer > max internal trigger buffer size

Value	Name	Description
0h	**[Default]**	Buffer Overflow Not Occurred
1		Buffer Overflow Occurred

OAG_OASTATUS - OAG Observation Architecture Status Register

	0	**Report Lost Error** Format: Enable _Custom_GTIReset: DEV This bit is set if the Report Trigger due to "Internal Report Trigger-1", "Internal Report Trigger-2" or "Timer Triggered" or "MMIO Trigger" to write out the counter values is dropped, while there is an ongoing report in progress. The report request is ignored and the counter continue to count. This bit will be set along with this MMIO Trigger Queue Full, in case of dropped request for MMIO trigger.

Value	Name	Description
0	**[Default]**	Report Lost Error Not Occurred
1		Report Lost Error Occurred

Programming Notes
This bit can be reset by SW by either soft reset or writing a 0 to it.

OAG Observation Architecture Tail Pointer

OAG_OATAILPTR - OAG Observation Architecture Tail Pointer

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB04h

This register allows software to program tail pointer and also indicates current tail pointer value.

DWord	Bit	Description
0	31:6	**Tail Pointer** _Custom_GTIReset: DEV Virtual address of the internal trigger based buffer that is updated for every 64B cacheline write to memory when reporting via internal report trigger. This pointer will not be updated for MI_REPORT_PERF_COUNT command based writes. **Programming Notes** Before enabling internally triggered performance counter reporting, SW must ensure that this address matches the Report Buffer Offset programmed in OABUFFER register (i.e. tail pointer must start at the beginning of the report buffer). SW must ensure that Tail pointer and the Head Pointer match before enabling internally triggered performance counter reporting.
	5:0	**Reserved** Format: PBC _Custom_GTIReset: DEV

OAG RC0 Any Engine Busy Free

OAG_RC0_ANY_ENGINE_BUSY_FREE - OAG RC0 Any Engine Busy Free	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB80h

This register counts the time that any engine is truly busy (not simply powered up). This counter is free running, always enabled and counting irrespective of OA enabled or disabled.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads. Unit in GPMusec. Refer to **Timestamp Bases** for details.

OAG RC0 Clocks Free

OAG_RC0_CLOCKS_FREE - OAG RC0 Clocks Free	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB90h

This register counts the clocks that GFX has been in RC0 in increments of 64 GFX clocks. This counter is free running, always enabled and counting irrespective of OA enabled or disabled.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG RC0 Hysteresis Free

OAG_RC0_HYSTERISIS_FREE - OAG RC0 Hysteresis Free	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	0DB84h

This register counts the time that any engine is truly busy (not simply powered up). This counter is free running, always enabled and counting irrespective of OA enabled or disabled.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG RC0 Wake Count Free

OAG_RC0_WAKE_COUNT_FREE - OAG RC0 Wake Count Free		
Register Space:	MMIO: 0/2/0	
Access:	RO	
Size (in bits):	32	
Address:	0DB88h	
This counter is free running, always enabled and counting irrespective of OA enabled or disabled.		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG RC6 Entry Count Free

OAG_RC6_ENTRY_COUNT_FREE - OAG RC6 Entry Count Free	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB9Ch

This register counts number rc6 entries. This counter is free running, always enabled and counting irrespective of OA enabled or disabled. This is an RPM mirrored register.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG RC6 Pre Engine Wake Overhead Free

OAG_RC6_PRE_ENGINE_WAKE_OVERHEAD_FREE - OAG RC6 Pre Engine Wake Overhead Free	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	0DB8Ch
This register counts the time between boot complete and Go=1. This counter is free running, always enabled and counting irrespective of OA enabled or disabled.	

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG RCPG Hysteresis Time Free

OAG_RCPG_HYSTERESIS_TIME_FREE - OAG RCPG Hysteresis Time Free

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DBE0h

This register counts the time RCPG hysteresis count is accumulating. This counter is free running, always enabled and counting irrespective of OA enabled or disabled.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Render Busy Free

OAG_RENDER_BUSY_FREE - OAG Render Busy Free	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DBDCh

This register counts the time that any render engine is busy. This counter is free running, always enabled and counting irrespective of OA enabled or disabled.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Render Wake Count Free

OAG_RENDER_WAKE_COUNT_FREE - OAG Render Wake Count Free	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DBE4h

This register counts the number of render wakes that have occurred. This counter is free running, always enabled and counting irrespective of OA enabled or disabled.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Slice Customizable Event Creation 0-0

OAG_SCEC0_0 - OAG Slice Customizable Event Creation 0-0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DC00h

This register is used to define the slice custom counter event 0, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.

DWord	Bit	Description
0	31:21	**Negate** Format: U11 _Custom_GTIReset: DEV The 11-bit array defined in this field allows input bus bits [10:0] to be individually negated in order to facilitate more complicated custom event creation (e.g. A & (!B | !C)). Note that LSB of this field affects bit 0 of the selected input bus. **Value** / **Name** / **Description** 0b / Pass-through / Input bit is passed through to comparator as is 1b / Negated / Input bit is negated before passing to comparator
	20:19	**Source Select** Format: U2 _Custom_GTIReset: DEV Selects the input signals to the Boolean event definition logic (see block diagram in the Custom Event Counters section). **Value** / **Name** / **Description** 01b / Prev Event / Selects the conditioned/flopped input from the last CEC block as the input bus to CEC0 block 11b / Reserved /
	18:3	**Compare Value** Format: U16 _Custom_GTIReset: DEV The value in this field is compared the 16-bit conditioned input bus that are fed into the comparator(see block diagram in the Custom Event Counters section). The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the custom event is asserted. This signal in turn can be counted by the B0 performance counter or fed into other CEC blocks.
	2:0	**Compare Function** Format: U3 _Custom_GTIReset: DEV This field selects the function used by the CEC0 comparator when comparing the compare value

OAG_SCEC0_0 - OAG Slice Customizable Event Creation 0-0

to the value active on the CEC0 conditioned input bus (see block diagram in the Custom Event Counters section).

Value	Name	Description
000b	Any Are Equal	Compare and assert output if any are equal (Can be used as OR function)
001b	Greater Than	Compare and assert output if greater than
010b	Equal	Compare and assert output if equal to (Can also be used as AND function)
011b	Greater Than or Equal	Compare and assert output if greater than or equal
100b	Less Than	Compare and assert output if less than
101b	Not Equal	Compare and assert output if not equal
110b	Less Than or Equal	Compare and assert output if less than or equal
111b	Reserved	

OAG Slice Customizable Event Creation 1-0

OAG_SCEC1_0 - OAG Slice Customizable Event Creation 1-0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DC08h

This register is used to define slice custom counter event 1, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.

DWord	Bit	Description
0	31:21	**Negate** Format: U11 _Custom_GTIReset: DEV The 11-bit array defined in this field allows input bus bits [10:0] to be individually negated in order to facilitate more complicated custom event creation (e.g. A & (!B | !C)). Note that LSB of this field affects bit 0 of the selected input bus. **Value** / **Name** / **Description** 0b / Pass-through / Input bit is passed through to comparator as is 1b / Negated / Input bit is negated before passing to comparator
	20:19	**Source Select** Format: U2 _Custom_GTIReset: DEV Selects the input signals to the Boolean event definition logic (see block diagram in the Custom Event Counters section). **Value** / **Name** / **Description** 01b / Prev Event / Selects the conditioned/flopped input from the previous CEC block as the input bus to this CEC block 11b / Reserved /
	18:3	**Compare Value** Format: U16 _Custom_GTIReset: DEV The value in this field is compared the 16-bit conditioned input bus that are fed into the comparator(see block diagram in the Custom Event Counters section). The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the custom event is asserted. This signal in turn can be counted by the B0 performance counter or fed into other CEC blocks.
	2:0	**Compare Function** Format: U3 _Custom_GTIReset: DEV This field selects the function used by the CEC comparator when comparing the compare value

OAG_SCEC1_0 - OAG Slice Customizable Event Creation 1-0

to the value active on the CEC conditioned input bus (see block diagram in the Custom Event Counters section).

Value	Name	Description
000b	Any Are Equal	Compare and assert output if any are equal (Can be used as OR function)
001b	Greater Than	Compare and assert output if greater than
010b	Equal	Compare and assert output if equal to (Can also be used as AND function)
011b	Greater Than or Equal	Compare and assert output if greater than or equal
100b	Less Than	Compare and assert output if less than
101b	Not Equal	Compare and assert output if not equal
110b	Less Than or Equal	Compare and assert output if less than or equal
111b	Reserved	

OAG Slice Customizable Event Creation 1-1

OAG_SCEC1_1 - OAG Slice Customizable Event Creation 1-1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DC0Ch

This register configures the input conditioning portion of slice CEC (custom event creation) block 1, bit definitions in this register refer to the CEC block diagram.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:16</td><td>Considerations
<table>
<tr><td>_Custom_GTIReset:</td><td>DEV</td></tr>
</table>
This 16-bit field allows individual bits of the bus selected as the input to CEC block to be delayed by 1 clock relative to the non-delayed bits in the bus (see block diagram in the Custom Event Counters section). See CEC0-1 register definition for an example use case.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0b</td><td>Live</td><td>Input bit is not delayed by 1 clock before event calculation</td></tr>
<tr><td>1b</td><td>Delayed</td><td>Input bit is delayed by 1 clock before event calculation</td></tr>
</table></td></tr>
<tr><td></td><td>15:0</td><td>Mask
<table>
<tr><td>_Custom_GTIReset:</td><td>DEV</td></tr>
</table>
This 16-bit field allows individual input bits to be ignored in custom event calculation. See block diagram in the Custom Event Counters section for more details on where this field is used.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0b</td><td>Unmasked</td><td>Input bit is considered in event calculation</td></tr>
<tr><td>1b</td><td>Masked</td><td>Input bit is ignored in event calculation</td></tr>
</table></td></tr>
</table>

OAG Slice Customizable Event Creation 2-0

OAG_SCEC2_0 - OAG Slice Customizable Event Creation 2-0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DC10h

This register is used to define custom counter event 2, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.

DWord	Bit	Description
0	31:21	**Negate** Format: U11 _Custom_GTIReset: DEV The 11-bit array defined in this field allows input bus bits [10:0] to be individually negated in order to facilitate more complicated custom event creation (e.g. A & (!B | !C)). Note that LSB of this field affects bit 0 of the selected input bus. Value / Name / Description: 0b / Pass-through / Input bit is passed through to comparator as is 1b / Negated / Input bit is negated before passing to comparator
	20:19	**Source Select** Format: U2 _Custom_GTIReset: DEV Selects the input signals to the Boolean event definition logic (see block diagram in the Custom Event Counters section). Value / Name / Description: 01b / Prev Event / Selects the conditioned/flopped input from the previous CEC block as the input bus to this CEC block 11b / Reserved /
	18:3	**Compare Value** Format: U16 _Custom_GTIReset: DEV The value in this field is compared the 16-bit conditioned input bus that are fed into the comparator(see block diagram in the Custom Event Counters section). The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the custom event is asserted. This signal in turn can be counted by the B0 performance counter or fed into other CEC blocks.
	2:0	**Compare Function** Format: U3 _Custom_GTIReset: DEV This field selects the function used by the CEC comparator when comparing the compare value

OAG_SCEC2_0 - OAG Slice Customizable Event Creation 2-0

to the value active on the CEC conditioned input bus (see block diagram in the Custom Event Counters section).

Value	Name	Description
000b	Any Are Equal	Compare and assert output if any are equal (Can be used as OR function)
001b	Greater Than	Compare and assert output if greater than
010b	Equal	Compare and assert output if equal to (Can also be used as AND function)
011b	Greater Than or Equal	Compare and assert output if greater than or equal
100b	Less Than	Compare and assert output if less than
101b	Not Equal	Compare and assert output if not equal
110b	Less Than or Equal	Compare and assert output if less than or equal
111b	Reserved	

OAG Slice Customizable Event Creation 2-1

OAG_SCEC2_1 - OAG Slice Customizable Event Creation 2-1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DC14h

This register configures the input conditioning portion of slice CEC (custom event creation) block 2, bit definitions in this register refer to the CEC block diagram.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:16</td><td>Considerations
<table>
<tr><td>_Custom_GTIReset:</td><td>DEV</td></tr>
</table>
This 16-bit field allows individual bits of the bus selected as the input to CEC block to be delayed by 1 clock relative to the non-delayed bits in the bus (see block diagram in the Custom Event Counters section). See CEC0-1 register definition for an example use case.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0b</td><td>Live</td><td>Input bit is not delayed by 1 clock before event calculation</td></tr>
<tr><td>1b</td><td>Delayed</td><td>Input bit is delayed by 1 clock before event calculation</td></tr>
</table></td></tr>
<tr><td></td><td>15:0</td><td>Mask
<table>
<tr><td>_Custom_GTIReset:</td><td>DEV</td></tr>
</table>
This 16-bit field allows individual input bits to be ignored in custom event calculation. See block diagram in the Custom Event Counters section for more details on where this field is used.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0b</td><td>Unmasked</td><td>Input bit is considered in event calculation</td></tr>
<tr><td>1b</td><td>Masked</td><td>Input bit is ignored in event calculation</td></tr>
</table></td></tr>
</table>

OAG Slice Customizable Event Creation 3-0

OAG_SCEC3_0 - OAG Slice Customizable Event Creation 3-0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DC18h

This register is used to define slice custom counter event 3, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.

DWord	Bit	Description
0	31:21	**Negate** Format: U11 _Custom_GTIReset: DEV The 11-bit array defined in this field allows input bus bits [10:0] to be individually negated in order to facilitate more complicated custom event creation (e.g. A & (!B | !C)). Note that LSB of this field affects bit 0 of the selected input bus. **Value** / **Name** / **Description** 0b / Pass-through / Input bit is passed through to comparator as is 1b / Negated / Input bit is negated before passing to comparator
	20:19	**Source Select** Format: U2 _Custom_GTIReset: DEV Selects the input signals to the Boolean event definition logic (see block diagram in the Custom Event Counters section). **Value** / **Name** / **Description** 01b / Prev Event / Selects the conditioned/flopped input from the previous CEC block as the input bus to this CEC block 11b / Reserved /
	18:3	**Compare Value** Format: U16 _Custom_GTIReset: DEV The value in this field is compared the 16-bit conditioned input bus that are fed into the comparator(see block diagram in the Custom Event Counters section). The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the custom event is asserted. This signal in turn can be counted by the B0 performance counter or fed into other CEC blocks.
	2:0	**Compare Function** Format: U3 _Custom_GTIReset: DEV This field selects the function used by the CEC comparator when comparing the compare value

OAG_SCEC3_0 - OAG Slice Customizable Event Creation 3-0

to the value active on the CEC conditioned input bus (see block diagram in the Custom Event Counters section).

Value	Name	Description
000b	Any Are Equal	Compare and assert output if any are equal (Can be used as OR function)
001b	Greater Than	Compare and assert output if greater than
010b	Equal	Compare and assert output if equal to (Can also be used as AND function)
011b	Greater Than or Equal	Compare and assert output if greater than or equal
100b	Less Than	Compare and assert output if less than
101b	Not Equal	Compare and assert output if not equal
110b	Less Than or Equal	Compare and assert output if less than or equal
111b	Reserved	

OAG Slice Customizable Event Creation 3-1

OAG_SCEC3_1 - OAG Slice Customizable Event Creation 3-1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DC1Ch

This register configures the input conditioning portion of Slice CEC (custom event creation) block 3, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.

DWord	Bit	Description
0	31:16	**Considerations** _Custom_GTIReset: DEV This 16-bit field allows individual bits of the bus selected as the input to CEC block to be delayed by 1 clock relative to the non-delayed bits in the bus (see block diagram in the Custom Event Counters section). See CEC0-1 register definition for an example use case. Value / Name / Description 0b / Live / Input bit is not delayed by 1 clock before event calculation 1b / Delayed / Input bit is delayed by 1 clock before event calculation
	15:0	**Mask** _Custom_GTIReset: DEV This 16-bit field allows individual input bits to be ignored in custom event calculation. See block diagram in the Custom Event Counters section for more details on where this field is used. Value / Name / Description 0b / Unmasked / Input bit is considered in event calculation 1b / Masked / Input bit is ignored in event calculation

OAG Slice Customizable Event Creation 4-0

OAG_SCEC4_0 - OAG Slice Customizable Event Creation 4-0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DC20h

This register is used to define slice custom counter event 4, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.

DWord	Bit	Description
0	31:21	**Negate** Format: U11 _Custom_GTIReset: DEV The 11-bit array defined in this field allows input bus bits [10:0] to be individually negated in order to facilitate more complicated custom event creation (e.g. A & (!B | !C)). Note that LSB of this field affects bit 0 of the selected input bus. Value / Name / Description: 0b / Pass-through / Input bit is passed through to comparator as is 1b / Negated / Input bit is negated before passing to comparator
	20:19	**Source Select** Format: U2 _Custom_GTIReset: DEV Selects the input signals to the Boolean event definition logic (see block diagram in the Custom Event Counters section). Value / Name / Description: 01b / Prev Event / Selects the conditioned/flopped input from the last CEC block as the input bus to CEC0 block 11b / Reserved /
	18:3	**Compare Value** Format: U16 _Custom_GTIReset: DEV The value in this field is compared the 16-bit conditioned input bus that are fed into the comparator(see block diagram in the Custom Event Counters section). The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the custom event is asserted. This signal in turn can be counted by the B0 performance counter or fed into other CEC blocks.
	2:0	**Compare Function** Format: U3 _Custom_GTIReset: DEV This field selects the function used by the CEC comparator when comparing the compare value

OAG_SCEC4_0 - OAG Slice Customizable Event Creation 4-0

to the value active on the CEC conditioned input bus (see block diagram in the Custom Event Counters section).

Value	Name	Description
000b	Any Are Equal	Compare and assert output if any are equal (Can be used as OR function)
001b	Greater Than	Compare and assert output if greater than
010b	Equal	Compare and assert output if equal to (Can also be used as AND function)
011b	Greater Than or Equal	Compare and assert output if greater than or equal
100b	Less Than	Compare and assert output if less than
101b	Not Equal	Compare and assert output if not equal
110b	Less Than or Equal	Compare and assert output if less than or equal
111b	Reserved	

OAG Slice Customizable Event Creation 5-0

OAG_SCEC5_0 - OAG Slice Customizable Event Creation 5-0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DC28h

This register is used to define slice custom counter event 5, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.

DWord	Bit	Description
0	31:21	**Negate** Format: U11 _Custom_GTIReset: DEV The 11-bit array defined in this field allows input bus bits [10:0] to be individually negated in order to facilitate more complicated custom event creation (e.g. A & (!B | !C)). Note that LSB of this field affects bit 0 of the selected input bus. Value / Name / Description: 0b / Pass-through / Input bit is passed through to comparator as is 1b / Negated / Input bit is negated before passing to comparator
	20:19	**Source Select** Format: U2 _Custom_GTIReset: DEV Selects the input signals to the Boolean event definition logic (see block diagram in the Custom Event Counters section). Value / Name / Description: 01b / Prev Event / Selects the conditioned/flopped input from the previous CEC block as the input bus to this CEC block 11b / Reserved /
	18:3	**Compare Value** Format: U16 _Custom_GTIReset: DEV The value in this field is compared the 16-bit conditioned input bus that are fed into the comparator(see block diagram in the Custom Event Counters section). The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the custom event is asserted. This signal in turn can be counted by the B0 performance counter or fed into other CEC blocks.
	2:0	**Compare Function** Format: U3 _Custom_GTIReset: DEV This field selects the function used by the CEC comparator when comparing the compare value

OAG_SCEC5_0 - OAG Slice Customizable Event Creation 5-0

to the value active on the CEC conditioned input bus (see block diagram in the Custom Event Counters section).

Value	Name	Description
000b	Any Are Equal	Compare and assert output if any are equal (Can be used as OR function)
001b	Greater Than	Compare and assert output if greater than
010b	Equal	Compare and assert output if equal to (Can also be used as AND function)
011b	Greater Than or Equal	Compare and assert output if greater than or equal
100b	Less Than	Compare and assert output if less than
101b	Not Equal	Compare and assert output if not equal
110b	Less Than or Equal	Compare and assert output if less than or equal
111b	Reserved	

OAG Slice Customizable Event Creation 5-1

OAG_SCEC5_1 - OAG Slice Customizable Event Creation 5-1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DC2Ch

This register configures the input conditioning portion of CEC (custom event creation) block 5, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.

DWord	Bit	Description
0	31:16	**Considerations** _Custom_GTIReset: DEV This 16-bit field allows individual bits of the bus selected as the input to CEC block to be delayed by 1 clock relative to the non-delayed bits in the bus (see block diagram in the Custom Event Counters section). See CEC0-1 register definition for an example use case. Value / Name / Description 0b / Live / Input bit is not delayed by 1 clock before event calculation 1b / Delayed / Input bit is delayed by 1 clock before event calculation
	15:0	**Mask** _Custom_GTIReset: DEV This 16-bit field allows individual input bits to be ignored in custom event calculation. See block diagram in the Custom Event Counters section for more details on where this field is used. Value / Name / Description 0b / Unmasked / Input bit is considered in event calculation 1b / Masked / Input bit is ignored in event calculation

OAG Slice Customizable Event Creation 6-0

OAG_SCEC6_0 - OAG Slice Customizable Event Creation 6-0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DC30h

This register is used to define slice custom counter event 6, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.

DWord	Bit	Description
0	31:21	**Negate** Format: U11 _Custom_GTIReset: DEV The 11-bit array defined in this field allows input bus bits [10:0] to be individually negated in order to facilitate more complicated custom event creation (e.g. A & (!B | !C)). Note that LSB of this field affects bit 0 of the selected input bus. **Value** / **Name** / **Description** 0b / Pass-through / Input bit is passed through to comparator as is 1b / Negated / Input bit is negated before passing to comparator
	20:19	**Source Select** Format: U2 _Custom_GTIReset: DEV Selects the input signals to the Boolean event definition logic (see block diagram in the Custom Event Counters section). **Value** / **Name** / **Description** 01b / Prev Event / Selects the conditioned/flopped input from the previous CEC block as the input bus to this CEC block 11b / Reserved /
	18:3	**Compare Value** Format: U16 _Custom_GTIReset: DEV The value in this field is compared the 16-bit conditioned input bus that are fed into the comparator(see block diagram in the Custom Event Counters section). The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the custom event is asserted. This signal in turn can be counted by the B0 performance counter or fed into other CEC blocks.
	2:0	**Compare Function** Format: U3 _Custom_GTIReset: DEV This field selects the function used by the CEC comparator when comparing the compare value

OAG_SCEC6_0 - OAG Slice Customizable Event Creation 6-0

to the value active on the CEC conditioned input bus (see block diagram in the Custom Event Counters section).

Value	Name	Description
000b	Any Are Equal	Compare and assert output if any are equal (Can be used as OR function)
001b	Greater Than	Compare and assert output if greater than
010b	Equal	Compare and assert output if equal to (Can also be used as AND function)
011b	Greater Than or Equal	Compare and assert output if greater than or equal
100b	Less Than	Compare and assert output if less than
101b	Not Equal	Compare and assert output if not equal
110b	Less Than or Equal	Compare and assert output if less than or equal
111b	Reserved	

OAG Slice Customizable Event Creation 6-1

OAG_SCEC6_1 - OAG Slice Customizable Event Creation 6-1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DC34h

This register configures the input conditioning portion of slice CEC (custom event creation) block 6, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.

DWord	Bit	Description
0	31:16	**Considerations** _Custom_GTIReset: DEV This 16-bit field allows individual bits of the bus selected as the input to CEC block to be delayed by 1 clock relative to the non-delayed bits in the bus (see block diagram in the Custom Event Counters section). See CEC0-1 register definition for an example use case. Value / Name / Description 0b / Live / Input bit is not delayed by 1 clock before event calculation 1b / Delayed / Input bit is delayed by 1 clock before event calculation
	15:0	**Mask** _Custom_GTIReset: DEV This 16-bit field allows individual input bits to be ignored in custom event calculation. See block diagram in the Custom Event Counters section for more details on where this field is used. Value / Name / Description 0b / Unmasked / Input bit is considered in event calculation 1b / Masked / Input bit is ignored in event calculation

OAG Slice Customizable Event Creation 7-0

OAG_SCEC7_0 - OAG Slice Customizable Event Creation 7-0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DC38h

This register is used to define slice custom counter event 7, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.

DWord	Bit	Description
0	31:21	**Negate** Format: U11 _Custom_GTIReset: DEV The 11-bit array defined in this field allows input bus bits [10:0] to be individually negated in order to facilitate more complicated custom event creation (e.g. A & (!B | !C)). Note that LSB of this field affects bit 0 of the selected input bus. **Value** / **Name** / **Description** 0b / Pass-through / Input bit is passed through to comparator as is 1b / Negated / Input bit is negated before passing to comparator
	20:19	**Source Select** Format: U2 _Custom_GTIReset: DEV Selects the input signals to the Boolean event definition logic (see block diagram in the Custom Event Counters section). **Value** / **Name** / **Description** 01b / Prev Event / Selects the conditioned/flopped input from the previous CEC block as the input bus to this CEC block 11b / Reserved /
	18:3	**Compare Value** Format: U16 _Custom_GTIReset: DEV The value in this field is compared the 16-bit conditioned input bus that are fed into the comparator(see block diagram in the Custom Event Counters section). The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the custom event is asserted. This signal in turn can be counted by the B0 performance counter or fed into other CEC blocks.
	2:0	**Compare Function** Format: U3 _Custom_GTIReset: DEV This field selects the function used by the CEC comparator when comparing the compare value

OAG_SCEC7_0 - OAG Slice Customizable Event Creation 7-0

to the value active on the CEC conditioned input bus (see block diagram in the Custom Event Counters section).

Value	Name	Description
000b	Any Are Equal	Compare and assert output if any are equal (Can be used as OR function)
001b	Greater Than	Compare and assert output if greater than
010b	Equal	Compare and assert output if equal to (Can also be used as AND function)
011b	Greater Than or Equal	Compare and assert output if greater than or equal
100b	Less Than	Compare and assert output if less than
101b	Not Equal	Compare and assert output if not equal
110b	Less Than or Equal	Compare and assert output if less than or equal
111b	Reserved	

OAG Slice Customizable Event Creation 7-1

OAG_SCEC7_1 - OAG Slice Customizable Event Creation 7-1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DC3Ch

This register configures the input conditioning portion of slice CEC (custom event creation) block 3, bit definitions in this register refer to the CEC block diagram in the Custom Event Counters section.

DWord	Bit	Description
0	31:16	**Considerations** _Custom_GTIReset: DEV This 16-bit field allows individual bits of the bus selected as the input to CEC block to be delayed by 1 clock relative to the non-delayed bits in the bus (see block diagram in the Custom Event Counters section). See CEC0-1 register definition for an example use case. Value / Name / Description 0b / Live / Input bit is not delayed by 1 clock before event calculation 1b / Delayed / Input bit is delayed by 1 clock before event calculation
	15:0	**Mask** _Custom_GTIReset: DEV This 16-bit field allows individual input bits to be ignored in custom event calculation. See block diagram in the Custom Event Counters section for more details on where this field is used. Value / Name / Description 0b / Unmasked / Input bit is considered in event calculation 1b / Masked / Input bit is ignored in event calculation

OAG SQ Full Occupancy count Free

OAG_SQ_FULL_OCCUPANCY_COUNT_FREE - OAG SQ Full Occupancy count Free	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB98h
This register counts the number of GFX clocks that SQ was full. This counter is free running, always enabled and counting irrespective of OA enabled or disabled.	

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG SQ Memory Bound Stall count Free

OAG_SQ_MBSTALL_COUNT_FREE - OAG SQ Memory Bound Stall count Free	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DB94h

This register counts the number of GFX clocks that SQ read has been stalled. This counter is free running, always enabled and counting irrespective of OA enabled or disabled.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Unslice Busy Free

OAG_UNSLICE_BUSY_FREE - OAG Unslice Busy Free	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0DBD4h

This register counts the number of times that unslice wakes up due to any unslice engine (currently WG, BLT, GuC, and media fixed functions)becoming busy. This counter is free running, always enabled and counting irrespective of OA enabled or disabled.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAG Unslice Wake Count Free

OAG_UNSLICE_WAKE_COUNT_FREE - OAG Unslice Wake Count Free		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DBD8h	
This register counts the number of times that unslice wakes up due to any unslice engine (currently WG, BLT, GuC, and media fixed functions)becoming busy. This counter is free running, always enabled and counting irrespective of OA enabled or disabled.		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAM GPU Ticks Counter

OAM_GPU_TICKS - OAM GPU Ticks Counter		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Reading this register returns the live value of the GPU ticks counter that is sampled and included in the performance counter report header. It increments every GFX clock and hence the number of increments per second changes with GFX frequency.		
DWord	**Bit**	**Description**
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the GPU tick counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAM GPU Ticks Counter Upper DWord

OAM_GPU_TICKS_UPPER - OAM GPU Ticks Counter Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

Reading this register returns the live value of the Upper DWord of GPU ticks counter that is sampled and included in the performance counter report header. It increments every GFX clock and hence the number of increments per second changes with GFX frequency.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32 of the GPU tick counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OA Programmable Event Source Select Register

OAG_OA_PESS - OA Programmable Event Source Select Register

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02B2Ch

The OA_PESS register is used by software to control programmable event P15 to P0 source selection generated from 'unslice' or 'aggregated P events generated from slice'. Bit[0] indicates selection for P0, bit[1] for P1 and so on.

DWord	Bit	Description
0	31:16	**Reserved** Default Value: 0000h Format: PBC _Custom_GTIReset: DEV
	15:0	**P event source select** _Custom_GTIReset: DEV

Value	Name	Description
0h	P event generated from unslice **[Default]**	Bit wise select for P[15:0] events generated from boolean logic present in unslice
1h	P event generated from slice	Bit wise select for P[15:0] events generated from boolean logic present in slice in an aggregated manner

OAR Aggregate Perf Counter A0

OAR_OAPERF_A0 - OAR Aggregate Perf Counter A0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02800h

This register reflects the count value of the OA Performance counter A0.

DWord	Bit	Description
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads. Value: 0x00000000 — **[Default]** Value: [0x00000001-0xFFFFFFFF]

OAR Aggregate Perf Counter A0 Upper DWord

OAR_OAPERF_A0_UPPER - OAR Aggregate Perf Counter A0 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02804h

This register enables the current live value of performance counter A0 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A1

OAR_OAPERF_A1 - OAR Aggregate Perf Counter A1		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02808h	
This register reflects the count value of the OA Performance counter A1. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A1 Upper DWord

OAR_OAPERF_A1_UPPER - OAR Aggregate Perf Counter A1 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0280Ch

This register enables the current live value of performance counter A1 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A2

OAR_OAPERF_A2 - OAR Aggregate Perf Counter A2		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02810h	
This register reflects the count value of the OA Performance counter A2. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A2 Upper DWord

OAR_OAPERF_A2_UPPER - OAR Aggregate Perf Counter A2 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02814h

This register enables the current live value of performance counter A2 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A3

OAR_OAPERF_A3 - OAR Aggregate Perf Counter A3		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02818h	
This register reflects the count value of the OA Performance counter A3. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A3 Upper DWord

OAR_OAPERF_A3_UPPER - OAR Aggregate Perf Counter A3 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0281Ch

This register enables the current live value of performance counter A3 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A4

OAR_OAPERF_A4 - OAR Aggregate Perf Counter A4		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02820h	
This register reflects the count value of the OA Performance counter A4. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A4 Upper DWord

OAR_OAPERF_A4_UPPER - OAR Aggregate Perf Counter A4 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02824h

This register enables the current live value of performance counter A4 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A5

OAR_OAPERF_A5 - OAR Aggregate Perf Counter A5		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02828h	
This register reflects the count value of the OA Performance counter A5. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A5 Upper DWord

OAR_OAPERF_A5_UPPER - OAR Aggregate Perf Counter A5 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0282Ch

This register enables the current live value of performance counter A5 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A6

OAR_OAPERF_A6 - OAR Aggregate Perf Counter A6		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02830h	
This register reflects the count value of the OA Performance counter A6. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A6 Upper DWord

OAR_OAPERF_A6_UPPER - OAR Aggregate Perf Counter A6 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02834h

This register enables the current live value of performance counter A6 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A7

OAR_OAPERF_A7 - OAR Aggregate Perf Counter A7		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02838h	
This register reflects the count value of the OA Performance counter A7. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A7 Upper DWord

OAR_OAPERF_A7_UPPER - OAR Aggregate Perf Counter A7 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0283Ch

This register enables the current live value of performance counter A7 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A8

OAR_OAPERF_A8 - OAR Aggregate Perf Counter A8		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02840h	
This register reflects the count value of the OA Performance counter A8. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A8 Upper DWord

OAR_OAPERF_A8_UPPER - OAR Aggregate Perf Counter A8 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02844h

This register enables the current live value of performance counter A8 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A9

OAR_OAPERF_A9 - OAR Aggregate Perf Counter A9		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02848h	
This register reflects the count value of the OA Performance counter A9. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A9 Upper DWord

OAR_OAPERF_A9_UPPER - OAR Aggregate Perf Counter A9 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0284Ch

This register enables the current live value of performance counter A9 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A10

OAR_OAPERF_A10 - OAR Aggregate Perf Counter A10		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02850h	
This register reflects the count value of the OA Performance counter A10. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A10 Upper DWord

OAR_OAPERF_A10_UPPER - OAR Aggregate Perf Counter A10 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02854h

This register enables the current live value of performance counter A10 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A11

OAR_OAPERF_A11 - OAR Aggregate Perf Counter A11		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02858h	
This register reflects the count value of the OA Performance counter A11. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A11 Upper DWord

OAR_OAPERF_A11_UPPER - OAR Aggregate Perf Counter A11 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0285Ch

This register enables the current live value of performance counter A11 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A12

OAR_OAPERF_A12 - OAR Aggregate Perf Counter A12		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02860h	
This register reflects the count value of the OA Performance counter A12. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A12 Upper DWord

OAR_OAPERF_A12_UPPER - OAR Aggregate Perf Counter A12 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02864h

This register enables the current live value of performance counter A12 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A13

OAR_OAPERF_A13 - OAR Aggregate Perf Counter A13		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02868h	
This register reflects the count value of the OA Performance counter A13. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A13 Upper DWord

OAR_OAPERF_A13_UPPER - OAR Aggregate Perf Counter A13 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0286Ch

This register enables the current live value of performance counter A13 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A14

OAR_OAPERF_A14 - OAR Aggregate Perf Counter A14		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02870h	
This register reflects the count value of the OA Performance counter A14. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A14 Upper DWord

OAR_OAPERF_A14_UPPER - OAR Aggregate Perf Counter A14 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02874h

This register enables the current live value of performance counter A14 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A15

OAR_OAPERF_A15 - OAR Aggregate Perf Counter A15		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02878h	
This register reflects the count value of the OA Performance counter A15. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A15 Upper DWord

OAR_OAPERF_A15_UPPER - OAR Aggregate Perf Counter A15 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0287Ch

This register enables the current live value of performance counter A15 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A16

OAR_OAPERF_A16 - OAR Aggregate Perf Counter A16		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02880h	
This register reflects the count value of the OA Performance counter A16. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A16 Upper DWord

OAR_OAPERF_A16_UPPER - OAR Aggregate Perf Counter A16 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02884h

This register enables the current live value of performance counter A16 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A17

OAR_OAPERF_A17 - OAR Aggregate Perf Counter A17		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02888h	
This register reflects the count value of the OA Performance counter A17. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A17 Upper DWord

OAR_OAPERF_A17_UPPER - OAR Aggregate Perf Counter A17 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0288Ch

This register enables the current live value of performance counter A17 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A18

OAR_OAPERF_A18 - OAR Aggregate Perf Counter A18		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02890h	
This register reflects the count value of the OA Performance counter A9. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A18 Upper DWord

OAR_OAPERF_A18_UPPER - OAR Aggregate Perf Counter A18 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02894h

This register enables the current live value of performance counter A18 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A19

OAR_OAPERF_A19 - OAR Aggregate Perf Counter A19		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02898h	
This register reflects the count value of the OA Performance counter A19. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A19 Upper DWord

OAR_OAPERF_A19_UPPER - OAR Aggregate Perf Counter A19 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0289Ch

This register enables the current live value of performance counter A19 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A20

OAR_OAPERF_A20 - OAR Aggregate Perf Counter A20		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	028A0h	
This register reflects the count value of the OA Performance counter A20. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A20 Upper DWord

OAR_OAPERF_A20_UPPER - OAR Aggregate Perf Counter A20 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	028A4h

This register enables the current live value of performance counter A20 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A21

OAR_OAPERF_A21 - OAR Aggregate Perf Counter A21		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	028A8h	
This register reflects the count value of the OA Performance counter A21. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A21 Upper DWord

OAR_OAPERF_A21_UPPER - OAR Aggregate Perf Counter A21 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	028ACh

This register enables the current live value of performance counter A21 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A22

OAR_OAPERF_A22 - OAR Aggregate Perf Counter A22		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	028B0h	
This register reflects the count value of the OA Performance counter A22. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A22 Upper DWord

OAR_OAPERF_A22_UPPER - OAR Aggregate Perf Counter A22 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	028B4h

This register enables the current live value of performance counter A22 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A23

OAR_OAPERF_A23 - OAR Aggregate Perf Counter A23		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	028B8h	
This register reflects the count value of the OA Performance counter A23. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A23 Upper DWord

OAR_OAPERF_A23_UPPER - OAR Aggregate Perf Counter A23 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	028BCh

This register enables the current live value of performance counter A23 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A24

OAR_OAPERF_A24 - OAR Aggregate Perf Counter A24		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	028C0h	
This register reflects the count value of the OA Performance counter A24. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A24 Upper DWord

OAR_OAPERF_A24_UPPER - OAR Aggregate Perf Counter A24 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	028C4h

This register enables the current live value of performance counter A24 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A25

OAR_OAPERF_A25 - OAR Aggregate Perf Counter A25		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	028C8h	
This register reflects the count value of the OA Performance counter A25. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A25 Upper DWord

OAR_OAPERF_A25_UPPER - OAR Aggregate Perf Counter A25 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	028CCh

This register enables the current live value of performance counter A25 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A26

OAR_OAPERF_A26 - OAR Aggregate Perf Counter A26		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	028D0h	
This register reflects the count value of the OA Performance counter A26. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A26 Upper DWord

OAR_OAPERF_A26_UPPER - OAR Aggregate Perf Counter A26 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	028D4h

This register enables the current live value of performance counter A26 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A27

OAR_OAPERF_A27 - OAR Aggregate Perf Counter A27		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	028D8h	
This register reflects the count value of the OA Performance counter A27. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A27 Upper DWord

OAR_OAPERF_A27_UPPER - OAR Aggregate Perf Counter A27 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	028DCh

This register enables the current live value of performance counter A27 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A28

OAR_OAPERF_A28 - OAR Aggregate Perf Counter A28		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	028E0h	
This register reflects the count value of the OA Performance counter A28. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A28 Upper DWord

OAR_OAPERF_A28_UPPER - OAR Aggregate Perf Counter A28 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	028E4h

This register enables the current live value of performance counter A28 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A29

OAR_OAPERF_A29 - OAR Aggregate Perf Counter A29		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	028E8h	
This register reflects the count value of the OA Performance counter A29. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A29 Upper DWord

OAR_OAPERF_A29_UPPER - OAR Aggregate Perf Counter A29 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	028ECh

This register enables the current live value of performance counter A29 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A30

OAR_OAPERF_A30 - OAR Aggregate Perf Counter A30		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	028F0h	
This register reflects the count value of the OA Performance counter A30. DefaultValue="00000000h"		
DWord	**Bit**	**Description**
0	31:0	**Considerations** _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A30 Upper DWord

OAR_OAPERF_A30_UPPER - OAR Aggregate Perf Counter A30 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	028F4h

This register enables the current live value of performance counter A30 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A31

OAR_OAPERF_A31 - OAR Aggregate Perf Counter A31		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	028F8h	
This register reflects the count value of the OA Performance counter A31		
DWord	**Bit**	**Description**
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A31 Upper DWord

OAR_OAPERF_A31_UPPER - OAR Aggregate Perf Counter A31 Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	028FCh

This register enables the current live value of performance counter A31 to be read. Please refer to the detailed description of the aggregating performance counters in the Aggregating Counters section for more details on what event is reported via this register.

DWord	Bit	Description
0	31:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:0	**Upper Value** Format: U8 _Custom_GTIReset: DEV This 8-bit field returns bits 39:32 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A32

OAR_OAPERF_A32 - OAR Aggregate Perf Counter A32		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02900h	
This register reflects the count value of the OA Performance counter A32		
DWord	**Bit**	**Description**
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A33

OAR_OAPERF_A33 - OAR Aggregate Perf Counter A33		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02904h	
This register reflects the count value of the OA Performance counter A33		
DWord	**Bit**	**Description**
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A34

OAR_OAPERF_A34 - OAR Aggregate Perf Counter A34		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02908h	
This register reflects the count value of the OA Performance Aggregating counter A34		
DWord	**Bit**	**Description**
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Aggregate Perf Counter A35

OAR_OAPERF_A35 - OAR Aggregate Perf Counter A35		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0290Ch	
This register reflects the count value of the OA Performance counter A35		
DWord	**Bit**	**Description**
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Boolean_Counter_B0

OAR_OAPERF_B0 - OAR Boolean_Counter_B0

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02920h

This register enables the current live value of performance counter B0 to be read. Since what conditions cause B0 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Boolean_Counter_B1

OAR_OAPERF_B1 - OAR Boolean_Counter_B1	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02924h

This register enables the current live value of performance counter B1 to be read. Since what conditions cause B1 to increment are defined by the programming of CEC1-0/CEC1-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Boolean_Counter_B2

OAR_OAPERF_B2 - OAR Boolean_Counter_B2	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02928h

This register enables the current live value of performance counter B2 to be read. Since what conditions cause B2 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Boolean_Counter_B3

OAR_OAPERF_B3 - OAR Boolean_Counter_B3	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0292Ch

This register enables the current live value of performance counter B3 to be read. Since what conditions cause B3 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Boolean_Counter_B4

OAR_OAPERF_B4 - OAR Boolean_Counter_B4	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02930h

This register enables the current live value of performance counter B4 to be read. Since what conditions cause B4 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Boolean_Counter_B5

OAR_OAPERF_B5 - OAR Boolean_Counter_B5	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02934h

This register enables the current live value of performance counter B5 to be read. Since what conditions cause B5 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Boolean_Counter_B6

OAR_OAPERF_B6 - OAR Boolean_Counter_B6	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02938h

This register enables the current live value of performance counter B6 to be read. Since what conditions cause B6 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Boolean_Counter_B7

OAR_OAPERF_B7 - OAR Boolean_Counter_B7	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0293Ch

This register enables the current live value of performance counter B7 to be read. Since what conditions cause B7 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

DWord	Bit	Description
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR GPU Ticks Counter

OAR_GPU_TICKS - OAR GPU Ticks Counter		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	02910h	
Reading this register returns the live value of the GPU ticks counter that is sampled and included in the performance counter report header. It increments every GFX clock and hence the number of increments per second changes with GFX frequency.		
DWord	**Bit**	**Description**
0	31:0	**Considerations** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 31:0 of the GPU tick counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR GPU Ticks Counter Upper DWord

OAR_GPU_TICKS_UPPER - OAR GPU Ticks Counter Upper DWord

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

Reading this register returns the live value of the Upper DWord of GPU ticks counter that is sampled and included in the performance counter report header. It increments every GFX clock and hence the number of increments per second changes with GFX frequency.

DWord	Bit	Description
0	31:0	**Upper Value** Format: U32 _Custom_GTIReset: DEV This 32-bit field returns bits 63:32 of the GPU tick counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAR Observation Architecture Control

OAR_OACONTROL - OAR Observation Architecture Control

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02960h
Name:	OAR Observation Architecture Control
ShortName:	OAR_OACONTROL

This register controls OAR functionality, report format, and context filtering. If OAR is enabled then it should be enabled in CS as well

DWord	Bit	Description
0	31:4	**Reserved** Format: PBC _Custom_GTIReset: DEV
	3:1	**Counter Select** _Custom_GTIReset: DEV This field selects which performance counter report format to use, please refer to Performance Counter Report Formats section for more details on the structure of the format.
	0	**Performance Counter Enable** Format: Enable _Custom_GTIReset: DEV Global performance counter enable. If clear, no counting will occur. MI_REPORT_PERF_COUNT is undefined when clear. **Programming Notes** "OAR Context Enable" mode bit in RenderEngne CTX_SR_CTL register must be set when "Programmer Counter Enable" is set. OAC and OAR concurrent enabling is not supported (OAG is recommended instead).

OAR Observation Architecture Status Register

OAR_OASTATUS - OAR Observation Architecture Status Register

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02968h
Name:	OAR Observation Architecture Status Register
ShortName:	OAR_OASTATUS

This register provides status of report buffer and overflow conditions as well as status of some flags which hardware needs for internal purpose. Software should not program these status bits. Status bits are active high flags (0=false, 1=true)

DWord	Bit	Description
0	31:22	**Reserved** Format: PBC _Custom_GTIReset: DEV
	21	**Start Trigger Flag 1** _Custom_GTIReset: DEV Value / Name: 0 **[Default]**; 1 **Programming Notes**: This bit is for HW internal use to context save /restore Start Trigger 1 occurrence On RC6 entry. Software should not program this bit.
	20	**Start Trigger Flag 2** _Custom_GTIReset: DEV Value / Name: 0 **[Default]**; 1 **Programming Notes**: This bit is for HW internal use to context save /restore Start Trigger 2 occurrence On RC6 entry. Software should not program this bit.
	19	**Report Trigger Flag 1** _Custom_GTIReset: DEV Value / Name

OAR_OASTATUS - OAR Observation Architecture Status Register

0	**[Default]**
1	

Programming Notes
This bit is for HW internal use to context save /restore Report Trigger 1 occurrence On RC6 entry. Software should not program this bit.

18 Report Trigger Flag 2

_Custom_GTIReset:	DEV

Value	Name
0	**[Default]**
1	

Programming Notes
This bit is for HW internal use to context save /restore Report Trigger 2 occurrence On RC6 entry. Software should not program this bit.

17:6 Reserved

Format:	PBC
_Custom_GTIReset:	DEV

5:2 Reserved

Format:	PBC
_Custom_GTIReset:	DEV

1 Accumulator Overflow

_Custom_GTIReset:	DEV

This field indicates that the one or more event accumulator inside the slice-OAunit has overflowed. Once set, this bit will remain set, until SW resets it by either soft reset or writing a 0 to it.

Value	Name	Description
0	**[Default]**	No overflow has occurred.
1		Overflow has occurred.

0 Counter Overflow

_Custom_GTIReset:	DEV

This bit is set if any of the counters overflows. This bit can be reset by SW by either soft reset or writing a 1 to it.

Value	Name	Description
0	**[Default]**	Counter Overflow not occurred
1		Counter Overflow occurred

Observation Architecture Control Context ID

OACTXID - Observation Architecture Control Context ID	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02364h
Name:	Observation Architecture Control Context ID
ShortName:	OACTXID

This register has the context details when Specific context Enable is set. This register is implemented in render command streamer and render context save/restored. This register should be initialized by SW appropriately on the very first submission of a context when OA is enabled.

DWord	Bit	Description
0	31:0	**Select Context ID** _Custom_GTIReset: DEV Specifies the context ID of the one context that affects the performance counters when "Specific Context Enable" bit is set. All other contexts are ignored. **Ring Buffer Mode of Scheduling:** Bits[31:12] represent the CCID and bits [11:0] must be zero. **Execlist mode of scheduling:** Bits[31:0] represent the context id.

Observation Architecture Control per Context

OACTXCONTROL - Observation Architecture Control per Context

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	02360h
Name:	Observation Architecture Control per Context
ShortName:	OACTXCONTROL

This register is implemented in render command streamer and render context save/restored. This register should be initialized by SW appropriately on the very first submission of a context when OA is enabled.

DWord	Bit	Description
0	31	**Reserved** Format: PBC _Custom_GTIReset: DEV
	30:8	**Reserved** Format: PBC _Custom_GTIReset: DEV
	7:2	**Timer Period** _Custom_GTIReset: DEV Specifies the period of the timer strobe as a function of the minimum TIME_STAMP resolution. The period is determined by selecting a specified bit from the TIME_STAMP register as follows:StrobePeriod = MinimumTimeStampPeriod * 2 **(TimerPeriod + 1)**The exponent is defined by this field. **Note:** The TIME_STAMP is not reset at start time so the phase of the strobe is not synchronized with the enable of the OA unit. This could result in approximately a full StrobePeriod elapsing prior to the first trigger. Usage for this mechanism should be time based periodic triggering, typically.
	1	**Timer Enable** _Custom_GTIReset: DEV This field enables the timer logic to output a periodic strobe, as defined by the Timer Period. When disabled the timer output is not asserted. Users should be aware that the timer-based report logic and the Boolean report trigger logic are logically OR'd together with no buffering of report triggers. This implies that only one performance counter report will be generated for clocks where both the timer-based report logic and the Boolean report trigger logic trigger a performance counter report. When "Specific Context Enable" bit is set to '1' in OACONTROL register, Timer based report trigger function gets enabled only for the selected context.When "Specific Context Enable" bit is set to '0' in OACONTROL register, Timer based report trigger function gets enabled for all

contexts.

Value	Name	Description
0h	Disable **[Default]**	Counter does not get written out on regular interval
1h	Enable	Counter gets written out on regular intervals, defined by the Timer Period

0 **Counter Stop-Resume Mechanism**

_Custom_GTIReset:	DEV

Value	Name	Description
1h		resume counting for all counters

Observation Architecture Head Pointer

OAG_OAHEADPTR - Observation Architecture Head Pointer		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	0DB00h	
This register allows SW to program head pointer.		
DWord	**Bit**	**Description**
0	31:6	**Head Pointer** _Custom_GTIReset: DEV Virtual address of the internal trigger based buffer that is updated by software after consuming reports from the report buffer. This pointer must be updated by SW only when using time or event-based report triggering. **Programming Notes** SW must ensure that Head Pointer and the Tail Pointer match before enabling internally triggered performance counter reporting.
	5:0	**Reserved** Format: PBC _Custom_GTIReset: DEV

OUTPUT_CSC_COEFF

OUTPUT_CSC_COEFF	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	192
_Custom_Display_DoubleBufferArmedBy:	Write to CSC_MODE
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank after armed

Address:	49050h-49067h
Name:	Pipe Output CSC Coefficients
ShortName:	OUTPUT_CSC_COEFF_A
Reset:	soft

Address:	49150h-49167h
Name:	Pipe Output CSC Coefficients
ShortName:	OUTPUT_CSC_COEFF_B
Reset:	soft

Address:	49250h-49267h
Name:	Pipe Output CSC Coefficients
ShortName:	OUTPUT_CSC_COEFF_C
Reset:	soft

Address:	49350h-49367h
Name:	Pipe Output CSC Coefficients
ShortName:	OUTPUT_CSC_COEFF_D
Reset:	soft

DWord	Bit	Description
0	31:16	**RY** Format: CSC COEFFICIENT FORMAT
	15:0	**GY** Format: CSC COEFFICIENT FORMAT
1	31:16	**BY** Format: CSC COEFFICIENT FORMAT
	15:0	**Reserved** Access: RO Format: MBZ
2	31:16	**RU** Format: CSC COEFFICIENT FORMAT

OUTPUT_CSC_COEFF		
	15:0	**GU** Format: CSC COEFFICIENT FORMAT
3	31:16	**BU** Format: CSC COEFFICIENT FORMAT
	15:0	**Reserved** Access: RO Format: MBZ
4	31:16	**RV** Format: CSC COEFFICIENT FORMAT
	15:0	**GV** Format: CSC COEFFICIENT FORMAT
5	31:16	**BV** Format: CSC COEFFICIENT FORMAT
	15:0	**Reserved** Access: RO Format: MBZ

OUTPUT_CSC_POSTOFF

OUTPUT_CSC_POSTOFF	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	96
_Custom_Display_DoubleBufferArmedBy:	Write to CSC_MODE
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank after armed

Address:	49074h-4907Fh
Name:	Pipe Output CSC Post-Offsets
ShortName:	OUTPUT_CSC_POSTOFF_A
Reset:	soft

Address:	49174h-4917Fh
Name:	Pipe Output CSC Post-Offsets
ShortName:	OUTPUT_CSC_POSTOFF_B
Reset:	soft

Address:	49274h-4927Fh
Name:	Pipe Output CSC Post-Offsets
ShortName:	OUTPUT_CSC_POSTOFF_C
Reset:	soft

Address:	49374h-4937Fh
Name:	Pipe Output CSC Post-Offsets
ShortName:	OUTPUT_CSC_POSTOFF_D
Reset:	soft

The post-offset is intended to add an offset from 0 on the Y or RGB channels and to convert UV channels from 2's complement to excess 0.5 as they exit pipe output color space conversion (CSC).

DWord	Bit	Description
0	31:13	**Reserved** Access: RO Format: MBZ
	12:0	**PostCSC High Offset** This value is used to give an offset to the high color channel as it exits CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).
1	31:13	**Reserved** Access: RO Format: MBZ

OUTPUT_CSC_POSTOFF		
	12:0	**PostCSC Medium Offset** This value is used to give an offset to the medium color channel as it exits CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).
2	31:13	**Reserved** Access: RO Format: MBZ
	12:0	**PostCSC Low Offset** This value is used to give an offset to the low color channel as it exits CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).

OUTPUT_CSC_PREOFF

OUTPUT_CSC_PREOFF	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	96
_Custom_Display_DoubleBufferArmedBy:	Write to CSC_MODE
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank after armed

Address:	49068h-49073h
Name:	Pipe Output CSC Pre-Offsets
ShortName:	OUTPUT_CSC_PREOFF_A
Reset:	soft

Address:	49168h-49173h
Name:	Pipe Output CSC Pre-Offsets
ShortName:	OUTPUT_CSC_PREOFF_B
Reset:	soft

Address:	49268h-49273h
Name:	Pipe Output CSC Pre-Offsets
ShortName:	OUTPUT_CSC_PREOFF_C
Reset:	soft

Address:	49368h-49373h
Name:	Pipe Output CSC Pre-Offsets
ShortName:	OUTPUT_CSC_PREOFF_D
Reset:	soft

The pre-offset is intended to remove an offset from 0 on the Y or RGB channels and to convert UV channels from excess 0.5 to 2's complement as they enter pipe output color space conversion (CSC).

DWord	Bit	Description
0	31:13	**Reserved** Access: RO Format: MBZ
	12:0	**PreCSC High Offset** This value is used to give an offset to the high color channel as it enters CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).
1	31:13	**Reserved** Access: RO Format: MBZ

OUTPUT_CSC_PREOFF		
	12:0	**PreCSC Medium Offset** This value is used to give an offset to the medium color channel as it enters CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).
2	31:13	**Reserved** Access: RO Format: MBZ
	12:0	**PreCSC Low Offset** This value is used to give an offset to the low color channel as it enters CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).

Outstanding Page Request Allocation

OPRA_0_2_0_PCI - Outstanding Page Request Allocation		
Register Space:	PCI: 0/2/0	
Size (in bits):	32	
Address:	0030Ch	
DWord	**Bit**	**Description**
0	31:0	**Outstanding Page Request Allocation** Default Value: 00000000000000000000000000000000b Access: R/W _Custom_GTIReset: BUS This register contains the number of outstanding page request messages the associated Page Request Interface is allowed to issue.

Outstanding Page Request Capacity

OPRC_0_2_0_PCI - Outstanding Page Request Capacity		
Register Space:	PCI: 0/2/0	
Size (in bits):	32	
Address:	00308h	
DWord	**Bit**	**Description**
0	31:0	**Outstanding Page Request Capacity** Default Value: 00000000000000001000000000000000b Access: RO _Custom_GTIReset: BUS This register contains the number of outstanding page request messages the associated Page Request Interface physically supports. This is the upper limit on the number of pages that can be usefully allocated to the Page Request Interface. Hardwired to 32,768 requests.

PAGE_FAULT_MODE

PAGE_FAULT_MODE - PAGE_FAULT_MODE

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	0E454h
Name:	PAGE_FAULT_MODE
ShortName:	PAGE_FAULT_MODE

This register is written as part of Context Submission to TDL. The data written is the lower 32 bits of the Context Descriptor Format structure.

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7:6	**FAULT_MODE** Access: WO **Fault Model:** Applicable only in advanced context

Value	Name	Description
00b	Fault and Hang **[Default]**	In Legacy Context mode, this is the only valid encoding.
01b	Fault and Halt	Restriction : Only valid in Advanced Context mode.
010b	Fault and Stream	Restriction : Only valid in Advanced Context mode.
Others	Reserved	

DWord	Bit	Description
	5:0	**Reserved** Access: RO Format: MBZ

Page Directory Pointer Descriptor - PDP0/PML4/PASID

PDP0 - Page Directory Pointer Descriptor - PDP0/PML4/PASID	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02270h-02277h
Name:	Page Directory Pointer Descriptor - PDP0/PML4/PASID
ShortName:	PDP0_RCSUNIT
Address:	18270h-18277h
Name:	Page Directory Pointer Descriptor - PDP0/PML4/PASID
ShortName:	PDP0_POCSUNIT
Address:	22270h-22277h
Name:	Page Directory Pointer Descriptor - PDP0/PML4/PASID
ShortName:	PDP0_BCSUNIT
Address:	1C0270h-1C0277h
Name:	Page Directory Pointer Descriptor - PDP0/PML4/PASID
ShortName:	PDP0_VCSUNIT0
Address:	1C4270h-1C4277h
Name:	Page Directory Pointer Descriptor - PDP0/PML4/PASID
ShortName:	PDP0_VCSUNIT1
Address:	1C8270h-1C8277h
Name:	Page Directory Pointer Descriptor - PDP0/PML4/PASID
ShortName:	PDP0_VECSUNIT0
Address:	1D0270h-1D0277h
Name:	Page Directory Pointer Descriptor - PDP0/PML4/PASID
ShortName:	PDP0_VCSUNIT2
Address:	1D4270h-1D4277h
Name:	Page Directory Pointer Descriptor - PDP0/PML4/PASID
ShortName:	PDP0_VCSUNIT3
Address:	1D8270h-1D8277h
Name:	Page Directory Pointer Descriptor - PDP0/PML4/PASID
ShortName:	PDP0_VECSUNIT1
Address:	1E0270h-1E0277h
Name:	Page Directory Pointer Descriptor - PDP0/PML4/PASID
ShortName:	PDP0_VCSUNIT4

PDP0 - Page Directory Pointer Descriptor - PDP0/PML4/PASID

Address:	1E4270h-1E4277h
Name:	Page Directory Pointer Descriptor - PDP0/PML4/PASID
ShortName:	PDP0_VCSUNIT5

Address:	1E8270h-1E8277h
Name:	Page Directory Pointer Descriptor - PDP0/PML4/PASID
ShortName:	PDP0_VECSUNIT2

Address:	1F0270h-1F0277h
Name:	Page Directory Pointer Descriptor - PDP0/PML4/PASID
ShortName:	PDP0_VCSUNIT6

Address:	1F4270h-1F4277h
Name:	Page Directory Pointer Descriptor - PDP0/PML4/PASID
ShortName:	PDP0_VCSUNIT7

Address:	1F8270h-1F8277h
Name:	Page Directory Pointer Descriptor - PDP0/PML4/PASID
ShortName:	PDP0_VECSUNIT3

Address:	1A270h-1A277h
Name:	Page Directory Pointer Descriptor - PDP0/PML4/PASID
ShortName:	PDP0_CCSUNIT0

PDP0/PML4/PASID: This register can contain three values which depend on the element descriptor definition. **PASID[19:0]:** Populated in the first 20bits of the register and selected when Advanced Context flag is set in the element descriptor in execlist mode of submission. This is not valid in ring buffer mode of scheduling. **PML4[38:12]:** Pointer to base address of PML4 and selected when Legacy Context flag is set and 64b address support is selected.**PDP0[38:12]:** Pointer to one of the four page directory pointer (lowest) and defines the first 0-1GB of memory mapping. This is valid when Legacy Context Flag is set and 64bit virtual addressing is not supported.*Note: This is a guest physical address.*

Programming Notes
Execlist Based Scheduling: SW should update PDP0/12/3 registers in context image with proper values before submitting the context to HW in execlist mode of scheduling. HW restores these registers as part of context restore to set the PPGTT access accordingly. PPGTT is always enabled in advanced context mode of execlist based scheduling and can be disabled only in legacy context mode. Privilege Access Bit in Element Descriptor controls the PPGTT enabling in legacy context mode.
Ring Buffer Based Scheduling: A write via MMIO to PDP0_DESCRIPROR (lower Dword) triggers the Page Directory Restore in HW when PPGTT is enabled. SW should ensure PDP1/2/3 registers are programmed appropriately prior to programming PDP0. PDP0_DESCRIPTOR lower dword should be programmed at the end. Per-Process GTT Enable Bit in GFX_MODE register controls the PPGTT enabling and disabling. Programming Per-Process GTT Enable Bit in GFX_MODE register doesn't enable/disable the PPGTT translation of memory access immediately; the change comes in to affect only when the Page Directory registers are programmed. Programming Per-Process GTT Enable Bit in GFX_MODE register bit must be followed by programming Page Directory Registers in order to enable or disable the PPGTT translation of memory access. PDP* DESCRIPTOR registers must always be programmed through MI_LOAD_REGISTER_IMMEDIATE command in ring buffer with

PDP0 - Page Directory Pointer Descriptor - PDP0/PML4/PASID

PDP0_DESCRIPRTOR lower dword written at the end. PDP0/12/3 registers are context save restored. PDP descriptors are context save restored per render context in RCS and must be programmed following MI_SET_CONTEXT command, in case of PDP descriptors programmed without context set (MI_SET_CONTEXT) will get lost on C6 entry/exit. PDP descriptors are context save restored in VCS, BCS and VECS engines and must be programmed following setup of CCID register, in case of PDP descriptors programmed without CCID set will get lost on C6 entry/exit. PDP descriptor registers should be programmed after ensuring the pipe is completely flushed and TLB's invalidated.

DWord	Bit	Description
0	63	**PD Load Busy** Access: RO This read-only field gets set when PDP0 is written to indicating Page Directory Restore activity is in progress and will get reset once the activity is completed.
	62:0	**PDP0 Descriptor**

Page Directory Pointer Descriptor - PDP1

PDP1 - Page Directory Pointer Descriptor - PDP1	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02278h-0227Fh
Name:	Page Directory Pointer Descriptor - PDP1
ShortName:	PDP1_RCSUNIT
Address:	18278h-1827Fh
Name:	Page Directory Pointer Descriptor - PDP1
ShortName:	PDP1_POCSUNIT
Address:	22278h-2227Fh
Name:	Page Directory Pointer Descriptor - PDP1
ShortName:	PDP1_BCSUNIT
Address:	1C0278h-1C027Fh
Name:	Page Directory Pointer Descriptor - PDP1
ShortName:	PDP1_VCSUNIT0
Address:	1C4278h-1C427Fh
Name:	Page Directory Pointer Descriptor - PDP1
ShortName:	PDP1_VCSUNIT1
Address:	1C8278h-1C827Fh
Name:	Page Directory Pointer Descriptor - PDP1
ShortName:	PDP1_VECSUNIT0
Address:	1D0278h-1D027Fh
Name:	Page Directory Pointer Descriptor - PDP1
ShortName:	PDP1_VCSUNIT2
Address:	1D4278h-1D427Fh
Name:	Page Directory Pointer Descriptor - PDP1
ShortName:	PDP1_VCSUNIT3
Address:	1D8278h-1D827Fh
Name:	Page Directory Pointer Descriptor - PDP1
ShortName:	PDP1_VECSUNIT1
Address:	1E0278h-1E027Fh
Name:	Page Directory Pointer Descriptor - PDP1
ShortName:	PDP1_VCSUNIT4

PDP1 - Page Directory Pointer Descriptor - PDP1

Address:	1E4278h-1E427Fh
Name:	Page Directory Pointer Descriptor - PDP1
ShortName:	PDP1_VCSUNIT5

Address:	1E8278h-1E827Fh
Name:	Page Directory Pointer Descriptor - PDP1
ShortName:	PDP1_VECSUNIT2

Address:	1F0278h-1F027Fh
Name:	Page Directory Pointer Descriptor - PDP1
ShortName:	PDP1_VCSUNIT6

Address:	1F4278h-1F427Fh
Name:	Page Directory Pointer Descriptor - PDP1
ShortName:	PDP1_VCSUNIT7

Address:	1F8278h-1F827Fh
Name:	Page Directory Pointer Descriptor - PDP1
ShortName:	PDP1_VECSUNIT3

Address:	1A278h-1A27Fh
Name:	Page Directory Pointer Descriptor - PDP1
ShortName:	PDP1_CCSUNIT0

PDP1[38:12]: Pointer to one of the four page directory pointer (lowest+1) and defines the first 1-2GB of memory mapping. This is valid when Legacy Context Flag is set and 64bit virtual addressing is not supported. *Note: This is a guest physical address.*

DWord	Bit	Description
0	63:0	**PDP1 Descriptor**

Page Directory Pointer Descriptor - PDP2

PDP2 - Page Directory Pointer Descriptor - PDP2	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02280h-02287h
Name:	Page Directory Pointer Descriptor - PDP2
ShortName:	PDP2_RCSUNIT
Address:	18280h-18287h
Name:	Page Directory Pointer Descriptor - PDP2
ShortName:	PDP2_POCSUNIT
Address:	22280h-22287h
Name:	Page Directory Pointer Descriptor - PDP2
ShortName:	PDP2_BCSUNIT
Address:	1C0280h-1C0287h
Name:	Page Directory Pointer Descriptor - PDP2
ShortName:	PDP2_VCSUNIT0
Address:	1C4280h-1C4287h
Name:	Page Directory Pointer Descriptor - PDP2
ShortName:	PDP2_VCSUNIT1
Address:	1C8280h-1C8287h
Name:	Page Directory Pointer Descriptor - PDP2
ShortName:	PDP2_VECSUNIT0
Address:	1D0280h-1D0287h
Name:	Page Directory Pointer Descriptor - PDP2
ShortName:	PDP2_VCSUNIT2
Address:	1D4280h-1D4287h
Name:	Page Directory Pointer Descriptor - PDP2
ShortName:	PDP2_VCSUNIT3
Address:	1D8280h-1D8287h
Name:	Page Directory Pointer Descriptor - PDP2
ShortName:	PDP2_VECSUNIT1
Address:	1E0280h-1E0287h
Name:	Page Directory Pointer Descriptor - PDP2
ShortName:	PDP2_VCSUNIT4

PDP2 - Page Directory Pointer Descriptor - PDP2

Address:	1E4280h-1E4287h
Name:	Page Directory Pointer Descriptor - PDP2
ShortName:	PDP2_VCSUNIT5

Address:	1E8280h-1E8287h
Name:	Page Directory Pointer Descriptor - PDP2
ShortName:	PDP2_VECSUNIT2

Address:	1F0280h-1F0287h
Name:	Page Directory Pointer Descriptor - PDP2
ShortName:	PDP2_VCSUNIT6

Address:	1F4280h-1F4287h
Name:	Page Directory Pointer Descriptor - PDP2
ShortName:	PDP2_VCSUNIT7

Address:	1F8280h-1F8287h
Name:	Page Directory Pointer Descriptor - PDP2
ShortName:	PDP2_VECSUNIT3

Address:	1A280h-1A287h
Name:	Page Directory Pointer Descriptor - PDP2
ShortName:	PDP2_CCSUNIT0

PDP2[38:12]: Pointer to one of the four page directory pointer (lowest+2) and defines the first 2-3GB of memory mapping. This is valid when Legacy Context Flag is set and 64bit virtual addressing is not supported. *Note: This is a guest physical address.*

DWord	Bit	Description
0	63:0	**PDP2 Descriptor**

Page Directory Pointer Descriptor - PDP3

PDP3 - Page Directory Pointer Descriptor - PDP3	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02288h-0228Fh
Name:	Page Directory Pointer Descriptor - PDP3
ShortName:	PDP3_RCSUNIT
Address:	18288h-1828Fh
Name:	Page Directory Pointer Descriptor - PDP3
ShortName:	PDP3_POCSUNIT
Address:	22288h-2228Fh
Name:	Page Directory Pointer Descriptor - PDP3
ShortName:	PDP3_BCSUNIT
Address:	1C0288h-1C028Fh
Name:	Page Directory Pointer Descriptor - PDP3
ShortName:	PDP3_VCSUNIT0
Address:	1C4288h-1C428Fh
Name:	Page Directory Pointer Descriptor - PDP3
ShortName:	PDP3_VCSUNIT1
Address:	1C8288h-1C828Fh
Name:	Page Directory Pointer Descriptor - PDP3
ShortName:	PDP3_VECSUNIT0
Address:	1D0288h-1D028Fh
Name:	Page Directory Pointer Descriptor - PDP3
ShortName:	PDP3_VCSUNIT2
Address:	1D4288h-1D428Fh
Name:	Page Directory Pointer Descriptor - PDP3
ShortName:	PDP3_VCSUNIT3
Address:	1D8288h-1D828Fh
Name:	Page Directory Pointer Descriptor - PDP3
ShortName:	PDP3_VECSUNIT1
Address:	1E0288h-1E028Fh
Name:	Page Directory Pointer Descriptor - PDP3
ShortName:	PDP3_VCSUNIT4

PDP3 - Page Directory Pointer Descriptor - PDP3

Address:	1E4288h-1E428Fh
Name:	Page Directory Pointer Descriptor - PDP3
ShortName:	PDP3_VCSUNIT5

Address:	1E8288h-1E828Fh
Name:	Page Directory Pointer Descriptor - PDP3
ShortName:	PDP3_VECSUNIT2

Address:	1F0288h-1F028Fh
Name:	Page Directory Pointer Descriptor - PDP3
ShortName:	PDP3_VCSUNIT6

Address:	1F4288h-1F428Fh
Name:	Page Directory Pointer Descriptor - PDP3
ShortName:	PDP3_VCSUNIT7

Address:	1F8288h-1F828Fh
Name:	Page Directory Pointer Descriptor - PDP3
ShortName:	PDP3_VECSUNIT3

Address:	1A288h-1A28Fh
Name:	Page Directory Pointer Descriptor - PDP3
ShortName:	PDP3_CCSUNIT0

PDP3[38:12]: Pointer to one of the four page directory pointer (lowest+3) and defines the first 3-4GB of memory mapping. This is valid when Legacy Context Flag is set and 64bit virtual addressing is not supported. *Note: This is a guest physical address.*

DWord	Bit	Description
0	63:0	**PDP3 Descriptor**

Page Req Queue Tail Shadow Register DW0

PRQTP_DW0 - Page Req Queue Tail Shadow Register DW0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00EC4h-00EC7h	
DWord	**Bit**	**Description**
0	31:0	**TailPtr** Access: R/W Shadow register for Page Req Queue Tail register DW0. Usage: GAM will provide the data which is readable via address F0C8.

Page Req Queue Tail Shadow Register DW1

PRQTP_DW1 - Page Req Queue Tail Shadow Register DW1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00EC8h-00ECBh	
DWord	**Bit**	**Description**
0	31:0	**TailPtr** Access: R/W Shadow register for Page Req Queue Tail register DW1. Usage: GAM will provide the data which is readable via address F0CC.

Page Request Control

PR_CTRL_0_2_0_PCI - Page Request Control

Register Space:	PCI: 0/2/0
Size (in bits):	16
Address:	00304h

DWord	Bit	Description
0	15:2	**Reserved** Access: RO Format: MBZ
	1	**Reset** Default Value: 0b Access: RO _Custom_GTIReset: BUS When the Enable field is clear, or is being cleared in the same register update that sets this field, writing a 1b to this field, clears the associated implementation dependent page request credit Counter and pending request state for the associated Page Request Interface. No action is initiated if this field is written to 0b or if this field is written with any value when the PRE field is set. Processor graphics does not use this field, and hardwires it as read-only (0).
	0	**Page-Request Enable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS When Set, indicates that the page request interface on the endpoint is allowed to make page requests. If both this field and the Stopped field in Page Request Status register are Clear, then the Page request interface will not issue new page requests, but has outstanding page requests for which page responses is not yet received. When this field transitions from 0 to 1, all the status fields in the Page-Request Status register are cleared. Enabling a page request interface that has not successfully stopped has indeterminate results.

Page Request Extended Capability Header

PR_EXTCAP_0_2_0_PCI - Page Request Extended Capability Header

Register Space:	PCI: 0/2/0
Size (in bits):	32
Address:	00300h

Page Request Extended Capability reports support for page-faults on Device-2, compliant to PCI-Express ATS 1.1 Specification

DWord	Bit	Description
0	31:20	**Next Capability Offset** Default Value: 000000000000b Access: RO Variant _Custom_GTIReset: BUS This is a hardwired pointer to the next item in the capabilities list. Value 000h (Default) indicates that this is the end of the PCI-Express Extended capability Linked List. When Graphics Virtualization is enabled, this field is hardwired to point to the next PCI Capability structure, the SRIOV Extended Capability Header at 320h. When Graphics Virtualization is disabled, this field will be hardwired to 000h to indicate the end of PCI-Express Extended capability Linked List.
	19:16	**Version** Default Value: 0001b Access: RO _Custom_GTIReset: BUS Hardwired to capability version 1.
	15:0	**Capability ID** Default Value: 0000000000010011b Access: RO _Custom_GTIReset: BUS Hardwired to the Page Request Extended Capability ID

Page Request Queue Address Register 0

PAGEREQ_QADDR_0 - Page Request Queue Address Register 0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	0F0D0h	
Register to configure the base address and size of the page request queue.		
DWord	**Bit**	**Description**
0	31:12	**Page Request Queue Base Register** Default Value: 00000h Access: R/W This field points to the base of 4KB aligned invalidation request queue. Hardware may ignore and not implement bits 63:HAW, where HAW is the host address width. Reads of this field return the value that was last programmed to it.
	11:3	**Reserved** Access: RO Format: MBZ
	2:0	**Queue Size** Default Value: 000b Access: R/W This field specifies the size of the page request queue. A value of X in this field indicates a page request queue of (2X) 4KB pages. The number of entries in the invalidation queue is 2(X+8).

Page Request Queue Address Register 1

PAGEREQ_QADDR_1 - Page Request Queue Address Register 1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	0F0D4h	
Register to configure the base address and size of the page request queue.		
DWord	**Bit**	**Description**
0	31:0	**Page Request Queue Base Register** Default Value: 00000000h Access: R/W This field points to the base of 4KB aligned invalidation request queue. Hardware may ignore and not implement bits 63:HAW, where HAW is the host address width. Reads of this field return the value that was last programmed to it.

Page Request Queue Head Register 0

PAGEREQ_QHEAD_0 - Page Request Queue Head Register 0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	0F0C0h	
Register indicating the page request queue head.		
DWord	**Bit**	**Description**
0	31:19	**Reserved** Access: RO Format: MBZ
	18:4	**Queue Head** Default Value: 000000000000000b Access: R/W Specifies the offset (128-bit aligned) to the page request queue for the command that is processed next by software. GFX implementation: GFX has to read the content of the Head pointer as tail pointer gets close to it to prevent overflows in page request queue.
	3:0	**Reserved** Access: RO Format: MBZ

Page Request Queue Head Register 1

PAGEREQ_QHEAD_1 - Page Request Queue Head Register 1	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	BUS
Address:	0F0C4h

Register indicating the page request queue head.

DWord	Bit	Description
0	31:0	**Page Request Queue Head Register 1 Reserved** Default Value: 00000000h Access: RO Bit[63:32]: Reserved.

Page Request Queue Tail Register 0

PAGEREQ_QTAIL_0 - Page Request Queue Tail Register 0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	0F0C8h	
Register indicating the page request queue tail.		
DWord	**Bit**	**Description**
0	31:1	**Queue Tail** Default Value: 00000000000000000000000000000000b Access: R/W Bit[31:19]: Reserved. Bit[18:4]: Specifies the offset (128-bit aligned) to the page request queue for the request that is written next by hardware. GFX Implementation: GT manages the tail pointer value as part of page requests. The value can be acquired as part of the RC6 exit. Bit[3:1]: Reserved.
	0	**Valid Bit** Default Value: 0b Access: R/W This bit can only be cleared by SW, which also clears the other fields.

Page Request Queue Tail Register 1

PAGEREQ_QTAIL_1 - Page Request Queue Tail Register 1			
Register Space:	MMIO: 0/2/0		
Size (in bits):	32		
_Custom_GTIReset:	BUS		
Address:	0F0CCh		
Register indicating the page request queue tail.			
DWord	**Bit**	**Description**	
0	31:0	**Page Request Queue Tail Register 1 Reserved**	
		Default Value:	00000000h
		Access:	RO
		Bit[63:32]: Reserved.	

Page Request Status

PR_STATUS_0_2_0_PCI - Page Request Status

Register Space: PCI: 0/2/0
Size (in bits): 16

Address: 00306h

DWord	Bit	Description
0	15	**PRG Response PASID Required** Default Value: 1b Access: RO _Custom_GTIReset: BUS If set, the Function expects a PASID TLP Prefix on PRG Response Messages when the corresponding page requests had a PASID TLP Prefix. If Clear, the function does not expect PASID TLP Prefixes on any PRG Response Message. Function behavior is undefined if this bit is Clear and the Function receives a PRG Response Message with a PASID TLP Prefix. Function behavior is undefined if this bit is Set and the Function receives a PRG Response Message with no PASID TLP Prefix when the corresponding Page Requests had a PASID TLP Prefix. This bit is RsvdZ if the Function does not support the PASID TLP Prefix.
	14:9	**Reserved** Access: RO Format: MBZ
	8	**Stopped** Access: RO _Custom_GTIReset: BUS When this field is Set, the associated page request interface has stopped issuing additional Page requests and that all previously issued Page requests have completed. When this field is clear the associate Page request interface either has not stopped or has stopped issuing new Page requests but has outstanding Page requests. Value: 1b — Name: **[Default]**
	7:2	**Reserved** Access: RO Format: MBZ
	1	**Unexpected Page Request Group Index** Default Value: 0b Access: R/W One Clear _Custom_GTIReset: BUS When Set, indicates the function received a PRG response message containing a PRG index that has no matching request, a response failure. This field is Set by the Function and cleared when a 1b is written to the field.

PR_STATUS_0_2_0_PCI - Page Request Status

	0	**Response Failure** Default Value: 0b Access: R/W One Clear _Custom_GTIReset: BUS When Set, indicates the function received a PRG response message indicating a response failure. The function expects no further response from the host (any received are ignored). This field is Set by the Function and cleared when a 1b is written to this field.

PAK_NUM_OF_SLICES

PAK_NUM_OF_SLICES - PAK_NUM_OF_SLICES		
Register Space:	MMIO: 0/2/0	
Access:	RO	
Size (in bits):	32	
Address:	12954h	
ShortName:	PAK_NUM_OF_SLICES1	
Address:	1C954h	
ShortName:	PAK_NUM_OF_SLICES2	

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Number of slices in a frame.** This field indicates number of slices in the current frame. This register is updated at the end of each slice.

PAK_Stream-Out Report (Errors)

PAK_ERR - PAK_Stream-Out Report (Errors)		
Register Space:	MMIO: 0/2/0	
Access:	RO	
Size (in bits):	32	
Address:	128E8h	

DWord	Bit	Description
0	31:22	**Reserved** Access: RO Format: MBZ
	21	**Incorrect IntraMBFlag in I-slice(AVCf)**
	20	**Out of Range Symbol Code(AVC/mpeg2)**
	19	**Incorrect MBType(AVC/mpeg2**
	18	**Motion Vectors are not inside the frame boundary(mpeg2)**
	17	**Scale code is zero(mpeg2)**
	16	**Incorrect DCTtype for given motionType(mpeg2)**
	15:8	**MB Y-position** This field indicates Macro Block(MB) Y- position where an error occurred while encoding.
	7:0	**MB X-position** This field indicates Macro Block(MB) X- position where an error occurred while encoding.

PAK_Stream-Out Report (Warnings)

PAK_WARN - PAK_Stream-Out Report (Warnings)		
Register Space:	MMIO: 0/2/0	
Access:	RO	
Size (in bits):	32	
Address:	128E4h	
DWord	**Bit**	**Description**
0	31:22	**Reserved** Access: RO Format: MBZ
	21	**Skip Run Greater Than 8192 (AVC)**
	20	**Incorrect SkipMB (AVC and mpeg2)**
	19	**Incorrect MV difference for dual-prime MB (mpeg2)**
	18	**End of Slice signal missing on last MB of a Row(mpeg2)**
	17	**Incorrect DCT type for field picture**
	16	**MVs are not within defined range by fcode**
	15:8	**MB Y-position**
	7:0	**MB X-position**

PAK Report Running Status

PAK_REPORT_STAT - PAK Report Running Status

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	128ECh

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**PAK Status** 0 – PAK engine is IDLE 1 – PAK engine is currently generating bit stream.

PAL_EXT_GC_MAX

PAL_EXT_GC_MAX	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	96
Address:	4A420h-4A42Bh
Name:	Pipe Extended Gamma Correction Max
ShortName:	PAL_EXT_GC_MAX_A
Reset:	soft
Address:	4AC20h-4AC2Bh
Name:	Pipe Extended Gamma Correction Max
ShortName:	PAL_EXT_GC_MAX_B
Reset:	soft
Address:	4B420h-4B42Bh
Name:	Pipe Extended Gamma Correction Max
ShortName:	PAL_EXT_GC_MAX_C
Reset:	soft
Address:	4BC20h-4BC2Bh
Name:	Pipe Extended Gamma Correction Max
ShortName:	PAL_EXT_GC_MAX_D
Reset:	soft

DWord	Bit	Description
0	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**Red Ext Max GC Point** Default Value: 11111111111111111b Format: U3.16 The extended point for red color channel gamma correction. This value is represented in a 3.16 format with 3 integer and 16 fractional bits.
1	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**Green Ext Max GC Point** Default Value: 11111111111111111b Format: U3.16 The extended point for green color channel gamma correction. This value is represented in a

<table>
<tr><th colspan="4">PAL_EXT_GC_MAX</th></tr>
<tr><td></td><td></td><td colspan="2">3.16 format with 3 integer and 16 fractional bits.</td></tr>
<tr><td rowspan="4">2</td><td rowspan="3">31:19</td><td colspan="2">Reserved</td></tr>
<tr><td>Access:</td><td>RO</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
<tr><td>18:0</td><td colspan="2">Blue Ext Max GC Point
Default Value: 11111111111111111b
Format: U3.16
The extended point for blue color channel gamma correction. This value is represented in a 3.16 format with 3 integer and 16 fractional bits.</td></tr>
</table>

PAL_EXT2_GC_MAX

PAL_EXT2_GC_MAX	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	96
Address:	4A430h-4A43Bh
Name:	Pipe Extended Second Gamma Correction Max
ShortName:	PAL_EXT2_GC_MAX_A
Reset:	soft
Address:	4AC30h-4AC3Bh
Name:	Pipe Extended Second Gamma Correction Max
ShortName:	PAL_EXT2_GC_MAX_B
Reset:	soft
Address:	4B430h-4B43Bh
Name:	Pipe Extended Second Gamma Correction Max
ShortName:	PAL_EXT2_GC_MAX_C
Reset:	soft
Address:	4BC30h-4BC3Bh
Name:	Pipe Extended Second Gamma Correction Max
ShortName:	PAL_EXT2_GC_MAX_D
Reset:	soft

DWord	Bit	Description
0	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**Red Ext Max GC Point** Default Value: 1111111111111111b Format: U3.16 The extended point for red color channel gamma correction. This value is represented in a 3.16 format with 3 integer and 16 fractional bits.
1	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**Green Ext Max GC Point** Default Value: 1111111111111111b Format: U3.16 The extended point for green color channel gamma correction. This value is represented in a

<table>
<tr><th colspan="3">PAL_EXT2_GC_MAX</th></tr>
<tr><td></td><td></td><td>3.16 format with 3 integer and 16 fractional bits.</td></tr>
<tr><td rowspan="2">2</td><td>31:19</td><td>Reserved<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>18:0</td><td>Blue Ext Max GC Point<table><tr><td>Default Value:</td><td>11111111111111111b</td></tr><tr><td>Format:</td><td>U3.16</td></tr></table>The extended point for blue color channel gamma correction. This value is represented in a 3.16 format with 3 integer and 16 fractional bits.</td></tr>
</table>

PAL_GC_MAX

PAL_GC_MAX	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	96
Address:	4A410h-4A41Bh
Name:	Pipe Gamma Correction Max
ShortName:	PAL_GC_MAX_A
Reset:	soft
Address:	4AC10h-4AC1Bh
Name:	Pipe Gamma Correction Max
ShortName:	PAL_GC_MAX_B
Reset:	soft
Address:	4B410h-4B41Bh
Name:	Pipe Gamma Correction Max
ShortName:	PAL_GC_MAX_C
Reset:	soft
Address:	4BC10h-4BC1Bh
Name:	Pipe Gamma Correction Max
ShortName:	PAL_GC_MAX_D
Reset:	soft

DWord	Bit	Description
0	31:17	**Reserved** Access: RO Format: MBZ
	16:0	**Red Max GC Point** Default Value: 10000000000000000b Format: U1.16 The 513th entry for the red color channel of the 12 bit interpolated gamma correction. This value is represented in a 1.16 format with 1 integer and 16 fractional bits. **Restriction** The value should always be programmed to be less than or equal to 1.0.
1	31:17	**Reserved** Access: RO Format: MBZ

PAL_GC_MAX

	16:0	**Green Max GC Point** Default Value: 10000000000000000b Format: U1.16 **Description** The 513th entry for the green color channel of the 12 bit interpolated gamma correction. This value is represented in a 1.16 format with 1 integer and 16 fractional bits. **Restriction** The value should always be programmed to be less than or equal to 1.0.
2	31:17	**Reserved** Access: RO Format: MBZ
	16:0	**Blue Max GC Point** Default Value: 10000000000000000b Format: U1.16 **Description** The 513th entry for the blue color channel of the 12 bit interpolated gamma correction. This value is represented in a 1.16 format with 1 integer and 16 fractional bits. **Restriction** The value should always be programmed to be less than or equal to 1.0.

PAL_LGC

PAL_LGC	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	4A000h-4A003h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_0_A
Reset:	soft
Address:	4A004h-4A007h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_1_A
Reset:	soft
Address:	4A008h-4A00Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_2_A
Reset:	soft
Address:	4A00Ch-4A00Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_3_A
Reset:	soft
Address:	4A010h-4A013h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_4_A
Reset:	soft
Address:	4A014h-4A017h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_5_A
Reset:	soft
Address:	4A018h-4A01Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_6_A
Reset:	soft
Address:	4A01Ch-4A01Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_7_A

PAL_LGC	
Reset:	soft
Address:	4A020h-4A023h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_8_A
Reset:	soft
Address:	4A024h-4A027h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_9_A
Reset:	soft
Address:	4A028h-4A02Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_10_A
Reset:	soft
Address:	4A02Ch-4A02Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_11_A
Reset:	soft
Address:	4A030h-4A033h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_12_A
Reset:	soft
Address:	4A034h-4A037h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_13_A
Reset:	soft
Address:	4A038h-4A03Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_14_A
Reset:	soft
Address:	4A03Ch-4A03Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_15_A
Reset:	soft
Address:	4A040h-4A043h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_16_A

PAL_LGC	
Reset:	soft
Address:	4A044h-4A047h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_17_A
Reset:	soft
Address:	4A048h-4A04Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_18_A
Reset:	soft
Address:	4A04Ch-4A04Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_19_A
Reset:	soft
Address:	4A050h-4A053h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_20_A
Reset:	soft
Address:	4A054h-4A057h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_21_A
Reset:	soft
Address:	4A058h-4A05Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_22_A
Reset:	soft
Address:	4A05Ch-4A05Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_23_A
Reset:	soft
Address:	4A060h-4A063h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_24_A
Reset:	soft
Address:	4A064h-4A067h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_25_A

PAL_LGC	
Reset:	soft
Address:	4A068h-4A06Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_26_A
Reset:	soft
Address:	4A06Ch-4A06Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_27_A
Reset:	soft
Address:	4A070h-4A073h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_28_A
Reset:	soft
Address:	4A074h-4A077h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_29_A
Reset:	soft
Address:	4A078h-4A07Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_30_A
Reset:	soft
Address:	4A07Ch-4A07Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_31_A
Reset:	soft
Address:	4A080h-4A083h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_32_A
Reset:	soft
Address:	4A084h-4A087h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_33_A
Reset:	soft
Address:	4A088h-4A08Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_34_A

PAL_LGC	
Reset:	soft
Address:	4A08Ch-4A08Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_35_A
Reset:	soft
Address:	4A090h-4A093h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_36_A
Reset:	soft
Address:	4A094h-4A097h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_37_A
Reset:	soft
Address:	4A098h-4A09Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_38_A
Reset:	soft
Address:	4A09Ch-4A09Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_39_A
Reset:	soft
Address:	4A0A0h-4A0A3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_40_A
Reset:	soft
Address:	4A0A4h-4A0A7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_41_A
Reset:	soft
Address:	4A0A8h-4A0ABh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_42_A
Reset:	soft
Address:	4A0ACh-4A0AFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_43_A

PAL_LGC	
Reset:	soft
Address:	4A0B0h-4A0B3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_44_A
Reset:	soft
Address:	4A0B4h-4A0B7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_45_A
Reset:	soft
Address:	4A0B8h-4A0BBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_46_A
Reset:	soft
Address:	4A0BCh-4A0BFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_47_A
Reset:	soft
Address:	4A0C0h-4A0C3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_48_A
Reset:	soft
Address:	4A0C4h-4A0C7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_49_A
Reset:	soft
Address:	4A0C8h-4A0CBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_50_A
Reset:	soft
Address:	4A0CCh-4A0CFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_51_A
Reset:	soft
Address:	4A0D0h-4A0D3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_52_A

PAL_LGC	
Reset:	soft
Address:	4A0D4h-4A0D7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_53_A
Reset:	soft
Address:	4A0D8h-4A0DBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_54_A
Reset:	soft
Address:	4A0DCh-4A0DFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_55_A
Reset:	soft
Address:	4A0E0h-4A0E3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_56_A
Reset:	soft
Address:	4A0E4h-4A0E7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_57_A
Reset:	soft
Address:	4A0E8h-4A0EBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_58_A
Reset:	soft
Address:	4A0ECh-4A0EFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_59_A
Reset:	soft
Address:	4A0F0h-4A0F3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_60_A
Reset:	soft
Address:	4A0F4h-4A0F7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_61_A

PAL_LGC	
Reset:	soft
Address:	4A0F8h-4A0FBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_62_A
Reset:	soft
Address:	4A0FCh-4A0FFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_63_A
Reset:	soft
Address:	4A100h-4A103h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_64_A
Reset:	soft
Address:	4A104h-4A107h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_65_A
Reset:	soft
Address:	4A108h-4A10Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_66_A
Reset:	soft
Address:	4A10Ch-4A10Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_67_A
Reset:	soft
Address:	4A110h-4A113h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_68_A
Reset:	soft
Address:	4A114h-4A117h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_69_A
Reset:	soft
Address:	4A118h-4A11Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_70_A

PAL_LGC	
Reset:	soft
Address:	4A11Ch-4A11Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_71_A
Reset:	soft
Address:	4A120h-4A123h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_72_A
Reset:	soft
Address:	4A124h-4A127h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_73_A
Reset:	soft
Address:	4A128h-4A12Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_74_A
Reset:	soft
Address:	4A12Ch-4A12Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_75_A
Reset:	soft
Address:	4A130h-4A133h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_76_A
Reset:	soft
Address:	4A134h-4A137h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_77_A
Reset:	soft
Address:	4A138h-4A13Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_78_A
Reset:	soft
Address:	4A13Ch-4A13Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_79_A

PAL_LGC	
Reset:	soft
Address:	4A140h-4A143h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_80_A
Reset:	soft
Address:	4A144h-4A147h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_81_A
Reset:	soft
Address:	4A148h-4A14Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_82_A
Reset:	soft
Address:	4A14Ch-4A14Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_83_A
Reset:	soft
Address:	4A150h-4A153h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_84_A
Reset:	soft
Address:	4A154h-4A157h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_85_A
Reset:	soft
Address:	4A158h-4A15Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_86_A
Reset:	soft
Address:	4A15Ch-4A15Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_87_A
Reset:	soft
Address:	4A160h-4A163h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_88_A

PAL_LGC	
Reset:	soft
Address:	4A164h-4A167h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_89_A
Reset:	soft
Address:	4A168h-4A16Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_90_A
Reset:	soft
Address:	4A16Ch-4A16Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_91_A
Reset:	soft
Address:	4A170h-4A173h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_92_A
Reset:	soft
Address:	4A174h-4A177h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_93_A
Reset:	soft
Address:	4A178h-4A17Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_94_A
Reset:	soft
Address:	4A17Ch-4A17Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_95_A
Reset:	soft
Address:	4A180h-4A183h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_96_A
Reset:	soft
Address:	4A184h-4A187h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_97_A

PAL_LGC	
Reset:	soft
Address:	4A188h-4A18Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_98_A
Reset:	soft
Address:	4A18Ch-4A18Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_99_A
Reset:	soft
Address:	4A190h-4A193h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_100_A
Reset:	soft
Address:	4A194h-4A197h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_101_A
Reset:	soft
Address:	4A198h-4A19Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_102_A
Reset:	soft
Address:	4A19Ch-4A19Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_103_A
Reset:	soft
Address:	4A1A0h-4A1A3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_104_A
Reset:	soft
Address:	4A1A4h-4A1A7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_105_A
Reset:	soft
Address:	4A1A8h-4A1ABh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_106_A

PAL_LGC	
Reset:	soft
Address:	4A1ACh-4A1AFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_107_A
Reset:	soft
Address:	4A1B0h-4A1B3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_108_A
Reset:	soft
Address:	4A1B4h-4A1B7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_109_A
Reset:	soft
Address:	4A1B8h-4A1BBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_110_A
Reset:	soft
Address:	4A1BCh-4A1BFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_111_A
Reset:	soft
Address:	4A1C0h-4A1C3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_112_A
Reset:	soft
Address:	4A1C4h-4A1C7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_113_A
Reset:	soft
Address:	4A1C8h-4A1CBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_114_A
Reset:	soft
Address:	4A1CCh-4A1CFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_115_A

PAL_LGC	
Reset:	soft
Address:	4A1D0h-4A1D3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_116_A
Reset:	soft
Address:	4A1D4h-4A1D7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_117_A
Reset:	soft
Address:	4A1D8h-4A1DBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_118_A
Reset:	soft
Address:	4A1DCh-4A1DFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_119_A
Reset:	soft
Address:	4A1E0h-4A1E3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_120_A
Reset:	soft
Address:	4A1E4h-4A1E7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_121_A
Reset:	soft
Address:	4A1E8h-4A1EBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_122_A
Reset:	soft
Address:	4A1ECh-4A1EFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_123_A
Reset:	soft
Address:	4A1F0h-4A1F3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_124_A

PAL_LGC	
Reset:	soft
Address:	4A1F4h-4A1F7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_125_A
Reset:	soft
Address:	4A1F8h-4A1FBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_126_A
Reset:	soft
Address:	4A1FCh-4A1FFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_127_A
Reset:	soft
Address:	4A200h-4A203h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_128_A
Reset:	soft
Address:	4A204h-4A207h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_129_A
Reset:	soft
Address:	4A208h-4A20Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_130_A
Reset:	soft
Address:	4A20Ch-4A20Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_131_A
Reset:	soft
Address:	4A210h-4A213h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_132_A
Reset:	soft
Address:	4A214h-4A217h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_133_A

PAL_LGC	
Reset:	soft
Address:	4A218h-4A21Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_134_A
Reset:	soft
Address:	4A21Ch-4A21Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_135_A
Reset:	soft
Address:	4A220h-4A223h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_136_A
Reset:	soft
Address:	4A224h-4A227h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_137_A
Reset:	soft
Address:	4A228h-4A22Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_138_A
Reset:	soft
Address:	4A22Ch-4A22Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_139_A
Reset:	soft
Address:	4A230h-4A233h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_140_A
Reset:	soft
Address:	4A234h-4A237h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_141_A
Reset:	soft
Address:	4A238h-4A23Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_142_A

PAL_LGC	
Reset:	soft
Address:	4A23Ch-4A23Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_143_A
Reset:	soft
Address:	4A240h-4A243h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_144_A
Reset:	soft
Address:	4A244h-4A247h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_145_A
Reset:	soft
Address:	4A248h-4A24Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_146_A
Reset:	soft
Address:	4A24Ch-4A24Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_147_A
Reset:	soft
Address:	4A250h-4A253h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_148_A
Reset:	soft
Address:	4A254h-4A257h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_149_A
Reset:	soft
Address:	4A258h-4A25Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_150_A
Reset:	soft
Address:	4A25Ch-4A25Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_151_A

PAL_LGC	
Reset:	soft
Address:	4A260h-4A263h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_152_A
Reset:	soft
Address:	4A264h-4A267h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_153_A
Reset:	soft
Address:	4A268h-4A26Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_154_A
Reset:	soft
Address:	4A26Ch-4A26Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_155_A
Reset:	soft
Address:	4A270h-4A273h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_156_A
Reset:	soft
Address:	4A274h-4A277h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_157_A
Reset:	soft
Address:	4A278h-4A27Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_158_A
Reset:	soft
Address:	4A27Ch-4A27Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_159_A
Reset:	soft
Address:	4A280h-4A283h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_160_A

PAL_LGC	
Reset:	soft
Address:	4A284h-4A287h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_161_A
Reset:	soft
Address:	4A288h-4A28Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_162_A
Reset:	soft
Address:	4A28Ch-4A28Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_163_A
Reset:	soft
Address:	4A290h-4A293h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_164_A
Reset:	soft
Address:	4A294h-4A297h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_165_A
Reset:	soft
Address:	4A298h-4A29Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_166_A
Reset:	soft
Address:	4A29Ch-4A29Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_167_A
Reset:	soft
Address:	4A2A0h-4A2A3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_168_A
Reset:	soft
Address:	4A2A4h-4A2A7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_169_A

PAL_LGC	
Reset:	soft
Address:	4A2A8h-4A2ABh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_170_A
Reset:	soft
Address:	4A2ACh-4A2AFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_171_A
Reset:	soft
Address:	4A2B0h-4A2B3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_172_A
Reset:	soft
Address:	4A2B4h-4A2B7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_173_A
Reset:	soft
Address:	4A2B8h-4A2BBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_174_A
Reset:	soft
Address:	4A2BCh-4A2BFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_175_A
Reset:	soft
Address:	4A2C0h-4A2C3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_176_A
Reset:	soft
Address:	4A2C4h-4A2C7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_177_A
Reset:	soft
Address:	4A2C8h-4A2CBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_178_A

PAL_LGC	
Reset:	soft
Address:	4A2CCh-4A2CFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_179_A
Reset:	soft
Address:	4A2D0h-4A2D3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_180_A
Reset:	soft
Address:	4A2D4h-4A2D7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_181_A
Reset:	soft
Address:	4A2D8h-4A2DBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_182_A
Reset:	soft
Address:	4A2DCh-4A2DFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_183_A
Reset:	soft
Address:	4A2E0h-4A2E3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_184_A
Reset:	soft
Address:	4A2E4h-4A2E7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_185_A
Reset:	soft
Address:	4A2E8h-4A2EBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_186_A
Reset:	soft
Address:	4A2ECh-4A2EFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_187_A

PAL_LGC	
Reset:	soft
Address:	4A2F0h-4A2F3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_188_A
Reset:	soft
Address:	4A2F4h-4A2F7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_189_A
Reset:	soft
Address:	4A2F8h-4A2FBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_190_A
Reset:	soft
Address:	4A2FCh-4A2FFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_191_A
Reset:	soft
Address:	4A300h-4A303h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_192_A
Reset:	soft
Address:	4A304h-4A307h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_193_A
Reset:	soft
Address:	4A308h-4A30Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_194_A
Reset:	soft
Address:	4A30Ch-4A30Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_195_A
Reset:	soft
Address:	4A310h-4A313h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_196_A

PAL_LGC	
Reset:	soft
Address:	4A314h-4A317h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_197_A
Reset:	soft
Address:	4A318h-4A31Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_198_A
Reset:	soft
Address:	4A31Ch-4A31Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_199_A
Reset:	soft
Address:	4A320h-4A323h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_200_A
Reset:	soft
Address:	4A324h-4A327h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_201_A
Reset:	soft
Address:	4A328h-4A32Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_202_A
Reset:	soft
Address:	4A32Ch-4A32Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_203_A
Reset:	soft
Address:	4A330h-4A333h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_204_A
Reset:	soft
Address:	4A334h-4A337h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_205_A

PAL_LGC	
Reset:	soft
Address:	4A338h-4A33Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_206_A
Reset:	soft
Address:	4A33Ch-4A33Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_207_A
Reset:	soft
Address:	4A340h-4A343h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_208_A
Reset:	soft
Address:	4A344h-4A347h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_209_A
Reset:	soft
Address:	4A348h-4A34Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_210_A
Reset:	soft
Address:	4A34Ch-4A34Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_211_A
Reset:	soft
Address:	4A350h-4A353h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_212_A
Reset:	soft
Address:	4A354h-4A357h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_213_A
Reset:	soft
Address:	4A358h-4A35Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_214_A

PAL_LGC	
Reset:	soft
Address:	4A35Ch-4A35Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_215_A
Reset:	soft
Address:	4A360h-4A363h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_216_A
Reset:	soft
Address:	4A364h-4A367h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_217_A
Reset:	soft
Address:	4A368h-4A36Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_218_A
Reset:	soft
Address:	4A36Ch-4A36Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_219_A
Reset:	soft
Address:	4A370h-4A373h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_220_A
Reset:	soft
Address:	4A374h-4A377h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_221_A
Reset:	soft
Address:	4A378h-4A37Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_222_A
Reset:	soft
Address:	4A37Ch-4A37Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_223_A

PAL_LGC	
Reset:	soft
Address:	4A380h-4A383h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_224_A
Reset:	soft
Address:	4A384h-4A387h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_225_A
Reset:	soft
Address:	4A388h-4A38Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_226_A
Reset:	soft
Address:	4A38Ch-4A38Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_227_A
Reset:	soft
Address:	4A390h-4A393h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_228_A
Reset:	soft
Address:	4A394h-4A397h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_229_A
Reset:	soft
Address:	4A398h-4A39Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_230_A
Reset:	soft
Address:	4A39Ch-4A39Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_231_A
Reset:	soft
Address:	4A3A0h-4A3A3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_232_A

PAL_LGC	
Reset:	soft
Address:	4A3A4h-4A3A7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_233_A
Reset:	soft
Address:	4A3A8h-4A3ABh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_234_A
Reset:	soft
Address:	4A3ACh-4A3AFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_235_A
Reset:	soft
Address:	4A3B0h-4A3B3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_236_A
Reset:	soft
Address:	4A3B4h-4A3B7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_237_A
Reset:	soft
Address:	4A3B8h-4A3BBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_238_A
Reset:	soft
Address:	4A3BCh-4A3BFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_239_A
Reset:	soft
Address:	4A3C0h-4A3C3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_240_A
Reset:	soft
Address:	4A3C4h-4A3C7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_241_A

PAL_LGC	
Reset:	soft
Address:	4A3C8h-4A3CBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_242_A
Reset:	soft
Address:	4A3CCh-4A3CFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_243_A
Reset:	soft
Address:	4A3D0h-4A3D3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_244_A
Reset:	soft
Address:	4A3D4h-4A3D7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_245_A
Reset:	soft
Address:	4A3D8h-4A3DBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_246_A
Reset:	soft
Address:	4A3DCh-4A3DFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_247_A
Reset:	soft
Address:	4A3E0h-4A3E3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_248_A
Reset:	soft
Address:	4A3E4h-4A3E7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_249_A
Reset:	soft
Address:	4A3E8h-4A3EBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_250_A

PAL_LGC	
Reset:	soft
Address:	4A3ECh-4A3EFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_251_A
Reset:	soft
Address:	4A3F0h-4A3F3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_252_A
Reset:	soft
Address:	4A3F4h-4A3F7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_253_A
Reset:	soft
Address:	4A3F8h-4A3FBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_254_A
Reset:	soft
Address:	4A3FCh-4A3FFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_255_A
Reset:	soft
Address:	4A800h-4A803h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_0_B
Reset:	soft
Address:	4A804h-4A807h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_1_B
Reset:	soft
Address:	4A808h-4A80Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_2_B
Reset:	soft
Address:	4A80Ch-4A80Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_3_B

PAL_LGC	
Reset:	soft
Address:	4A810h-4A813h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_4_B
Reset:	soft
Address:	4A814h-4A817h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_5_B
Reset:	soft
Address:	4A818h-4A81Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_6_B
Reset:	soft
Address:	4A81Ch-4A81Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_7_B
Reset:	soft
Address:	4A820h-4A823h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_8_B
Reset:	soft
Address:	4A824h-4A827h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_9_B
Reset:	soft
Address:	4A828h-4A82Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_10_B
Reset:	soft
Address:	4A82Ch-4A82Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_11_B
Reset:	soft
Address:	4A830h-4A833h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_12_B

PAL_LGC	
Reset:	soft
Address:	4A834h-4A837h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_13_B
Reset:	soft
Address:	4A838h-4A83Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_14_B
Reset:	soft
Address:	4A83Ch-4A83Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_15_B
Reset:	soft
Address:	4A840h-4A843h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_16_B
Reset:	soft
Address:	4A844h-4A847h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_17_B
Reset:	soft
Address:	4A848h-4A84Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_18_B
Reset:	soft
Address:	4A84Ch-4A84Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_19_B
Reset:	soft
Address:	4A850h-4A853h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_20_B
Reset:	soft
Address:	4A854h-4A857h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_21_B

PAL_LGC	
Reset:	soft
Address:	4A858h-4A85Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_22_B
Reset:	soft
Address:	4A85Ch-4A85Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_23_B
Reset:	soft
Address:	4A860h-4A863h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_24_B
Reset:	soft
Address:	4A864h-4A867h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_25_B
Reset:	soft
Address:	4A868h-4A86Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_26_B
Reset:	soft
Address:	4A86Ch-4A86Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_27_B
Reset:	soft
Address:	4A870h-4A873h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_28_B
Reset:	soft
Address:	4A874h-4A877h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_29_B
Reset:	soft
Address:	4A878h-4A87Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_30_B

PAL_LGC	
Reset:	soft
Address:	4A87Ch-4A87Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_31_B
Reset:	soft
Address:	4A880h-4A883h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_32_B
Reset:	soft
Address:	4A884h-4A887h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_33_B
Reset:	soft
Address:	4A888h-4A88Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_34_B
Reset:	soft
Address:	4A88Ch-4A88Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_35_B
Reset:	soft
Address:	4A890h-4A893h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_36_B
Reset:	soft
Address:	4A894h-4A897h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_37_B
Reset:	soft
Address:	4A898h-4A89Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_38_B
Reset:	soft
Address:	4A89Ch-4A89Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_39_B

PAL_LGC	
Reset:	soft
Address:	4A8A0h-4A8A3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_40_B
Reset:	soft
Address:	4A8A4h-4A8A7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_41_B
Reset:	soft
Address:	4A8A8h-4A8ABh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_42_B
Reset:	soft
Address:	4A8ACh-4A8AFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_43_B
Reset:	soft
Address:	4A8B0h-4A8B3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_44_B
Reset:	soft
Address:	4A8B4h-4A8B7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_45_B
Reset:	soft
Address:	4A8B8h-4A8BBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_46_B
Reset:	soft
Address:	4A8BCh-4A8BFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_47_B
Reset:	soft
Address:	4A8C0h-4A8C3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_48_B

PAL_LGC	
Reset:	soft
Address:	4A8C4h-4A8C7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_49_B
Reset:	soft
Address:	4A8C8h-4A8CBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_50_B
Reset:	soft
Address:	4A8CCh-4A8CFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_51_B
Reset:	soft
Address:	4A8D0h-4A8D3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_52_B
Reset:	soft
Address:	4A8D4h-4A8D7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_53_B
Reset:	soft
Address:	4A8D8h-4A8DBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_54_B
Reset:	soft
Address:	4A8DCh-4A8DFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_55_B
Reset:	soft
Address:	4A8E0h-4A8E3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_56_B
Reset:	soft
Address:	4A8E4h-4A8E7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_57_B

PAL_LGC	
Reset:	soft
Address:	4A8E8h-4A8EBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_58_B
Reset:	soft
Address:	4A8ECh-4A8EFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_59_B
Reset:	soft
Address:	4A8F0h-4A8F3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_60_B
Reset:	soft
Address:	4A8F4h-4A8F7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_61_B
Reset:	soft
Address:	4A8F8h-4A8FBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_62_B
Reset:	soft
Address:	4A8FCh-4A8FFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_63_B
Reset:	soft
Address:	4A900h-4A903h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_64_B
Reset:	soft
Address:	4A904h-4A907h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_65_B
Reset:	soft
Address:	4A908h-4A90Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_66_B

PAL_LGC	
Reset:	soft
Address:	4A90Ch-4A90Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_67_B
Reset:	soft
Address:	4A910h-4A913h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_68_B
Reset:	soft
Address:	4A914h-4A917h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_69_B
Reset:	soft
Address:	4A918h-4A91Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_70_B
Reset:	soft
Address:	4A91Ch-4A91Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_71_B
Reset:	soft
Address:	4A920h-4A923h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_72_B
Reset:	soft
Address:	4A924h-4A927h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_73_B
Reset:	soft
Address:	4A928h-4A92Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_74_B
Reset:	soft
Address:	4A92Ch-4A92Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_75_B

PAL_LGC	
Reset:	soft
Address:	4A930h-4A933h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_76_B
Reset:	soft
Address:	4A934h-4A937h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_77_B
Reset:	soft
Address:	4A938h-4A93Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_78_B
Reset:	soft
Address:	4A93Ch-4A93Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_79_B
Reset:	soft
Address:	4A940h-4A943h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_80_B
Reset:	soft
Address:	4A944h-4A947h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_81_B
Reset:	soft
Address:	4A948h-4A94Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_82_B
Reset:	soft
Address:	4A94Ch-4A94Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_83_B
Reset:	soft
Address:	4A950h-4A953h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_84_B

PAL_LGC	
Reset:	soft
Address:	4A954h-4A957h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_85_B
Reset:	soft
Address:	4A958h-4A95Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_86_B
Reset:	soft
Address:	4A95Ch-4A95Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_87_B
Reset:	soft
Address:	4A960h-4A963h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_88_B
Reset:	soft
Address:	4A964h-4A967h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_89_B
Reset:	soft
Address:	4A968h-4A96Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_90_B
Reset:	soft
Address:	4A96Ch-4A96Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_91_B
Reset:	soft
Address:	4A970h-4A973h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_92_B
Reset:	soft
Address:	4A974h-4A977h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_93_B

PAL_LGC	
Reset:	soft
Address:	4A978h-4A97Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_94_B
Reset:	soft
Address:	4A97Ch-4A97Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_95_B
Reset:	soft
Address:	4A980h-4A983h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_96_B
Reset:	soft
Address:	4A984h-4A987h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_97_B
Reset:	soft
Address:	4A988h-4A98Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_98_B
Reset:	soft
Address:	4A98Ch-4A98Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_99_B
Reset:	soft
Address:	4A990h-4A993h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_100_B
Reset:	soft
Address:	4A994h-4A997h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_101_B
Reset:	soft
Address:	4A998h-4A99Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_102_B

PAL_LGC	
Reset:	soft
Address:	4A99Ch-4A99Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_103_B
Reset:	soft
Address:	4A9A0h-4A9A3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_104_B
Reset:	soft
Address:	4A9A4h-4A9A7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_105_B
Reset:	soft
Address:	4A9A8h-4A9ABh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_106_B
Reset:	soft
Address:	4A9ACh-4A9AFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_107_B
Reset:	soft
Address:	4A9B0h-4A9B3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_108_B
Reset:	soft
Address:	4A9B4h-4A9B7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_109_B
Reset:	soft
Address:	4A9B8h-4A9BBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_110_B
Reset:	soft
Address:	4A9BCh-4A9BFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_111_B

PAL_LGC	
Reset:	soft
Address:	4A9C0h-4A9C3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_112_B
Reset:	soft
Address:	4A9C4h-4A9C7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_113_B
Reset:	soft
Address:	4A9C8h-4A9CBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_114_B
Reset:	soft
Address:	4A9CCh-4A9CFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_115_B
Reset:	soft
Address:	4A9D0h-4A9D3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_116_B
Reset:	soft
Address:	4A9D4h-4A9D7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_117_B
Reset:	soft
Address:	4A9D8h-4A9DBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_118_B
Reset:	soft
Address:	4A9DCh-4A9DFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_119_B
Reset:	soft
Address:	4A9E0h-4A9E3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_120_B

PAL_LGC	
Reset:	soft
Address:	4A9E4h-4A9E7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_121_B
Reset:	soft
Address:	4A9E8h-4A9EBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_122_B
Reset:	soft
Address:	4A9ECh-4A9EFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_123_B
Reset:	soft
Address:	4A9F0h-4A9F3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_124_B
Reset:	soft
Address:	4A9F4h-4A9F7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_125_B
Reset:	soft
Address:	4A9F8h-4A9FBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_126_B
Reset:	soft
Address:	4A9FCh-4A9FFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_127_B
Reset:	soft
Address:	4AA00h-4AA03h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_128_B
Reset:	soft
Address:	4AA04h-4AA07h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_129_B

PAL_LGC	
Reset:	soft
Address:	4AA08h-4AA0Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_130_B
Reset:	soft
Address:	4AA0Ch-4AA0Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_131_B
Reset:	soft
Address:	4AA10h-4AA13h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_132_B
Reset:	soft
Address:	4AA14h-4AA17h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_133_B
Reset:	soft
Address:	4AA18h-4AA1Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_134_B
Reset:	soft
Address:	4AA1Ch-4AA1Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_135_B
Reset:	soft
Address:	4AA20h-4AA23h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_136_B
Reset:	soft
Address:	4AA24h-4AA27h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_137_B
Reset:	soft
Address:	4AA28h-4AA2Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_138_B

PAL_LGC	
Reset:	soft
Address:	4AA2Ch-4AA2Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_139_B
Reset:	soft
Address:	4AA30h-4AA33h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_140_B
Reset:	soft
Address:	4AA34h-4AA37h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_141_B
Reset:	soft
Address:	4AA38h-4AA3Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_142_B
Reset:	soft
Address:	4AA3Ch-4AA3Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_143_B
Reset:	soft
Address:	4AA40h-4AA43h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_144_B
Reset:	soft
Address:	4AA44h-4AA47h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_145_B
Reset:	soft
Address:	4AA48h-4AA4Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_146_B
Reset:	soft
Address:	4AA4Ch-4AA4Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_147_B

PAL_LGC	
Reset:	soft
Address:	4AA50h-4AA53h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_148_B
Reset:	soft
Address:	4AA54h-4AA57h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_149_B
Reset:	soft
Address:	4AA58h-4AA5Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_150_B
Reset:	soft
Address:	4AA5Ch-4AA5Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_151_B
Reset:	soft
Address:	4AA60h-4AA63h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_152_B
Reset:	soft
Address:	4AA64h-4AA67h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_153_B
Reset:	soft
Address:	4AA68h-4AA6Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_154_B
Reset:	soft
Address:	4AA6Ch-4AA6Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_155_B
Reset:	soft
Address:	4AA70h-4AA73h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_156_B

PAL_LGC	
Reset:	soft
Address:	4AA74h-4AA77h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_157_B
Reset:	soft
Address:	4AA78h-4AA7Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_158_B
Reset:	soft
Address:	4AA7Ch-4AA7Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_159_B
Reset:	soft
Address:	4AA80h-4AA83h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_160_B
Reset:	soft
Address:	4AA84h-4AA87h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_161_B
Reset:	soft
Address:	4AA88h-4AA8Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_162_B
Reset:	soft
Address:	4AA8Ch-4AA8Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_163_B
Reset:	soft
Address:	4AA90h-4AA93h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_164_B
Reset:	soft
Address:	4AA94h-4AA97h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_165_B

PAL_LGC	
Reset:	soft
Address:	4AA98h-4AA9Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_166_B
Reset:	soft
Address:	4AA9Ch-4AA9Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_167_B
Reset:	soft
Address:	4AAA0h-4AAA3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_168_B
Reset:	soft
Address:	4AAA4h-4AAA7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_169_B
Reset:	soft
Address:	4AAA8h-4AAABh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_170_B
Reset:	soft
Address:	4AAACh-4AAAFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_171_B
Reset:	soft
Address:	4AAB0h-4AAB3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_172_B
Reset:	soft
Address:	4AAB4h-4AAB7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_173_B
Reset:	soft
Address:	4AAB8h-4AABBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_174_B

PAL_LGC	
Reset:	soft
Address:	4AABCh-4AABFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_175_B
Reset:	soft
Address:	4AAC0h-4AAC3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_176_B
Reset:	soft
Address:	4AAC4h-4AAC7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_177_B
Reset:	soft
Address:	4AAC8h-4AACBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_178_B
Reset:	soft
Address:	4AACCh-4AACFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_179_B
Reset:	soft
Address:	4AAD0h-4AAD3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_180_B
Reset:	soft
Address:	4AAD4h-4AAD7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_181_B
Reset:	soft
Address:	4AAD8h-4AADBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_182_B
Reset:	soft
Address:	4AADCh-4AADFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_183_B

PAL_LGC	
Reset:	soft
Address:	4AAE0h-4AAE3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_184_B
Reset:	soft
Address:	4AAE4h-4AAE7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_185_B
Reset:	soft
Address:	4AAE8h-4AAEBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_186_B
Reset:	soft
Address:	4AAECh-4AAEFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_187_B
Reset:	soft
Address:	4AAF0h-4AAF3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_188_B
Reset:	soft
Address:	4AAF4h-4AAF7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_189_B
Reset:	soft
Address:	4AAF8h-4AAFBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_190_B
Reset:	soft
Address:	4AAFCh-4AAFFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_191_B
Reset:	soft
Address:	4AB00h-4AB03h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_192_B

PAL_LGC	
Reset:	soft
Address:	4AB04h-4AB07h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_193_B
Reset:	soft
Address:	4AB08h-4AB0Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_194_B
Reset:	soft
Address:	4AB0Ch-4AB0Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_195_B
Reset:	soft
Address:	4AB10h-4AB13h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_196_B
Reset:	soft
Address:	4AB14h-4AB17h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_197_B
Reset:	soft
Address:	4AB18h-4AB1Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_198_B
Reset:	soft
Address:	4AB1Ch-4AB1Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_199_B
Reset:	soft
Address:	4AB20h-4AB23h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_200_B
Reset:	soft
Address:	4AB24h-4AB27h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_201_B

PAL_LGC	
Reset:	soft
Address:	4AB28h-4AB2Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_202_B
Reset:	soft
Address:	4AB2Ch-4AB2Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_203_B
Reset:	soft
Address:	4AB30h-4AB33h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_204_B
Reset:	soft
Address:	4AB34h-4AB37h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_205_B
Reset:	soft
Address:	4AB38h-4AB3Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_206_B
Reset:	soft
Address:	4AB3Ch-4AB3Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_207_B
Reset:	soft
Address:	4AB40h-4AB43h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_208_B
Reset:	soft
Address:	4AB44h-4AB47h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_209_B
Reset:	soft
Address:	4AB48h-4AB4Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_210_B

PAL_LGC	
Reset:	soft
Address:	4AB4Ch-4AB4Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_211_B
Reset:	soft
Address:	4AB50h-4AB53h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_212_B
Reset:	soft
Address:	4AB54h-4AB57h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_213_B
Reset:	soft
Address:	4AB58h-4AB5Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_214_B
Reset:	soft
Address:	4AB5Ch-4AB5Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_215_B
Reset:	soft
Address:	4AB60h-4AB63h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_216_B
Reset:	soft
Address:	4AB64h-4AB67h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_217_B
Reset:	soft
Address:	4AB68h-4AB6Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_218_B
Reset:	soft
Address:	4AB6Ch-4AB6Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_219_B

PAL_LGC	
Reset:	soft
Address:	4AB70h-4AB73h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_220_B
Reset:	soft
Address:	4AB74h-4AB77h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_221_B
Reset:	soft
Address:	4AB78h-4AB7Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_222_B
Reset:	soft
Address:	4AB7Ch-4AB7Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_223_B
Reset:	soft
Address:	4AB80h-4AB83h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_224_B
Reset:	soft
Address:	4AB84h-4AB87h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_225_B
Reset:	soft
Address:	4AB88h-4AB8Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_226_B
Reset:	soft
Address:	4AB8Ch-4AB8Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_227_B
Reset:	soft
Address:	4AB90h-4AB93h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_228_B

PAL_LGC	
Reset:	soft
Address:	4AB94h-4AB97h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_229_B
Reset:	soft
Address:	4AB98h-4AB9Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_230_B
Reset:	soft
Address:	4AB9Ch-4AB9Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_231_B
Reset:	soft
Address:	4ABA0h-4ABA3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_232_B
Reset:	soft
Address:	4ABA4h-4ABA7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_233_B
Reset:	soft
Address:	4ABA8h-4ABABh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_234_B
Reset:	soft
Address:	4ABACh-4ABAFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_235_B
Reset:	soft
Address:	4ABB0h-4ABB3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_236_B
Reset:	soft
Address:	4ABB4h-4ABB7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_237_B

PAL_LGC	
Reset:	soft
Address:	4ABB8h-4ABBBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_238_B
Reset:	soft
Address:	4ABBCh-4ABBFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_239_B
Reset:	soft
Address:	4ABC0h-4ABC3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_240_B
Reset:	soft
Address:	4ABC4h-4ABC7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_241_B
Reset:	soft
Address:	4ABC8h-4ABCBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_242_B
Reset:	soft
Address:	4ABCCh-4ABCFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_243_B
Reset:	soft
Address:	4ABD0h-4ABD3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_244_B
Reset:	soft
Address:	4ABD4h-4ABD7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_245_B
Reset:	soft
Address:	4ABD8h-4ABDBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_246_B

PAL_LGC	
Reset:	soft
Address:	4ABDCh-4ABDFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_247_B
Reset:	soft
Address:	4ABE0h-4ABE3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_248_B
Reset:	soft
Address:	4ABE4h-4ABE7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_249_B
Reset:	soft
Address:	4ABE8h-4ABEBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_250_B
Reset:	soft
Address:	4ABECh-4ABEFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_251_B
Reset:	soft
Address:	4ABF0h-4ABF3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_252_B
Reset:	soft
Address:	4ABF4h-4ABF7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_253_B
Reset:	soft
Address:	4ABF8h-4ABFBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_254_B
Reset:	soft
Address:	4ABFCh-4ABFFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_255_B

PAL_LGC	
Reset:	soft
Address:	4B000h-4B003h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_0_C
Reset:	soft
Address:	4B004h-4B007h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_1_C
Reset:	soft
Address:	4B008h-4B00Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_2_C
Reset:	soft
Address:	4B00Ch-4B00Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_3_C
Reset:	soft
Address:	4B010h-4B013h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_4_C
Reset:	soft
Address:	4B014h-4B017h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_5_C
Reset:	soft
Address:	4B018h-4B01Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_6_C
Reset:	soft
Address:	4B01Ch-4B01Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_7_C
Reset:	soft
Address:	4B020h-4B023h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_8_C

PAL_LGC	
Reset:	soft
Address:	4B024h-4B027h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_9_C
Reset:	soft
Address:	4B028h-4B02Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_10_C
Reset:	soft
Address:	4B02Ch-4B02Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_11_C
Reset:	soft
Address:	4B030h-4B033h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_12_C
Reset:	soft
Address:	4B034h-4B037h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_13_C
Reset:	soft
Address:	4B038h-4B03Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_14_C
Reset:	soft
Address:	4B03Ch-4B03Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_15_C
Reset:	soft
Address:	4B040h-4B043h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_16_C
Reset:	soft
Address:	4B044h-4B047h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_17_C

PAL_LGC	
Reset:	soft
Address:	4B048h-4B04Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_18_C
Reset:	soft
Address:	4B04Ch-4B04Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_19_C
Reset:	soft
Address:	4B050h-4B053h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_20_C
Reset:	soft
Address:	4B054h-4B057h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_21_C
Reset:	soft
Address:	4B058h-4B05Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_22_C
Reset:	soft
Address:	4B05Ch-4B05Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_23_C
Reset:	soft
Address:	4B060h-4B063h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_24_C
Reset:	soft
Address:	4B064h-4B067h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_25_C
Reset:	soft
Address:	4B068h-4B06Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_26_C

PAL_LGC	
Reset:	soft
Address:	4B06Ch-4B06Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_27_C
Reset:	soft
Address:	4B070h-4B073h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_28_C
Reset:	soft
Address:	4B074h-4B077h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_29_C
Reset:	soft
Address:	4B078h-4B07Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_30_C
Reset:	soft
Address:	4B07Ch-4B07Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_31_C
Reset:	soft
Address:	4B080h-4B083h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_32_C
Reset:	soft
Address:	4B084h-4B087h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_33_C
Reset:	soft
Address:	4B088h-4B08Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_34_C
Reset:	soft
Address:	4B08Ch-4B08Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_35_C

PAL_LGC	
Reset:	soft
Address:	4B090h-4B093h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_36_C
Reset:	soft
Address:	4B094h-4B097h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_37_C
Reset:	soft
Address:	4B098h-4B09Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_38_C
Reset:	soft
Address:	4B09Ch-4B09Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_39_C
Reset:	soft
Address:	4B0A0h-4B0A3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_40_C
Reset:	soft
Address:	4B0A4h-4B0A7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_41_C
Reset:	soft
Address:	4B0A8h-4B0ABh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_42_C
Reset:	soft
Address:	4B0ACh-4B0AFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_43_C
Reset:	soft
Address:	4B0B0h-4B0B3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_44_C

PAL_LGC	
Reset:	soft
Address:	4B0B4h-4B0B7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_45_C
Reset:	soft
Address:	4B0B8h-4B0BBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_46_C
Reset:	soft
Address:	4B0BCh-4B0BFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_47_C
Reset:	soft
Address:	4B0C0h-4B0C3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_48_C
Reset:	soft
Address:	4B0C4h-4B0C7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_49_C
Reset:	soft
Address:	4B0C8h-4B0CBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_50_C
Reset:	soft
Address:	4B0CCh-4B0CFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_51_C
Reset:	soft
Address:	4B0D0h-4B0D3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_52_C
Reset:	soft
Address:	4B0D4h-4B0D7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_53_C

PAL_LGC	
Reset:	soft
Address:	4B0D8h-4B0DBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_54_C
Reset:	soft
Address:	4B0DCh-4B0DFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_55_C
Reset:	soft
Address:	4B0E0h-4B0E3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_56_C
Reset:	soft
Address:	4B0E4h-4B0E7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_57_C
Reset:	soft
Address:	4B0E8h-4B0EBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_58_C
Reset:	soft
Address:	4B0ECh-4B0EFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_59_C
Reset:	soft
Address:	4B0F0h-4B0F3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_60_C
Reset:	soft
Address:	4B0F4h-4B0F7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_61_C
Reset:	soft
Address:	4B0F8h-4B0FBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_62_C

PAL_LGC	
Reset:	soft
Address:	4B0FCh-4B0FFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_63_C
Reset:	soft
Address:	4B100h-4B103h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_64_C
Reset:	soft
Address:	4B104h-4B107h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_65_C
Reset:	soft
Address:	4B108h-4B10Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_66_C
Reset:	soft
Address:	4B10Ch-4B10Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_67_C
Reset:	soft
Address:	4B110h-4B113h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_68_C
Reset:	soft
Address:	4B114h-4B117h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_69_C
Reset:	soft
Address:	4B118h-4B11Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_70_C
Reset:	soft
Address:	4B11Ch-4B11Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_71_C

PAL_LGC	
Reset:	soft
Address:	4B120h-4B123h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_72_C
Reset:	soft
Address:	4B124h-4B127h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_73_C
Reset:	soft
Address:	4B128h-4B12Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_74_C
Reset:	soft
Address:	4B12Ch-4B12Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_75_C
Reset:	soft
Address:	4B130h-4B133h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_76_C
Reset:	soft
Address:	4B134h-4B137h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_77_C
Reset:	soft
Address:	4B138h-4B13Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_78_C
Reset:	soft
Address:	4B13Ch-4B13Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_79_C
Reset:	soft
Address:	4B140h-4B143h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_80_C

PAL_LGC	
Reset:	soft
Address:	4B144h-4B147h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_81_C
Reset:	soft
Address:	4B148h-4B14Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_82_C
Reset:	soft
Address:	4B14Ch-4B14Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_83_C
Reset:	soft
Address:	4B150h-4B153h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_84_C
Reset:	soft
Address:	4B154h-4B157h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_85_C
Reset:	soft
Address:	4B158h-4B15Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_86_C
Reset:	soft
Address:	4B15Ch-4B15Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_87_C
Reset:	soft
Address:	4B160h-4B163h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_88_C
Reset:	soft
Address:	4B164h-4B167h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_89_C

PAL_LGC	
Reset:	soft
Address:	4B168h-4B16Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_90_C
Reset:	soft
Address:	4B16Ch-4B16Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_91_C
Reset:	soft
Address:	4B170h-4B173h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_92_C
Reset:	soft
Address:	4B174h-4B177h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_93_C
Reset:	soft
Address:	4B178h-4B17Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_94_C
Reset:	soft
Address:	4B17Ch-4B17Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_95_C
Reset:	soft
Address:	4B180h-4B183h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_96_C
Reset:	soft
Address:	4B184h-4B187h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_97_C
Reset:	soft
Address:	4B188h-4B18Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_98_C

PAL_LGC	
Reset:	soft
Address:	4B18Ch-4B18Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_99_C
Reset:	soft
Address:	4B190h-4B193h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_100_C
Reset:	soft
Address:	4B194h-4B197h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_101_C
Reset:	soft
Address:	4B198h-4B19Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_102_C
Reset:	soft
Address:	4B19Ch-4B19Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_103_C
Reset:	soft
Address:	4B1A0h-4B1A3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_104_C
Reset:	soft
Address:	4B1A4h-4B1A7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_105_C
Reset:	soft
Address:	4B1A8h-4B1ABh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_106_C
Reset:	soft
Address:	4B1ACh-4B1AFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_107_C

PAL_LGC	
Reset:	soft
Address:	4B1B0h-4B1B3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_108_C
Reset:	soft
Address:	4B1B4h-4B1B7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_109_C
Reset:	soft
Address:	4B1B8h-4B1BBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_110_C
Reset:	soft
Address:	4B1BCh-4B1BFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_111_C
Reset:	soft
Address:	4B1C0h-4B1C3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_112_C
Reset:	soft
Address:	4B1C4h-4B1C7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_113_C
Reset:	soft
Address:	4B1C8h-4B1CBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_114_C
Reset:	soft
Address:	4B1CCh-4B1CFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_115_C
Reset:	soft
Address:	4B1D0h-4B1D3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_116_C

PAL_LGC	
Reset:	soft
Address:	4B1D4h-4B1D7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_117_C
Reset:	soft
Address:	4B1D8h-4B1DBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_118_C
Reset:	soft
Address:	4B1DCh-4B1DFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_119_C
Reset:	soft
Address:	4B1E0h-4B1E3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_120_C
Reset:	soft
Address:	4B1E4h-4B1E7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_121_C
Reset:	soft
Address:	4B1E8h-4B1EBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_122_C
Reset:	soft
Address:	4B1ECh-4B1EFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_123_C
Reset:	soft
Address:	4B1F0h-4B1F3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_124_C
Reset:	soft
Address:	4B1F4h-4B1F7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_125_C

PAL_LGC	
Reset:	soft
Address:	4B1F8h-4B1FBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_126_C
Reset:	soft
Address:	4B1FCh-4B1FFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_127_C
Reset:	soft
Address:	4B200h-4B203h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_128_C
Reset:	soft
Address:	4B204h-4B207h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_129_C
Reset:	soft
Address:	4B208h-4B20Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_130_C
Reset:	soft
Address:	4B20Ch-4B20Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_131_C
Reset:	soft
Address:	4B210h-4B213h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_132_C
Reset:	soft
Address:	4B214h-4B217h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_133_C
Reset:	soft
Address:	4B218h-4B21Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_134_C

PAL_LGC	
Reset:	soft
Address:	4B21Ch-4B21Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_135_C
Reset:	soft
Address:	4B220h-4B223h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_136_C
Reset:	soft
Address:	4B224h-4B227h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_137_C
Reset:	soft
Address:	4B228h-4B22Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_138_C
Reset:	soft
Address:	4B22Ch-4B22Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_139_C
Reset:	soft
Address:	4B230h-4B233h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_140_C
Reset:	soft
Address:	4B234h-4B237h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_141_C
Reset:	soft
Address:	4B238h-4B23Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_142_C
Reset:	soft
Address:	4B23Ch-4B23Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_143_C

PAL_LGC	
Reset:	soft
Address:	4B240h-4B243h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_144_C
Reset:	soft
Address:	4B244h-4B247h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_145_C
Reset:	soft
Address:	4B248h-4B24Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_146_C
Reset:	soft
Address:	4B24Ch-4B24Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_147_C
Reset:	soft
Address:	4B250h-4B253h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_148_C
Reset:	soft
Address:	4B254h-4B257h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_149_C
Reset:	soft
Address:	4B258h-4B25Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_150_C
Reset:	soft
Address:	4B25Ch-4B25Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_151_C
Reset:	soft
Address:	4B260h-4B263h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_152_C

PAL_LGC	
Reset:	soft
Address:	4B264h-4B267h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_153_C
Reset:	soft
Address:	4B268h-4B26Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_154_C
Reset:	soft
Address:	4B26Ch-4B26Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_155_C
Reset:	soft
Address:	4B270h-4B273h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_156_C
Reset:	soft
Address:	4B274h-4B277h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_157_C
Reset:	soft
Address:	4B278h-4B27Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_158_C
Reset:	soft
Address:	4B27Ch-4B27Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_159_C
Reset:	soft
Address:	4B280h-4B283h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_160_C
Reset:	soft
Address:	4B284h-4B287h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_161_C

PAL_LGC	
Reset:	soft
Address:	4B288h-4B28Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_162_C
Reset:	soft
Address:	4B28Ch-4B28Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_163_C
Reset:	soft
Address:	4B290h-4B293h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_164_C
Reset:	soft
Address:	4B294h-4B297h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_165_C
Reset:	soft
Address:	4B298h-4B29Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_166_C
Reset:	soft
Address:	4B29Ch-4B29Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_167_C
Reset:	soft
Address:	4B2A0h-4B2A3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_168_C
Reset:	soft
Address:	4B2A4h-4B2A7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_169_C
Reset:	soft
Address:	4B2A8h-4B2ABh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_170_C

PAL_LGC	
Reset:	soft
Address:	4B2ACh-4B2AFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_171_C
Reset:	soft
Address:	4B2B0h-4B2B3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_172_C
Reset:	soft
Address:	4B2B4h-4B2B7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_173_C
Reset:	soft
Address:	4B2B8h-4B2BBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_174_C
Reset:	soft
Address:	4B2BCh-4B2BFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_175_C
Reset:	soft
Address:	4B2C0h-4B2C3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_176_C
Reset:	soft
Address:	4B2C4h-4B2C7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_177_C
Reset:	soft
Address:	4B2C8h-4B2CBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_178_C
Reset:	soft
Address:	4B2CCh-4B2CFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_179_C

PAL_LGC	
Reset:	soft
Address:	4B2D0h-4B2D3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_180_C
Reset:	soft
Address:	4B2D4h-4B2D7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_181_C
Reset:	soft
Address:	4B2D8h-4B2DBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_182_C
Reset:	soft
Address:	4B2DCh-4B2DFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_183_C
Reset:	soft
Address:	4B2E0h-4B2E3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_184_C
Reset:	soft
Address:	4B2E4h-4B2E7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_185_C
Reset:	soft
Address:	4B2E8h-4B2EBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_186_C
Reset:	soft
Address:	4B2ECh-4B2EFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_187_C
Reset:	soft
Address:	4B2F0h-4B2F3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_188_C

PAL_LGC	
Reset:	soft
Address:	4B2F4h-4B2F7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_189_C
Reset:	soft
Address:	4B2F8h-4B2FBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_190_C
Reset:	soft
Address:	4B2FCh-4B2FFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_191_C
Reset:	soft
Address:	4B300h-4B303h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_192_C
Reset:	soft
Address:	4B304h-4B307h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_193_C
Reset:	soft
Address:	4B308h-4B30Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_194_C
Reset:	soft
Address:	4B30Ch-4B30Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_195_C
Reset:	soft
Address:	4B310h-4B313h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_196_C
Reset:	soft
Address:	4B314h-4B317h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_197_C

PAL_LGC	
Reset:	soft
Address:	4B318h-4B31Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_198_C
Reset:	soft
Address:	4B31Ch-4B31Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_199_C
Reset:	soft
Address:	4B320h-4B323h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_200_C
Reset:	soft
Address:	4B324h-4B327h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_201_C
Reset:	soft
Address:	4B328h-4B32Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_202_C
Reset:	soft
Address:	4B32Ch-4B32Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_203_C
Reset:	soft
Address:	4B330h-4B333h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_204_C
Reset:	soft
Address:	4B334h-4B337h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_205_C
Reset:	soft
Address:	4B338h-4B33Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_206_C

PAL_LGC	
Reset:	soft
Address:	4B33Ch-4B33Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_207_C
Reset:	soft
Address:	4B340h-4B343h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_208_C
Reset:	soft
Address:	4B344h-4B347h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_209_C
Reset:	soft
Address:	4B348h-4B34Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_210_C
Reset:	soft
Address:	4B34Ch-4B34Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_211_C
Reset:	soft
Address:	4B350h-4B353h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_212_C
Reset:	soft
Address:	4B354h-4B357h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_213_C
Reset:	soft
Address:	4B358h-4B35Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_214_C
Reset:	soft
Address:	4B35Ch-4B35Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_215_C

PAL_LGC	
Reset:	soft
Address:	4B360h-4B363h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_216_C
Reset:	soft
Address:	4B364h-4B367h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_217_C
Reset:	soft
Address:	4B368h-4B36Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_218_C
Reset:	soft
Address:	4B36Ch-4B36Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_219_C
Reset:	soft
Address:	4B370h-4B373h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_220_C
Reset:	soft
Address:	4B374h-4B377h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_221_C
Reset:	soft
Address:	4B378h-4B37Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_222_C
Reset:	soft
Address:	4B37Ch-4B37Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_223_C
Reset:	soft
Address:	4B380h-4B383h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_224_C

PAL_LGC	
Reset:	soft
Address:	4B384h-4B387h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_225_C
Reset:	soft
Address:	4B388h-4B38Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_226_C
Reset:	soft
Address:	4B38Ch-4B38Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_227_C
Reset:	soft
Address:	4B390h-4B393h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_228_C
Reset:	soft
Address:	4B394h-4B397h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_229_C
Reset:	soft
Address:	4B398h-4B39Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_230_C
Reset:	soft
Address:	4B39Ch-4B39Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_231_C
Reset:	soft
Address:	4B3A0h-4B3A3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_232_C
Reset:	soft
Address:	4B3A4h-4B3A7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_233_C

PAL_LGC	
Reset:	soft
Address:	4B3A8h-4B3ABh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_234_C
Reset:	soft
Address:	4B3ACh-4B3AFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_235_C
Reset:	soft
Address:	4B3B0h-4B3B3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_236_C
Reset:	soft
Address:	4B3B4h-4B3B7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_237_C
Reset:	soft
Address:	4B3B8h-4B3BBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_238_C
Reset:	soft
Address:	4B3BCh-4B3BFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_239_C
Reset:	soft
Address:	4B3C0h-4B3C3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_240_C
Reset:	soft
Address:	4B3C4h-4B3C7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_241_C
Reset:	soft
Address:	4B3C8h-4B3CBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_242_C

PAL_LGC	
Reset:	soft
Address:	4B3CCh-4B3CFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_243_C
Reset:	soft
Address:	4B3D0h-4B3D3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_244_C
Reset:	soft
Address:	4B3D4h-4B3D7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_245_C
Reset:	soft
Address:	4B3D8h-4B3DBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_246_C
Reset:	soft
Address:	4B3DCh-4B3DFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_247_C
Reset:	soft
Address:	4B3E0h-4B3E3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_248_C
Reset:	soft
Address:	4B3E4h-4B3E7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_249_C
Reset:	soft
Address:	4B3E8h-4B3EBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_250_C
Reset:	soft
Address:	4B3ECh-4B3EFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_251_C

PAL_LGC	
Reset:	soft
Address:	4B3F0h-4B3F3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_252_C
Reset:	soft
Address:	4B3F4h-4B3F7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_253_C
Reset:	soft
Address:	4B3F8h-4B3FBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_254_C
Reset:	soft
Address:	4B3FCh-4B3FFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_255_C
Reset:	soft
Address:	4B800h-4B803h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_0_D
Reset:	soft
Address:	4B804h-4B807h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_1_D
Reset:	soft
Address:	4B808h-4B80Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_2_D
Reset:	soft
Address:	4B80Ch-4B80Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_3_D
Reset:	soft
Address:	4B810h-4B813h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_4_D

PAL_LGC	
Reset:	soft
Address:	4B814h-4B817h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_5_D
Reset:	soft
Address:	4B818h-4B81Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_6_D
Reset:	soft
Address:	4B81Ch-4B81Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_7_D
Reset:	soft
Address:	4B820h-4B823h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_8_D
Reset:	soft
Address:	4B824h-4B827h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_9_D
Reset:	soft
Address:	4B828h-4B82Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_10_D
Reset:	soft
Address:	4B82Ch-4B82Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_11_D
Reset:	soft
Address:	4B830h-4B833h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_12_D
Reset:	soft
Address:	4B834h-4B837h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_13_D

PAL_LGC	
Reset:	soft
Address:	4B838h-4B83Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_14_D
Reset:	soft
Address:	4B83Ch-4B83Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_15_D
Reset:	soft
Address:	4B840h-4B843h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_16_D
Reset:	soft
Address:	4B844h-4B847h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_17_D
Reset:	soft
Address:	4B848h-4B84Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_18_D
Reset:	soft
Address:	4B84Ch-4B84Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_19_D
Reset:	soft
Address:	4B850h-4B853h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_20_D
Reset:	soft
Address:	4B854h-4B857h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_21_D
Reset:	soft
Address:	4B858h-4B85Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_22_D

PAL_LGC	
Reset:	soft
Address:	4B85Ch-4B85Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_23_D
Reset:	soft
Address:	4B860h-4B863h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_24_D
Reset:	soft
Address:	4B864h-4B867h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_25_D
Reset:	soft
Address:	4B868h-4B86Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_26_D
Reset:	soft
Address:	4B86Ch-4B86Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_27_D
Reset:	soft
Address:	4B870h-4B873h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_28_D
Reset:	soft
Address:	4B874h-4B877h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_29_D
Reset:	soft
Address:	4B878h-4B87Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_30_D
Reset:	soft
Address:	4B87Ch-4B87Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_31_D

PAL_LGC	
Reset:	soft
Address:	4B880h-4B883h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_32_D
Reset:	soft
Address:	4B884h-4B887h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_33_D
Reset:	soft
Address:	4B888h-4B88Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_34_D
Reset:	soft
Address:	4B88Ch-4B88Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_35_D
Reset:	soft
Address:	4B890h-4B893h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_36_D
Reset:	soft
Address:	4B894h-4B897h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_37_D
Reset:	soft
Address:	4B898h-4B89Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_38_D
Reset:	soft
Address:	4B89Ch-4B89Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_39_D
Reset:	soft
Address:	4B8A0h-4B8A3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_40_D

PAL_LGC	
Reset:	soft
Address:	4B8A4h-4B8A7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_41_D
Reset:	soft
Address:	4B8A8h-4B8ABh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_42_D
Reset:	soft
Address:	4B8ACh-4B8AFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_43_D
Reset:	soft
Address:	4B8B0h-4B8B3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_44_D
Reset:	soft
Address:	4B8B4h-4B8B7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_45_D
Reset:	soft
Address:	4B8B8h-4B8BBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_46_D
Reset:	soft
Address:	4B8BCh-4B8BFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_47_D
Reset:	soft
Address:	4B8C0h-4B8C3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_48_D
Reset:	soft
Address:	4B8C4h-4B8C7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_49_D

PAL_LGC	
Reset:	soft
Address:	4B8C8h-4B8CBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_50_D
Reset:	soft
Address:	4B8CCh-4B8CFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_51_D
Reset:	soft
Address:	4B8D0h-4B8D3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_52_D
Reset:	soft
Address:	4B8D4h-4B8D7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_53_D
Reset:	soft
Address:	4B8D8h-4B8DBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_54_D
Reset:	soft
Address:	4B8DCh-4B8DFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_55_D
Reset:	soft
Address:	4B8E0h-4B8E3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_56_D
Reset:	soft
Address:	4B8E4h-4B8E7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_57_D
Reset:	soft
Address:	4B8E8h-4B8EBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_58_D

PAL_LGC	
Reset:	soft
Address:	4B8ECh-4B8EFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_59_D
Reset:	soft
Address:	4B8F0h-4B8F3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_60_D
Reset:	soft
Address:	4B8F4h-4B8F7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_61_D
Reset:	soft
Address:	4B8F8h-4B8FBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_62_D
Reset:	soft
Address:	4B8FCh-4B8FFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_63_D
Reset:	soft
Address:	4B900h-4B903h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_64_D
Reset:	soft
Address:	4B904h-4B907h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_65_D
Reset:	soft
Address:	4B908h-4B90Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_66_D
Reset:	soft
Address:	4B90Ch-4B90Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_67_D

PAL_LGC	
Reset:	soft
Address:	4B910h-4B913h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_68_D
Reset:	soft
Address:	4B914h-4B917h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_69_D
Reset:	soft
Address:	4B918h-4B91Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_70_D
Reset:	soft
Address:	4B91Ch-4B91Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_71_D
Reset:	soft
Address:	4B920h-4B923h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_72_D
Reset:	soft
Address:	4B924h-4B927h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_73_D
Reset:	soft
Address:	4B928h-4B92Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_74_D
Reset:	soft
Address:	4B92Ch-4B92Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_75_D
Reset:	soft
Address:	4B930h-4B933h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_76_D

PAL_LGC	
Reset:	soft
Address:	4B934h-4B937h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_77_D
Reset:	soft
Address:	4B938h-4B93Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_78_D
Reset:	soft
Address:	4B93Ch-4B93Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_79_D
Reset:	soft
Address:	4B940h-4B943h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_80_D
Reset:	soft
Address:	4B944h-4B947h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_81_D
Reset:	soft
Address:	4B948h-4B94Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_82_D
Reset:	soft
Address:	4B94Ch-4B94Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_83_D
Reset:	soft
Address:	4B950h-4B953h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_84_D
Reset:	soft
Address:	4B954h-4B957h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_85_D

PAL_LGC	
Reset:	soft
Address:	4B958h-4B95Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_86_D
Reset:	soft
Address:	4B95Ch-4B95Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_87_D
Reset:	soft
Address:	4B960h-4B963h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_88_D
Reset:	soft
Address:	4B964h-4B967h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_89_D
Reset:	soft
Address:	4B968h-4B96Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_90_D
Reset:	soft
Address:	4B96Ch-4B96Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_91_D
Reset:	soft
Address:	4B970h-4B973h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_92_D
Reset:	soft
Address:	4B974h-4B977h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_93_D
Reset:	soft
Address:	4B978h-4B97Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_94_D

PAL_LGC	
Reset:	soft
Address:	4B97Ch-4B97Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_95_D
Reset:	soft
Address:	4B980h-4B983h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_96_D
Reset:	soft
Address:	4B984h-4B987h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_97_D
Reset:	soft
Address:	4B988h-4B98Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_98_D
Reset:	soft
Address:	4B98Ch-4B98Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_99_D
Reset:	soft
Address:	4B990h-4B993h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_100_D
Reset:	soft
Address:	4B994h-4B997h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_101_D
Reset:	soft
Address:	4B998h-4B99Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_102_D
Reset:	soft
Address:	4B99Ch-4B99Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_103_D

PAL_LGC	
Reset:	soft
Address:	4B9A0h-4B9A3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_104_D
Reset:	soft
Address:	4B9A4h-4B9A7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_105_D
Reset:	soft
Address:	4B9A8h-4B9ABh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_106_D
Reset:	soft
Address:	4B9ACh-4B9AFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_107_D
Reset:	soft
Address:	4B9B0h-4B9B3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_108_D
Reset:	soft
Address:	4B9B4h-4B9B7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_109_D
Reset:	soft
Address:	4B9B8h-4B9BBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_110_D
Reset:	soft
Address:	4B9BCh-4B9BFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_111_D
Reset:	soft
Address:	4B9C0h-4B9C3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_112_D

PAL_LGC	
Reset:	soft
Address:	4B9C4h-4B9C7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_113_D
Reset:	soft
Address:	4B9C8h-4B9CBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_114_D
Reset:	soft
Address:	4B9CCh-4B9CFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_115_D
Reset:	soft
Address:	4B9D0h-4B9D3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_116_D
Reset:	soft
Address:	4B9D4h-4B9D7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_117_D
Reset:	soft
Address:	4B9D8h-4B9DBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_118_D
Reset:	soft
Address:	4B9DCh-4B9DFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_119_D
Reset:	soft
Address:	4B9E0h-4B9E3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_120_D
Reset:	soft
Address:	4B9E4h-4B9E7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_121_D

PAL_LGC	
Reset:	soft
Address:	4B9E8h-4B9EBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_122_D
Reset:	soft
Address:	4B9ECh-4B9EFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_123_D
Reset:	soft
Address:	4B9F0h-4B9F3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_124_D
Reset:	soft
Address:	4B9F4h-4B9F7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_125_D
Reset:	soft
Address:	4B9F8h-4B9FBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_126_D
Reset:	soft
Address:	4B9FCh-4B9FFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_127_D
Reset:	soft
Address:	4BA00h-4BA03h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_128_D
Reset:	soft
Address:	4BA04h-4BA07h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_129_D
Reset:	soft
Address:	4BA08h-4BA0Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_130_D

PAL_LGC	
Reset:	soft
Address:	4BA0Ch-4BA0Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_131_D
Reset:	soft
Address:	4BA10h-4BA13h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_132_D
Reset:	soft
Address:	4BA14h-4BA17h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_133_D
Reset:	soft
Address:	4BA18h-4BA1Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_134_D
Reset:	soft
Address:	4BA1Ch-4BA1Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_135_D
Reset:	soft
Address:	4BA20h-4BA23h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_136_D
Reset:	soft
Address:	4BA24h-4BA27h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_137_D
Reset:	soft
Address:	4BA28h-4BA2Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_138_D
Reset:	soft
Address:	4BA2Ch-4BA2Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_139_D

PAL_LGC	
Reset:	soft
Address:	4BA30h-4BA33h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_140_D
Reset:	soft
Address:	4BA34h-4BA37h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_141_D
Reset:	soft
Address:	4BA38h-4BA3Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_142_D
Reset:	soft
Address:	4BA3Ch-4BA3Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_143_D
Reset:	soft
Address:	4BA40h-4BA43h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_144_D
Reset:	soft
Address:	4BA44h-4BA47h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_145_D
Reset:	soft
Address:	4BA48h-4BA4Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_146_D
Reset:	soft
Address:	4BA4Ch-4BA4Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_147_D
Reset:	soft
Address:	4BA50h-4BA53h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_148_D

PAL_LGC	
Reset:	soft
Address:	4BA54h-4BA57h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_149_D
Reset:	soft
Address:	4BA58h-4BA5Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_150_D
Reset:	soft
Address:	4BA5Ch-4BA5Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_151_D
Reset:	soft
Address:	4BA60h-4BA63h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_152_D
Reset:	soft
Address:	4BA64h-4BA67h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_153_D
Reset:	soft
Address:	4BA68h-4BA6Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_154_D
Reset:	soft
Address:	4BA6Ch-4BA6Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_155_D
Reset:	soft
Address:	4BA70h-4BA73h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_156_D
Reset:	soft
Address:	4BA74h-4BA77h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_157_D

PAL_LGC	
Reset:	soft
Address:	4BA78h-4BA7Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_158_D
Reset:	soft
Address:	4BA7Ch-4BA7Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_159_D
Reset:	soft
Address:	4BA80h-4BA83h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_160_D
Reset:	soft
Address:	4BA84h-4BA87h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_161_D
Reset:	soft
Address:	4BA88h-4BA8Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_162_D
Reset:	soft
Address:	4BA8Ch-4BA8Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_163_D
Reset:	soft
Address:	4BA90h-4BA93h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_164_D
Reset:	soft
Address:	4BA94h-4BA97h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_165_D
Reset:	soft
Address:	4BA98h-4BA9Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_166_D

PAL_LGC	
Reset:	soft
Address:	4BA9Ch-4BA9Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_167_D
Reset:	soft
Address:	4BAA0h-4BAA3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_168_D
Reset:	soft
Address:	4BAA4h-4BAA7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_169_D
Reset:	soft
Address:	4BAA8h-4BAABh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_170_D
Reset:	soft
Address:	4BAACh-4BAAFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_171_D
Reset:	soft
Address:	4BAB0h-4BAB3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_172_D
Reset:	soft
Address:	4BAB4h-4BAB7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_173_D
Reset:	soft
Address:	4BAB8h-4BABBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_174_D
Reset:	soft
Address:	4BABCh-4BABFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_175_D

PAL_LGC	
Reset:	soft
Address:	4BAC0h-4BAC3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_176_D
Reset:	soft
Address:	4BAC4h-4BAC7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_177_D
Reset:	soft
Address:	4BAC8h-4BACBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_178_D
Reset:	soft
Address:	4BACCh-4BACFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_179_D
Reset:	soft
Address:	4BAD0h-4BAD3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_180_D
Reset:	soft
Address:	4BAD4h-4BAD7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_181_D
Reset:	soft
Address:	4BAD8h-4BADBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_182_D
Reset:	soft
Address:	4BADCh-4BADFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_183_D
Reset:	soft
Address:	4BAE0h-4BAE3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_184_D

PAL_LGC	
Reset:	soft
Address:	4BAE4h-4BAE7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_185_D
Reset:	soft
Address:	4BAE8h-4BAEBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_186_D
Reset:	soft
Address:	4BAECh-4BAEFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_187_D
Reset:	soft
Address:	4BAF0h-4BAF3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_188_D
Reset:	soft
Address:	4BAF4h-4BAF7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_189_D
Reset:	soft
Address:	4BAF8h-4BAFBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_190_D
Reset:	soft
Address:	4BAFCh-4BAFFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_191_D
Reset:	soft
Address:	4BB00h-4BB03h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_192_D
Reset:	soft
Address:	4BB04h-4BB07h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_193_D

PAL_LGC	
Reset:	soft
Address:	4BB08h-4BB0Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_194_D
Reset:	soft
Address:	4BB0Ch-4BB0Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_195_D
Reset:	soft
Address:	4BB10h-4BB13h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_196_D
Reset:	soft
Address:	4BB14h-4BB17h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_197_D
Reset:	soft
Address:	4BB18h-4BB1Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_198_D
Reset:	soft
Address:	4BB1Ch-4BB1Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_199_D
Reset:	soft
Address:	4BB20h-4BB23h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_200_D
Reset:	soft
Address:	4BB24h-4BB27h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_201_D
Reset:	soft
Address:	4BB28h-4BB2Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_202_D

PAL_LGC	
Reset:	soft
Address:	4BB2Ch-4BB2Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_203_D
Reset:	soft
Address:	4BB30h-4BB33h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_204_D
Reset:	soft
Address:	4BB34h-4BB37h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_205_D
Reset:	soft
Address:	4BB38h-4BB3Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_206_D
Reset:	soft
Address:	4BB3Ch-4BB3Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_207_D
Reset:	soft
Address:	4BB40h-4BB43h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_208_D
Reset:	soft
Address:	4BB44h-4BB47h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_209_D
Reset:	soft
Address:	4BB48h-4BB4Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_210_D
Reset:	soft
Address:	4BB4Ch-4BB4Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_211_D

PAL_LGC	
Reset:	soft
Address:	4BB50h-4BB53h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_212_D
Reset:	soft
Address:	4BB54h-4BB57h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_213_D
Reset:	soft
Address:	4BB58h-4BB5Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_214_D
Reset:	soft
Address:	4BB5Ch-4BB5Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_215_D
Reset:	soft
Address:	4BB60h-4BB63h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_216_D
Reset:	soft
Address:	4BB64h-4BB67h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_217_D
Reset:	soft
Address:	4BB68h-4BB6Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_218_D
Reset:	soft
Address:	4BB6Ch-4BB6Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_219_D
Reset:	soft
Address:	4BB70h-4BB73h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_220_D

PAL_LGC	
Reset:	soft
Address:	4BB74h-4BB77h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_221_D
Reset:	soft
Address:	4BB78h-4BB7Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_222_D
Reset:	soft
Address:	4BB7Ch-4BB7Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_223_D
Reset:	soft
Address:	4BB80h-4BB83h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_224_D
Reset:	soft
Address:	4BB84h-4BB87h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_225_D
Reset:	soft
Address:	4BB88h-4BB8Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_226_D
Reset:	soft
Address:	4BB8Ch-4BB8Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_227_D
Reset:	soft
Address:	4BB90h-4BB93h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_228_D
Reset:	soft
Address:	4BB94h-4BB97h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_229_D

PAL_LGC	
Reset:	soft
Address:	4BB98h-4BB9Bh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_230_D
Reset:	soft
Address:	4BB9Ch-4BB9Fh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_231_D
Reset:	soft
Address:	4BBA0h-4BBA3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_232_D
Reset:	soft
Address:	4BBA4h-4BBA7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_233_D
Reset:	soft
Address:	4BBA8h-4BBABh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_234_D
Reset:	soft
Address:	4BBACh-4BBAFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_235_D
Reset:	soft
Address:	4BBB0h-4BBB3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_236_D
Reset:	soft
Address:	4BBB4h-4BBB7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_237_D
Reset:	soft
Address:	4BBB8h-4BBBBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_238_D

PAL_LGC	
Reset:	soft
Address:	4BBBCh-4BBBFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_239_D
Reset:	soft
Address:	4BBC0h-4BBC3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_240_D
Reset:	soft
Address:	4BBC4h-4BBC7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_241_D
Reset:	soft
Address:	4BBC8h-4BBCBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_242_D
Reset:	soft
Address:	4BBCCh-4BBCFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_243_D
Reset:	soft
Address:	4BBD0h-4BBD3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_244_D
Reset:	soft
Address:	4BBD4h-4BBD7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_245_D
Reset:	soft
Address:	4BBD8h-4BBDBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_246_D
Reset:	soft
Address:	4BBDCh-4BBDFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_247_D

PAL_LGC	
Reset:	soft
Address:	4BBE0h-4BBE3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_248_D
Reset:	soft
Address:	4BBE4h-4BBE7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_249_D
Reset:	soft
Address:	4BBE8h-4BBEBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_250_D
Reset:	soft
Address:	4BBECh-4BBEFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_251_D
Reset:	soft
Address:	4BBF0h-4BBF3h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_252_D
Reset:	soft
Address:	4BBF4h-4BBF7h
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_253_D
Reset:	soft
Address:	4BBF8h-4BBFBh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_254_D
Reset:	soft
Address:	4BBFCh-4BBFFh
Name:	Pipe Legacy Palette
ShortName:	PAL_LGC_255_D
Reset:	soft
There are 256 instances of this register format per display pipe.	

Restriction
This register must be written only as a full 32 bit dword. Byte or word writes are not supported.

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**Red Legacy Palette Entry** Default Value: UUh Red legacy palette entry value.
	15:8	**Green Legacy Palette Entry** Default Value: UUh Green legacy palette entry value.
	7:0	**Blue Legacy Palette Entry** Default Value: UUh Blue legacy palette entry value.

PAL_PREC_DATA

PAL_PREC_DATA	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	4A404h-4A407h
Name:	Pipe Precision Palette Data
ShortName:	PAL_PREC_DATA_A
Reset:	soft
Address:	4AC04h-4AC07h
Name:	Pipe Precision Palette Data
ShortName:	PAL_PREC_DATA_B
Reset:	soft
Address:	4B404h-4B407h
Name:	Pipe Precision Palette Data
ShortName:	PAL_PREC_DATA_C
Reset:	soft
Address:	4BC04h-4BC07h
Name:	Pipe Precision Palette Data
ShortName:	PAL_PREC_DATA_D
Reset:	soft

These are the precision palette entries used for the 10 bpc, split, and 12 bpc gamma.The Precision Palette Index Value indicates the precision palette location to be accessed through this register.

Programming Notes

For 10 bpc, program with the color 10 bit palette entry fraction value. For 12 bpc gamma odd indexes, program with the upper 10 bits of the color palette entry fraction value. For 12 bpc gamma even indexes, program the MSbs with the lower 6 bits of the color palette entry fraction value, then program all 0s in the LSbs. For split gamma indexes 0 to 511, program with the first gamma (before CSC) color 10 bit palette entry fraction value. For split gamma indexes 512 to 1023, program with the second gamma (after CSC) color 10 bit palette entry fraction value.

Restriction

This register must be written only as a full 32 bit dword. Byte or word writes are not supported.

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ

PAL_PREC_DATA		
	29:20	**Red Precision Palette Entry** Default Value: UUUUUUUUUUb Red precision palette entry value.
	19:10	**Green Precision Palette Entry** Default Value: UUUUUUUUUUb Green precision palette entry value.
	9:0	**Blue Precision Palette Entry** Default Value: UUUUUUUUUUb Blue precision palette entry value.

PAL_PREC_INDEX

PAL_PREC_INDEX	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	4A400h-4A403h
Name:	Pipe Precision Palette Index
ShortName:	PAL_PREC_INDEX_A
Reset:	soft
Address:	4AC00h-4AC03h
Name:	Pipe Precision Palette Index
ShortName:	PAL_PREC_INDEX_B
Reset:	soft
Address:	4B400h-4B403h
Name:	Pipe Precision Palette Index
ShortName:	PAL_PREC_INDEX_C
Reset:	soft
Address:	4BC00h-4BC03h
Name:	Pipe Precision Palette Index
ShortName:	PAL_PREC_INDEX_D
Reset:	soft

This index controls access to the array of precision palette data values.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15	**Index Auto Increment** This field enables the index auto increment. 0b – No Increment – Do not automatically increment the index value. 1b – Auto Increment – Increment the index value with each read or write to the data register.
	14:10	**Reserved** Access: RO Format: MBZ

PAL_PREC_INDEX		
	9:0	**Index Value** This field indicates the data location to be accessed through the data register. This value can be automatically incremented by a read or a write to the data register if the index auto increment bit is set. When automatically incrementing, the current automatically calculated index value can be read here, and the index will roll over to 0 after reaching the end of the allowed range. **Value** / **Name** [0,1023] /

PAL_PREC_MULTI_SEG_DATA

PAL_PREC_MULTI_SEG_DATA	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	4A40Ch-4A40Fh
Name:	Pipe Precision Multi Segment Palette Data
ShortName:	PAL_PREC_MULTI_SEG_DATA_A
Reset:	soft
Address:	4AC0Ch-4AC0Fh
Name:	Pipe Precision Multi Segment Palette Data
ShortName:	PAL_PREC_MULTI_SEG_DATA_B
Reset:	soft
Address:	4B40Ch-4B40Fh
Name:	Pipe Precision Multi Segment Palette Data
ShortName:	PAL_PREC_MULTI_SEG_DATA_C
Reset:	soft
Address:	4BC0Ch-4BC0Fh
Name:	Pipe Precision Multi Segment Palette Data
ShortName:	PAL_PREC_MULTI_SEG_DATA_D
Reset:	soft

These are the precision palette entries used for the multi segment gamma.The Precision Palette Index Value indicates the precision palette location to be accessed through this register.

Programming Notes

For 10 bpc, program with the color 10 bit palette entry fraction value. For 12 bpc gamma odd indexes, program with the upper 10 bits of the color palette entry fraction value. For 12 bpc gamma even indexes, program the MSbs with the lower 6 bits of the color palette entry fraction value, then program all 0s in the LSbs.

Restriction

This register must be written only as a full 32 bit dword. Byte or word writes are not supported.

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29:20	**Red Precision Palette Entry** Default Value: UUUUUUUUUUb Red precision palette entry value.

PAL_PREC_MULTI_SEG_DATA		
	19:10	**Green Precision Palette Entry** Default Value: UUUUUUUUUUb Green precision palette entry value.
	9:0	**Blue Precision Palette Entry** Default Value: UUUUUUUUUUb Blue precision palette entry value.

PAL_PREC_MULTI_SEG_INDEX

PAL_PREC_MULTI_SEG_INDEX	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	4A408h-4A40Bh
Name:	Pipe Precision Multi Segment Palette Index
ShortName:	PAL_PREC_MULTI_SEG_INDEX_A
Reset:	soft
Address:	4AC08h-4AC0Bh
Name:	Pipe Precision Multi Segment Palette Index
ShortName:	PAL_PREC_MULTI_SEG_INDEX_B
Reset:	soft
Address:	4B408h-4B40Bh
Name:	Pipe Precision Multi Segment Palette Index
ShortName:	PAL_PREC_MULTI_SEG_INDEX_C
Reset:	soft
Address:	4BC08h-4BC0Bh
Name:	Pipe Precision Multi Segment Palette Index
ShortName:	PAL_PREC_MULTI_SEG_INDEX_D
Reset:	soft

This index controls access to the array of precision palette data values used in the multi-segment gamma mode.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15	**Index Auto Increment** This field enables the index auto increment. Value 0b – Name: No Increment – Description: Do not automatically increment the index value. Value 1b – Name: Auto Increment – Description: Increment the index value with each read or write to the data register.
	14:5	**Reserved** Access: RO Format: MBZ

PAL_PREC_MULTI_SEG_INDEX		
	4:0	**Index Value** This field indicates the data location to be accessed through the data register. This value can be automatically incremented by a read or a write to the data register if the index auto increment bit is set. When automatically incrementing, the current automatically calculated index value can be read here, and the index will roll over to 0 after reaching the end of the allowed range. **Value** / **Name** [0,17] /

PASID Capability

PASID_CAP_0_2_0_PCI - PASID Capability

Register Space:	PCI: 0/2/0
Size (in bits):	16
Address:	00104h

PASID capability reports support for Process Address Space ID(PASID) on Device-2, compliant to PCI-Express PASID ECN.

DWord	Bit	Description
0	15:13	**Reserved** Access: RO Format: MBZ
	12:8	**Maximum PASID Width** Default Value: 10100b Access: RO _Custom_GTIReset: BUS Indicates the width of the PASID field supported by the Endpoint. Hardwired to 14h to indicate support for all PASID values (20 bits).
	7:3	**Reserved** Access: RO Format: MBZ
	2	**Privilege Mode Supported** Default Value: 0b Access: RO _Custom_GTIReset: BUS Hardwired to 0, the Endpoint supports operating in Non-privileged mode only, and will never request privileged mode in requests-with-PASID.
	1	**Execute Permission Supported** Access: RO _Custom_GTIReset: BUS **Description**: Hardwired to 0, the Endpoint supports requests-with-PASID that requests execute permission. **Value** 0b — **Name** **[Default]**
	0	**Reserved** Access: RO Format: MBZ

PASID Control

PASID_CTRL_0_2_0_PCI - PASID Control

Register Space: PCI: 0/2/0
Size (in bits): 16

Address: 00106h

Process Address Space ID (PASID) control for Device-2.

DWord	Bit	Description
0	15:3	**Reserved** Access: RO Format: MBZ
	2	**Privileged Mode Enable** Default Value: 0b Access: RO _Custom_GTIReset: BUS Hardwired to 0, the Endpoint is not permitted to request privileged mode in requests-with-PASID.
	1	**Execute Permission Enable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS If Set, the Endpoint is permitted to request execute permission in requests-with-PASID. If Clear, the Endpoint is not permitted to do so. Behavior is undefined if this bit changes value when ATS Enable field in ATS Capability is Set. Processor graphics does not use this field. Software is expected to Set this field before configuring extended-context-entry for Device-2 with the Execute Request Enable field Set.
	0	**PASID Enable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS If Set, the Endpoint is permitted to generate requests-with-PASID. If Clear, the Endpoint is not permitted to do so. Behavior is undefined if this bit changes value when ATS Enable field in ATS Capability is Set. If Privileged Mode Supported field in PASID Capability register is Clear, then this field is treated as Reserved(0).Processor graphics does not use this field. Software is expected to Set this field before configuring extended-context-entry for Device-2 with Supervisor Request Enable field Set. For compatibility reasons, this field is implemented as RW.

PASID Extended Capability Header

PASID_EXTCAP_0_2_0_PCI - PASID Extended Capability Header

Register Space:	PCI: 0/2/0
Size (in bits):	32
Address:	00100h

PASID capability reports support for Process Address Space ID(PASID) on Device-2, compliant to PCI-Express PASID ECN.

DWord	Bit	Description
0	31:20	**Next Capability Offset** Default Value: 001000000000b Access: RO _Custom_GTIReset: BUS This is a hardwired pointer to the next item in the capabilities list.
	19:16	**Version** Default Value: 0001b Access: RO _Custom_GTIReset: BUS Hardwired to capability version 1.
	15:0	**Capability ID** Default Value: 0000000000011011b Access: RO _Custom_GTIReset: BUS Hardwired to the PASID Extended Capability ID

PAT Index

PAT_INDEX - PAT Index

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV

Address:	04800h
Name:	PAT INDEX 0
ShortName:	PAT_INDEX_0

Address:	04804h
Name:	PAT INDEX 1
ShortName:	PAT_INDEX_1

Address:	04808h
Name:	PAT INDEX 2
ShortName:	PAT_INDEX_2

Address:	0480Ch
Name:	PAT INDEX 3
ShortName:	PAT_INDEX_3

Address:	04810h
Name:	PAT INDEX 4
ShortName:	PAT_INDEX_4

Address:	04814h
Name:	PAT INDEX 5
ShortName:	PAT_INDEX_5

Address:	04818h
Name:	PAT INDEX 6
ShortName:	PAT_INDEX_6

Address:	0481Ch
Name:	PAT INDEX 7
ShortName:	PAT_INDEX_7

DWord	Bit	Description
0	31:4	**Reserved** Access: RO Format: MBZ
	3:2	**Reserved** Access: RO Format: MBZ

PAT_INDEX - PAT Index		
	1:0	**Mem Type** Default Value: 11b Access: R/W 00: Uncacheable(UC) 01: Write Combining(WC) 10: Write through(WT) 11: Write back(WB)

PCI Command

PCICMD_0_2_0_PCI - PCI Command

Register Space: PCI: 0/2/0
Size (in bits): 16

Address: 00004h

This 16-bit register provides basic control over the IGD's ability to respond to PCI cycles. The PCICMD Register in the IGD disables the IGD PCI compliant primary accesses to main memory.

DWord	Bit	Description
0	15:11	**Reserved** Access: RO Format: MBZ
	10	**Interrupt Disable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS This bit disables the device from asserting INTx#. 0: Enable the assertion of this device's INTx# signal. 1: Disable the assertion of this device's INTx# signal. DO_INTx messages will not be sent to DMI.
	9	**Fast Back-to-Back** Default Value: 0b Access: RO _Custom_GTIReset: BUS Not Implemented. Hardwired to 0.
	8	**SERR Enable** Default Value: 0b Access: RO _Custom_GTIReset: BUS Not Implemented. Hardwired to 0.
	7	**Wait Cycle Control** Default Value: 0b Access: RO _Custom_GTIReset: BUS Not Implemented. Hardwired to 0.
	6	**Parity Error Enable** Default Value: 0b Access: RO _Custom_GTIReset: BUS Not Implemented. Hardwired to 0. Since the IGD belongs to the category of devices that does

PCICMD_0_2_0_PCI - PCI Command

		not corrupt programs or data in system memory or hard drives, the IGD ignores any parity error that it detects and continues with normal operation.
	5	**Video Palette Snooping** Default Value: 0b Access: RO _Custom_GTIReset: BUS This bit is hardwired to 0 to disable snooping.
	4	**Memory Write and Invalidate Enable** Default Value: 0b Access: RO _Custom_GTIReset: BUS Hardwired to 0. The IGD does not support memory write and invalidate commands.
	3	**Special Cycle Enable** Default Value: 0b Access: RO _Custom_GTIReset: BUS This bit is hardwired to 0. The IGD ignores Special cycles.
	2	**Bus Primary Enable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS 0: Disable IGD bus primaring. 1: Enable the IGD to function as a PCI compliant primary.
	1	**Memory Access Enable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS This bit controls the IGD's response to memory space accesses. 0: Disable. 1: Enable.
	0	**I/O Access Enable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS This bit controls the IGD's response to I/O space accesses. 0: Disable. 1: Enable. This field is RO 1'b0 if DEV2CTL[0] IOBARDIS at offset 0x58 is 1b.

PCI Express Capability

PCIECAP_0_2_0_PCI - PCI Express Capability

Register Space: PCI: 0/2/0
Size (in bits): 16

Address: 00072h

PCI Express Capability

DWord	Bit	Description
0	15:14	**Reserved** Access: RO Format: MBZ
	13:9	**Interrupt Message Number** Default Value: 00000b Access: RO _Custom_GTIReset: BUS This field indicates which MSI vector is used for the interrupt message generated in association with any of the status bits of this Capability structure. Since this device only supports one MSI vector, this field is hardwired to 0.
	8	**Slot Implemented** Default Value: 0b Access: RO _Custom_GTIReset: BUS This field is hardwired to 0 for an endpoint device.
	7:4	**Device Type** Default Value: 1001b Access: RO _Custom_GTIReset: BUS This field is hardwired to 9h to indicate a Root Complex Integrated Endpoint.
	3:0	**Capability Version** Default Value: 0010b Access: RO _Custom_GTIReset: BUS This field is hardwired to 2h to indicate Functions compliant to PCI Express 3.0 Base Specification.

PCI Express Capability Header

PCIECAPHDR_0_2_0_PCI - PCI Express Capability Header		
Register Space:	PCI: 0/2/0	
Size (in bits):	16	
Address:	00070h	
PCI Express Capability Header		
DWord	**Bit**	**Description**
0	15:8	**Next Capability Pointer** Default Value: 10101100b Access: RO _Custom_GTIReset: BUS This field is hardwired to point to the next PCI Capability structure, the MSI Capabilities at ACh.
	7:0	**Capability Identifier** Default Value: 00010000b Access: RO _Custom_GTIReset: BUS This field is hardwired to 10h to indicate that this is a PCI Express Capability structure.

PCI Express Capability Structure

DEVICESTS_0_2_0_PCI - PCI Express Capability Structure

Register Space:	PCI: 0/2/0
Size (in bits):	16
Address:	0007Ah

PCI Express Capability Structure

DWord	Bit	Description
0	15:6	**Reserved** Access: RO Format: MBZ
	5	**Transactions Pending** Default Value: 0b Access: RO _Custom_GTIReset: BUS When Set, this bit indicates that the Function has issued Non-Posted Requests that have not been completed. A Function reports this bit is cleared only when all outstanding Non-Posted Requests have completed or have been terminated by the Completion Timeout mechanism. This bit must also be cleared upon the completion of an FLR.
	4	**Aux Power Detected** Default Value: 0b Access: RO _Custom_GTIReset: BUS Functions that require Aux power report this bit as Set if Aux power is detected by the Function. Hardwired to 0b, the integrated graphics device does not require Aux power.
	3	**Unsupported Request Detected** Default Value: 0b Access: RO _Custom_GTIReset: BUS This bit indicates the Function received an Unsupported Request. Hardwired to 0b, the Root Complex Integrated Endpoint graphics device does not use the PCI Express error reporting mechanism.
	2	**Fatal Error Detected** Default Value: 0b Access: R/WC _Custom_GTIReset: BUS This bit indicates status of fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control Register.

DEVICESTS_0_2_0_PCI - PCI Express Capability Structure

	1	**Non-Fatal Error Detected** Default Value: 0b Access: R/WC _Custom_GTIReset: BUS This bit indicates status of non fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control Register.
	0	**Correctable Error Detected** Default Value: 0b Access: R/WC _Custom_GTIReset: BUS This bit indicates status of correctable errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control Register.

PCI Express Device Control

DEVICECTL_0_2_0_PCI - PCI Express Device Control

Register Space:	PCI: 0/2/0
Size (in bits):	16
Address:	00078h

PCI Express Device Control

DWord	Bit	Description
0	15	**Initiate Function Level Reset** Default Value: 0b Access: R/W _Custom_GTIReset: BUS A write of 1b initiates Function Level Reset to the Function. During FLR, a read will return 1b since device 2 reads abort. If a local panel is powered on and configured to power down on reset, the FLR will typically take several hundred milliseconds to complete. The worst possible, although unrealistic, delay is 5 seconds.
	14:12	**Max Read Request Size** Default Value: 000b Access: RO _Custom_GTIReset: BUS Functions that do not generate Read Requests larger than 128 bytes and Functions that do not generate Read Requests on their own behalf are permitted to implement this field as Read Only (RO) with a value of 000b.
	11	**Enable No Snoop** Default Value: 0b Access: RO _Custom_GTIReset: BUS This bit is permitted to be hardwired to 0b if a Function would never Set the No Snoop attribute in transactions it initiates. The graphics device never generates a PCI Express TLP.
	10	**Aux Power PM Enable** Default Value: 0b Access: RO _Custom_GTIReset: BUS Functions that do not implement this capability hardwire this bit to 0b.
	9	**Phantom Functions Enable** Default Value: 0b Access: RO _Custom_GTIReset: BUS Functions that do not implement this capability hardwire this bit to 0b.

DEVICECTL_0_2_0_PCI - PCI Express Device Control

	8	**Extended Tag field Enable** Default Value: 0b Access: RO _Custom_GTIReset: BUS Functions that do not implement this capability hardwire this bit to 0b.
	7:5	**Max Payload Size** Default Value: 000b Access: RO _Custom_GTIReset: BUS Functions that support only the 128-byte max payload size are permitted to hardwire this field to 000b.
	4	**Enable Relaxed Ordering** Default Value: 0b Access: RO _Custom_GTIReset: BUS A Function is permitted to hardwire this bit to 0b if it never sets the Relaxed Ordering attribute in transactions it initiates as a Requester. The graphics device never generates a PCI Express TLP.
	3	**Unsupported Request Response Enable** Default Value: 0b Access: RO _Custom_GTIReset: BUS A Root Complex Integrated Endpoint that is not associated with a Root Complex Event Collector is permitted to hardwire this bit to 0b.
	2	**Fatal Error Enable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS This bit, in conjunction with other bits, controls sending ERR_FATAL Messages (see Section 6.2.5 and Section 6.2.6 for details).
	1	**Non-Fatal Error Enable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS This bit, in conjunction with other bits, controls sending ERR_NONFATAL Messages (see Section 6.2.5 and Section 6.2.6 for details).

DEVICECTL_0_2_0_PCI - PCI Express Device Control		
	0	**Correctable Error Enable** Default Value: 0b Access: R/W _Custom_GTIReset: BUS This bit, in conjunction with other bits, controls sending ERR_COR Messages (see Section 6.2.5 and Section 6.2.6 for details).

PCI Mirror of GMCH Graphics Control

MGGC0_0_2_0_PCI - PCI Mirror of GMCH Graphics Control

Register Space:	PCI: 0/2/0
Size (in bits):	16
Address:	00050h

Mirror of GGC register from GTTMMADR Space at offset 0x108040.

DWord	Bit	Description
0	15:8	**GMS** Default Value: 05h Access: RO Variant _Custom_GTIReset: BUS This field is used to select the amount of Main Memory that is pre-allocated to support the Internal Graphics device in VGA (non-linear) and Native (linear) modes. It corresponds to DSM (Data Stolen Memory region) region. The BIOS ensures that memory is pre-allocated only when Internal graphics is enabled. Hardware does not clear or set any of these bits automatically based on IGD being disabled/enabled. BIOS Requirement: BIOS must not set this field to 0h if IVD (bit 1 of this register) is 0. BIOS Requirement: Given new sizes allow down to 8MB allocation, BIOS has to ensure there is sufficient space for WOPCM and basic GFX Stolen functions. 00h:0MB 01h:32MB 02h:64MB 03h:96MB 04h:128MB 05h:160MB 06h:192MB 07h:224MB 08h:256MB 09h:288MB 0Ah:320MB 0Bh:352MB 0Ch:384MB 0Dh:416MB 0Eh:448MB 0Fh:480MB 10h:512MB 11h - 1Fh: Reserved 20h:1024MB 21h - 2Fh: Reserved 30h:1536MB 31h - 3Fh: Reserved 40h: 2048MB 41h - EFh: Reserved

F0h: 4MB
F1h: 8MB
F2h: 12MB
F3h: 16MB
F4h: 20MB
F5h: 24MB
F6h: 28MB
F7h: 32MB
F8h: 36MB
F9h: 40MB
FAh: 44MB
FBh: 48MB
FCh: 52MB
FDh: 56MB
FEh: 60MB
FFh: Reserved Hardware functionality in case of programming this value to Reserved is not guaranteed.

7:6 **GGMS**

Default Value:	00b
Access:	RO Variant
_Custom_GTIReset:	BUS

This field is used to select the amount of Main Memory that is pre-allocated to support the Internal Graphics Translation Table. The BIOS ensures that memory is pre-allocated only when Internal graphics is enabled.
GSM is assumed to be a contiguous physical DRAM space with DSM, and BIOS needs to allocate a contiguous memory chunk. Hardware will derive the base of GSM from DSM only using the GSM size programmed in the register.
Hardware functionality in case of programming this value to Reserved is not guaranteed.
0x0:No Preallocated Memory
0x1:2MB of Preallocated Memory
0x2:4MB of Preallocated Memory
0x3:8MB of Preallocated Memory

5:3 **Reserved**

Access:	RO
Format:	MBZ

2 **VAMEN**

Default Value:	0b
Access:	RO Variant
_Custom_GTIReset:	BUS

Enables the use of the iGFX engines for Versatile Acceleration.
0 - iGFX engines are in iGFX Mode. Device 2 Class Code is 030000h.
1 - iGFX engines are in Versatile Acceleration Mode. Device 2 Class Code is 038000h.

MGGC0_0_2_0_PCI - PCI Mirror of GMCH Graphics Control

	1	**IVD** Default Value: 0b Access: RO Variant _Custom_GTIReset: BUS 0: Enable. Device 2 (IGD) claims VGA memory and IO cycles, the Sub-Class Code within Device 2 Class Code register is 00. 1: Disable. Device 2 (IGD) does not claim VGA cycles (Mem and IO), and the Sub- Class Code field within Device 2 function 0 Class Code register is 80. BIOS Requirement: BIOS must not set this bit to 0 if the GMS field (bits 7:3 of this register) pre-allocates no memory.
	0	**SPARE** Default Value: 0b Access: RO Variant _Custom_GTIReset: BUS Note: This bit was maintained as a placeholder for compatibility. Prior, it locked the register.

PCI Status

PCISTS2_0_2_0_PCI - PCI Status

Register Space: PCI: 0/2/0
Size (in bits): 16

Address: 00006h

PCISTS is a 16-bit status register that reports the occurrence of a PCI compliant primary abort and PCI compliant target abort. PCISTS also indicates the DEVSEL# timing that has been set by the IGD.

DWord	Bit	Description
0	15	**Detected Parity Error** Default Value: 0b Access: RO _Custom_GTIReset: BUS Since the IGD does not detect parity, this bit is always hardwired to 0.
	14	**Signaled System Error** Default Value: 0b Access: RO _Custom_GTIReset: BUS The IGD never asserts SERR#, therefore this bit is hardwired to 0.
	13	**Received Primary Abort Status** Default Value: 0b Access: RO _Custom_GTIReset: BUS The IGD never gets a Primary Abort, therefore this bit is hardwired to 0.
	12	**Received Target Abort Status** Default Value: 0b Access: RO _Custom_GTIReset: BUS The IGD never gets a Target Abort, therefore this bit is hardwired to 0.
	11	**Signaled Target Abort Status** Default Value: 0b Access: RO _Custom_GTIReset: BUS Hardwired to 0. The IGD does not use target abort semantics.
	10:9	**DEVSEL Timing** Default Value: 00b Access: RO _Custom_GTIReset: BUS Hardwired to 00.

PCISTS2_0_2_0_PCI - PCI Status

8	**Primary Data Parity Error Detected**
	Default Value: 0b Access: RO _Custom_GTIReset: BUS
	Since Parity Error Response is hardwired to disabled, and the IGD does not do any parity detection, this bit is hardwired to 0.
7	**Fast Back-to-Back**
	Default Value: 0b Access: RO _Custom_GTIReset: BUS
	Hardwired to 0 to be compliant to PCI Express Base Spec (rev 3.0).
6	**User Defined Format**
	Default Value: 0b Access: RO _Custom_GTIReset: BUS
	Hardwired to 0.
5	**66 MHz PCI Capable**
	Default Value: 0b Access: RO _Custom_GTIReset: BUS
	Hardwired to 0.
4	**Capability List**
	Default Value: 1b Access: RO _Custom_GTIReset: BUS
	This bit is hardwired to 1 to indicate that the register at 34h provides an offset into the function's PCI Configuration Space containing a pointer to the location of the first item in the list.
3	**Interrupt Status**
	Default Value: 0b Access: RO Variant _Custom_GTIReset: BUS
	This bit reflects the state of the interrupt in the device. Only when the Interrupt Disable bit in the command register is a 0 and this Interrupt Status bit is a 1, will the devices INTx# signal be asserted.
2:0	**Reserved**
	Access: RO Format: MBZ

PCU Interrupt Definition

PCU Interrupt Definition

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	444E0h-444EFh
Name:	PCU Interrupts
ShortName:	PCU_INTERRUPT

This table indicates which events are mapped to each bit of the PCU Interrupt registers.
0x444E0 = ISR
0x444E4 = IMR
0x444E8 = IIR
0x444EC = IER

DWord	Bit	Description
0	31	**DDIA DC9 HPD** _Custom_GTIReset: BUS This field indicates DDIA hotplug activity was detected during DC9.
	30	**DDIB DC9 HPD** _Custom_GTIReset: BUS This field indicates DDIB hotplug activity was detected during DC9.
	29	**DDIC DC9 HPD** _Custom_GTIReset: BUS This field indicates DDIC hotplug activity was detected during DC9.
	28	**Spare_28** _Custom_GTIReset: BUS Spare bit
	27	**Spare_27** _Custom_GTIReset: BUS Spare bit
	26	**Spare_26** _Custom_GTIReset: BUS Spare bit
	25	**PCU_Pcode2driver_Mailbox_Event** _Custom_GTIReset: BUS
	24	**PCU_Thermal_Event** _Custom_GTIReset: BUS
	23	**Spare_23** _Custom_GTIReset: BUS Spare bit

PCU Interrupt Definition			
	22	**Spare_22**	
		_Custom_GTIReset:	BUS
		Spare bit	
	21	**Spare_21**	
		_Custom_GTIReset:	BUS
		Spare bit	
	20	**Spare_20**	
		_Custom_GTIReset:	BUS
		Spare bit	
	19	**Spare_19**	
		_Custom_GTIReset:	BUS
		Spare bit	
	18	**Spare_18**	
		_Custom_GTIReset:	BUS
		Spare bit	
	17	**Spare_17**	
		_Custom_GTIReset:	BUS
		Spare bit	
	16	**Spare_16**	
		_Custom_GTIReset:	BUS
		Spare bit	
	15	**Spare_15**	
		_Custom_GTIReset:	BUS
		Spare bit	
	14	**Spare_14**	
		_Custom_GTIReset:	BUS
		Spare bit	
	13	**Spare_13**	
		_Custom_GTIReset:	BUS
		Spare bit	
	12	**Spare_12**	
		_Custom_GTIReset:	BUS
		Spare bit	
	11	**Spare_11**	
		_Custom_GTIReset:	BUS
		Spare bit	
	10	**Spare_10**	
		_Custom_GTIReset:	BUS
		Spare bit	

PCU Interrupt Definition		
	9	**Spare_9** _Custom_GTIReset: BUS Spare bit
	8	**Spare_8** _Custom_GTIReset: BUS Spare bit
	7	**Spare_7** _Custom_GTIReset: BUS Spare bit
	6	**Spare_6** _Custom_GTIReset: BUS Spare bit
	5	**Spare_5** _Custom_GTIReset: BUS Spare bit
	4	**Spare_4** _Custom_GTIReset: BUS Spare bit
	3	**Spare_3** _Custom_GTIReset: BUS Spare bit
	2	**Spare_2** _Custom_GTIReset: BUS Spare bit
	1	**Spare_1** _Custom_GTIReset: BUS KVMR Release Display Enable - This field indicates that KVMR is no longer requesting driver to enable a display output.
	0	**Spare_0** _Custom_GTIReset: BUS KVMR Request Display Enable -This field indicates that KVMR is requesting driver to enable a display output.

PHY_MISC

PHY_MISC	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	64C00h-64C03h
Name:	PHY_MISC_A
ShortName:	PHY_MISC_A
Reset:	global
Address:	64C04h-64C07h
Name:	PHY_MISC_B
ShortName:	PHY_MISC_B
Reset:	global
Address:	64C08h-64C0Bh
Name:	PHY_MISC_C
ShortName:	PHY_MISC_C
Reset:	global

This register is on the ungated clock and the chip reset, not the FLR.

DWord	Bit	Description
0	31:28	**DE to IO Misc** Default Value: 0010b
	27:24	**IO to DE Misc** Access: RO
	23	**DE to IO Comp Pwr Down**
	22	**Spare 22**
	21	**Spare 21**
	20	**Spare 20**
	19:12	**Reserved** Access: RO Format: MBZ
	11:4	**Reserved** Access: RO Format: MBZ
	3:0	**Reserved** Access: RO Format: MBZ

Pinned Surface Mapping Size Register

PINNED_SURFACE_SIZE - Pinned Surface Mapping Size Register

Register Space: MMIO: 0/2/0

Size (in bits): 32

Pinned Surface Mapping Size register

DWord	Bit	Description
0	31:12	**Size** Default Value: 0000000000000000000b Access: R/W _Custom_GTIReset: DEV Size of the surface(s) in 4KB pages. Must be > 0.
	11:1	**Reserved** Access: RO Format: MBZ
	0	**Valid** Default Value: 0b Access: R/W _Custom_GTIReset: DEV 1'b0: This entry is not valid 1'b1: This entry is valid

Pinned Surface Per Process Virtual Address Base Register

PINNED_SURFACE_PPBASE_H - Pinned Surface Per Process Virtual Address Base Register

Register Space: MMIO: 0/2/0
Size (in bits): 32

Pinned Surface Per Process Virtual Address Base register

DWord	Bit	Description
0	31:0	**Per Process Virtual Base_H** Default Value: 0000h Access: R/W _Custom_GTIReset: DEV Per Process Virtual Address Base - PPBASE[47:16] 4KB Page number of the surface(s) in Per-Process Virtual Address Space (ie, index into the PPGTT)

Pinned Surface Process ID Register

PINNED_SURFACE_PROCESS_ID - Pinned Surface Process ID Register		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Pinned Surface Process ID register		
DWord	**Bit**	**Description**
0	31:0	**Process ID** Default Value: 0000h Access: R/W _Custom_GTIReset: DEV ID of the Process that owns the pinned surfaces represented by this register.

Pinned Surface Virtual Address Base Register

PINNED_SURFACE_ADDR_BASE - Pinned Surface Virtual Address Base Register

Register Space: MMIO: 0/2/0
Size (in bits): 32

Pinned Surface Virtual Address Base register

DWord	Bit	Description
0	31:28	**Per Process Virtual Base_L** Default Value: 0000b Access: R/W _Custom_GTIReset: DEV Per Process Virtual Address Base - PPBASE[15:12] 4KB Page number of the surface(s) in Per-Process Virtual Address Space (ie, index into the PPGTT)
	27:20	**Reserved** Access: RO Format: MBZ
	19:0	**Global Virtual Address Base** Default Value: 00000000000000000000b Access: R/W _Custom_GTIReset: DEV 4KB Page number of the surface(s) in Global Virtual Address Space (ie, index into the GGTT). The entire range from Base to Base+Size-1 must fall within the Pinned Range of the GGTT.

PIPE_ARB_CTL

PIPE_ARB_CTL	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank OR pipe disabled

Address:	70028h-7002Bh
Name:	Pipe Arbiter Control
ShortName:	PIPE_ARB_CTL_A
Reset:	soft

Address:	71028h-7102Bh
Name:	Pipe Arbiter Control
ShortName:	PIPE_ARB_CTL_B
Reset:	soft

Address:	72028h-7202Bh
Name:	Pipe Arbiter Control
ShortName:	PIPE_ARB_CTL_C
Reset:	soft

Address:	73028h-7302Bh
Name:	Pipe Arbiter Control
ShortName:	PIPE_ARB_CTL_D
Reset:	soft

There is one instance of this register per pipe.

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30:21	**Reserved** Access: RO Format: MBZ
	20	**Disable Weighted Arbitration** This field disables the weighted pipe slice arbitration. 0b – Enable **[Default]** 1b – Disable
	19	**Reserved**

PIPE_ARB_CTL

18:16	**Additional Slots** These additional Slots gets added to each arbitration cycle during which the clients gets serviced in a round robin manner. A programmed value of 1b results in 1 additional slot.
15:14	**Reserved** Access: RO Format: MBZ
13	**Use Programmed Slots** When this field is set, HW uses the Slots programmed in the PLANE_CTL register instead of the HW defaults.
12	**Disable Block Valid Check** The field disables the block valid check done at pipe arbiter. 0b: Enable 1b: Disable
11:10	**DSB Arbitration Interval** This field defines the DSB requests service interval in the pipe arbitration. 00b: 16 clocks 01b: 32 clocks 10b: 64 clocks **[Default]** 11b: 128 clocks
9:8	**Request Vs Data Arbitration** This field selects the arbitration weightage for the Streamer and the DDB requests. 00b: Allow 1 Streamer requests every 2 DDB requests. 01b: Allow 1 Streamer requests every 4 DDB requests. 10b **[Default]**: Allow 1 Streamer requests every 8 DDB requests. 11b: Allow 1 Streamer requests every 16 DDB requests.
7:6	**Reserved** Access: RO Format: MBZ
5:0	**Frame Start Drain Delay** This field contains the time, in microseconds, the pipe waits before draining the data from the Display Buffer. [0-31] 15 **[Default]**

PIPE_BOTTOM_COLOR

PIPE_BOTTOM_COLOR

Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank OR pipe disabled

Address:	70034h-70037h
Name:	Pipe Bottom Color
ShortName:	PIPE_BOTTOM_COLOR_A
Reset:	soft

Address:	71034h-71037h
Name:	Pipe Bottom Color
ShortName:	PIPE_BOTTOM_COLOR_B
Reset:	soft

Address:	72034h-72037h
Name:	Pipe Bottom Color
ShortName:	PIPE_BOTTOM_COLOR_C
Reset:	soft

Address:	73034h-73037h
Name:	Pipe Bottom Color
ShortName:	PIPE_BOTTOM_COLOR_D
Reset:	soft

This register sets the color that appears underneath the bottom most plane in the pipe blender Z-order. The value for each color channel is represented in an unsigned 0.10 format with 0 integer and 10 fractional bits.

DWord	Bit	Description
0	31	**Pipe Gamma Enable** This bit enables pipe gamma correction for the bottom color. This field is deprecated. Value: 0b — Name: Disable Value: 1b — Name: Enable
	30	**Pipe CSC Enable** This bit enables pipe color space conversion for the bottom color. This field is deprecated.

PIPE_BOTTOM_COLOR

		Value / **Name** 0b — Disable 1b — Enable
	29:20	**V R Bottom Color** Format: U0.10 This field sets the bottom color for the V or Red channel.
	19:10	**Y G Bottom Color** Format: U0.10 This field sets the bottom color for the Y or Green channel.
	9:0	**U B Bottom Color** Format: U0.10 This field sets the bottom color for the U or Blue channel.

PIPE_DMCSCANLINECOMP

PIPE_DMCSCANLINECOMP

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	7000Ch-7000Fh
Name:	Pipe Scan Line Compare for DMC
ShortName:	PIPE_DMCSCANLINECOMP_A
Reset:	soft
Address:	7100Ch-7100Fh
Name:	Pipe Scan Line Compare for DMC
ShortName:	PIPE_DMCSCANLINECOMP_B
Reset:	soft
Address:	7200Ch-7200Fh
Name:	Pipe Scan Line Compare for DMC
ShortName:	PIPE_DMCSCANLINECOMP_C
Reset:	soft
Address:	7300Ch-7300Fh
Name:	Pipe Scan Line Compare for DMC
ShortName:	PIPE_DMCSCANLINECOMP_D
Reset:	soft

DWord	Bit	Description
0	31	**Enable Compare** This field enables the scan line compare for DMC event generation. When this register is written with this bit set to 1b, the display engine will, trigger a scan line event after reaching the programmed scan line number. It will do the same on every frame. Value / Name: 0b – Do nothing 1b – Enable compare Restriction: Do not enable this register if the event is not needed in the DMC.
	30:20	**Reserved** Access: RO Format: MBZ
	19:0	**Scan Line Value** This field specifies the ending scan line number of the scan line window.

PIPE_DSS_CTL1

PIPE_DSS_CTL1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	78000h-78003h
Name:	PIPE DSS Control 1
ShortName:	PIPE_DSS_CTL1_PA
Reset:	soft
Address:	78200h-78203h
Name:	PIPE DSS Control 1
ShortName:	PIPE_DSS_CTL1_PB
Reset:	soft
Address:	78400h-78403h
Name:	PIPE DSS Control 1
ShortName:	PIPE_DSS_CTL1_PC
Reset:	soft
Address:	78600h-78603h
Name:	PIPE DSS Control 1
ShortName:	PIPE_DSS_CTL1_PD
Reset:	soft

Display stream splitter

DWord	Bit	Description
0	31	**Splitter Enable** This field enables stream splitting. This bit must be set to enable MSO configuration. **Value** / **Name** 0b – Disable 1b – Enable **Restriction** Splitter enable is supported for pipe A only.
	30	**Joiner Enable** This field enables stream joiner after compression. **Value** / **Name** 0b – Disable

PIPE_DSS_CTL1

	1b	Enable

29 Big_Joiner_Enable

When big_joiner_enable is '1', this dssunit will be working with another dssunit in adjacent pipe either as a primary or as a secondary.

Value	Name
0b	Disable
1b	Enable

28 Primary_Big_Joiner_Enable

This bit indicates that this pipe is the primary/secondary when Big_Joiner_Enable bit is set in this register.

Value	Name
0b	Secondary
1b	Primary

27 Reserved

26:25 Reserved

Access:	RO
Format:	MBZ

24 Dual Link Mode

This field selects the split pattern. Applicable only if splitter mode is enabled through DSS configuration bits.

Value	Name
0b	Front-Back mode
1b	Interleave mode

23:20 Reserved

Access:	RO
Format:	MBZ

19:16 Overlap

MIPI use case (mainly dual link mode):
This field specifies the number of pixels of overlap.
1 to 15 = valid integer number of overlap pixels.
0 = Sink device requires no overlap pixels.
eDP use case:
This field specifies the number of overlap pixels the sink device uses in the active data.
1 to 8 = valid integer number of overlap pixels.
0 = Sink device requires no overlap pixels.

15:12 Reserved

Access:	RO
Format:	MBZ

PIPE_DSS_CTL1		
	11:0	**Left DL buffer Target Depth** This field indicates the number of pixels to hold in the secondary link buffer before enabling the timing generator, so the Primary and Secondary client controllers are in sync. Valid only when operating in front back dual link mode. Value should only be programmed for the Secondary client controller. If bit 31 is set then the target for the Secondary controller must be non-zero. Maximum value is 1440 decimal.

PIPE_DSS_CTL2

PIPE_DSS_CTL2

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	78004h-78007h
Name:	PIPE DSS Control 2
ShortName:	PIPE_DSS_CTL2_PA
Reset:	soft
Address:	78204h-78207h
Name:	PIPE DSS Control 2
ShortName:	PIPE_DSS_CTL2_PB
Reset:	soft
Address:	78404h-78407h
Name:	PIPE DSS Control 2
ShortName:	PIPE_DSS_CTL2_PC
Reset:	soft
Address:	78604h-78607h
Name:	PIPE DSS Control 2
ShortName:	PIPE_DSS_CTL2_PD
Reset:	soft

Display stream splitter

DWord	Bit	Description
0	31	**Left Branch VDSC Enable** This bit enables Display Stream Compression on left branch. Its double buffered to rising edge of vblank. Restriction : Display stream compression is supported for pipe active sizes up to 4096 pixels. Value / Name: 0b – Disable 1b – Enable
	30	**Reserved** Access: RO Format: MBZ
	29:27	**Reserved** Access: RO Format: MBZ

PIPE_DSS_CTL2

	26	**Spare 26**
	25	**Spare 25**
	24	**Spare 24**
	23	**Reserved** Access: RO Format: MBZ
	22:19	**Reserved** Access: RO Format: MBZ
	18:17	**Reserved** Access: RO Format: MBZ
	16	**Reserved** Access: RO Format: MBZ
	15	**Right Branch VDSC Enable** Display stream compression on right branch enable/disable. It is double buffered on rising edge of vblank. Restriction : Display stream compression is supported for pipe active sizes up to 4096 pixels. **Value** / **Name** 0b – Disable 1b – Enable
	14	**Spare 14**
	13	**Spare 13**
	12	**Spare 12**
	11:0	**Right DL Buffer Target Depth** This field indicates the number of pixels to hold in the secondary link buffer before enabling the timing generator, so the Primary and Secondary client controllers are in sync. Valid only when operating in front back dual link mode. Value should only be programmed for the Secondary controller. If bit 31 is set then the target for the Secondary controller must be non-zero. Maximum value is 1440 decimal. Default is 0.

PIPE_FLIPCNT

PIPE_FLIPCNT	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	70044h-70047h
Name:	Pipe Flip Count
ShortName:	PIPE_FLIPCNT_A
Reset:	soft
Address:	71044h-71047h
Name:	Pipe Flip Count
ShortName:	PIPE_FLIPCNT_B
Reset:	soft
Address:	72044h-72047h
Name:	Pipe Flip Count
ShortName:	PIPE_FLIPCNT_C
Reset:	soft
Address:	73044h-73047h
Name:	Pipe Flip Count
ShortName:	PIPE_FLIPCNT_D
Reset:	soft

DWord	Bit	Description
0	31:0	**Pipe Flip Counter** This field provides read back of the display pipe flip counter. The counter increments on the start of each flip to the selected plane of this pipe. The start of flip is when the plane surface address is updated, not when the flip completes. The flip can be through command streamer asynchronous and synchronous flips or MMIO writes to the plane 1 surface address. It rolls over back to 0 after (2^32)-1 flips. Pipe flipcounter is restricted to one plane at a time. The plane select is programmed in PIPE_MISC2->Flip Timestamp Plane Select.

PIPE_FLIPDONETMSTMP

PIPE_FLIPDONETMSTMP	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	70054h-70057h
Name:	Pipe Flip Done Time Stamp
ShortName:	PIPE_FLIPDONETMSTMP_A
Reset:	soft
Address:	71054h-71057h
Name:	Pipe Flip Done Time Stamp
ShortName:	PIPE_FLIPDONETMSTMP_B
Reset:	soft
Address:	72054h-72057h
Name:	Pipe Flip Done Time Stamp
ShortName:	PIPE_FLIPDONETMSTMP_C
Reset:	soft
Address:	73054h-73057h
Name:	Pipe Flip Done Time Stamp
ShortName:	PIPE_FLIPDONETMSTMP_D
Reset:	soft

DWord	Bit	Description
0	31:0	**Pipe Flip Done Time Stamp** This field provides read back of the display pipe flip done time stamp. The time stamp value is sampled when hardware latches on to the new surface and the flip done gets sent. The flip can be through command streamer asynchronous and synchronous flips or MMIO writes to the plane. The TIMESTAMP_CTR register has the current time stamp value. Flip time stamp sampling is restricted to one plane at a time. The plane select is programmed in PIPE_MISC2->Flip Timestamp Plane Select.

PIPE_FLIPTMSTMP

PIPE_FLIPTMSTMP	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	7004Ch-7004Fh
Name:	Pipe Flip Time Stamp
ShortName:	PIPE_FLIPTMSTMP_A
Reset:	soft
Address:	7104Ch-7104Fh
Name:	Pipe Flip Time Stamp
ShortName:	PIPE_FLIPTMSTMP_B
Reset:	soft
Address:	7204Ch-7204Fh
Name:	Pipe Flip Time Stamp
ShortName:	PIPE_FLIPTMSTMP_C
Reset:	soft
Address:	7304Ch-7304Fh
Name:	Pipe Flip Time Stamp
ShortName:	PIPE_FLIPTMSTMP_D
Reset:	soft

DWord	Bit	Description
0	31:0	**Pipe Flip Time Stamp** This field provides read back of the display pipe flip time stamp. The time stamp value is sampled on the start of each flip. The start of flip is when the plane surface address is updated, not when the flip completes. The flip can be through command streamer asynchronous and synchronous flips or MMIO writes. The TIMESTAMP_CTR register has the current time stamp value. Writes to this register will overwrite and update the time stamp value. Flip time stamp sampling is restricted to one plane at a time. The plane select is programmed in PIPE_MISC2->Flip Timestamp Plane Select.

PIPE_FRMCNT

PIPE_FRMCNT	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	70040h-70043h
Name:	Pipe Frame Count
ShortName:	PIPE_FRMCNT_A
Reset:	soft
Address:	71040h-71043h
Name:	Pipe Frame Count
ShortName:	PIPE_FRMCNT_B
Reset:	soft
Address:	72040h-72043h
Name:	Pipe Frame Count
ShortName:	PIPE_FRMCNT_C
Reset:	soft
Address:	73040h-73043h
Name:	Pipe Frame Count
ShortName:	PIPE_FRMCNT_D
Reset:	soft

DWord	Bit	Description
0	31:0	**Pipe Frame Counter** Provides read back of the display pipe frame counter. This counter increments on every start of vertical blank and rolls over back to 0 after (2^32)-1 frames.

PIPE_FRMTMSTMP

PIPE_FRMTMSTMP	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	70048h-7004Bh
Name:	Pipe Frame Time Stamp
ShortName:	PIPE_FRMTMSTMP_A
Reset:	soft
Address:	71048h-7104Bh
Name:	Pipe Frame Time Stamp
ShortName:	PIPE_FRMTMSTMP_B
Reset:	soft
Address:	72048h-7204Bh
Name:	Pipe Frame Time Stamp
ShortName:	PIPE_FRMTMSTMP_C
Reset:	soft
Address:	73048h-7304Bh
Name:	Pipe Frame Time Stamp
ShortName:	PIPE_FRMTMSTMP_D
Reset:	soft

DWord	Bit	Description
0	31:0	**Pipe Frame Time Stamp** This field provides read back of the display pipe frame time stamp. The time stamp value is sampled at every start of vertical blank. The TIMESTAMP_CTR register has the current time stamp value. Writes to this register will overwrite and update the time stamp value.

PIPE_ISOCREQ

PIPE_ISOCREQ - PIPE_ISOCREQ

Register Space:	MMIO: 0/2/0
Size (in bits):	64
Address:	70010h-70017h
Name:	Pipe Isoch Request
ShortName:	PIPE_ISOCREQ_A
Reset:	soft
Address:	71010h-71017h
Name:	Pipe Isoch Request
ShortName:	PIPE_ISOCREQ_B
Reset:	soft
Address:	72010h-72017h
Name:	Pipe Isoch Request
ShortName:	PIPE_ISOCREQ_C
Reset:	soft
Address:	73010h-73017h
Name:	Pipe Isoch Request
ShortName:	PIPE_ISOCREQ_D
Reset:	soft

When enabled, the write to DWord 1 (higher address DWord) of this register triggers an IsocReq to be sent with the last written values from both DWords.

DWord	Bit	Description
0	31:16	**LTR** Access: R/W This field specifies the latency tolerance (LTR) for this pipe in microseconds.
	15:0	**Bandwidth** Access: R/W This field specifies the bandwidth requirement for this pipe in multiples of 100 MB/s.
1	31	**Enable** Access: R/W This field enables IsocReq to be sent when DWord 1 of this register is written. Value: 1b — Name: Enable Value: 0b — Name: Disable

PIPE_ISOCREQ - PIPE_ISOCREQ		
	30:8	**Reserved** Access: RO Format: MBZ
	7:0	**Delay** Access: R/W This field specifies the downwards transition delay for this pipe in milliseconds.

PIPE_ISOCREQ_OFFSET

PIPE_ISOCREQ_OFFSET - PIPE_ISOCREQ_OFFSET

Register Space:	MMIO: 0/2/0
Size (in bits):	64

Address:	70018h-7001Fh
Name:	Pipe Isoch Offset addition
ShortName:	PIPE_ISOCREQ_OFFSET_A
Reset:	soft

Address:	71018h-7101Fh
Name:	Pipe Isoch Offset addition
ShortName:	PIPE_ISOCREQ_OFFSET_B
Reset:	soft

Address:	72018h-7201Fh
Name:	Pipe Isoch Offset addition
ShortName:	PIPE_ISOCREQ_OFFSET_C
Reset:	soft

Address:	73018h-7301Fh
Name:	Pipe Isoch Offset addition
ShortName:	PIPE_ISOCREQ_OFFSET_D
Reset:	soft

The values programmed in this register will added as offsets to the LTR, BW and Delay. SW must ensure to clear these registers if no offset is desired to be added.
For example, If LTR offset is programmed to 0x0100 in this register, then this value is added to the LTR value programmed in PIPE_ISOCREQ.

DWord	Bit	Description
0	31:16	**LTR offset** Access: R/W This field specifies the latency tolerance (LTR) offset to be added to LTR value in PIPE_ISOCREQ for this pipe in microseconds.
	15:0	**Bandwidth offset** Access: R/W This field specifies the bandwidth requirement offset to be added to BW value in the PIPE_ISOCREQ register
1	31:8	**Reserved** Access: RO Format: MBZ

PIPE_ISOCREQ_OFFSET - PIPE_ISOCREQ_OFFSET		
	7:0	**Delay Offset** Access: R/W This field specifies the offset to be added to the downwards transition delay programmed in the PIPE_ISOCREQ.

PIPE_MISC

PIPE_MISC	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank OR pipe disabled

Address:	70030h-70033h
Name:	Pipe Miscellaneous
ShortName:	PIPE_MISC_A
Reset:	soft

Address:	71030h-71033h
Name:	Pipe Miscellaneous
ShortName:	PIPE_MISC_B
Reset:	soft

Address:	72030h-72033h
Name:	Pipe Miscellaneous
ShortName:	PIPE_MISC_C
Reset:	soft

Address:	73030h-73033h
Name:	Pipe Miscellaneous
ShortName:	PIPE_MISC_D
Reset:	soft

DWord	Bit	Description
0	31:30	**Stereo Mask Pipe Int** This field controls which pipe vertical timing (vertical blank, scan line, and vertical sync) events will be reported in interrupts during stereo 3D mode.

Value	Name	Description
00b	Mask None	No masking. Report both the left and right eye vertical events.
01b	Mask Left	Mask the left eye vertical events. Only report right eye events.
10b	Mask Right	Mask the right eye vertical events. Only report left eye events.
11b	Reserved	Reserved

Restriction
This field must be programmed prior to enabling stereo 3D mode and must not be changed while stereo 3D is enabled. In the stacked frame mode the vertical sync is not generated in the gap between left and right eye images, and the scan line count increments across the entire tall frame, so masking may cause unexpected behavior for those events.

PIPE_MISC

29:28 **Stereo Mask Pipe Render**

This field controls which pipe vertical timing (vertical blank, scan line, and vertical sync) events will be reported in render responses during stereo 3D mode.

Value	Name	Description
00b	Mask None	No masking. Report both the left and right eye vertical events.
01b	Mask Left	Mask the left eye vertical events. Only report right eye events.
10b	Mask Right	Mask the right eye vertical events. Only report left eye events.
11b	Reserved	Reserved

Restriction
This field must be programmed prior to enabling stereo 3D mode and must not be changed while stereo 3D is enabled. In the stacked frame mode the vertical sync is not generated in the gap between left and right eye images, and the scan line count increments across the entire tall frame, so masking may cause unexpected behavior for those events.

27 **YUV420 Enable**

This field enables YUV420 output from this pipe. This is only for use with HDMI and DP.

Value	Name
0b	Disable
1b	Enable

Restriction
This field must be programmed prior to enabling the transcoder attached to this pipe.

26 **YUV420 Mode**

This field specifies the mode in which YUV420 pixels are generated by this pipe.

Value	Name
0b	Bypass
1b	Full blend

25 **Pipe Gamma Input Clamp Disable**

This field controls the pipe post csc gamma input clamp operation. When this bit is set to 0b the negative pixel values get clamped to zero at the gamma input.

Value	Name
0b	Enable **[Default]**
1b	Disable

24 **Allow Double Buffer Update Disable**

Access:	R/W

This field controls whether double buffer updates are allowed to be disabled for the double buffered pipe registers listed below. The DOUBLE_BUFFER_CTL register can be configured to globally disable double buffer updates for those resources that allow them to be disabled.

PIPE_MISC

		Value / **Name** 0b — Not Allowed 1b — Allowed **[Default]**
	23	**HDR Mode** This field enables the HDR mode, allowing for higher precision output from the HDR supporting planes and bypassing the SDR planes in blending. In addition to setting bit 8 of this register (Pixel Rounding),this bit must be set to 1b to passthrough the frame buffer pixels unmodified across the pipe. **Value** / **Name** 0b — Disable 1b — Enable
	22	**Change Mask for LDPST** This field controls the change tracking for the LACE. Change tracking can be used by PSR/SRD and WD **Value** / **Name** 0b — Not Masked 1b — Masked
	21	**Change Mask for Register Write** This field controls change tracking for the pipe register write. Change tracking can be used by PSR/SRD and WD. **Value** / **Name** 0b — Not Masked 1b — Masked
	20	**Change Mask for Vblank Vsync Int** This field controls change tracking for the vblank or vsync interrupt enable. Change tracking can be used by PSR/SRD and WD. **Value** / **Name** 0b — Not Masked 1b — Masked
	19	**Reserved**
	18	**Reserved**
	17	**Reserved**
	16	**Reserved**
	15:14	**Rotation Info** This field indicates to internal KVMR screen capture that the display has been rotated through software or hardware rotation. Select the closest value if the rotation is not an exact multiple of 90 degrees. Hardware rotation of the display output is controlled through the plane control registers, not through this field.

PIPE_MISC

Value	Name	Description
00b	None	No rotation on this pipe
01b	90	90 degree rotation on this pipe
10b	180	180 degree rotation on this pipe
11b	270	270 degree rotation on this pipe

Restriction
This field must be programmed in order for internal KVMR screen capture to work correctly when display is rotated by software or hardware.

13 **Reserved**

Access:	RO
Format:	MBZ

12 **OLED Compensation**

This field enables the OLED compensation on the pipe. When this bit is set, plane 5 is used as the OLED compensation plane with up to 10 bits per channel precision. OLED compensation must be used only when the pipe is configured to output RGB format.
The OLED compensation plane size must be same as the pipe active size.

Value	Name
0b	Disable
1b	Enable

11 **Pipe output color space select**

This field indicates the output color space.This field affects the values of the pipe border and some capture functions.This field does not affect the planes, pipe CSC, or ports.

Value	Name
0b	RGB
1b	YUV

Restriction
This field must be set to match the color space that will be output from the pipe CSC or output from the planes if they pipe CSC is bypassed.

10 **xvYCC Color Range Limit**

This field limits the color range of the pipe output to 1 to 254 for 8-bit components, 4 to 1019 for 10bit components, and 16 to 4079 for 12-bit components. Values outside of the range will be clamped to fit within the range.

Value	Name	Description
0b	Full	Do not limit the range
1b	Limit	Limit range

PIPE_MISC

9	**Pixel Extension** This field controls how the pixel extension is handled in the pipe. In scenarios where the frame buffers bpc is larger or equal to the port output bpc, this bit may be programmed to 'Zero Extend'.

Value	Name
0b	MSB Extend **[Default]**
1b	Zero Extend

8	**Pixel Rounding** This field controls the pixel rounding at the end of the pipe. This bit must be set to 1b to passthrough the frame buffer pixels unmodified across the pipe.

Value	Name
0b	Round Up **[Default]**
1b	Truncate

7:5	**Dithering BPC** This field selects the number of bits per color to be used in dithering.

Value	Name	Description
000b	8 bpc	8 bits per color
001b	10 bpc	10 bits per color
010b	6 bpc	6 bits per color
Others	Reserved	Reserved

Programming Notes
When dithering is enabled, the value selected here should match the bits per color selected in the Transcoder DDI Function Control register attached to this pipe.

4	**Dithering enable** This field enables dithering.

Value	Name
0b	Disable
1b	Enable

3:2	**Dithering type** This field selects the dithering type.

Value	Name	Description
00b	Spatial	Spatial
01b	ST1	Spatio-Temporal 1
10b	ST2	Spatio-Temporal 2
11b	Temporal	Temporal

PIPE_MISC		
	1	**Reserved** Access: RO Format: MBZ
	0	**Reserved**

PIPE_MISC2

PIPE_MISC2	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank OR pipe disabled

Address:	7002Ch-7002Fh
Name:	Pipe Miscellaneous 2
ShortName:	PIPE_MISC2_A
Reset:	soft

Address:	7102Ch-7102Fh
Name:	Pipe Miscellaneous 2
ShortName:	PIPE_MISC2_B
Reset:	soft

Address:	7202Ch-7202Fh
Name:	Pipe Miscellaneous 2
ShortName:	PIPE_MISC2_C
Reset:	soft

Address:	7302Ch-7302Fh
Name:	Pipe Miscellaneous 2
ShortName:	PIPE_MISC2_D
Reset:	soft

There is one instance of this register per pipe.

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23:20	**TLB Throttle** Default Value: 8 This field specifies how often the TLB requests are sent. If the programmed value is x, TLBs requests are sent once in x clocks if there are competing data requests.
	19:16	**Reserved** Access: RO Format: MBZ
	15:12	**IPC Demote Req Chunk Size** Default Value: 8 This field limits the request burst sizes while in IPC demote. A value of 0 disables the inflight

limit.

11 Reserved

Access:	RO
Format:	MBZ

10:9 Reserved

Access:	RO
Format:	MBZ

8 ASFU Flip exception

Value	Name	Description
1b	mask	Add exception for Flip for global register update event and Pipe register update event.
0b	No mask	Do not add exception for Flip for global register update event and Pipe register update event.

7 Reserved

Access:	RO
Format:	MBZ

6:4 Scanline Plane Select

This field specifies the plane for which scanline compare fetch line is captured. A programmed value of 0b selects plane 1.

Value	Name
[0h-6h]	

3 Reserved

Access:	RO
Format:	MBZ

2:0 Flip Info Plane Select

This field specifies the plane for which flip information is captured. A programmed value of 0b selects plane 1.

Value	Name
[0h-6h]	

PIPE_MISC3

PIPE_MISC3		
Register Space:	MMIO: 0/2/0	
Access:	Double Buffered	
Size (in bits):	32	
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank OR pipe disabled	
There is one instance of this register per pipe.		
DWord	**Bit**	**Description**
0	31:16	**Reserved**
	15:0	**Reserved** Access: RO Format: MBZ

PIPE_SCANLINE

PIPE_SCANLINE

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	70000h-70003h
Name:	Pipe Scan Line
ShortName:	PIPE_SCANLINE_A
Reset:	soft
Address:	71000h-71003h
Name:	Pipe Scan Line
ShortName:	PIPE_SCANLINE_B
Reset:	soft
Address:	72000h-72003h
Name:	Pipe Scan Line
ShortName:	PIPE_SCANLINE_C
Reset:	soft
Address:	73000h-73003h
Name:	Pipe Scan Line
ShortName:	PIPE_SCANLINE_D
Reset:	soft

This register enables the read back of the pipe vertical line counter. The value increments at the leading edge of HSYNC. The value resets to line zero at the first active line of the display. In interlaced display timings, the scan line counter provides the current line in the field. One field can have a total number of lines that is one greater than the other field.

DWord	Bit	Description
0	31	**Current Field** This is an indication of the current display field.

Value	Name	Description
0b	Odd	First field (odd field)
1b	Even	Second field (even field)

30:20	**Reserved**

Access:	RO
Format:	MBZ

19:0	**Line Counter for Display** This is an indication of the current display scan line.

PIPE_SCANLINE		
		Programming Notes
		The line count value is from the display output timing generator, representing the scan line currently being output to a receiver. Due to buffering within the display engine, the line being fetched (read) from the frame buffer is not directly linked to the line being output. It is possible for the fetched line to be hundreds of lines ahead of the timing generator output line.

PIPE_SCANLINECOMP

PIPE_SCANLINECOMP	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	70004h-70007h
Name:	Pipe Scan Line Compare
ShortName:	PIPE_SCANLINECOMP_A
Reset:	soft
Address:	71004h-71007h
Name:	Pipe Scan Line Compare
ShortName:	PIPE_SCANLINECOMP_B
Reset:	soft
Address:	72004h-72007h
Name:	Pipe Scan Line Compare
ShortName:	PIPE_SCANLINECOMP_C
Reset:	soft
Address:	73004h-73007h
Name:	Pipe Scan Line Compare
ShortName:	PIPE_SCANLINECOMP_D
Reset:	soft

This register is used to initiate a display scan line compare. This MMIO driven scan line compare can not be used at the same time as the command streamer driven scan line compare on the same pipe. When this register is written with the Initiate Compare bit set to 1b, the Display Engine (DE) will start comparing the display pipe or plane (selectable) current scan line value (current scan line) with the start scan line value (current scan line >= start scan line) and the end scan line value (current scan line <= end scan line) to decide if the pipe scan line is inside or outside the scan line window of interest. DE will wait until the current scan line is either outside (Inclusive mode) or inside (Exclusive mode) the scan line window, then trigger a scan line event and stop any further comparing. The scan line event can cause display to send a scan line compare response to the command streamer, (used for releasing a MI_WAIT_FOR_EVENT on scan line window), if unmasked in the DERRMR mask register 0x44050. The scan line event can also cause display to generate a scan line compare interrupt, if the interrupt registers are configured for that. The value programmed should be the desired value - 1, so for scan line 0, the value programmed is vertical total, and for scan line 1, the value programmed is 0. The programmable range can include the vertical blank. In interlaced display timings, the current scan line is the current line of the current interlaced field. Either MMIO or a MI_LOAD_REGISTER_IMM command can be used to unmask the scan line render response 0x44050. That can be done anytime before programming this register. There is one instance of this register per pipe.

Restriction
A new scan line compare must not be started until after the previous compare has finished. The end scan line value must be greater than or equal to the start scan line value.When using LRI care must be taken to follow all

PIPE_SCANLINECOMP

the programming rules for LRI targeting the display engine.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31</td><td>Initiate Compare
This field initiates the scan line compare. When this register is written with this bit set to 1b, the display engine will do one complete comparison cycle, trigger a scan line event, then stop comparing.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0b</td><td>Do nothing</td></tr>
<tr><td>1b</td><td>Initiate compare</td></tr>
</table>
<table>
<tr><th>Restriction</th></tr>
<tr><td>Do not write this register again until after any previous scan line compare has completed.</td></tr>
</table></td></tr>
<tr><td></td><td>30</td><td>Inclusive Exclusive Select
This field selects whether the scan line compare is done in inclusive mode, where display triggers the scan line event when outside the scan line window, or inclusive mode, where display triggers when inside the window.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0b</td><td>Exclusive</td><td>Exclusive mode: trigger scan line event when inside the scan line window</td></tr>
<tr><td>1b</td><td>Inclusive</td><td>Inclusive mode: trigger scan line event when outside the scan line window</td></tr>
</table></td></tr>
<tr><td></td><td>29</td><td>Counter Select
This field selects whether the scan line compare is done using the pipe timing generator scanline counter or a plane scanline counter. The pipe timing generator counts the scanlines being output from display. The plane counts the scan lines being fetched from the frame buffer.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0b</td><td>Timing generator</td><td>Use the scanline count from the pipe timing generator</td></tr>
<tr><td>1b</td><td>Plane</td><td>Use the scanline count from plane selected in PIPE_MISC2.</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>Due to buffering within the display engine, the line being fetched from the frame buffer is not directly linked to the line being output. It is possible for the fetched line to be hundreds of lines ahead of the timing generator output line. The plane scan line count more closely represents what data is currently being fetched by the plane.</td></tr>
</table></td></tr>
<tr><td></td><td>28:16</td><td>Start Scan Line
This field specifies the starting scan line number of the scan line window.</td></tr>
<tr><td></td><td>15</td><td>Render Response Destination
This bit indicates what destination to send the scan line event render response to.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0b</td><td>CS</td><td>Send scan line event response to CS</td></tr>
<tr><td>1b</td><td>BCS</td><td>Send scan line event response to BCS</td></tr>
</table></td></tr>
</table>

PIPE_SCANLINECOMP		
	14:13	**Reserved** Access: RO Format: MBZ
	12:0	**End Scan Line** This field specifies the ending scan line number of the scan line window.

PIPE_SEAM_EXCESS

PIPE_SEAM_EXCESS - PIPE_SEAM_EXCESS	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank OR pipe disabled
Address:	60020h-60023h
Name:	Pipe Seam Excess
ShortName:	PIPE_SEAM_EXCESS_A
Reset:	soft
Address:	61020h-61023h
Name:	Pipe Seam Excess
ShortName:	PIPE_SEAM_EXCESS_B
Reset:	soft
Address:	62020h-62023h
Name:	Pipe Seam Excess
ShortName:	PIPE_SEAM_EXCESS_C
Reset:	soft
Address:	63020h-63023h
Name:	Pipe Seam Excess
ShortName:	PIPE_SEAM_EXCESS_D
Reset:	soft

This register defines the number of excess pixels within the Pipe window (on the right or left) that the Scaler will need to remove from the post scaled image.
When an image is split across two Pipes, scaled, and then joined at the Port, the Scalers within each Pipe will operate on a split image that contains overlap pixels around where the final seam will be to facilitate a seamless join at the Port. For example, if the left portion of an image is being scaled in Pipe A and the right portion of the image is being scaled in Pipe B, then there will be an excess number of pixels (i.e. overlap pixels) on the right side of the Pipe A image and an excess number of pixels on the left side of the Pipe B image. The overlap pixels of the window within each of the Pipes need to be dropped by the Scaler before they are delivered to the Port.
Notes:

1. Dropping of the overlap/excess pixels is done at the very end of the Pipe within the Scaler regardless of whether a Scaler is bound to the Pipe, or not.
2. The values programmed within this register are one-based (i.e. a programming of 1 equals 1 pixel of excess)
3. The values programmed within this register will be added to the Horizontal Active programming of the TRANS_HTOTAL register of the port bound to this pipe. I.e. the pipe will see a Horizontal size equal to Horizontal Active + Left Excess Amount + Right Excess Amount

PIPE_SEAM_EXCESS - PIPE_SEAM_EXCESS

Restriction :

1. The number of excess pixels cannot exceed the size of the horizontal blank, otherwise there will not be enough time to throw them away before starting the next line and the image will be corrupted
2. Pillarbox boarders must be even
3. The source size on each pipe, including pre-scale excess, must be a multiple of 2. When the Pipe output format is YUV 420 with full blend, the source size is required to be a multiple of 4.

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**Right Excess Amount** This field defines the number of excess pixels to drop, if any, on the right side of the Pipe window
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**Left Excess Amount** This field defines the number of excess pixels to drop, if any, on the left side of the Pipe window

PIPE_SRCSZ

PIPE_SRCSZ	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank

Address:	6001Ch-6001Fh
Name:	Pipe Source Image Size
ShortName:	PIPE_SRCSZ_A
Reset:	soft

Address:	6101Ch-6101Fh
Name:	Pipe Source Image Size
ShortName:	PIPE_SRCSZ_B
Reset:	soft

Address:	6201Ch-6201Fh
Name:	Pipe Source Image Size
ShortName:	PIPE_SRCSZ_C
Reset:	soft

Address:	6301Ch-6301Fh
Name:	Pipe Source Image Size
ShortName:	PIPE_SRCSZ_D
Reset:	soft

There is one instance of this register for each pipe.

Programming Notes

In VGA display mode, this register is ignored and the VGA size from the VGA registers is used instead.

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**Horizontal Source Size** This field specifies Horizontal Source Size. This determines the horizontal size of the image created by the display planes. This field is programmed to the number of pixels desired minus one. **Restriction** • This register must always be programmed to the same value as the Horizontal Active, except when panel fitting is enabled or DSC pixel replication is enabled. • Refer to PS_CTRL for size restrictions when panel fitting is enabled.

PIPE_SRCSZ		
		• Horizontal source size must always be even. The programmed value must be odd.
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**Vertical Source Size** This field specifies Vertical Source Size. This determines the vertical size of the image created by the display planes. This field is programmed to the number of lines desired minus one. **Restriction** Vertical source sizes larger than 4320 lines are not supported. This register must always be programmed to the same value as the Vertical Active, except when panel fitting is enabled. Refer to PS_CTRL for size restrictions when panel fitting is enabled.

PIPE_STATUS

PIPE_STATUS	
Register Space:	MMIO: 0/2/0
Access:	R/WC
Size (in bits):	32
CrashLogSaved:	true
CrashLogPriority:	1
CrashLogVisibility:	public
ExternalLongName:	DE Pipe Status
ExternalDescription:	Display engine pipe status
Address:	70058h-7005Bh
Name:	Pipe Status
ShortName:	PIPE_STATUS_A
Reset:	soft
Address:	71058h-7105Bh
Name:	Pipe Status
ShortName:	PIPE_STATUS_B
Reset:	soft
Address:	72058h-7205Bh
Name:	Pipe Status
ShortName:	PIPE_STATUS_C
Reset:	soft
Address:	73058h-7305Bh
Name:	Pipe Status
ShortName:	PIPE_STATUS_D
Reset:	soft

DWord	Bit	Description
0	31	**Underrun** Access: R/WC This field indicates that there is an underrun on the transcoder attached to this pipe.
	30	**Vblank** Access: R/WC The field is set at the start of the vertical blank of the transcoder attached to this pipe.
	29	**Frame start** Access: R/WC The field is set at the frame start of the transcoder attached to this pipe.
	28	**Not Used 28**

PIPE_STATUS		
	27	**Not Used 27**
	26	**Not Used 26** Access: R/WC
	25	**Not Used 25** Access: R/WC
	24	**Not Used 24** Access: R/WC
	23	**Not Used 23** Access: R/WC
	22	**Not Used 22** Access: R/WC
	21	**Not Used 21** Access: R/WC
	20	**Not Used 20** Access: R/WC
	19	**Not Used 19** Access: R/WC
	18	**Not Used 18** Access: R/WC
	17	**Not Used 17** Access: R/WC
	16	**Not Used 16** Access: R/WC
	15	**Not Used 15** Access: R/WC
	14	**Not Used 14** Access: R/WC
	13	**Not Used 13** Access: R/WC
	12	**Not Used 12** Access: R/WC
	11	**Not Used 11** Access: R/WC
	10	**Not Used 10** Access: R/WC

PIPE_STATUS		
	9	**Not Used 9** Access: R/WC
	8	**Not Used 8** Access: R/WC
	7	**Not Used 7** Access: R/WC
	6	**BW Credits Pending At VBlank** Access: R/WC A '1' indicates that the there are some pending MBUS BW-Credits at the start of VBlank. Sticky bit cleared by a write of '1'
	5	**B Credits Pending At VBlank** Access: R/WC A '1' indicates that the there are some pending MBUS B-Credits at the start of VBlank. Sticky bit cleared by a write of '1'
	4	**A Credits Pending At VBlank** Access: R/WC A '1' indicates that the there are some pending MBUS A-Credits at the start of VBlank. Sticky bit cleared by a write of '1'
	3	**Not Used 3** Access: R/WC
	2	**Not used 2** Access: R/WC
	1	**Valid Block At FrameStart** Access: R/WC A '1' indicates that a valid block is still present in Display Buffer at frame start. Sticky bit cleared by a write of '1'.
	0	**Valid Block Overwritten** Access: R/WC A '1' indicates that a valid block in Display Buffer was overwritten. Sticky bit cleared by a write of '1'.

PIPEDMC_CONTROL

PIPEDMC_CONTROL - PIPEDMC_CONTROL	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	45250h-45253h
Name:	Pipe DMC Control
ShortName:	PIPEDMC_CONTROL_A
Reset:	soft
Address:	45254h-45257h
Name:	Pipe DMC Control
ShortName:	PIPEDMC_CONTROL_B
Reset:	soft
Address:	45258h-4525Bh
Name:	Pipe DMC Control
ShortName:	PIPEDMC_CONTROL_C
Reset:	soft
Address:	4525Ch-4525Fh
Name:	Pipe DMC Control
ShortName:	PIPEDMC_CONTROL_D
Reset:	soft

This Register is to add pipe DMC enable. To use pipeDMC driver must enable pipeDMC first.

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**pipedmc_enable** Access: R/W Setting this bit enables the pipeDMC.

Pixel Shader Scheduling Mode

PSS_MODE - Pixel Shader Scheduling Mode

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	07038h
Name:	Pixel Shader Scheduling Mode
ShortName:	PSS_MODE

Mode registers for Pixel Shader Scheduling.

DWord	Bit	Description
0	31:16	**Mask** Access: WO _Custom_GTIReset: DEV Must be set to modify corresponding bit in Bits 15:0 (All implemented bits)
	15:14	**Reserved** Access: R/W Format: PBC _Custom_GTIReset: DEV
	13:10	**Maximum Allowed PS Thread Dependencies** Default Value: 8h Access: R/W Format: U4 _Custom_GTIReset: DEV Sets the maximum number of dependencies that a pixel shader thread can have. If a thread requires more than this number, thread dispatch will be stalled until enough threads that this thread is dependent on have retired. Since maximum number of dependencies in a Pixel Shader is 8 (for fused-SIMD16, corresponding to 8 2X2 pixel block; fused-SIMD32 corresponding to 8 4x2 pixel blocks), programming a value higher than 0X8 is illegal. If HW chooses to launch lower SIMD thread e.g. fused-SIMD8, it will not have more than 4 dependencies. Hence, programming 0x8 guarantees maximum dependencies for all SIMD width dispatches.
	9	**3D Scoreboard Address XOR Disable** Access: R/W Format: Disable _Custom_GTIReset: DEV

Value	Name	Description
0h	Enabled **[Default]**	3D Scoreboard Address calculation includes XOR of higher bits
1h	Disabled	3D Scoreboard Address calculation based on lower address bits only

PSS_MODE - Pixel Shader Scheduling Mode

8 Reserved

Access:	R/W
Format:	PBC
_Custom_GTIReset:	DEV

7 Stalling behavior disable for PS Dispatch for PTBR

Access:	R/W
_Custom_GTIReset:	DEV

This bit, when enabled, allows PS dispatch function to not stall when no TILE ID is free for incoming new tile access under PTBR. Although this behavior is non stalling, it has a potential to reduce number of color pixel blocks from tile cache in certain cases.

Value	Name	Description
0	**[Default]**	This is a default value and default behavior is to have PS dispatch stall when there are no free TILE_IDs available for incoming new access on a new TILE under PTBR.
1		When this bit is set, PS dispatch does not stall for a new incoming tile but simply uses the next TILE_ID without waiting for it to be free. SW should enable this bit if there are perf downsides with stalling behavior and additional discard BW is not significant.

6:5 Thread Scheduler Mode

Access:	R/W
Format:	U2
_Custom_GTIReset:	DEV

Selects the PSD selection policy

Value	Name	Description
0h	8x8 Round-Robin **[Default]**	Scheduler alternates PSDs to load entire 8x8
1h	8x8 Fixed	Scheduler statically hashes each 8x8 based on xy address
2h	8x8 Dynamic	At the start of each 8x8, scheduler selects a PSD to load entire 8x8 based on EU load
3h	Reserved	

4 Disable 4x4 schedule optimization

Access:	R/W
Format:	Disable
_Custom_GTIReset:	DEV

If set, prevent scheduler from attempting to pack 4x4 at SIMD16 alignment.

Value	Name
0h	4x4 packed at SIMD16 alignment if possible **[Default]**
1h	scheduler does not consider 4x4 when packing threads

PSS_MODE - Pixel Shader Scheduling Mode

	3	**Disable 4x2 schedule optimization** Access: R/W Format: Disable _Custom_GTIReset: DEV If set, prevent scheduler from attempting to pack 4x2 at SIMD8 alignment. **Value** / **Name** 0h: 4x2 packed at SIMD8 alignment if possible **[Default]** 1h: scheduler does not consider 4x2 when packing threads
	2	**Disable fused thread scheduling** Access: R/W Format: Disable _Custom_GTIReset: DEV If set, scheduler will only populate the first thread of a fused thread. A fused thread will still be dispatched, but will only contain a single thread. **Value** / **Name** 0h: Fused thread enabled **[Default]** 1h: Fused thread disabled
	1:0	**Limit maximum number of polys per fused-thread** Access: R/W Format: U2 _Custom_GTIReset: DEV Limit the scheduler to pack no more than the indicated number polys into a single fused-thread. **Value** / **Name** 0h: Polys per fused-thread is not limited **[Default]** 1h: Maximum polys per fused-thread limited to 1 2h: Maximum polys per fused-thread limited to 2 3h: Maximum polys per fused-thread limited to 3

Pixel Shader Scheduling Mode 2

PSS_MODE2 - Pixel Shader Scheduling Mode 2

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	0703Ch
Name:	Pixel Shader Scheduling Mode
ShortName:	PSS_MODE2

Mode register for Pixel Shader Scheduling.

DWord	Bit	Description
0	31:16	**Mask** Access: WO _Custom_GTIReset: DEV Must be set to modify corresponding bit in Bits 15:0 (All implemented bits)
	15:9	**Reserved** Access: R/W Format: PBC _Custom_GTIReset: DEV
	8	**PS Load Bias match size** Access: R/W _Custom_GTIReset: DEV Select the granularity of xy compare for determining if the incoming 8x8 matches history buffer used for **8x8 Match PS Load Bias**. Value: 0h — Name: 8x8 **[Default]** Value: 1h — Name: 16x16
	7:6	**8x8 Match PS Load Bias** Access: R/W Format: U2 _Custom_GTIReset: DEV When scheduler is in dynamic mode, PSS will be create a load imbalance by the selected number of threads when a new 8x8 matches history buffer or any partially collected threads. This will direct matching 8x8 to the same PSD when load is not extremely skewed. Value: 0h — Name: 2_7 **[Default]** Value: 1h — Name: 4_7 Value: 2h — Name: 8_7 Value: 3h — Name: 0_7

PSS_MODE2 - Pixel Shader Scheduling Mode 2

5 Scoreboard Stall Flush Control

Access:	R/W
_Custom_GTIReset:	DEV

Value	Name	Description
0h	Start Thread Group **[Default]**	pipe_control with **Stall at Scoreboard** set creates a new thread group instead of stalling at scoreboard. PS threads launched after pipe_control will have sendc dependency to all threads before pipe_control. This is a pipelined flush and will not drain EUs of PS threads.
1h	Scoreboard Stall	pipe_control with **Stall at Scoreboard** set stalls at PSS scoreboard until all previous threads have completed. This will drain EUs of all PS threads.

4 Thread Group Dependency Control

Access:	R/W
_Custom_GTIReset:	DEV

Value	Name
0h	Use 9th dependency to enforce ordering between thread groups. **[Default]**
1h	Perform stall at PSSunit when thread group changes.

3 Over-subscribe PS thread allocation

Access:	R/W
Format:	Enable
_Custom_GTIReset:	DEV

Value	Name
0h	Number of enabled threads does not affect PS thread scheduling **[Default]**
1h	Number of PS threads scheduled per-PSD will not exceed number of enabled threads

2:0 PS Dispatch Time-out Values

Access:	R/W
_Custom_GTIReset:	DEV

Valid values other than 111, allow SW to program number of clocks PS dispatch function wait to combine fuse thread sequence (aka bundle).
Setting 111 disables the time-out mechanism and will wait indefinitely for an event to end the collection of pixels.

Value	Name	Description	Programming Notes
0h	32_2	PS dispatch function waits for 32 clocks for gathering more pixels in the current bundle.	
1h	64	PS dispatch function waits for 64	

PSS_MODE2 - Pixel Shader Scheduling Mode 2

				clocks for gathering more pixels in the current bundle.	
		2h	128	PS dispatch function waits for 128 clocks for gathering more pixels in the current bundle.	
		3h	256	PS dispatch function waits for 256 clocks for gathering more pixels in the current bundle.	
		4h	512	PS dispatch function waits for 512 clocks for gathering more pixels in the current bundle.	
		5h	1024	PS dispatch function waits for 1024 clocks for gathering more pixels in the current bundle.	
		6h	2048 **[Default]**	PS dispatch function waits for 2048 clocks for gathering more pixels in the current bundle.	
		7h		PSD waits indefinitely to assemble a bundle before dispatching a PS.	This value may not be programmed if non-promoted depth or stencil is enabled.

PLANE_AUX_DIST

PLANE_AUX_DIST	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF or plane not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled; after armed
Address:	704C0h-704C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_4_A
Reset:	soft
Address:	705C0h-705C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_5_A
Reset:	soft
Address:	706C0h-706C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_6_A
Reset:	soft
Address:	707C0h-707C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_7_A
Reset:	soft
Address:	714C0h-714C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_4_B
Reset:	soft
Address:	715C0h-715C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_5_B
Reset:	soft
Address:	716C0h-716C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_6_B
Reset:	soft
Address:	717C0h-717C3h

PLANE_AUX_DIST	
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_7_B
Reset:	soft
Address:	724C0h-724C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_4_C
Reset:	soft
Address:	725C0h-725C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_5_C
Reset:	soft
Address:	726C0h-726C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_6_C
Reset:	soft
Address:	727C0h-727C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_7_C
Reset:	soft
Address:	734C0h-734C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_4_D
Reset:	soft
Address:	735C0h-735C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_5_D
Reset:	soft
Address:	736C0h-736C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_6_D
Reset:	soft
Address:	737C0h-737C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_7_D
Reset:	soft
Address:	701C0h-701C3h

PLANE_AUX_DIST	
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_1_A
Reset:	soft
Address:	702C0h-702C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_2_A
Reset:	soft
Address:	703C0h-703C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_3_A
Reset:	soft
Address:	711C0h-711C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_1_B
Reset:	soft
Address:	712C0h-712C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_2_B
Reset:	soft
Address:	713C0h-713C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_3_B
Reset:	soft
Address:	721C0h-721C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_1_C
Reset:	soft
Address:	722C0h-722C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_2_C
Reset:	soft
Address:	723C0h-723C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_3_C
Reset:	soft
Address:	731C0h-731C3h

PLANE_AUX_DIST

Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_1_D
Reset:	soft

Address:	732C0h-732C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_2_D
Reset:	soft

Address:	733C0h-733C3h
Name:	Plane Auxiliary Surface Distance
ShortName:	PLANE_AUX_DIST_3_D
Reset:	soft

This register is used to specify the distance from the main surface base address and the stride of the auxiliary surface.Unlike the surface base address, this register value cannot be updated through flips.

DWord	Bit	Description
0	31:12	**Auxiliary Surface Distance** When using a compressed surface, this field represents the distance of the control surface in 4K pages, where a value of [31:12] = 1 represents one 4K page.
	11:10	**Reserved** Access: RO Format: MBZ
	9:0	**Auxiliary Surface Stride** **Description** This field is unused. Leave at default value.

PLANE_BUF_CFG

PLANE_BUF_CFG	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF or plane not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank, plane not enabled, or pipe not enabled
Address:	7017Ch-7017Fh
Name:	Cursor Buffer Config
ShortName:	CUR_BUF_CFG_A
Reset:	soft
Address:	7117Ch-7117Fh
Name:	Cursor Buffer Config
ShortName:	CUR_BUF_CFG_B
Reset:	soft
Address:	7217Ch-7217Fh
Name:	Cursor Buffer Config
ShortName:	CUR_BUF_CFG_C
Reset:	soft
Address:	7317Ch-7317Fh
Name:	Cursor Buffer Config
ShortName:	CUR_BUF_CFG_D
Reset:	soft
Address:	7057Ch-7057Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_4_A
Reset:	soft
Address:	7067Ch-7067Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_5_A
Reset:	soft
Address:	7077Ch-7077Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_6_A
Reset:	soft
Address:	7087Ch-7087Fh

PLANE_BUF_CFG	
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_7_A
Reset:	soft
Address:	7157Ch-7157Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_4_B
Reset:	soft
Address:	7167Ch-7167Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_5_B
Reset:	soft
Address:	7177Ch-7177Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_6_B
Reset:	soft
Address:	7187Ch-7187Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_7_B
Reset:	soft
Address:	7257Ch-7257Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_4_C
Reset:	soft
Address:	7267Ch-7267Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_5_C
Reset:	soft
Address:	7277Ch-7277Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_6_C
Reset:	soft
Address:	7287Ch-7287Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_7_C
Reset:	soft
Address:	7357Ch-7357Fh

PLANE_BUF_CFG	
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_4_D
Reset:	soft
Address:	7367Ch-7367Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_5_D
Reset:	soft
Address:	7377Ch-7377Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_6_D
Reset:	soft
Address:	7387Ch-7387Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_7_D
Reset:	soft
Address:	7027Ch-7027Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_1_A
Reset:	soft
Address:	7037Ch-7037Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_2_A
Reset:	soft
Address:	7047Ch-7047Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_3_A
Reset:	soft
Address:	7127Ch-7127Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_1_B
Reset:	soft
Address:	7137Ch-7137Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_2_B
Reset:	soft
Address:	7147Ch-7147Fh

PLANE_BUF_CFG

Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_3_B
Reset:	soft

Address:	7227Ch-7227Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_1_C
Reset:	soft

Address:	7237Ch-7237Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_2_C
Reset:	soft

Address:	7247Ch-7247Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_3_C
Reset:	soft

Address:	7327Ch-7327Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_1_D
Reset:	soft

Address:	7337Ch-7337Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_2_D
Reset:	soft

Address:	7347Ch-7347Fh
Name:	Plane Buffer Config
ShortName:	PLANE_BUF_CFG_3_D
Reset:	soft

DWord	Bit	Description
0	31:28	**Reserved** Access: RO Format: MBZ
	27	**Reserved** Access: RO Format: MBZ
	26:16	**Buffer End** Default Value: 000h

PLANE_BUF_CFG		
		This field contains the buffer end position for this plane.
	15:12	**Reserved** Access: RO Format: MBZ
	11	**Reserved** Access: RO Format: MBZ
	10:0	**Buffer Start** Default Value: 000h This field contains the buffer start position for this plane.

PLANE_CC_VAL

PLANE_CC_VAL	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	64
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF or plane not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled; after armed
Address:	704B4h-704BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_4_A
Reset:	soft
Address:	705B4h-705BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_5_A
Reset:	soft
Address:	706B4h-706BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_6_A
Reset:	soft
Address:	707B4h-707BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_7_A
Reset:	soft
Address:	714B4h-714BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_4_B
Reset:	soft
Address:	715B4h-715BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_5_B
Reset:	soft
Address:	716B4h-716BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_6_B
Reset:	soft
Address:	717B4h-717BBh

PLANE_CC_VAL	
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_7_B
Reset:	soft
Address:	724B4h-724BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_4_C
Reset:	soft
Address:	725B4h-725BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_5_C
Reset:	soft
Address:	726B4h-726BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_6_C
Reset:	soft
Address:	727B4h-727BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_7_C
Reset:	soft
Address:	734B4h-734BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_4_D
Reset:	soft
Address:	735B4h-735BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_5_D
Reset:	soft
Address:	736B4h-736BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_6_D
Reset:	soft
Address:	737B4h-737BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_7_D
Reset:	soft
Address:	701B4h-701BBh

PLANE_CC_VAL	
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_1_A
Reset:	soft
Address:	702B4h-702BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_2_A
Reset:	soft
Address:	703B4h-703BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_3_A
Reset:	soft
Address:	711B4h-711BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_1_B
Reset:	soft
Address:	712B4h-712BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_2_B
Reset:	soft
Address:	713B4h-713BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_3_B
Reset:	soft
Address:	721B4h-721BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_1_C
Reset:	soft
Address:	722B4h-722BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_2_C
Reset:	soft
Address:	723B4h-723BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_3_C
Reset:	soft
Address:	731B4h-731BBh

PLANE_CC_VAL

Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_1_D
Reset:	soft

Address:	732B4h-732BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_2_D
Reset:	soft

Address:	733B4h-733BBh
Name:	Plane Clear Color Value
ShortName:	PLANE_CC_VAL_3_D
Reset:	soft

This register programs the clear color value to be used with render decompression. The value is used only when render decompression and clear color are both enabled in the plane control register.
The register value can be updated when flipping to a new surface with new clear color value. It does not need to be updated if the new surface has the same clear color value as the previous surface.

DWord	Bit	Description
0	31:0	**Clear Color Value DW0** This field gives the 32 bit value of the clear color.
1	31:0	**Clear Color Value DW1** This field gives the upper 32 bit value of the clear color. This field is used only with 64 bits formats, ignored otherwise.

PLANE_COLOR_CTL

PLANE_COLOR_CTL	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF or plane not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled; after armed
Address:	704CCh-704CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_4_A
Reset:	soft
Address:	705CCh-705CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_5_A
Reset:	soft
Address:	706CCh-706CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_6_A
Reset:	soft
Address:	707CCh-707CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_7_A
Reset:	soft
Address:	714CCh-714CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_4_B
Reset:	soft
Address:	715CCh-715CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_5_B
Reset:	soft
Address:	716CCh-716CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_6_B
Reset:	soft
Address:	717CCh-717CFh

PLANE_COLOR_CTL	
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_7_B
Reset:	soft
Address:	724CCh-724CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_4_C
Reset:	soft
Address:	725CCh-725CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_5_C
Reset:	soft
Address:	726CCh-726CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_6_C
Reset:	soft
Address:	727CCh-727CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_7_C
Reset:	soft
Address:	734CCh-734CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_4_D
Reset:	soft
Address:	735CCh-735CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_5_D
Reset:	soft
Address:	736CCh-736CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_6_D
Reset:	soft
Address:	737CCh-737CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_7_D
Reset:	soft
Address:	701CCh-701CFh

PLANE_COLOR_CTL	
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_1_A
Reset:	soft
Address:	702CCh-702CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_2_A
Reset:	soft
Address:	703CCh-703CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_3_A
Reset:	soft
Address:	711CCh-711CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_1_B
Reset:	soft
Address:	712CCh-712CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_2_B
Reset:	soft
Address:	713CCh-713CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_3_B
Reset:	soft
Address:	721CCh-721CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_1_C
Reset:	soft
Address:	722CCh-722CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_2_C
Reset:	soft
Address:	723CCh-723CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_3_C
Reset:	soft
Address:	731CCh-731CFh

PLANE_COLOR_CTL

Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_1_D
Reset:	soft

Address:	732CCh-732CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_2_D
Reset:	soft

Address:	733CCh-733CFh
Name:	Plane Color Control
ShortName:	PLANE_COLOR_CTL_3_D
Reset:	soft

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Pipe Gamma Enable** This bit enables pipe gamma correction for the plane pixel data. This field is deprecated. Use 'GAMMA_MODE.Post CSC Gamma Enable' for enabling pipe gamma across all pixels from all planes. Value 0b: Disable Value 1b: Enable
	29	**Remove YUV Offset** This field controls whether the plane removes or preserves the 1/2 offset on U and V components when the source pixel format is YUV and the plane YUV to RGB CSC is disabled. This bit has no effect on RGB source pixel formats Value 0b: Remove: Remove 1/2 offset on UV components Value 1b: Preserve: Preserve 1/2 offset on UV components
	28	**YUV Range Correction Disable** Setting this bit disables the YUV range correction logic inside the plane. The range correction logic is used to expand the compressed range YUV to full range YUV. The Y channel is expanded from the 8 bit +16 to +235 range to full range. The U and V channels are expanded from the 8 bit -112 to +112 range to full range. Extended range values will be preserved after the expansion. This bit has no effect on RGB source pixel formats since they automatically bypass range correction.

PLANE_COLOR_CTL

Value	Name
0b	Enable
1b	Disable

27:24 **Reserved**

Access:	RO
Format:	MBZ

23 **Pipe CSC Enable**

This bit enables pipe color space conversion and the pipe pre color space conversion gamma for the plane pixel data. This is separate from the color conversion logic within the plane.

This field is deprecated.
Use 'CSC_MODE.Pipe CSC Enable', 'GAMMA_MODE.Pre CSC Gamma Enable' for enabling pipe color space conversion and gamma respectively across all pixels from all planes. Plane CSC must be used for plane specific color space conversion.

Value	Name
0b	Disable
1b	Enable

22 **Reserved**

21 **Plane CSC Enable**

This field enables the plane color space conversion. This field applies only to planes 1 through 3.

Value	Name
0b	Disable
1b	Enable

20 **Plane Input CSC Enable**

This field enables the plane input color space conversion. This field applies only to planes 1 through 3.

Value	Name
0b	Disable
1b	Enable

19:17 **Plane CSC Mode**

Description
This field specifies the mode of plane color space conversion operation.
This is used only for planes 4 through 7. For planes 1 through 3, CSC is programmed in PLANE_CSC_* registers.

Value	Name	Description
000b	Bypass	Pixel data bypasses the plane color space conversion
001b	YUV601 to RGB601	YUV BT.601 to RGB BT.601 conversion.

010b	YUV709 to RGB709	YUV BT.709 to RGB BT.709 conversion.
011b	YUV2020 to RGB2020	YUV BT.2020 to RGB BT.2020 conversion.
100b	RGB709 to RGB2020	RGB BT.709 to RGB BT.2020 conversion.

16 **Reserved**

Access:	RO
Format:	MBZ

15 **Plane Post CSC Gamma Multi Segment Enable**

This bit enables plane post CSC gamma multi segment processing. It is only used for HDR tone Mapping. It is only valid if Plane Gamma (bit[13]) is enabled.

Value	Name
1b	Enable
0b	Disable **[Default]**

14 **Plane Pre CSC Gamma Enable**

This bit controls plane internal pre-CSC gamma correction.

Value	Name
1b	Enable
0b	Disable

13 **Plane Gamma Disable**

This bit controls plane internal post-CSC gamma correction.

Value	Name
1b	Disable
0b	Enable

12 **Plane Gamma Mode**

This field specifies the plane gamma mode of operation. This field is ignored if plane gamma is disabled.

Value	Name	Description
0b	Direct **[Default]**	Direct mode is used for regular plane gamma programming. Lookup is based on incoming pixel individual r, g, b values. The output is a computed by lookup of two nearest points and interpolation.
1b	Multiply	[] Multiple mode is used when plane gamma is used for HDR tone mapping. Lookup is based on a pseudo luminance of the incoming pixel calculated using Lin = 0.25*Red input + 0.625*Green input + 0.125*Blue input. An adjustment factor 'F' is computed by lookup of two nearest points and interpolation. Output is computed by multiplying each color channel with the adjustment factor F.

11 **Plane Gamma Multiplier Precision**

This field specifies the plane gamma entry format in the multiplier mode. This field is ignored in the direct lookup mode. The gamma entries can be programmed in either unsigned 0.24 format or unsigned 8.16 format.

PLANE_COLOR_CTL

Value	Name
0b	U0.24 **[Default]**
1b	U8.16

10:6 **Reserved**

Access:	RO
Format:	MBZ

5:4 **Alpha Mode**

This field controls how the plane will use per pixel alpha data from frame buffer. Constant plane alpha is defined in PLANE_KEYMSK and PLANE_KEYMAX registers.

RGB 64-bit - only alpha in 0-1 range supported with 8 bit granularity.
RGB 64-bit UINT - only 8 upper bits of alpha used.
RGB 2:10:10:10 - 2 bit alpha expanded out to 8 bit to give full range of opacity.
XR_BIAS 10:10:10 - 2 bit alpha expanded out to 8 bit to give full range of opacity.

Value	Name	Description
00b	Disable	Alpha channel ignored.
10b	Enable with SW pre-multiply	Alpha channel used. Color channels should be pre-multiplied with alpha by software.
11b	Enable with HW pre-multiply	Alpha channel used. Color channels will be pre-multiplied with alpha by hardware.

Restriction
Per pixel alpha is supported only with RGB pixel formats.
FBC is not compatible with per pixel alpha.

3:0 **Reserved**

Access:	RO
Format:	MBZ

PLANE_CSC_COEFF

PLANE_CSC_COEFF	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	192
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank after armed
Address:	70210h-70227h
Name:	Plane CSC Coefficients
ShortName:	PLANE_CSC_COEFF_1_A
Reset:	soft
Address:	70310h-70327h
Name:	Plane CSC Coefficients
ShortName:	PLANE_CSC_COEFF_2_A
Reset:	soft
Address:	70410h-70427h
Name:	Plane CSC Coefficients
ShortName:	PLANE_CSC_COEFF_3_A
Reset:	soft
Address:	71210h-71227h
Name:	Plane CSC Coefficients
ShortName:	PLANE_CSC_COEFF_1_B
Reset:	soft
Address:	71310h-71327h
Name:	Plane CSC Coefficients
ShortName:	PLANE_CSC_COEFF_2_B
Reset:	soft
Address:	71410h-71427h
Name:	Plane CSC Coefficients
ShortName:	PLANE_CSC_COEFF_3_B
Reset:	soft
Address:	72210h-72227h
Name:	Plane CSC Coefficients
ShortName:	PLANE_CSC_COEFF_1_C
Reset:	soft
Address:	72310h-72327h

PLANE_CSC_COEFF

Name:	Plane CSC Coefficients
ShortName:	PLANE_CSC_COEFF_2_C
Reset:	soft

Address:	72410h-72427h
Name:	Plane CSC Coefficients
ShortName:	PLANE_CSC_COEFF_3_C
Reset:	soft

Address:	73210h-73227h
Name:	Plane CSC Coefficients
ShortName:	PLANE_CSC_COEFF_1_D
Reset:	soft

Address:	73310h-73327h
Name:	Plane CSC Coefficients
ShortName:	PLANE_CSC_COEFF_2_D
Reset:	soft

Address:	73410h-73427h
Name:	Plane CSC Coefficients
ShortName:	PLANE_CSC_COEFF_3_D
Reset:	soft

Programming Notes

Refer to Color Space Conversion page for programming details and examples.

DWord	Bit	Description
0	31:16	**RY** Format: CSC COEFFICIENT FORMAT
	15:0	**GY** Format: CSC COEFFICIENT FORMAT
1	31:16	**BY** Format: CSC COEFFICIENT FORMAT
	15:0	**Reserved** Access: RO Format: MBZ
2	31:16	**RU** Format: CSC COEFFICIENT FORMAT
	15:0	**GU** Format: CSC COEFFICIENT FORMAT
3	31:16	**BU**

PLANE_CSC_COEFF		
		Format: CSC COEFFICIENT FORMAT
	15:0	**Reserved** Access: RO Format: MBZ
4	31:16	**RV** Format: CSC COEFFICIENT FORMAT
	15:0	**GV** Format: CSC COEFFICIENT FORMAT
5	31:16	**BV** Format: CSC COEFFICIENT FORMAT
	15:0	**Reserved** Access: RO Format: MBZ

PLANE_CSC_POSTOFF

PLANE_CSC_POSTOFF	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	96
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank after armed
Address:	70234h-7023Fh
Name:	Plane CSC Post-offset
ShortName:	PLANE_CSC_POSTOFF_1_A
Reset:	soft
Address:	70334h-7033Fh
Name:	Plane CSC Post-offset
ShortName:	PLANE_CSC_POSTOFF_2_A
Reset:	soft
Address:	70434h-7043Fh
Name:	Plane CSC Post-offset
ShortName:	PLANE_CSC_POSTOFF_3_A
Reset:	soft
Address:	71234h-7123Fh
Name:	Plane CSC Post-offset
ShortName:	PLANE_CSC_POSTOFF_1_B
Reset:	soft
Address:	71334h-7133Fh
Name:	Plane CSC Post-offset
ShortName:	PLANE_CSC_POSTOFF_2_B
Reset:	soft
Address:	71434h-7143Fh
Name:	Plane CSC Post-offset
ShortName:	PLANE_CSC_POSTOFF_3_B
Reset:	soft
Address:	72234h-7223Fh
Name:	Plane CSC Post-offset
ShortName:	PLANE_CSC_POSTOFF_1_C
Reset:	soft
Address:	72334h-7233Fh

PLANE_CSC_POSTOFF

Name:	Plane CSC Post-offset
ShortName:	PLANE_CSC_POSTOFF_2_C
Reset:	soft

Address:	72434h-7243Fh
Name:	Plane CSC Post-offset
ShortName:	PLANE_CSC_POSTOFF_3_C
Reset:	soft

Address:	73234h-7323Fh
Name:	Plane CSC Post-offset
ShortName:	PLANE_CSC_POSTOFF_1_D
Reset:	soft

Address:	73334h-7333Fh
Name:	Plane CSC Post-offset
ShortName:	PLANE_CSC_POSTOFF_2_D
Reset:	soft

Address:	73434h-7343Fh
Name:	Plane CSC Post-offset
ShortName:	PLANE_CSC_POSTOFF_3_D
Reset:	soft

The post-offset is intended to add an offset from 0 on the Y or RGB channels and to convert UV channels from 2's complement to excess 0.5 as they exit plane color space conversion (CSC).

DWord	Bit	Description
0	31:13	**Reserved** Access: RO Format: MBZ
	12:0	**PostCSC High Offset** This value is used to give an offset to the high color channel as it exits CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).
1	31:13	**Reserved** Access: RO Format: MBZ
	12:0	**PostCSC Medium Offset** This value is used to give an offset to the medium color channel as it exits CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).
2	31:13	**Reserved** Access: RO Format: MBZ

PLANE_CSC_POSTOFF		
	12:0	**PostCSC Low Offset** This value is used to give an offset to the low color channel as it exits CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).

PLANE_CSC_PREOFF

PLANE_CSC_PREOFF	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	96
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank after armed
Address:	70228h-70233h
Name:	Plane CSC Pre-offset
ShortName:	PLANE_CSC_PREOFF_1_A
Reset:	soft
Address:	70328h-70333h
Name:	Plane CSC Pre-offset
ShortName:	PLANE_CSC_PREOFF_2_A
Reset:	soft
Address:	70428h-70433h
Name:	Plane CSC Pre-offset
ShortName:	PLANE_CSC_PREOFF_3_A
Reset:	soft
Address:	71228h-71233h
Name:	Plane CSC Pre-offset
ShortName:	PLANE_CSC_PREOFF_1_B
Reset:	soft
Address:	71328h-71333h
Name:	Plane CSC Pre-offset
ShortName:	PLANE_CSC_PREOFF_2_B
Reset:	soft
Address:	71428h-71433h
Name:	Plane CSC Pre-offset
ShortName:	PLANE_CSC_PREOFF_3_B
Reset:	soft
Address:	72228h-72233h
Name:	Plane CSC Pre-offset
ShortName:	PLANE_CSC_PREOFF_1_C
Reset:	soft
Address:	72328h-72333h

PLANE_CSC_PREOFF

Name:	Plane CSC Pre-offset
ShortName:	PLANE_CSC_PREOFF_2_C
Reset:	soft

Address:	72428h-72433h
Name:	Plane CSC Pre-offset
ShortName:	PLANE_CSC_PREOFF_3_C
Reset:	soft

Address:	73228h-73233h
Name:	Plane CSC Pre-offset
ShortName:	PLANE_CSC_PREOFF_1_D
Reset:	soft

Address:	73328h-73333h
Name:	Plane CSC Pre-offset
ShortName:	PLANE_CSC_PREOFF_2_D
Reset:	soft

Address:	73428h-73433h
Name:	Plane CSC Pre-offset
ShortName:	PLANE_CSC_PREOFF_3_D
Reset:	soft

The pre-offset is intended to remove an offset from 0 on the Y or RGB channels and to convert UV channels from excess 0.5 to 2's complement as they enter plane color space conversion (CSC).
RGB modes:Red is in the High channel, Green in Medium, and Blue in Low.
YUV modes:V is in the High channel, Y in Medium, and U in Low.

DWord	Bit	Description
0	31:13	**Reserved** Access: RO Format: MBZ
	12:0	**PreCSC High Offset** This value is used to give an offset to the high color channel as it enters CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).
1	31:13	**Reserved** Access: RO Format: MBZ
	12:0	**PreCSC Medium Offset** This value is used to give an offset to the medium color channel as it enters CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).

PLANE_CSC_PREOFF		
2	31:13	**Reserved** Access: RO Format: MBZ
	12:0	**PreCSC Low Offset** This value is used to give an offset to the low color channel as it enters CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).

PLANE_CTL

PLANE_CTL	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF or plane not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled; after armed
Address:	70480h-70483h
Name:	Plane Control
ShortName:	PLANE_CTL_4_A
Reset:	soft
Address:	70580h-70583h
Name:	Plane Control
ShortName:	PLANE_CTL_5_A
Reset:	soft
Address:	70680h-70683h
Name:	Plane Control
ShortName:	PLANE_CTL_6_A
Reset:	soft
Address:	70780h-70783h
Name:	Plane Control
ShortName:	PLANE_CTL_7_A
Reset:	soft
Address:	71480h-71483h
Name:	Plane Control
ShortName:	PLANE_CTL_4_B
Reset:	soft
Address:	71580h-71583h
Name:	Plane Control
ShortName:	PLANE_CTL_5_B
Reset:	soft
Address:	71680h-71683h
Name:	Plane Control
ShortName:	PLANE_CTL_6_B
Reset:	soft
Address:	71780h-71783h

PLANE_CTL	
Name:	Plane Control
ShortName:	PLANE_CTL_7_B
Reset:	soft
Address:	72480h-72483h
Name:	Plane Control
ShortName:	PLANE_CTL_4_C
Reset:	soft
Address:	72580h-72583h
Name:	Plane Control
ShortName:	PLANE_CTL_5_C
Reset:	soft
Address:	72680h-72683h
Name:	Plane Control
ShortName:	PLANE_CTL_6_C
Reset:	soft
Address:	72780h-72783h
Name:	Plane Control
ShortName:	PLANE_CTL_7_C
Reset:	soft
Address:	73480h-73483h
Name:	Plane Control
ShortName:	PLANE_CTL_4_D
Reset:	soft
Address:	73580h-73583h
Name:	Plane Control
ShortName:	PLANE_CTL_5_D
Reset:	soft
Address:	73680h-73683h
Name:	Plane Control
ShortName:	PLANE_CTL_6_D
Reset:	soft
Address:	73780h-73783h
Name:	Plane Control
ShortName:	PLANE_CTL_7_D
Reset:	soft
Address:	70180h-70183h

PLANE_CTL	
Name:	Plane Control
ShortName:	PLANE_CTL_1_A
Reset:	soft
Address:	70280h-70283h
Name:	Plane Control
ShortName:	PLANE_CTL_2_A
Reset:	soft
Address:	70380h-70383h
Name:	Plane Control
ShortName:	PLANE_CTL_3_A
Reset:	soft
Address:	71180h-71183h
Name:	Plane Control
ShortName:	PLANE_CTL_1_B
Reset:	soft
Address:	71280h-71283h
Name:	Plane Control
ShortName:	PLANE_CTL_2_B
Reset:	soft
Address:	71380h-71383h
Name:	Plane Control
ShortName:	PLANE_CTL_3_B
Reset:	soft
Address:	72180h-72183h
Name:	Plane Control
ShortName:	PLANE_CTL_1_C
Reset:	soft
Address:	72280h-72283h
Name:	Plane Control
ShortName:	PLANE_CTL_2_C
Reset:	soft
Address:	72380h-72383h
Name:	Plane Control
ShortName:	PLANE_CTL_3_C
Reset:	soft
Address:	73180h-73183h

PLANE_CTL

Name:	Plane Control
ShortName:	PLANE_CTL_1_D
Reset:	soft

Address:	73280h-73283h
Name:	Plane Control
ShortName:	PLANE_CTL_2_D
Reset:	soft

Address:	73380h-73383h
Name:	Plane Control
ShortName:	PLANE_CTL_3_D
Reset:	soft

The pipe scaler can be attached to a plane to scale the plane output before blending.

Restriction
Refer to 'Plane Capability and Interoperability' page for plane capabilities and restrictions.

DWord	Bit	Description
0	31	**Plane Enable** When this bit is set, the plane will generate pixels for display. When cleared to zero, plane memory fetches cease and plane output is transparent. Value: 0b — Name: Disable Value: 1b — Name: Enable
	30:28	**Pipe Slice Arbitration Slots** This field specifies the number of slots allocated to this plane in pipe slice request arbitration. This field is ignored when the 'PIPE_SLICE_ARBITRATION_CTL->Use Programmed Slots' is not set. This field is zero based; a programmed value of 0 results in 1 slot allocation.
	27:23	**Source Pixel Format** This field selects the source pixel format for the plane. Before entering the blender, each source format is converted to the pipe pixel format. The 8-bpp indexed format will always use the pipe palette. In planar YUV formats Y samples appear first in memory followed by interleaved UV samples. YUV 4:2:2 byte order is programmed separately. YUV 4:2:0 and YUV 4:4:4 byte order is not programmable. RGB color order is programmed separately for some formats. Refer to Universal Plane, Plane Pixel Formats section for color channel bit mappings. YUV 4:2:0 P010, P012 and P016 formats share the same 16 bpc memory layout but use 10, 12 and 16 bits per channel respectively. The color values are stored in the most significant bits. 64-bit formats supported only on the HDR planes. P01x output is only allowed from HDR planes.

Value	Name	Description
00000b	YUV 422 Packed 8 bpc	YUV 4:2:2 packed, 8 bpc
00010b	YUV 420 Planar 8 bpc	YUV 4:2:0 Planar, 8 bpc - NV12
00100b	RGB 2101010	RGB 2:10:10:10, 32 bit.
00110b	YUV 420 Planar 10 bpc	YUV 4:2:0 Planar, 10 bpc - P010
01000b	RGB 8888	RGB 8:8:8:8, 32 bit
01010b	YUV 420 Planar 12 bpc	YUV 4:2:0 Planar 12 bpc - P012
01100b	RGB 16161616 Float	RGB 16:16:16:16 Floating Point, 64 bit (FP16)
01110b	YUV 420 Planar 16 bpc	YUV 4:2:0 Planar, 16 bpc - P016
10000b	YUV 444 Packed 8 bpc	YUV 4:4:4 packed (MSB-X:Y:U:V), 8bpc
10100b	RGB 2101010 XR_BIAS	RGB 2:10:10:10 Extended Range Bias (MSB-X:B:G:R), 32 bit
11000b	Indexed 8 bit	Indexed 8-bit
11100b	RGB 565	RGB 5:6:5 (MSB-R:G:B), 16 bit
00001b	YUV 422 Packed 10 bpc	YUV 4:2:2 packed, 10 bpc - Y210
00011b	YUV 422 Packed 12 bpc	YUV 4:2:2 packed, 12 bpc - Y212
00101b	YUV 422 Packed 16 bpc	YUV 4:2:2 packed, 16 bpc - Y216
00111b	YUV 444 Packed 10 bpc	YUV 4:4:4 packed (MSB-X:V:Y:U), 10 bpc - Y410
01001b	YUV 444 Packed 12 bpc	YUV 4:4:4 packed (MSB-X:V:Y:U), 12 bpc - Y412
01011b	YUV 444 Packed 16 bpc	YUV 4:4:4 packed (MSB-X:V:Y:U), 16 bpc - Y416

Restriction
Plane scaling is not compatible with the Indexed 8-bit, XR_BIAS source pixel formats.

22:21 **Key Enable**

This field enables color keying. The key color, range, channel enables, and mask are programmed in PLANE_KEYVAL, PLANE_KEYMSK, and PLANE_KEYMAX.

Value	Name	Description
00b	Disable	Disable keying for this plane.
01b	Source Key Enable	This plane's pixels will be checked for a key match. The blend between this plane and the plane below will treat the key matched pixels as transparent.
10b	Destination Key Enable	This plane's pixels will be checked for a key match. The blend between this plane and the plane above will treat the pixels above as opaque only where this plane is key matched and the plane above is opaque. When plane gamma is enabled, the gamma processing may shift the pixel color values sent to blender and may cause it to not match the key color as desired. The recommendation is to use the pipe gamma when destination keying is enabled.
11b	Source Key Window	This plane's pixels will be checked for a key match. The blend between this plane and the plane below will treat the key matched pixels as

PLANE_CTL

	Enable	transparent only where the plane below is opaque.

Restriction
Plane color keying is not compatible with the Indexed 8-bit pixel format. Destination key/Source Key Window should be enabled only on one set (a pair) of planes, per pipe, at a time. Source key and Source Key Window must not be enabled on the bottom most active plane. Destination key must not be enabled on the top most active plane.

20 **RGB Color Order**

This field is used to select the color order when using RGB data formats, except RGB 32-bit XR_BIAS 10:10:10 and 16-bit BGRX 5:6:5. For other formats, this field is ignored.

Value	Name	Description
0b	BGRX	BGRX (MSB-X:R:G:B)
1b	RGBX	RGBX (MSB-X:B:G:R)

19 **Planar YUV420 component**

This field selects the planar YUV420 component for the plane when NV12/P0xx source pixel formats is used. This field must be set to '0b' for other (YUV non-planar/RGB) surface formats.

Value	Name	Description
0b	UV	Planes 1 to 5 can be configured as UV plane. Planes 6 and 7 must not be configured as a UV plane.
1b	Y	Planes 6 and 7 can be configured as Y plane. Planes 1 to 5 must not be configured as a Y plane.

18 **Reserved**

Access:	RO
Format:	MBZ

17:16 **YUV 422 Byte Order**

This field is used to select the byte order for YUV 4:2:2 8bpc data formats. For other formats, this field is ignored.

Value	Name	Description
00b	YUYV	YUYV (MSB-V:Y2:U:Y1)
01b	UYVY	UYVY (MSB-Y2:V:Y1:U)
10b	YVYU	YVYU (MSB-U:Y2:V:Y1)
11b	VYUY	VYUY (MSB-Y2:U:Y1:V)

15 **Render Decomp**

This bit enables the Display decompression of Render compressed surfaces.

Value	Name
0b	Disable
1b	Enable

PLANE_CTL

Restriction
Color Clear is supported.
Only the Left-right cache-line pair decompression is supported. The compressed surface should be Y (Legacy) or Y F Tiled. Decompression is not supported with 90/270 degree rotation.
Decompression is supported with RGB8888, RGB1010102 and FP16 formats.
Decompression is supported on all planes and pipes.

14 **Reserved**

Access:	RO
Format:	MBZ

13 **Clear Color Disable**

Description
This field disables the render decompression clear color mode. It is ignored when the Render Decomp field is disabled. The color value must be programmed in PLANE_CC_VAL before flipping to the surface that uses clear color value.

Value	Name
1b	Disable
0b	Enable **[Default]**

12:10 **Tiled Surface**

This field indicates that the surface data is in tiled memory. This bit may be updated through MMIO writes or through a command streamer initiated synchronous flip.

Value	Name
000b	Linear memory
001b	Tile X memory
100b	Tile Y (Legacy) memory

Restriction
Interlaced mode is not supported with Y Tiling. Tile Ys is not supported.

9 **Async Address Update Enable**

Access:	R/W

This bit will enable asynchronous updates of the plane surface address when written by MMIO (MMIO asynchronous flips). The surface address will change as soon as possible. This bit is not double buffered and the changes will apply immediately. When performing an asynchronous update, only the plane surface can be updated. Changes to stride, pixel, format, compression, FBC, etc. are not allowed.

Value	Name	Description
0b	Sync	Surface Address MMIO writes will update synchronous to start of vertical blank
1b	Async	Surface Address MMIO writes will update asynchronous to start of vertical blank

Restriction
No command streamer (ring) flips to this plane are allowed when this bit is enabled. Command streamer flips will set a similar bit in the flip message that it sends to display.
Asynchronous S3D flips are not allowed.
Each async surface address write must be followed by a wait for flip done indication before writing the surface address register again.

8 **Horizontal Flip**

This field controls the horizontal flipping of the plane. When horizontal flipping is enabled with rotation, the horizontal flip operation is logically performed first followed by rotation. For further information refer to "Universal Plane" section.

Value	Name
0b	Disable **[Default]**
1b	Enable

Restriction
Horizontal flip is not supported with linear surface formats.

7:6 **Stereo Surface Vblank Mask**

This field controls which vertical blank (left eye, right eye, or both) will be used for the plane surface address double-buffering during stereo 3D mode. This field is ignored when not in stereo 3D mode.

Value	Name	Description
00b	Mask None	Both the left and right eye vertical blanks will be used.
01b	Mask Left	Mask the left eye vertical blank. Only the right eye vertical blank will be used.
10b	Mask Right	Mask the right eye vertical blank. Only the left eye vertical blank will be used.

5 **Reserved**

4 **Media Decomp**

Description
This bit enables the Display decompression of Media compressed surfaces. 'Media Decomp' and 'Render Decomp' are mutually exclusive and must not enabled at the same time for a given plane.
Media decompression is supported with NV12, P0xx, YUV422, YUV444, RGB8888, RGB1010102 and FP16 formats.

PLANE_CTL

Value	Name
0b	Disable
1b	Enable

3 **Allow Double Buffer Update Disable**

Access:	R/W

This field controls whether double buffer updates are allowed to be disabled for this plane. The DOUBLE_BUFFER_CTL register can be configured to globally disable double buffer updates for resources that allow them to be disabled. This field applies only to the plane registers that supports double buffering. Scaler registers used for plane scaling purposes are not included in this.

Value	Name
0b	Not Allowed
1b	Allowed **[Default]**

2 **Reserved**

Access:	RO
Format:	MBZ

1:0 **Plane Rotation**

This field controls hardware rotation of the plane.

Value	Name
00b	No rotation
01b	90 degree rotation
10b	180 degree rotation
11b	270 degree rotation

Programming Notes
Hardware does not change the plane position when rotation is enabled. Software may need to adjust the plane position to match the physical orientation of the display.

Restriction
90/270 degree rotation requires the surface to be Y Tiled. Interlaced mode is not supported with 90/270 degree rotation. Render-Display decompression is not supported with 90/270 degree rotation.
90/270 rotation is supported with plane width (pre-rotation) up to 4096 pixels.

PLANE_CUS_CTL

PLANE_CUS_CTL	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF or plane not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled; after armed
Address:	701C8h-701CBh
Name:	Plane Chroma Upsampler Control
ShortName:	PLANE_CUS_CTL_1_A
Reset:	soft
Address:	702C8h-702CBh
Name:	Plane Chroma Upsampler Control
ShortName:	PLANE_CUS_CTL_2_A
Reset:	soft
Address:	703C8h-703CBh
Name:	Plane Chroma Upsampler Control
ShortName:	PLANE_CUS_CTL_3_A
Reset:	soft
Address:	711C8h-711CBh
Name:	Plane Chroma Upsampler Control
ShortName:	PLANE_CUS_CTL_1_B
Reset:	soft
Address:	712C8h-712CBh
Name:	Plane Chroma Upsampler Control
ShortName:	PLANE_CUS_CTL_2_B
Reset:	soft
Address:	713C8h-713CBh
Name:	Plane Chroma Upsampler Control
ShortName:	PLANE_CUS_CTL_3_B
Reset:	soft
Address:	721C8h-721CBh
Name:	Plane Chroma Upsampler Control
ShortName:	PLANE_CUS_CTL_1_C
Reset:	soft
Address:	722C8h-722CBh

PLANE_CUS_CTL

Name:	Plane Chroma Upsampler Control
ShortName:	PLANE_CUS_CTL_2_C
Reset:	soft

Address:	723C8h-723CBh
Name:	Plane Chroma Upsampler Control
ShortName:	PLANE_CUS_CTL_3_C
Reset:	soft

Address:	731C8h-731CBh
Name:	Plane Chroma Upsampler Control
ShortName:	PLANE_CUS_CTL_1_D
Reset:	soft

Address:	732C8h-732CBh
Name:	Plane Chroma Upsampler Control
ShortName:	PLANE_CUS_CTL_2_D
Reset:	soft

Address:	733C8h-733CBh
Name:	Plane Chroma Upsampler Control
ShortName:	PLANE_CUS_CTL_3_D
Reset:	soft

Description

This register programs the chroma upsampler for processing pixel streams from hybrid planar YUV 420 (NV12, P0xx) surfaces.
This dedicated chroma upsampling capability is available only in Planes 1 through 3. Planes 4 and 5 must use plane scaler (PS_CTRL) for chroma upsampling.
The following table shows phase programming for frequently used YUV420 to YUV444 chroma upsampling scenarios.

YUV 420Chroma Siting	Horz Phase	Vert Phase	Programmed Horz Initial Phase	Programmed Horz Initial Phase Sign	Programmed Vert Initial Phase	Programmed Vert Initial Phase Sign
Top Left	0	0	0	0	0	0
Top	-0.25	0	0.25	1	0	0
Left (MPEG-2)	0	-0.25	0	0	0.25	1
Center (MPEG-1)	-0.25	-0.25	0.25	1	0.25	1

Restriction :
When the Chroma upsampler is enabled, then:

1. The maximum horizontal plane size allowed is 4096 pixels
2. The minimum horizontal plane size allowed is 8 pixels

PLANE_CUS_CTL

3. The minimum vertical plane size allowed is 4 lines
4. The horizontal and vertical plane size should be even

DWord	Bit	Description
0	31	**Chroma Upsampler Enable** This field enables the plane chroma upsampler for handling hybrid planar YUV 420 (NV12, P0xx) formats. Value: 0b – Disable; 1b – Enable
	30	**Y Binding** This field defines the Y plane from where the chroma upsampler will receive the Y pixels stream when processing hybrid planar YUV 420 (NV12, P0xx) formats. Value: 0b – Plane 6; 1b – Plane 7
	29:24	**Reserved** Access: RO Format: MBZ
	23	**Reserved**
	22	**Reserved** Access: RO Format: MBZ
	21:20	**Reserved**
	19	**Horz Initial Phase Sign** This field is defines the direction of the horizontal initial phase adjustment on the UV stream during upsampling. A positive initial phase will have an effect of shifting the UV pixels to the right with respect to the Y pixels whereas a negative initial phase will have an effect of shifting left. The sign bit must be zero if the initial phase is zero. Value: 0b – Positive Initial Phase; 1b – Negative Initial Phase
	18	**Reserved** Access: RO Format: MBZ
	17:16	**Horz Initial Phase** This field defines the horizontal initial phase adjustment required on the UV stream during

upsampling. This field should be programmed based on the YUV 420 chroma siting.

Value	Name
00b	0
01b	0.25
10b	0.5
11b	Reserved

15 **Vert Initial Phase Sign**

This field is defines the direction of the vertical initial phase adjustment on the UV stream during upsampling.
A positive initial phase will have an effect of shifting the UV pixels down with respect to the Y pixels whereas a negative initial phase will have an effect of shifting up.
The sign bit must be zero if the initial phase is zero.

Value	Name
0b	Positive Initial Phase
1b	Negative Initial Phase

14 **Reserved**

Access:	RO
Format:	MBZ

13:12 **Vert Initial Phase**

This field defines the vertical initial phase adjustment required on the UV stream during upsampling. This field should be programmed based on the YUV 420 chroma siting.

Value	Name
00b	0
01b	0.25
10b	0.5
11b	Reserved

11 **Reserved**

10:9 **Power Up Delay**

This field indicates the wait (in CD clocks) between powering up the line buffer arrays.

8 **Reserved**

7:6 **Reserved**

Access:	RO
Format:	MBZ

5 **ECC Single Error**

Access:	R/WC

This field indicates that an single bit error encountered at the ECC logic. Hardware will correct the single bit errors. Hardware will set the bit; SW can clear with a write of 1.

PLANE_CUS_CTL		
	4	**ECC Double Error** Access: R/WC This field indicates that an double bit error encountered at the ECC logic. Hardware will not correct the double bit errors. Hardware will set the bit; SW can clear with a write of 1.
	3:1	**Reserved** Access: RO Format: MBZ
	0	**Power Up In Progress** Access: RO This field is set when the chroma upsampler line buffers are being powered up. Chroma upsampler cannot handle pixel traffic when this bit is set.

PLANE_INPUT_CSC_COEFF

PLANE_INPUT_CSC_COEFF	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	192
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank after armed
Address:	701E0h-701F7h
Name:	Plane Input CSC Coefficients
ShortName:	PLANE_INPUT_CSC_COEFF_1_A
Reset:	soft
Address:	702E0h-702F7h
Name:	Plane Input CSC Coefficients
ShortName:	PLANE_INPUT_CSC_COEFF_2_A
Reset:	soft
Address:	703E0h-703F7h
Name:	Plane Input CSC Coefficients
ShortName:	PLANE_INPUT_CSC_COEFF_3_A
Reset:	soft
Address:	711E0h-711F7h
Name:	Plane Input CSC Coefficients
ShortName:	PLANE_INPUT_CSC_COEFF_1_B
Reset:	soft
Address:	712E0h-712F7h
Name:	Plane Input CSC Coefficients
ShortName:	PLANE_INPUT_CSC_COEFF_2_B
Reset:	soft
Address:	713E0h-713F7h
Name:	Plane Input CSC Coefficients
ShortName:	PLANE_INPUT_CSC_COEFF_3_B
Reset:	soft
Address:	721E0h-721F7h
Name:	Plane Input CSC Coefficients
ShortName:	PLANE_INPUT_CSC_COEFF_1_C
Reset:	soft

PLANE_INPUT_CSC_COEFF

Address:	722E0h-722F7h
Name:	Plane Input CSC Coefficients
ShortName:	PLANE_INPUT_CSC_COEFF_2_C
Reset:	soft

Address:	723E0h-723F7h
Name:	Plane Input CSC Coefficients
ShortName:	PLANE_INPUT_CSC_COEFF_3_C
Reset:	soft

Address:	731E0h-731F7h
Name:	Plane Input CSC Coefficients
ShortName:	PLANE_INPUT_CSC_COEFF_1_D
Reset:	soft

Address:	732E0h-732F7h
Name:	Plane Input CSC Coefficients
ShortName:	PLANE_INPUT_CSC_COEFF_2_D
Reset:	soft

Address:	733E0h-733F7h
Name:	Plane Input CSC Coefficients
ShortName:	PLANE_INPUT_CSC_COEFF_3_D
Reset:	soft

DWord	Bit	Description
0	31:16	**RY** Format: CSC COEFFICIENT FORMAT
	15:0	**GY** Format: CSC COEFFICIENT FORMAT
1	31:16	**BY** Format: CSC COEFFICIENT FORMAT
	15:0	**Reserved** Access: RO Format: MBZ
2	31:16	**RU** Format: CSC COEFFICIENT FORMAT
	15:0	**GU** Format: CSC COEFFICIENT FORMAT
3	31:16	**BU** Format: CSC COEFFICIENT FORMAT

<table>
<tr><th colspan="3">PLANE_INPUT_CSC_COEFF</th></tr>
<tr><td></td><td>15:0</td><td>Reserved<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td rowspan="2">4</td><td>31:16</td><td>RV<table><tr><td>Format:</td><td>CSC COEFFICIENT FORMAT</td></tr></table></td></tr>
<tr><td>15:0</td><td>GV<table><tr><td>Format:</td><td>CSC COEFFICIENT FORMAT</td></tr></table></td></tr>
<tr><td rowspan="2">5</td><td>31:16</td><td>BV<table><tr><td>Format:</td><td>CSC COEFFICIENT FORMAT</td></tr></table></td></tr>
<tr><td>15:0</td><td>Reserved<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

PLANE_INPUT_CSC_POSTOFF

PLANE_INPUT_CSC_POSTOFF	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	96
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank after armed
Address:	70204h-7020Fh
Name:	Plane Input CSC Post-offset
ShortName:	PLANE_INPUT_CSC_POSTOFF_1_A
Reset:	soft
Address:	70304h-7030Fh
Name:	Plane Input CSC Post-offset
ShortName:	PLANE_INPUT_CSC_POSTOFF_2_A
Reset:	soft
Address:	70404h-7040Fh
Name:	Plane Input CSC Post-offset
ShortName:	PLANE_INPUT_CSC_POSTOFF_3_A
Reset:	soft
Address:	71204h-7120Fh
Name:	Plane Input CSC Post-offset
ShortName:	PLANE_INPUT_CSC_POSTOFF_1_B
Reset:	soft
Address:	71304h-7130Fh
Name:	Plane Input CSC Post-offset
ShortName:	PLANE_INPUT_CSC_POSTOFF_2_B
Reset:	soft
Address:	71404h-7140Fh
Name:	Plane Input CSC Post-offset
ShortName:	PLANE_INPUT_CSC_POSTOFF_3_B
Reset:	soft
Address:	72204h-7220Fh
Name:	Plane Input CSC Post-offset
ShortName:	PLANE_INPUT_CSC_POSTOFF_1_C
Reset:	soft
Address:	72304h-7230Fh

PLANE_INPUT_CSC_POSTOFF

Name:	Plane Input CSC Post-offset
ShortName:	PLANE_INPUT_CSC_POSTOFF_2_C
Reset:	soft

Address:	72404h-7240Fh
Name:	Plane Input CSC Post-offset
ShortName:	PLANE_INPUT_CSC_POSTOFF_3_C
Reset:	soft

Address:	73204h-7320Fh
Name:	Plane Input CSC Post-offset
ShortName:	PLANE_INPUT_CSC_POSTOFF_1_D
Reset:	soft

Address:	73304h-7330Fh
Name:	Plane Input CSC Post-offset
ShortName:	PLANE_INPUT_CSC_POSTOFF_2_D
Reset:	soft

Address:	73404h-7340Fh
Name:	Plane Input CSC Post-offset
ShortName:	PLANE_INPUT_CSC_POSTOFF_3_D
Reset:	soft

The post-offset is intended to add an offset from 0 on the Y or RGB channels and to convert UV channels from 2's complement to excess 0.5 as they exit plane input color space conversion (CSC).

DWord	Bit	Description
0	31:13	**Reserved** Access: RO Format: MBZ
	12:0	**PostCSC High Offset** This value is used to give an offset to the high color channel as it exits CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).
1	31:13	**Reserved** Access: RO Format: MBZ
	12:0	**PostCSC Medium Offset** This value is used to give an offset to the medium color channel as it exits CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).
2	31:13	**Reserved** Access: RO Format: MBZ

PLANE_INPUT_CSC_POSTOFF		
	12:0	**PostCSC Low Offset** This value is used to give an offset to the low color channel as it exits CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).

PLANE_INPUT_CSC_PREOFF

PLANE_INPUT_CSC_PREOFF	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	96
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank after armed
Address:	701F8h-70203h
Name:	Plane Input CSC Pre-offset
ShortName:	PLANE_INPUT_CSC_PREOFF_1_A
Reset:	soft
Address:	702F8h-70303h
Name:	Plane Input CSC Pre-offset
ShortName:	PLANE_INPUT_CSC_PREOFF_2_A
Reset:	soft
Address:	703F8h-70403h
Name:	Plane Input CSC Pre-offset
ShortName:	PLANE_INPUT_CSC_PREOFF_3_A
Reset:	soft
Address:	711F8h-71203h
Name:	Plane Input CSC Pre-offset
ShortName:	PLANE_INPUT_CSC_PREOFF_1_B
Reset:	soft
Address:	712F8h-71303h
Name:	Plane Input CSC Pre-offset
ShortName:	PLANE_INPUT_CSC_PREOFF_2_B
Reset:	soft
Address:	713F8h-71403h
Name:	Plane Input CSC Pre-offset
ShortName:	PLANE_INPUT_CSC_PREOFF_3_B
Reset:	soft
Address:	721F8h-72203h
Name:	Plane Input CSC Pre-offset
ShortName:	PLANE_INPUT_CSC_PREOFF_1_C
Reset:	soft
Address:	722F8h-72303h

PLANE_INPUT_CSC_PREOFF

Name:	Plane Input CSC Pre-offset
ShortName:	PLANE_INPUT_CSC_PREOFF_2_C
Reset:	soft

Address:	723F8h-72403h
Name:	Plane Input CSC Pre-offset
ShortName:	PLANE_INPUT_CSC_PREOFF_3_C
Reset:	soft

Address:	731F8h-73203h
Name:	Plane Input CSC Pre-offset
ShortName:	PLANE_INPUT_CSC_PREOFF_1_D
Reset:	soft

Address:	732F8h-73303h
Name:	Plane Input CSC Pre-offset
ShortName:	PLANE_INPUT_CSC_PREOFF_2_D
Reset:	soft

Address:	733F8h-73403h
Name:	Plane Input CSC Pre-offset
ShortName:	PLANE_INPUT_CSC_PREOFF_3_D
Reset:	soft

The pre-offset is intended to remove an offset from 0 on the Y or RGB channels and to convert UV channels from excess 0.5 to 2's complement as they enter plane input color space conversion (CSC).

DWord	Bit	Description
0	31:13	**Reserved** Access: RO Format: MBZ
	12:0	**PreCSC High Offset** This value is used to give an offset to the high color channel as it enters CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).
1	31:13	**Reserved** Access: RO Format: MBZ
	12:0	**PreCSC Medium Offset** This value is used to give an offset to the medium color channel as it enters CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).
2	31:13	**Reserved** Access: RO Format: MBZ

PLANE_INPUT_CSC_PREOFF		
	12:0	**PreCSC Low Offset** This value is used to give an offset to the low color channel as it enters CSC logic. The value is a 2's complement fraction allowing offsets between -1 and +1 (exclusive).

PLANE_KEYMAX

PLANE_KEYMAX	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF or plane not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank, pipe not enabled, or plane not enabled
Address:	704A0h-704A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_4_A
Reset:	soft
Address:	705A0h-705A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_5_A
Reset:	soft
Address:	706A0h-706A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_6_A
Reset:	soft
Address:	707A0h-707A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_7_A
Reset:	soft
Address:	714A0h-714A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_4_B
Reset:	soft
Address:	715A0h-715A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_5_B
Reset:	soft
Address:	716A0h-716A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_6_B
Reset:	soft
Address:	717A0h-717A3h

PLANE_KEYMAX	
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_7_B
Reset:	soft
Address:	724A0h-724A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_4_C
Reset:	soft
Address:	725A0h-725A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_5_C
Reset:	soft
Address:	726A0h-726A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_6_C
Reset:	soft
Address:	727A0h-727A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_7_C
Reset:	soft
Address:	734A0h-734A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_4_D
Reset:	soft
Address:	735A0h-735A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_5_D
Reset:	soft
Address:	736A0h-736A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_6_D
Reset:	soft
Address:	737A0h-737A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_7_D
Reset:	soft
Address:	701A0h-701A3h

PLANE_KEYMAX	
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_1_A
Reset:	soft
Address:	702A0h-702A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_2_A
Reset:	soft
Address:	703A0h-703A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_3_A
Reset:	soft
Address:	711A0h-711A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_1_B
Reset:	soft
Address:	712A0h-712A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_2_B
Reset:	soft
Address:	713A0h-713A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_3_B
Reset:	soft
Address:	721A0h-721A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_1_C
Reset:	soft
Address:	722A0h-722A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_2_C
Reset:	soft
Address:	723A0h-723A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_3_C
Reset:	soft
Address:	731A0h-731A3h

PLANE_KEYMAX

Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_1_D
Reset:	soft

Address:	732A0h-732A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_2_D
Reset:	soft

Address:	733A0h-733A3h
Name:	Plane Key Color Max
ShortName:	PLANE_KEYMAX_3_D
Reset:	soft

When plane source is YUV, this register specifies the maximum YUV key value to be used together with the minimum YUV key value and the channel enables to determine if the plane matches the key. When plane source is RGB, this register is not used.

DWord	Bit	Description
0	31:24	**Plane Alpha Value** Specifies the plane alpha value when plane alpha is enabled in PLANE_KEYMSK register.
	23:16	**V Key Max Value** Specifies the maximum key value for the V channel.
	15:8	**Y Key Max Value** Specifies the maximum key value for the Y channel.
	7:0	**U Key Max Value** Specifies the maximum key value for the U channel.

PLANE_KEYMSK

PLANE_KEYMSK	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF or plane not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank, pipe not enabled, or plane not enabled
Address:	70498h-7049Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_4_A
Reset:	soft
Address:	70598h-7059Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_5_A
Reset:	soft
Address:	70698h-7069Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_6_A
Reset:	soft
Address:	70798h-7079Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_7_A
Reset:	soft
Address:	71498h-7149Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_4_B
Reset:	soft
Address:	71598h-7159Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_5_B
Reset:	soft
Address:	71698h-7169Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_6_B
Reset:	soft
Address:	71798h-7179Bh

PLANE_KEYMSK	
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_7_B
Reset:	soft
Address:	72498h-7249Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_4_C
Reset:	soft
Address:	72598h-7259Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_5_C
Reset:	soft
Address:	72698h-7269Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_6_C
Reset:	soft
Address:	72798h-7279Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_7_C
Reset:	soft
Address:	73498h-7349Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_4_D
Reset:	soft
Address:	73598h-7359Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_5_D
Reset:	soft
Address:	73698h-7369Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_6_D
Reset:	soft
Address:	73798h-7379Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_7_D
Reset:	soft
Address:	70198h-7019Bh

PLANE_KEYMSK	
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_1_A
Reset:	soft
Address:	70298h-7029Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_2_A
Reset:	soft
Address:	70398h-7039Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_3_A
Reset:	soft
Address:	71198h-7119Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_1_B
Reset:	soft
Address:	71298h-7129Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_2_B
Reset:	soft
Address:	71398h-7139Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_3_B
Reset:	soft
Address:	72198h-7219Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_1_C
Reset:	soft
Address:	72298h-7229Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_2_C
Reset:	soft
Address:	72398h-7239Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_3_C
Reset:	soft
Address:	73198h-7319Bh

PLANE_KEYMSK

Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_1_D
Reset:	soft

Address:	73298h-7329Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_2_D
Reset:	soft

Address:	73398h-7339Bh
Name:	Plane Key Mask
ShortName:	PLANE_KEYMSK_3_D
Reset:	soft

DWord	Bit	Description
0	31	**Plane Alpha Enable** Enables the plane alpha. Color channels will be pre-multiplied by hardware with the plane alpha value from PLANE_KEYMAX register. Per-pixel alpha is defined in the PLANE_COLOR_CTL register. Value: 0b — Name: Disable Value: 1b — Name: Enable
	30:27	**Reserved** Access: RO Format: MBZ
	26	**V or R Key Channel Enable** Enables the V/Red channel for key comparison. A disabled channel will be ignored when determining a key match. Value: 0b — Name: Disable Value: 1b — Name: Enable
	25	**Y or G Key Channel Enable** Enables the Y/Green channel for key comparison. A disabled channel will be ignored when determining a key match. Value: 0b — Name: Disable Value: 1b — Name: Enable
	24	**U or B Key Channel Enable** Enables the U/Blue channel for key comparison. A disabled channel will be ignored when determining a key match. Value: 0b — Name: Disable

PLANE_KEYMSK			
		1b	Enable
	23:16	**R Key Mask Value** Specifies the key mask for the Red channel. A zero bit in the mask indicates that the corresponding bit will be ignored when determining a key match.	
	15:8	**G Key Mask Value** Specifies the key mask for the Green channel. A zero bit in the mask indicates that the corresponding bit will be ignored when determining a key match.	
	7:0	**B Key Mask Value** Specifies the key mask for the Blue channel. A zero bit in the mask indicates that the corresponding bit will be ignored when determining a key match.	

PLANE_KEYVAL

PLANE_KEYVAL	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF or plane not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank, pipe not enabled, or plane not enabled
Address:	70494h-70497h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_4_A
Reset:	soft
Address:	70594h-70597h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_5_A
Reset:	soft
Address:	70694h-70697h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_6_A
Reset:	soft
Address:	70794h-70797h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_7_A
Reset:	soft
Address:	71494h-71497h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_4_B
Reset:	soft
Address:	71594h-71597h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_5_B
Reset:	soft
Address:	71694h-71697h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_6_B
Reset:	soft
Address:	71794h-71797h

PLANE_KEYVAL	
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_7_B
Reset:	soft
Address:	72494h-72497h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_4_C
Reset:	soft
Address:	72594h-72597h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_5_C
Reset:	soft
Address:	72694h-72697h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_6_C
Reset:	soft
Address:	72794h-72797h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_7_C
Reset:	soft
Address:	73494h-73497h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_4_D
Reset:	soft
Address:	73594h-73597h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_5_D
Reset:	soft
Address:	73694h-73697h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_6_D
Reset:	soft
Address:	73794h-73797h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_7_D
Reset:	soft
Address:	70194h-70197h

PLANE_KEYVAL	
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_1_A
Reset:	soft
Address:	70294h-70297h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_2_A
Reset:	soft
Address:	70394h-70397h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_3_A
Reset:	soft
Address:	71194h-71197h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_1_B
Reset:	soft
Address:	71294h-71297h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_2_B
Reset:	soft
Address:	71394h-71397h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_3_B
Reset:	soft
Address:	72194h-72197h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_1_C
Reset:	soft
Address:	72294h-72297h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_2_C
Reset:	soft
Address:	72394h-72397h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_3_C
Reset:	soft
Address:	73194h-73197h

PLANE_KEYVAL

Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_1_D
Reset:	soft

Address:	73294h-73297h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_2_D
Reset:	soft

Address:	73394h-73397h
Name:	Plane Key Color
ShortName:	PLANE_KEYVAL_3_D
Reset:	soft

When plane source is YUV, this register specifies the minimum YUV key value to be used together with the maximum YUV key value and the channel enables to determine if the plane matches the key. When plane source is RGB, this register specifies the RGB key value to be used together with the channel masks to determine if the plane matches the key. RGB key matches can only occur for positive pixel values in the 0 to 1 range. Extended range pixel values will not match. MSB bits are used for comparison.

Restriction : Keying is not supported in HDR mode.

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**V Min or R Key Value** Specifies the minimum key value for the V channel or the compare value for Red channel.
	15:8	**Y Min or G Key Value** Specifies the minimum key value for the Y channel or the compare value for Green channel.
	7:0	**U Min or B Key Value** Specifies the minimum key value for the U channel or the compare value for Blue channel.

PLANE_LEFT_SURF

PLANE_LEFT_SURF	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF or plane not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of left or right eye vertical blank (selectable), pipe not enabled. or plane not enabled; after armed
Address:	704B0h-704B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_4_A
Reset:	soft
Address:	705B0h-705B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_5_A
Reset:	soft
Address:	706B0h-706B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_6_A
Reset:	soft
Address:	707B0h-707B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_7_A
Reset:	soft
Address:	714B0h-714B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_4_B
Reset:	soft
Address:	715B0h-715B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_5_B
Reset:	soft
Address:	716B0h-716B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_6_B
Reset:	soft

PLANE_LEFT_SURF	
Address:	717B0h-717B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_7_B
Reset:	soft
Address:	724B0h-724B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_4_C
Reset:	soft
Address:	725B0h-725B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_5_C
Reset:	soft
Address:	726B0h-726B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_6_C
Reset:	soft
Address:	727B0h-727B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_7_C
Reset:	soft
Address:	734B0h-734B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_4_D
Reset:	soft
Address:	735B0h-735B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_5_D
Reset:	soft
Address:	736B0h-736B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_6_D
Reset:	soft
Address:	737B0h-737B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_7_D
Reset:	soft

PLANE_LEFT_SURF	
Address:	701B0h-701B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_1_A
Reset:	soft
Address:	702B0h-702B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_2_A
Reset:	soft
Address:	703B0h-703B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_3_A
Reset:	soft
Address:	711B0h-711B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_1_B
Reset:	soft
Address:	712B0h-712B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_2_B
Reset:	soft
Address:	713B0h-713B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_3_B
Reset:	soft
Address:	721B0h-721B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_1_C
Reset:	soft
Address:	722B0h-722B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_2_C
Reset:	soft
Address:	723B0h-723B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_3_C
Reset:	soft

PLANE_LEFT_SURF

Address:	731B0h-731B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_1_D
Reset:	soft

Address:	732B0h-732B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_2_D
Reset:	soft

Address:	733B0h-733B3h
Name:	Plane Left Surface Base Address
ShortName:	PLANE_LEFT_SURF_3_D
Reset:	soft

Restriction
This register must be programmed with a valid address prior to enabling stereo 3D on this pipe.

DWord	Bit	Description
0	31:12	**Left Surface Base Address** Format: GraphicsAddress[31:12] This address specifies the stereo 3D left eye surface base address bits 31:12. **Restriction** This surface must have the same stride, tiling, and panning offset parameters as the right eye surface and meet all the same restrictions.
	11:0	**Reserved** Access: RO Format: MBZ

PLANE_OFFSET

PLANE_OFFSET	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF or plane not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank, pipe not enabled, or plane not enabled
Address:	704A4h-704A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_4_A
Reset:	soft
Address:	705A4h-705A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_5_A
Reset:	soft
Address:	706A4h-706A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_6_A
Reset:	soft
Address:	707A4h-707A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_7_A
Reset:	soft
Address:	714A4h-714A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_4_B
Reset:	soft
Address:	715A4h-715A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_5_B
Reset:	soft
Address:	716A4h-716A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_6_B
Reset:	soft
Address:	717A4h-717A7h

PLANE_OFFSET	
Name:	Plane Offset
ShortName:	PLANE_OFFSET_7_B
Reset:	soft
Address:	724A4h-724A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_4_C
Reset:	soft
Address:	725A4h-725A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_5_C
Reset:	soft
Address:	726A4h-726A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_6_C
Reset:	soft
Address:	727A4h-727A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_7_C
Reset:	soft
Address:	734A4h-734A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_4_D
Reset:	soft
Address:	735A4h-735A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_5_D
Reset:	soft
Address:	736A4h-736A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_6_D
Reset:	soft
Address:	737A4h-737A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_7_D
Reset:	soft
Address:	701A4h-701A7h

PLANE_OFFSET	
Name:	Plane Offset
ShortName:	PLANE_OFFSET_1_A
Reset:	soft
Address:	702A4h-702A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_2_A
Reset:	soft
Address:	703A4h-703A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_3_A
Reset:	soft
Address:	711A4h-711A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_1_B
Reset:	soft
Address:	712A4h-712A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_2_B
Reset:	soft
Address:	713A4h-713A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_3_B
Reset:	soft
Address:	721A4h-721A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_1_C
Reset:	soft
Address:	722A4h-722A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_2_C
Reset:	soft
Address:	723A4h-723A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_3_C
Reset:	soft
Address:	731A4h-731A7h

PLANE_OFFSET	
Name:	Plane Offset
ShortName:	PLANE_OFFSET_1_D
Reset:	soft
Address:	732A4h-732A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_2_D
Reset:	soft
Address:	733A4h-733A7h
Name:	Plane Offset
ShortName:	PLANE_OFFSET_3_D
Reset:	soft
Address:	7089Ch-7089Fh
Name:	Selective Fetch Plane Offset
ShortName:	SEL_FETCH_PLANE_OFFSET_1_A
Reset:	soft
Address:	708BCh-708BFh
Name:	Selective Fetch Plane Offset
ShortName:	SEL_FETCH_PLANE_OFFSET_2_A
Reset:	soft
Address:	708DCh-708DFh
Name:	Selective Fetch Plane Offset
ShortName:	SEL_FETCH_PLANE_OFFSET_3_A
Reset:	soft
Address:	708FCh-708FFh
Name:	Selective Fetch Plane Offset
ShortName:	SEL_FETCH_PLANE_OFFSET_4_A
Reset:	soft
Address:	7092Ch-7092Fh
Name:	Selective Fetch Plane Offset
ShortName:	SEL_FETCH_PLANE_OFFSET_5_A
Reset:	soft
Address:	7094Ch-7094Fh
Name:	Selective Fetch Plane Offset
ShortName:	SEL_FETCH_PLANE_OFFSET_6_A
Reset:	soft
Address:	7096Ch-7096Fh

PLANE_OFFSET

Name: Selective Fetch Plane Offset

ShortName: SEL_FETCH_PLANE_OFFSET_7_A

Reset: soft

This register specifies the panning for the plane surface. The start position is specified in this register as a (x, y) offset from the beginning of the surface. When performing 180 rotation, hardware will internally add the plane size to the offsets so the plane will start displaying from the bottom right corner of the image.

Restriction

Plane Size + Plane Offset should not exceed the surface <stride/width> (in pixels)
X and Y offset restrictions are specified in the following table. For formats not specified in the table, both odd and even offsets are supported.

PixelFormat	Rotate	Start X Position	Start Y Position
YUV 420 Planar - NV12	All	Even	Even
YUV 420 Planar - P01x	All	Even	Even
YUV 422	All	Even	Even

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**Start Y Position** The Start Y Position or the Y Offset is the vertical offset in lines of the beginning of the active display plane relative to the display surface.
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**Start X Position** The Start X Position or the X Offset is the horizontal offset in pixels of the beginning of the active display plane relative to the display surface.

PLANE_PIXEL_NORMALIZE

PLANE_PIXEL_NORMALIZE	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank after armed
Address:	701A8h-701ABh
Name:	Plane Pixel Normalize
ShortName:	PLANE_PIXEL_NORMALIZE_1_A
Reset:	soft
Address:	702A8h-702ABh
Name:	Plane Pixel Normalize
ShortName:	PLANE_PIXEL_NORMALIZE_2_A
Reset:	soft
Address:	703A8h-703ABh
Name:	Plane Pixel Normalize
ShortName:	PLANE_PIXEL_NORMALIZE_3_A
Reset:	soft
Address:	711A8h-711ABh
Name:	Plane Pixel Normalize
ShortName:	PLANE_PIXEL_NORMALIZE_1_B
Reset:	soft
Address:	712A8h-712ABh
Name:	Plane Pixel Normalize
ShortName:	PLANE_PIXEL_NORMALIZE_2_B
Reset:	soft
Address:	713A8h-713ABh
Name:	Plane Pixel Normalize
ShortName:	PLANE_PIXEL_NORMALIZE_3_B
Reset:	soft
Address:	721A8h-721ABh
Name:	Plane Pixel Normalize
ShortName:	PLANE_PIXEL_NORMALIZE_1_C
Reset:	soft
Address:	722A8h-722ABh

PLANE_PIXEL_NORMALIZE

Name:	Plane Pixel Normalize
ShortName:	PLANE_PIXEL_NORMALIZE_2_C
Reset:	soft

Address:	723A8h-723ABh
Name:	Plane Pixel Normalize
ShortName:	PLANE_PIXEL_NORMALIZE_3_C
Reset:	soft

Address:	731A8h-731ABh
Name:	Plane Pixel Normalize
ShortName:	PLANE_PIXEL_NORMALIZE_1_D
Reset:	soft

Address:	732A8h-732ABh
Name:	Plane Pixel Normalize
ShortName:	PLANE_PIXEL_NORMALIZE_2_D
Reset:	soft

Address:	733A8h-733ABh
Name:	Plane Pixel Normalize
ShortName:	PLANE_PIXEL_NORMALIZE_3_D
Reset:	soft

DWord	Bit	Description
0	31	**Enable** This field enables the normalization of FP16 pixels with the specified normalization factor.
	30:16	**Reserved** Access: RO Format: MBZ
	15:0	**Normalization Factor** **Description** This field specifies the normalization factor in the FP16 format. This programmed value is multiplied with the input pixel value and normalized to range -4.0 to 4.0, exclusive. Out of bound values get clamped to be within the range from -4.0 to 4.0, exclusive. The programmed half float value must be a positive and not de-normalized, zero or NAN.

PLANE_POS

PLANE_POS	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF or plane not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled; after armed
Address:	7048Ch-7048Fh
Name:	Plane Position
ShortName:	PLANE_POS_4_A
Reset:	soft
Address:	7058Ch-7058Fh
Name:	Plane Position
ShortName:	PLANE_POS_5_A
Reset:	soft
Address:	7068Ch-7068Fh
Name:	Plane Position
ShortName:	PLANE_POS_6_A
Reset:	soft
Address:	7078Ch-7078Fh
Name:	Plane Position
ShortName:	PLANE_POS_7_A
Reset:	soft
Address:	7148Ch-7148Fh
Name:	Plane Position
ShortName:	PLANE_POS_4_B
Reset:	soft
Address:	7158Ch-7158Fh
Name:	Plane Position
ShortName:	PLANE_POS_5_B
Reset:	soft
Address:	7168Ch-7168Fh
Name:	Plane Position
ShortName:	PLANE_POS_6_B
Reset:	soft
Address:	7178Ch-7178Fh

PLANE_POS	
Name:	Plane Position
ShortName:	PLANE_POS_7_B
Reset:	soft
Address:	7248Ch-7248Fh
Name:	Plane Position
ShortName:	PLANE_POS_4_C
Reset:	soft
Address:	7258Ch-7258Fh
Name:	Plane Position
ShortName:	PLANE_POS_5_C
Reset:	soft
Address:	7268Ch-7268Fh
Name:	Plane Position
ShortName:	PLANE_POS_6_C
Reset:	soft
Address:	7278Ch-7278Fh
Name:	Plane Position
ShortName:	PLANE_POS_7_C
Reset:	soft
Address:	7348Ch-7348Fh
Name:	Plane Position
ShortName:	PLANE_POS_4_D
Reset:	soft
Address:	7358Ch-7358Fh
Name:	Plane Position
ShortName:	PLANE_POS_5_D
Reset:	soft
Address:	7368Ch-7368Fh
Name:	Plane Position
ShortName:	PLANE_POS_6_D
Reset:	soft
Address:	7378Ch-7378Fh
Name:	Plane Position
ShortName:	PLANE_POS_7_D
Reset:	soft
Address:	7018Ch-7018Fh

PLANE_POS	
Name:	Plane Position
ShortName:	PLANE_POS_1_A
Reset:	soft
Address:	7028Ch-7028Fh
Name:	Plane Position
ShortName:	PLANE_POS_2_A
Reset:	soft
Address:	7038Ch-7038Fh
Name:	Plane Position
ShortName:	PLANE_POS_3_A
Reset:	soft
Address:	7118Ch-7118Fh
Name:	Plane Position
ShortName:	PLANE_POS_1_B
Reset:	soft
Address:	7128Ch-7128Fh
Name:	Plane Position
ShortName:	PLANE_POS_2_B
Reset:	soft
Address:	7138Ch-7138Fh
Name:	Plane Position
ShortName:	PLANE_POS_3_B
Reset:	soft
Address:	7218Ch-7218Fh
Name:	Plane Position
ShortName:	PLANE_POS_1_C
Reset:	soft
Address:	7228Ch-7228Fh
Name:	Plane Position
ShortName:	PLANE_POS_2_C
Reset:	soft
Address:	7238Ch-7238Fh
Name:	Plane Position
ShortName:	PLANE_POS_3_C
Reset:	soft
Address:	7318Ch-7318Fh

PLANE_POS	
Name:	Plane Position
ShortName:	PLANE_POS_1_D
Reset:	soft
Address:	7328Ch-7328Fh
Name:	Plane Position
ShortName:	PLANE_POS_2_D
Reset:	soft
Address:	7338Ch-7338Fh
Name:	Plane Position
ShortName:	PLANE_POS_3_D
Reset:	soft
Address:	70894h-70897h
Name:	Selective Fetch Plane Position
ShortName:	SEL_FETCH_PLANE_POS_1_A
Reset:	soft
Address:	708B4h-708B7h
Name:	Selective Fetch Plane Position
ShortName:	SEL_FETCH_PLANE_POS_2_A
Reset:	soft
Address:	708D4h-708D7h
Name:	Selective Fetch Plane Position
ShortName:	SEL_FETCH_PLANE_POS_3_A
Reset:	soft
Address:	708F4h-708F7h
Name:	Selective Fetch Plane Position
ShortName:	SEL_FETCH_PLANE_POS_4_A
Reset:	soft
Address:	70924h-70927h
Name:	Selective Fetch Plane Position
ShortName:	SEL_FETCH_PLANE_POS_5_A
Reset:	soft
Address:	70944h-70947h
Name:	Selective Fetch Plane Position
ShortName:	SEL_FETCH_PLANE_POS_6_A
Reset:	soft
Address:	70964h-70967h

PLANE_POS

Name:	Selective Fetch Plane Position
ShortName:	SEL_FETCH_PLANE_POS_7_A
Reset:	soft

This register specifies the screen position of the plane. The origin of the plane position is always the upper left corner of the display pipe source image area. When plane scaling is not enabled on this plane, this is the position of the plane when blended with other planes on this pipe. When plane scaling is enabled on this plane, the scaler window position is the position of the plane when blended with other planes on this pipe. When performing rotation, the plane image is rotated by hardware, but the position is not, so it must be adjusted if it is desired to maintain the same apparent position on a physically rotated display.

Restriction

When plane scaling is not enabled on this plane, the plane must be completely contained within the pipe source area. Pipe source size >= plane position + plane size. When plane scaling is enabled on this plane, the X and Y positions must be programmed to 0.

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**Y Position** This specifies the vertical position of the plane upper left corner in lines.
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**X Position** This specifies the horizontal position of the plane upper left corner in pixels.

PLANE_POST_CSC_GAMC_DATA

PLANE_POST_CSC_GAMC_DATA	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF or plane not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled; after armed
Address:	704DCh-704DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_4_A
Reset:	soft
Address:	705DCh-705DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_5_A
Reset:	soft
Address:	706DCh-706DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_6_A
Reset:	soft
Address:	707DCh-707DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_7_A
Reset:	soft
Address:	714DCh-714DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_4_B
Reset:	soft
Address:	715DCh-715DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_5_B
Reset:	soft
Address:	716DCh-716DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_6_B
Reset:	soft
Address:	717DCh-717DFh

PLANE_POST_CSC_GAMC_DATA	
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_7_B
Reset:	soft
Address:	724DCh-724DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_4_C
Reset:	soft
Address:	725DCh-725DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_5_C
Reset:	soft
Address:	726DCh-726DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_6_C
Reset:	soft
Address:	727DCh-727DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_7_C
Reset:	soft
Address:	734DCh-734DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_4_D
Reset:	soft
Address:	735DCh-735DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_5_D
Reset:	soft
Address:	736DCh-736DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_6_D
Reset:	soft
Address:	737DCh-737DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_7_D
Reset:	soft
PLANE_POST_CSC_GAMC_INDEX and PLANE_POST_CSC_GAMC_DATA registers are used to program the values	

PLANE_POST_CSC_GAMC_DATA

that determine the characteristics of the gamma correction for the plane pixel data after plane Color Space Conversion.
The gamma correction curve is represented by specifying a set of gamma entry reference points spaced equally along the curve for values between -1 and 1. For extended values there are extended gamma entry reference points. All input values are clamped to the greater than -7.0 and less than 7.0 range before the gamma calculation.
For input values greater than or equal to 0 and less than 1.0, the input value is used to linearly interpolate between two adjacent points of the first 33 gamma entries to create the result value. The first 32 entries are stored as 16 bits per color in an unsigned 0.16 format with 0 integer and16 fractional. The 33rd, 34th and 35th entries are stored as19 bits per color in an unsigned 3.16 format with 3 integer and16 fractional bits.
For input values greater than or equal to 1.0 and less than 3.0, the input value is used to linearly interpolate between the 33rd and 34th gamma entries to create the result value.
For input values greater than or equal to 3.0 and less than 7.0, the input value is used to linearly interpolate between the 34th and 35th gamma entries to create the result value.
For negative input values, gamma is mirrored along the X-axis, giving the same result as positive input values, except for a negative sign. When gamma input may be negative, the first gamma point should be programmed to a value of 0.0 in order to have a symmetric mirroring.
Plane Post-CSC Gamma correction gets enabled or disabled based on the 'Plane Gamma Disable' bit in the 'PLANE_COLOR_CTL' register. The same set of values is used for gamma correction of the red, blue and green channels.

Programming Notes

To program the gamma correction entries, calculate the desired gamma curve for inputs from 0 to 7.0. For inputs of 0 to 1.0, multiply the input value by 32 to find the gamma entry number, then store the desired gamma result in that entry. For inputs greater than 1.0 and less than or equal to 3.0, store the result for an input of 3.0 in the 34th gamma entry. For inputs greater than 3.0 and less than or equal to 7.0, store the result for an input of 7.0 in the 35th gamma entry.

Restriction

The gamma curve must be flat or increasing, never decreasing in Direct mode. The gamma correction registers should only be updated when the plane is off, otherwise screen artifacts may show temporarily.

DWord	Bit	Description
0	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**Gamma Value** Default Value: 00000000000000000b Format: U3.16

PLANE_POST_CSC_GAMC_DATA_ENH

PLANE_POST_CSC_GAMC_DATA_ENH	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	701DCh-701DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_ENH_1_A
Reset:	soft
Address:	702DCh-702DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_ENH_2_A
Reset:	soft
Address:	703DCh-703DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_ENH_3_A
Reset:	soft
Address:	711DCh-711DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_ENH_1_B
Reset:	soft
Address:	712DCh-712DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_ENH_2_B
Reset:	soft
Address:	713DCh-713DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_ENH_3_B
Reset:	soft
Address:	721DCh-721DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_ENH_1_C
Reset:	soft
Address:	722DCh-722DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_ENH_2_C

PLANE_POST_CSC_GAMC_DATA_ENH	
Reset:	soft
Address:	723DCh-723DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_ENH_3_C
Reset:	soft
Address:	731DCh-731DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_ENH_1_D
Reset:	soft
Address:	732DCh-732DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_ENH_2_D
Reset:	soft
Address:	733DCh-733DFh
Name:	Plane Post CSC Gamma Data
ShortName:	PLANE_POST_CSC_GAMC_DATA_ENH_3_D
Reset:	soft

PLANE_POST_CSC_GAMC_INDEX and PLANE_POST_CSC_GAMC_DATA registers are used to program the values that determine the characteristics of the gamma correction for the plane pixel data before Color Space Conversion. Additional gamma correction can be done after the Color Space Conversion, if needed.
The gamma correction curve is represented by specifying a set of gamma entry reference points spaced equally along the curve for values between -1 and 1. For extended values there are extended gamma entry reference points. All input values are clamped to the greater than -7.0 and less than 7.0 range before the gamma calculation.
For input values greater than or equal to 0 and less than 1.0, the input value is used to linearly interpolate between two adjacent points of the first 33 gamma entries to create the result value. The first 32 entries are stored as 24 bits per color in an unsigned 0.24 format with 0 integer and 24 fractional. The 33rd, 34th and 35th entries are stored as27 bits per color in an unsigned 3.24 format with 3 integer and 24 fractional bits.
For input values greater than or equal to 1.0 and less than 3.0, the input value is used to linearly interpolate between the 33rd and 34th gamma entries to create the result value.
For input values greater than or equal to 3.0 and less than 7.0, the input value is used to linearly interpolate between the 34th and 35th gamma entries to create the result value.
For negative input values, gamma is mirrored along the X-axis, giving the same result as positive input values, except for a negative sign. When gamma input may be negative, the first gamma point should be programmed to a value of 0.0 in order to have a symmetric mirroring.
Pre-CSC Gamma correction gets enabled or disabled based on the "Pipe CSC Enable" bit in the plane control register. The same set of values is used for gamma correction of the red, blue and green channels.
See Pipe Gamma for an example gamma curve diagram.
To program the gamma correction entries, calculate the desired gamma curve for inputs from 0 to 7.0. For inputs of 0 to 1.0, multiply the input value by 32 to find the gamma entry number, then store the desired gamma result in that entry. For inputs greater than 1.0 and less than or equal to 3.0, store the result for an input of 3.0 in the

PLANE_POST_CSC_GAMC_DATA_ENH

34th gamma entry. For inputs greater than 3.0 and less than or equal to 7.0, store the result for an input of 7.0 in the 35th gamma entry.
For HDR tone mapping usages, only the first 33 entries get used. The entries are used either in an unsigned 0.24 format or unsigned 8.16 format based on PLANE_COLOR_CTL->Plane Gamma Multiplier Precision programming.

Restriction
The gamma curve must be flat or increasing, never decreasing when used in the direct lookup mode. The gamma correction registers should only be updated when the plane is off, otherwise screen artifacts may show temporarily.

DWord	Bit	Description
0	31:27	**Reserved** Access: RO Format: MBZ
	26:0	**Gamma Value** Default Value: 00000000000000000000000000b Format: U3.24

PLANE_POST_CSC_GAMC_INDEX

PLANE_POST_CSC_GAMC_INDEX	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF or plane not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled; after armed
Address:	704D8h-704DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_4_A
Reset:	soft
Address:	705D8h-705DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_5_A
Reset:	soft
Address:	706D8h-706DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_6_A
Reset:	soft
Address:	707D8h-707DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_7_A
Reset:	soft
Address:	714D8h-714DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_4_B
Reset:	soft
Address:	715D8h-715DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_5_B
Reset:	soft
Address:	716D8h-716DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_6_B
Reset:	soft
Address:	717D8h-717DBh

PLANE_POST_CSC_GAMC_INDEX	
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_7_B
Reset:	soft
Address:	724D8h-724DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_4_C
Reset:	soft
Address:	725D8h-725DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_5_C
Reset:	soft
Address:	726D8h-726DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_6_C
Reset:	soft
Address:	727D8h-727DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_7_C
Reset:	soft
Address:	734D8h-734DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_4_D
Reset:	soft
Address:	735D8h-735DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_5_D
Reset:	soft
Address:	736D8h-736DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_6_D
Reset:	soft
Address:	737D8h-737DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_7_D
Reset:	soft

DWord	Bit	Description

PLANE_POST_CSC_GAMC_INDEX

31:11 Reserved

Access:	RO
Format:	MBZ

10 Index Auto Increment

This field enables the index auto increment.

Value	Name	Description
0b	No Increment	Do not automatically increment the index value.
1b	Auto Increment **[Default]**	Increment the index value with each read or write to the data register.

9:6 Reserved

Access:	RO
Format:	MBZ

5:0 Index Value

Access:	Write/Read Status

This index controls access to the array of plane pre color space conversion gamma values.
This value can be automatically incremented by a read or a write to the data register if the index auto increment bit is set.
When automatically incrementing, the index will roll over to 0 after writing or reading the entire allowed range.
While in auto increment mode, after performing reads or writes to only part of the range, the auto increment bit must be cleared before resetting the index value.

Value	Name
[0,34]	

PLANE_POST_CSC_GAMC_INDEX_ENH

PLANE_POST_CSC_GAMC_INDEX_ENH	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	701D8h-701DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_ENH_1_A
Reset:	soft
Address:	702D8h-702DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_ENH_2_A
Reset:	soft
Address:	703D8h-703DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_ENH_3_A
Reset:	soft
Address:	711D8h-711DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_ENH_1_B
Reset:	soft
Address:	712D8h-712DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_ENH_2_B
Reset:	soft
Address:	713D8h-713DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_ENH_3_B
Reset:	soft
Address:	721D8h-721DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_ENH_1_C
Reset:	soft
Address:	722D8h-722DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_ENH_2_C

PLANE_POST_CSC_GAMC_INDEX_ENH

Reset:	soft
Address:	723D8h-723DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_ENH_3_C
Reset:	soft
Address:	731D8h-731DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_ENH_1_D
Reset:	soft
Address:	732D8h-732DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_ENH_2_D
Reset:	soft
Address:	733D8h-733DBh
Name:	Plane Post CSC Gamma Index
ShortName:	PLANE_POST_CSC_GAMC_INDEX_ENH_3_D
Reset:	soft

DWord	Bit	Description
0	31:11	**Reserved** Access: RO Format: MBZ
	10	**Index Auto Increment** This field enables the index auto increment. Value / Name / Description: 0b — No Increment — Do not automatically increment the index value. 1b — Auto Increment **[Default]** — Increment the index value with each read or write to the data register.
	9:6	**Reserved** Access: RO Format: MBZ
	5:0	**Index Value** Access: Write/Read Status This index controls access to the array of plane pre color space conversion gamma values. This value can be automatically incremented by a read or a write to the data register if the index auto increment bit is set. When automatically incrementing, the current automatically calculated index value can be read here, and the index will roll over to 0 after reaching the end of the allowed range. While in auto increment mode, after performing reads or writes to only part of the range, the

PLANE_POST_CSC_GAMC_INDEX_ENH		
		auto increment bit must be cleared before resetting the index value. Value: [0,34] Name:

PLANE_POST_CSC_GAMC_SEG0_DATA_ENH

PLANE_POST_CSC_GAMC_SEG0_DATA_ENH	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	70164h-70167h
Name:	Plane Post CSC Gamma Segment0 Data
ShortName:	PLANE_POST_CSC_GAMC_SEG0_DATA_ENH_1_A
Reset:	soft
Address:	70264h-70267h
Name:	Plane Post CSC Gamma Segment0 Data
ShortName:	PLANE_POST_CSC_GAMC_SEG0_DATA_ENH_2_A
Reset:	soft
Address:	70364h-70367h
Name:	Plane Post CSC Gamma Segment0 Data
ShortName:	PLANE_POST_CSC_GAMC_SEG0_DATA_ENH_3_A
Reset:	soft
Address:	71164h-71167h
Name:	Plane Post CSC Gamma Segment0 Data
ShortName:	PLANE_POST_CSC_GAMC_SEG0_DATA_ENH_1_B
Reset:	soft
Address:	71264h-71267h
Name:	Plane Post CSC Gamma Segment0 Data
ShortName:	PLANE_POST_CSC_GAMC_SEG0_DATA_ENH_2_B
Reset:	soft
Address:	71364h-71367h
Name:	Plane Post CSC Gamma Segment0 Data
ShortName:	PLANE_POST_CSC_GAMC_SEG0_DATA_ENH_3_B
Reset:	soft
Address:	72164h-72167h
Name:	Plane Post CSC Gamma Segment0 Data
ShortName:	PLANE_POST_CSC_GAMC_SEG0_DATA_ENH_1_C
Reset:	soft
Address:	72264h-72267h
Name:	Plane Post CSC Gamma Segment0 Data
ShortName:	PLANE_POST_CSC_GAMC_SEG0_DATA_ENH_2_C

PLANE_POST_CSC_GAMC_SEG0_DATA_ENH

Reset:	soft
Address:	72364h-72367h
Name:	Plane Post CSC Gamma Segment0 Data
ShortName:	PLANE_POST_CSC_GAMC_SEG0_DATA_ENH_3_C
Reset:	soft
Address:	73164h-73167h
Name:	Plane Post CSC Gamma Segment0 Data
ShortName:	PLANE_POST_CSC_GAMC_SEG0_DATA_ENH_1_D
Reset:	soft
Address:	73264h-73267h
Name:	Plane Post CSC Gamma Segment0 Data
ShortName:	PLANE_POST_CSC_GAMC_SEG0_DATA_ENH_2_D
Reset:	soft
Address:	73364h-73367h
Name:	Plane Post CSC Gamma Segment0 Data
ShortName:	PLANE_POST_CSC_GAMC_SEG0_DATA_ENH_3_D
Reset:	soft

PLANE_POST_CSC_GAMC_SEG0_INDEX and PLANE_POST_CSC_GAMC_SEG0_DATA registers are used to program the segment 0 values of the HDR tone mapping curve.
The entries are stored as 24 bits per color in an unsigned 0.24 format with 0 integer and 24 fractional.

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23:0	**Gamma Value** Default Value: 000000000000000000000000b Format: U0.24

PLANE_POST_CSC_GAMC_SEG0_INDEX_ENH

PLANE_POST_CSC_GAMC_SEG0_INDEX_ENH	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	70160h-70163h
Name:	Plane Post CSC Gamma Segment0 Index
ShortName:	PLANE_POST_CSC_GAMC_SEG0_INDEX_ENH_1_A
Reset:	soft
Address:	70260h-70263h
Name:	Plane Post CSC Gamma Segment0 Index
ShortName:	PLANE_POST_CSC_GAMC_SEG0_INDEX_ENH_2_A
Reset:	soft
Address:	70360h-70363h
Name:	Plane Post CSC Gamma Segment0 Index
ShortName:	PLANE_POST_CSC_GAMC_SEG0_INDEX_ENH_3_A
Reset:	soft
Address:	71160h-71163h
Name:	Plane Post CSC Gamma Segment0 Index
ShortName:	PLANE_POST_CSC_GAMC_SEG0_INDEX_ENH_1_B
Reset:	soft
Address:	71260h-71263h
Name:	Plane Post CSC Gamma Segment0 Index
ShortName:	PLANE_POST_CSC_GAMC_SEG0_INDEX_ENH_2_B
Reset:	soft
Address:	71360h-71363h
Name:	Plane Post CSC Gamma Segment0 Index
ShortName:	PLANE_POST_CSC_GAMC_SEG0_INDEX_ENH_3_B
Reset:	soft
Address:	72160h-72163h
Name:	Plane Post CSC Gamma Segment0 Index
ShortName:	PLANE_POST_CSC_GAMC_SEG0_INDEX_ENH_1_C
Reset:	soft
Address:	72260h-72263h
Name:	Plane Post CSC Gamma Segment0 Index
ShortName:	PLANE_POST_CSC_GAMC_SEG0_INDEX_ENH_2_C

PLANE_POST_CSC_GAMC_SEG0_INDEX_ENH

Reset:	soft
Address:	72360h-72363h
Name:	Plane Post CSC Gamma Segment0 Index
ShortName:	PLANE_POST_CSC_GAMC_SEG0_INDEX_ENH_3_C
Reset:	soft
Address:	73160h-73163h
Name:	Plane Post CSC Gamma Segment0 Index
ShortName:	PLANE_POST_CSC_GAMC_SEG0_INDEX_ENH_1_D
Reset:	soft
Address:	73260h-73263h
Name:	Plane Post CSC Gamma Segment0 Index
ShortName:	PLANE_POST_CSC_GAMC_SEG0_INDEX_ENH_2_D
Reset:	soft
Address:	73360h-73363h
Name:	Plane Post CSC Gamma Segment0 Index
ShortName:	PLANE_POST_CSC_GAMC_SEG0_INDEX_ENH_3_D
Reset:	soft

DWord	Bit	Description
0	31:11	**Reserved** Access: RO Format: MBZ
	10	**Index Auto Increment** This field enables the index auto increment. Value / Name / Description 0b / No Increment / Do not automatically increment the index value. 1b / Auto Increment **[Default]** / Increment the index value with each read or write to the data register.
	9:4	**Reserved** Access: RO Format: MBZ
	3:0	**Index Value** Access: Write/Read Status This index controls access to the segment 0of plane post color space conversion gamma values. This value can be automatically incremented by a read or a write to the data register if the index auto increment bit is set. When automatically incrementing, the current automatically calculated index value can be read here, and the index will roll over to 0 after reaching the end of the allowed range. While in auto increment mode, after performing reads or writes to only part of the range, the

PLANE_POST_CSC_GAMC_SEG0_INDEX_ENH		
		auto increment bit must be cleared before resetting the index value. Value: [0,8] Name:

PLANE_PRE_CSC_GAMC_DATA

PLANE_PRE_CSC_GAMC_DATA	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF or plane not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled; after armed
Address:	704D4h-704D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_4_A
Reset:	soft
Address:	705D4h-705D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_5_A
Reset:	soft
Address:	706D4h-706D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_6_A
Reset:	soft
Address:	707D4h-707D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_7_A
Reset:	soft
Address:	714D4h-714D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_4_B
Reset:	soft
Address:	715D4h-715D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_5_B
Reset:	soft
Address:	716D4h-716D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_6_B
Reset:	soft
Address:	717D4h-717D7h

PLANE_PRE_CSC_GAMC_DATA	
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_7_B
Reset:	soft
Address:	724D4h-724D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_4_C
Reset:	soft
Address:	725D4h-725D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_5_C
Reset:	soft
Address:	726D4h-726D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_6_C
Reset:	soft
Address:	727D4h-727D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_7_C
Reset:	soft
Address:	734D4h-734D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_4_D
Reset:	soft
Address:	735D4h-735D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_5_D
Reset:	soft
Address:	736D4h-736D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_6_D
Reset:	soft
Address:	737D4h-737D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_7_D
Reset:	soft
PLANE_PRE_CSC_GAMC_INDEX and PLANE_PRE_CSC_GAMC_DATA registers are used to program the values that	

PLANE_PRE_CSC_GAMC_DATA

determine the characteristics of the gamma correction for the plane pixel data before Color Space Conversion. Additional gamma correction can be done after the Color Space Conversion, if needed.
The gamma correction curve is represented by specifying a set of gamma entry reference points spaced equally along the curve for values between -1 and 1. For extended values there are extended gamma entry reference points. All input values are clamped to the greater than -7.0 and less than 7.0 range before the gamma calculation.
For input values greater than or equal to 0 and less than 1.0, the input value is used to linearly interpolate between two adjacent points of the first 33 gamma entries to create the result value. The first 32 entries are stored as16 bits per color in an unsigned 0.16 format with 0 integer and 16 fractional. The 33rd, 34th and 35th entries are stored as 19 bits per color in an unsigned 3.16 format with 3 integer and 16 fractional bits.
For input values greater than or equal to 1.0 and less than 3.0, the input value is used to linearly interpolate between the 33rd and 34th gamma entries to create the result value.
For input values greater than or equal to 3.0 and less than 7.0, the input value is used to linearly interpolate between the 34th and 35th gamma entries to create the result value.
For negative input values, gamma is mirrored along the X-axis, giving the same result as positive input values, except for a negative sign. When gamma input may be negative, the first gamma point should be programmed to a value of 0.0 in order to have a symmetric mirroring.
Plane Pre-CSC Gamma correction gets enabled or disabled based on the 'Plane Pre CSC Gamma Enable'bit in the 'PLANE_COLOR_CTL'register. The same set of values is used for gamma correction of the red, blue and green channels.

Programming Notes

To program the gamma correction entries, calculate the desired gamma curve for inputs from 0 to 7.0. For inputs of 0 to 1.0, multiply the input value by 32 to find the gamma entry number, then store the desired gamma result in that entry. For inputs greater than 1.0 and less than or equal to 3.0, store the result for an input of 3.0 in the 34th gamma entry. For inputs greater than 3.0 and less than or equal to 7.0, store the result for an input of 7.0 in the 35th gamma entry.

Restriction

The gamma curve must be flat or increasing, never decreasing. The gamma correction registers should only be updated when the plane is off, otherwise screen artifacts may show temporarily.

DWord	Bit	Description
0	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**Gamma Value** Default Value: 0000000000000000b Format: U3.16

PLANE_PRE_CSC_GAMC_DATA_ENH

PLANE_PRE_CSC_GAMC_DATA_ENH	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
Address:	701D4h-701D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_ENH_1_A
Reset:	soft
Address:	702D4h-702D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_ENH_2_A
Reset:	soft
Address:	703D4h-703D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_ENH_3_A
Reset:	soft
Address:	711D4h-711D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_ENH_1_B
Reset:	soft
Address:	712D4h-712D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_ENH_2_B
Reset:	soft
Address:	713D4h-713D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_ENH_3_B
Reset:	soft
Address:	721D4h-721D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_ENH_1_C
Reset:	soft
Address:	722D4h-722D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_ENH_2_C

PLANE_PRE_CSC_GAMC_DATA_ENH

Reset:	soft

Address:	723D4h-723D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_ENH_3_C
Reset:	soft

Address:	731D4h-731D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_ENH_1_D
Reset:	soft

Address:	732D4h-732D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_ENH_2_D
Reset:	soft

Address:	733D4h-733D7h
Name:	Plane Pre CSC Gamma Data
ShortName:	PLANE_PRE_CSC_GAMC_DATA_ENH_3_D
Reset:	soft

PLANE_PRE_CSC_GAMC_INDEX and PLANE_PRE_CSC_GAMC_DATA registers are used to program the values that determine the characteristics of the gamma correction for the plane pixel data before Color Space Conversion. Additional gamma correction can be done after the Color Space Conversion, if needed.
The gamma correction curve is represented by specifying a set of gamma entry reference points spaced equally along the curve for values between -1 and 1. For extended values there are extended gamma entry reference points. All input values are clamped to the greater than -7.0 and less than 7.0 range before the gamma calculation.
For input values greater than or equal to 0 and less than 1.0, the input value is used to linearly interpolate between two adjacent points of the first 129 gamma entries to create the result value. The first 128 entries are stored as 24 bits per color in an unsigned 0.24 format with 0 integer and 24 fractional. The 129th, 130th and 131th entries are stored as 27 bits per color in an unsigned 3.24 format with 3 integer and 24 fractional bits.
For input values greater than or equal to 1.0 and less than 3.0, the input value is used to linearly interpolate between the 129th and 130th gamma entries to create the result value.
For input values greater than or equal to 3.0 and less than 7.0, the input value is used to linearly interpolate between the 130th and 131st gamma entries to create the result value.
For negative input values, gamma is mirrored along the X-axis, giving the same result as positive input values, except for a negative sign. When gamma input may be negative, the first gamma point should be programmed to a value of 0.0 in order to have a symmetric mirroring.
Pre-CSC Gamma correction gets enabled or disabled based on the "Plane Pre CSC Gamma Enable" bit in the plane color control register. The same set of values is used for gamma correction of the red, blue and green channels.
See Pipe Gamma for an example gamma curve diagram.

Programming Notes

PLANE_PRE_CSC_GAMC_DATA_ENH

To program the gamma correction entries, calculate the desired gamma curve for inputs from 0 to 7.0. For inputs of 0 to 1.0, multiply the input value by 128 to find the gamma entry number, then store the desired gamma result in that entry. For inputs greater than 1.0 and less than or equal to 3.0, store the result for an input of 3.0 in the 129th gamma entry. For inputs greater than 3.0 and less than or equal to 7.0, store the result for an input of 7.0 in the 130th gamma entry.

Restriction
The gamma curve must be flat or increasing, never decreasing. The gamma correction registers should only be updated when the plane is off, otherwise screen artifacts may show temporarily.

DWord	Bit	Description
0	31:27	**Reserved** Access: RO Format: MBZ
	26:0	**Gamma Value** Default Value: 00000000000000000000000000b Format: U3.24

PLANE_PRE_CSC_GAMC_INDEX

PLANE_PRE_CSC_GAMC_INDEX	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF or plane not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled; after armed
Address:	704D0h-704D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_4_A
Reset:	soft
Address:	705D0h-705D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_5_A
Reset:	soft
Address:	706D0h-706D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_6_A
Reset:	soft
Address:	707D0h-707D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_7_A
Reset:	soft
Address:	714D0h-714D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_4_B
Reset:	soft
Address:	715D0h-715D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_5_B
Reset:	soft
Address:	716D0h-716D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_6_B
Reset:	soft
Address:	717D0h-717D3h

PLANE_PRE_CSC_GAMC_INDEX	
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_7_B
Reset:	soft
Address:	724D0h-724D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_4_C
Reset:	soft
Address:	725D0h-725D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_5_C
Reset:	soft
Address:	726D0h-726D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_6_C
Reset:	soft
Address:	727D0h-727D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_7_C
Reset:	soft
Address:	734D0h-734D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_4_D
Reset:	soft
Address:	735D0h-735D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_5_D
Reset:	soft
Address:	736D0h-736D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_6_D
Reset:	soft
Address:	737D0h-737D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_7_D
Reset:	soft

DWord	Bit	Description

PLANE_PRE_CSC_GAMC_INDEX

31:11 Reserved

Access:	RO
Format:	MBZ

10 Index Auto Increment

This field enables the index auto increment.

Value	Name	Description
0b	No Increment	Do not automatically increment the index value.
1b	Auto Increment **[Default]**	Increment the index value with each read or write to the data register.

9:6 Reserved

Access:	RO
Format:	MBZ

5:0 Index Value

Access:	Write/Read Status

This index controls access to the array of plane pre color space conversion gamma values.
This value can be automatically incremented by a read or a write to the data register if the index auto increment bit is set.
When automatically incrementing, the index will roll over to 0 after writing or reading the entire allowed range.
While in auto increment mode, after performing reads or writes to only part of the range, the auto increment bit must be cleared before resetting the index value.

Value	Name
[0,34]	

PLANE_PRE_CSC_GAMC_INDEX_ENH

PLANE_PRE_CSC_GAMC_INDEX_ENH	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
Address:	701D0h-701D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_ENH_1_A
Reset:	soft
Address:	702D0h-702D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_ENH_2_A
Reset:	soft
Address:	703D0h-703D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_ENH_3_A
Reset:	soft
Address:	711D0h-711D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_ENH_1_B
Reset:	soft
Address:	712D0h-712D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_ENH_2_B
Reset:	soft
Address:	713D0h-713D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_ENH_3_B
Reset:	soft
Address:	721D0h-721D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_ENH_1_C
Reset:	soft
Address:	722D0h-722D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_ENH_2_C

PLANE_PRE_CSC_GAMC_INDEX_ENH

Reset:	soft

Address:	723D0h-723D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_ENH_3_C
Reset:	soft

Address:	731D0h-731D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_ENH_1_D
Reset:	soft

Address:	732D0h-732D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_ENH_2_D
Reset:	soft

Address:	733D0h-733D3h
Name:	Plane Pre CSC Gamma Index
ShortName:	PLANE_PRE_CSC_GAMC_INDEX_ENH_3_D
Reset:	soft

DWord	Bit	Description
0	31:11	**Reserved** Access: RO Format: MBZ
	10	**Index Auto Increment** This field enables the index auto increment. Value 0b – No Increment – Do not automatically increment the index value. Value 1b – Auto Increment **[Default]** – Increment the index value with each read or write to the data register.
	9:8	**Reserved** Access: RO Format: MBZ
	7:0	**Index Value** Access: Write/Read Status This index controls access to the array of plane pre color space conversion gamma values. This value can be automatically incremented by a read or a write to the data register if the index auto increment bit is set. When automatically incrementing, the index will roll over to 0 after writing or reading the entire allowed range. While in auto increment mode, after performing reads or writes to only part of the range, the

PLANE_PRE_CSC_GAMC_INDEX_ENH		
		auto increment bit must be cleared before resetting the index value. **Value** | **Name** [0,130] |

PLANE_SIZE

PLANE_SIZE	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF or plane not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled; after armed
Address:	70490h-70493h
Name:	Plane Size
ShortName:	PLANE_SIZE_4_A
Reset:	soft
Address:	70590h-70593h
Name:	Plane Size
ShortName:	PLANE_SIZE_5_A
Reset:	soft
Address:	70690h-70693h
Name:	Plane Size
ShortName:	PLANE_SIZE_6_A
Reset:	soft
Address:	70790h-70793h
Name:	Plane Size
ShortName:	PLANE_SIZE_7_A
Reset:	soft
Address:	71490h-71493h
Name:	Plane Size
ShortName:	PLANE_SIZE_4_B
Reset:	soft
Address:	71590h-71593h
Name:	Plane Size
ShortName:	PLANE_SIZE_5_B
Reset:	soft
Address:	71690h-71693h
Name:	Plane Size
ShortName:	PLANE_SIZE_6_B
Reset:	soft
Address:	71790h-71793h

PLANE_SIZE	
Name:	Plane Size
ShortName:	PLANE_SIZE_7_B
Reset:	soft
Address:	72490h-72493h
Name:	Plane Size
ShortName:	PLANE_SIZE_4_C
Reset:	soft
Address:	72590h-72593h
Name:	Plane Size
ShortName:	PLANE_SIZE_5_C
Reset:	soft
Address:	72690h-72693h
Name:	Plane Size
ShortName:	PLANE_SIZE_6_C
Reset:	soft
Address:	72790h-72793h
Name:	Plane Size
ShortName:	PLANE_SIZE_7_C
Reset:	soft
Address:	73490h-73493h
Name:	Plane Size
ShortName:	PLANE_SIZE_4_D
Reset:	soft
Address:	73590h-73593h
Name:	Plane Size
ShortName:	PLANE_SIZE_5_D
Reset:	soft
Address:	73690h-73693h
Name:	Plane Size
ShortName:	PLANE_SIZE_6_D
Reset:	soft
Address:	73790h-73793h
Name:	Plane Size
ShortName:	PLANE_SIZE_7_D
Reset:	soft
Address:	70190h-70193h

PLANE_SIZE	
Name:	Plane Size
ShortName:	PLANE_SIZE_1_A
Reset:	soft
Address:	70290h-70293h
Name:	Plane Size
ShortName:	PLANE_SIZE_2_A
Reset:	soft
Address:	70390h-70393h
Name:	Plane Size
ShortName:	PLANE_SIZE_3_A
Reset:	soft
Address:	71190h-71193h
Name:	Plane Size
ShortName:	PLANE_SIZE_1_B
Reset:	soft
Address:	71290h-71293h
Name:	Plane Size
ShortName:	PLANE_SIZE_2_B
Reset:	soft
Address:	71390h-71393h
Name:	Plane Size
ShortName:	PLANE_SIZE_3_B
Reset:	soft
Address:	72190h-72193h
Name:	Plane Size
ShortName:	PLANE_SIZE_1_C
Reset:	soft
Address:	72290h-72293h
Name:	Plane Size
ShortName:	PLANE_SIZE_2_C
Reset:	soft
Address:	72390h-72393h
Name:	Plane Size
ShortName:	PLANE_SIZE_3_C
Reset:	soft
Address:	73190h-73193h

PLANE_SIZE	
Name:	Plane Size
ShortName:	PLANE_SIZE_1_D
Reset:	soft
Address:	73290h-73293h
Name:	Plane Size
ShortName:	PLANE_SIZE_2_D
Reset:	soft
Address:	73390h-73393h
Name:	Plane Size
ShortName:	PLANE_SIZE_3_D
Reset:	soft
Address:	70898h-7089Bh
Name:	Selective Fetch Plane Size
ShortName:	SEL_FETCH_PLANE_SIZE_1_A
Reset:	soft
Address:	708B8h-708BBh
Name:	Selective Fetch Plane Size
ShortName:	SEL_FETCH_PLANE_SIZE_2_A
Reset:	soft
Address:	708D8h-708DBh
Name:	Selective Fetch Plane Size
ShortName:	SEL_FETCH_PLANE_SIZE_3_A
Reset:	soft
Address:	708F8h-708FBh
Name:	Selective Fetch Plane Size
ShortName:	SEL_FETCH_PLANE_SIZE_4_A
Reset:	soft
Address:	70928h-7092Bh
Name:	Selective Fetch Plane Size
ShortName:	SEL_FETCH_PLANE_SIZE_5_A
Reset:	soft
Address:	70948h-7094Bh
Name:	Selective Fetch Plane Size
ShortName:	SEL_FETCH_PLANE_SIZE_6_A
Reset:	soft
Address:	70968h-7096Bh

PLANE_SIZE

Name: Selective Fetch Plane Size

ShortName: SEL_FETCH_PLANE_SIZE_7_A

Reset: soft

This register specifies the plane source size, the size of the image fetched from the frame buffer. When plane scaling is not enabled on this plane, this is the size of the plane when blended with other planes on this pipe. When plane scaling is enabled on this plane, the scaler window size is the size of the plane when blended with other planes on this pipe.

Restriction

When plane scaling is not enabled on this plane, the plane must be completely contained within the pipe source area. Pipe source size >= plane position + plane size.

For OLED compensation plane size restrictions, refer to PIPE_MISC->OLED Compensation (bit[12]).

Height and Width restrictions are specified in the following table. For formats not specified in the table, both odd and even sizes are supported.

PixelFormat	Rotate	Width	Height
YUV 420 Planar - NV12	All	Even	Even
YUV 420 Planar - P01x	All	Even	Even
YUV 422	All	Even	Even
RGB565	90, 270	Even	Even

If Plane Scaling or using the Chroma Up-Sampler (CUS) for this plane, please refer to **PS_CTRL** or **PLANE_CUS_CTL** respectively, for further size restrictions.

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**Height** This specifies the height of the plane in lines. The value in the register is the height minus one. **Restriction** The height must be at least one line when non-interlaced, two lines when interlaced. The height is limited to maximum of 4320 lines. Refer to size restrictions within PS_CTRL when plane scaling is enabled.
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**Width** This specifies the width of the plane in pixels. The value in the register is the width minus one. **Restriction** The width must be even (programmed value odd) when YUV 4:2:2 or YUV 4:2:0 source pixel format is used. For YUV4:2:0 (NV12), the UV plane must be greater than or equal to 8 and the Y plane must be greater than or equal to 16.

<table>
<tr><th colspan="3">PLANE_SIZE</th></tr>
<tr><td></td><td></td><td>The width must be greater than or equal to 4 for 32bpp, YUV212 and YUV216 formats, greater than or equal to 8 for 16bpp formats, and greater than or equal to 16 for 8bpp formats.
The width must be greater than or equal to 2 for 64bpp formats.
The width must be greater than or equal to 8 for P010, P012 and P016 formats.
The width should be less than or equal to the stride in pixels.

For planar YUV 420 formats, refer to chroma upsampler size restrictions in PLANE_CUS_CTL register.
<table><tr><th>Tiling format</th><th>Bytes per pixel</th><th>Max Width supported in pixels</th></tr><tr><td>Linear, X Tiling</td><td>1,2,4,8</td><td>5120</td></tr><tr><td>Y Tiling</td><td>1,2,4,8</td><td>5120</td></tr></table></td></tr>
</table>

PLANE_STRIDE

PLANE_STRIDE	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF or plane not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled; after armed
Address:	70488h-7048Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_4_A
Reset:	soft
Address:	70588h-7058Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_5_A
Reset:	soft
Address:	70688h-7068Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_6_A
Reset:	soft
Address:	70788h-7078Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_7_A
Reset:	soft
Address:	71488h-7148Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_4_B
Reset:	soft
Address:	71588h-7158Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_5_B
Reset:	soft
Address:	71688h-7168Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_6_B
Reset:	soft
Address:	71788h-7178Bh

PLANE_STRIDE	
Name:	Plane Stride
ShortName:	PLANE_STRIDE_7_B
Reset:	soft
Address:	72488h-7248Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_4_C
Reset:	soft
Address:	72588h-7258Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_5_C
Reset:	soft
Address:	72688h-7268Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_6_C
Reset:	soft
Address:	72788h-7278Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_7_C
Reset:	soft
Address:	73488h-7348Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_4_D
Reset:	soft
Address:	73588h-7358Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_5_D
Reset:	soft
Address:	73688h-7368Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_6_D
Reset:	soft
Address:	73788h-7378Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_7_D
Reset:	soft
Address:	70188h-7018Bh

PLANE_STRIDE	
Name:	Plane Stride
ShortName:	PLANE_STRIDE_1_A
Reset:	soft
Address:	70288h-7028Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_2_A
Reset:	soft
Address:	70388h-7038Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_3_A
Reset:	soft
Address:	71188h-7118Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_1_B
Reset:	soft
Address:	71288h-7128Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_2_B
Reset:	soft
Address:	71388h-7138Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_3_B
Reset:	soft
Address:	72188h-7218Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_1_C
Reset:	soft
Address:	72288h-7228Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_2_C
Reset:	soft
Address:	72388h-7238Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_3_C
Reset:	soft
Address:	73188h-7318Bh

PLANE_STRIDE

Name:	Plane Stride
ShortName:	PLANE_STRIDE_1_D
Reset:	soft

Address:	73288h-7328Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_2_D
Reset:	soft

Address:	73388h-7338Bh
Name:	Plane Stride
ShortName:	PLANE_STRIDE_3_D
Reset:	soft

This register may be updated through MMIO writes or through command streamer initiated synchronous flips.

DWord	Bit	Description
0	31:18	**Reserved** Access: RO Format: MBZ
	17:12	**Reserved** Access: RO Format: MBZ
	11	**Reserved** Access: RO Format: MBZ
	10:0	**Stride** This field specifies the stride for the plane. The field is used to determine the line to line increment for the plane. For Linear memory, this field specifies the stride in chunks of 64 bytes (1 cache line). If the programmed value is 100, the actual stride = 100 * 64 = 6400 bytes. For X-Tiled & Y-Tiled memory, this field specifies the stride in number of tiles. For Tile X, if the programmed value is 10, the actual stride = 10 * 512 (X tile width) = 5120 bytes. For Tile Y, if the programmed value is 10, the actual stride = 10 * 128 (Y tile width) = 1280 bytes. **Tile Format** / **Width in bytes** Tile X: 512 Tile Y (legacy): 128 Restriction : For YUV planar (NV12 or P0xx) plane pixel formats, the stride calculated in bytes should be equal for the Y and UV surfaces. The stride in bytes must not exceed the of the size of 8K pixels.

PLANE_STRIDE

Tile Format	Pixel Format	Maximum Stride in tiles
Linear	64 bpp pixel format	1024
	32 bpp pixel format	512
	16 bpp pixel format	256
	8 bpp pixel format	128
X Tiling	64 bpp pixel format	128
	32 bpp pixel format	64
	16 bpp pixel format	32
	8 bpp pixel format	16
Y Tiling (Legacy)	64 bpp pixel format	512
	32 bpp pixel format	256
	16 bpp pixel format	128
	8 bpp pixel format	64

PLANE_SURF

PLANE_SURF	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of left or right eye vertical blank (selectable), pipe not enabled, or plane not enabled
Address:	7049Ch-7049Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_4_A
Reset:	soft
Address:	7059Ch-7059Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_5_A
Reset:	soft
Address:	7069Ch-7069Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_6_A
Reset:	soft
Address:	7079Ch-7079Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_7_A
Reset:	soft
Address:	7149Ch-7149Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_4_B
Reset:	soft
Address:	7159Ch-7159Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_5_B
Reset:	soft
Address:	7169Ch-7169Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_6_B
Reset:	soft
Address:	7179Ch-7179Fh

PLANE_SURF	
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_7_B
Reset:	soft
Address:	7249Ch-7249Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_4_C
Reset:	soft
Address:	7259Ch-7259Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_5_C
Reset:	soft
Address:	7269Ch-7269Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_6_C
Reset:	soft
Address:	7279Ch-7279Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_7_C
Reset:	soft
Address:	7349Ch-7349Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_4_D
Reset:	soft
Address:	7359Ch-7359Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_5_D
Reset:	soft
Address:	7369Ch-7369Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_6_D
Reset:	soft
Address:	7379Ch-7379Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_7_D
Reset:	soft
Address:	7019Ch-7019Fh

PLANE_SURF	
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_1_A
Reset:	soft
Address:	7029Ch-7029Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_2_A
Reset:	soft
Address:	7039Ch-7039Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_3_A
Reset:	soft
Address:	7119Ch-7119Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_1_B
Reset:	soft
Address:	7129Ch-7129Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_2_B
Reset:	soft
Address:	7139Ch-7139Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_3_B
Reset:	soft
Address:	7219Ch-7219Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_1_C
Reset:	soft
Address:	7229Ch-7229Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_2_C
Reset:	soft
Address:	7239Ch-7239Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_3_C
Reset:	soft
Address:	7319Ch-7319Fh

PLANE_SURF

Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_1_D
Reset:	soft

Address:	7329Ch-7329Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_2_D
Reset:	soft

Address:	7339Ch-7339Fh
Name:	Plane Surface Base Address
ShortName:	PLANE_SURF_3_D
Reset:	soft

Writes to this register arm primary registers for this pipe. A write to this register is considered a flip and can cause a flip done interrupt if the interrupt registers are configured for that. The values in this register may be updated through MMIO writes or through command streamer initiated flips. Synchronous updates (synchronous command streamer flips or synchronous MMIO writes) will update the plane surface values at the start of the next vertical blank. Asynchronous updates (asynchronous command streamer flips or asynchronous MMIO writes) will update the plane surface values at the next TLB request or at the start of the next vertical blank. Stereo 3D synchronous updates (stereo 3D command streamer flips or synchronous MMIO writes while stereo 3D is enabled) will update at the start of either the left or right eye vertical blank, selectable by the plane control register stereo surface vblank mask.

Double buffering control does not apply to PLANE_SURF updates that occur when the plane is disabled. An interrupt event is generated immediately when the PLANE_SURF is written. If the interrupt is unmasked, the interrupt is logged in the IIR.

Asynchronous flip completion time depends greatly on how much data has been prefetched for power savings, and can take up to 1 full frame to complete. For faster flip completion, disable FBC and render compression and allocate a small amount of data buffer for the plane.

DWord	Bit	Description
0	31:12	**Surface Base Address** Format: GraphicsAddress[31:12]
	11	**Reserved**
	10:7	**Reserved** Access: RO Format: MBZ
	6:4	**Reserved**
	3	**Ring Flip Source** This bit indicates if the source of the last ring flip was CS or BCS. This will determine where the flip done response is sent. Value: 0b — Name: CS

PLANE_SURF			
		1b	BCS
	2	**Reserved**	
	1:0	**Reserved**	
		Access:	RO
		Format:	MBZ

PLANE_SURFLIVE

PLANE_SURFLIVE	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	704ACh-704AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_4_A
Reset:	soft
Address:	704BCh-704BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_4_A
Reset:	soft
Address:	705ACh-705AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_5_A
Reset:	soft
Address:	705BCh-705BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_5_A
Reset:	soft
Address:	706ACh-706AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_6_A
Reset:	soft
Address:	706BCh-706BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_6_A
Reset:	soft
Address:	707ACh-707AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_7_A
Reset:	soft
Address:	707BCh-707BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_7_A
Reset:	soft

PLANE_SURFLIVE	
Address:	714ACh-714AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_4_B
Reset:	soft
Address:	714BCh-714BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_4_B
Reset:	soft
Address:	715ACh-715AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_5_B
Reset:	soft
Address:	715BCh-715BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_5_B
Reset:	soft
Address:	716ACh-716AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_6_B
Reset:	soft
Address:	716BCh-716BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_6_B
Reset:	soft
Address:	717ACh-717AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_7_B
Reset:	soft
Address:	717BCh-717BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_7_B
Reset:	soft
Address:	724ACh-724AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_4_C
Reset:	soft

PLANE_SURFLIVE	
Address:	724BCh-724BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_4_C
Reset:	soft
Address:	725ACh-725AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_5_C
Reset:	soft
Address:	725BCh-725BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_5_C
Reset:	soft
Address:	726ACh-726AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_6_C
Reset:	soft
Address:	726BCh-726BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_6_C
Reset:	soft
Address:	727ACh-727AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_7_C
Reset:	soft
Address:	727BCh-727BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_7_C
Reset:	soft
Address:	734ACh-734AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_4_D
Reset:	soft
Address:	734BCh-734BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_4_D
Reset:	soft

PLANE_SURFLIVE	
Address:	735ACh-735AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_5_D
Reset:	soft
Address:	735BCh-735BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_5_D
Reset:	soft
Address:	736ACh-736AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_6_D
Reset:	soft
Address:	736BCh-736BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_6_D
Reset:	soft
Address:	737ACh-737AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_7_D
Reset:	soft
Address:	737BCh-737BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_7_D
Reset:	soft
Address:	701ACh-701AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_1_A
Reset:	soft
Address:	701BCh-701BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_1_A
Reset:	soft
Address:	702ACh-702AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_2_A
Reset:	soft

PLANE_SURFLIVE	
Address:	702BCh-702BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_2_A
Reset:	soft
Address:	703ACh-703AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_3_A
Reset:	soft
Address:	703BCh-703BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_3_A
Reset:	soft
Address:	711ACh-711AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_1_B
Reset:	soft
Address:	711BCh-711BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_1_B
Reset:	soft
Address:	712ACh-712AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_2_B
Reset:	soft
Address:	712BCh-712BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_2_B
Reset:	soft
Address:	713ACh-713AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_3_B
Reset:	soft
Address:	713BCh-713BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_3_B
Reset:	soft

PLANE_SURFLIVE	
Address:	721ACh-721AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_1_C
Reset:	soft
Address:	721BCh-721BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_1_C
Reset:	soft
Address:	722ACh-722AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_2_C
Reset:	soft
Address:	722BCh-722BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_2_C
Reset:	soft
Address:	723ACh-723AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_3_C
Reset:	soft
Address:	723BCh-723BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_3_C
Reset:	soft
Address:	731ACh-731AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_1_D
Reset:	soft
Address:	731BCh-731BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_1_D
Reset:	soft
Address:	732ACh-732AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_2_D
Reset:	soft

PLANE_SURFLIVE

Address:	732BCh-732BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_2_D
Reset:	soft
Address:	733ACh-733AFh
Name:	Plane Live Surface Base Address
ShortName:	PLANE_SURFLIVE_3_D
Reset:	soft
Address:	733BCh-733BFh
Name:	Plane Live Left Surface Base Address
ShortName:	PLANE_LEFT_SURFLIVE_3_D
Reset:	soft

There is one instance of this register for each plane.

DWord	Bit	Description
0	31:12	**Live Surface Base Address** Access: RO This gives the live value of the surface base address as being currently used for the plane.
	11	**Reserved** Access: RO Format: MBZ
	10:9	**Reserved**
	8:0	**Reserved** Access: RO Format: MBZ

PLANE_WM

PLANE_WM	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF/CUR_BASE or plane/cursor not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank, plane not enabled, or pipe not enabled
Address:	70140h-70143h
Name:	Cursor Watermarks
ShortName:	CUR_WM_0_A
Reset:	soft
Address:	70144h-70147h
Name:	Cursor Watermarks
ShortName:	CUR_WM_1_A
Reset:	soft
Address:	70148h-7014Bh
Name:	Cursor Watermarks
ShortName:	CUR_WM_2_A
Reset:	soft
Address:	7014Ch-7014Fh
Name:	Cursor Watermarks
ShortName:	CUR_WM_3_A
Reset:	soft
Address:	70150h-70153h
Name:	Cursor Watermarks
ShortName:	CUR_WM_4_A
Reset:	soft
Address:	70154h-70157h
Name:	Cursor Watermarks
ShortName:	CUR_WM_5_A
Reset:	soft
Address:	70158h-7015Bh
Name:	Cursor Watermarks
ShortName:	CUR_WM_6_A
Reset:	soft
Address:	7015Ch-7015Fh

PLANE_WM	
Name:	Cursor Watermarks
ShortName:	CUR_WM_7_A
Reset:	soft
Address:	70168h-7016Bh
Name:	Cursor Transition Watermark
ShortName:	CUR_WM_TRANS_A
Reset:	soft
Address:	71140h-71143h
Name:	Cursor Watermarks
ShortName:	CUR_WM_0_B
Reset:	soft
Address:	71144h-71147h
Name:	Cursor Watermarks
ShortName:	CUR_WM_1_B
Reset:	soft
Address:	71148h-7114Bh
Name:	Cursor Watermarks
ShortName:	CUR_WM_2_B
Reset:	soft
Address:	7114Ch-7114Fh
Name:	Cursor Watermarks
ShortName:	CUR_WM_3_B
Reset:	soft
Address:	71150h-71153h
Name:	Cursor Watermarks
ShortName:	CUR_WM_4_B
Reset:	soft
Address:	71154h-71157h
Name:	Cursor Watermarks
ShortName:	CUR_WM_5_B
Reset:	soft
Address:	71158h-7115Bh
Name:	Cursor Watermarks
ShortName:	CUR_WM_6_B
Reset:	soft
Address:	7115Ch-7115Fh

PLANE_WM	
Name:	Cursor Watermarks
ShortName:	CUR_WM_7_B
Reset:	soft
Address:	71168h-7116Bh
Name:	Cursor Transition Watermark
ShortName:	CUR_WM_TRANS_B
Reset:	soft
Address:	72140h-72143h
Name:	Cursor Watermarks
ShortName:	CUR_WM_0_C
Reset:	soft
Address:	72144h-72147h
Name:	Cursor Watermarks
ShortName:	CUR_WM_1_C
Reset:	soft
Address:	72148h-7214Bh
Name:	Cursor Watermarks
ShortName:	CUR_WM_2_C
Reset:	soft
Address:	7214Ch-7214Fh
Name:	Cursor Watermarks
ShortName:	CUR_WM_3_C
Reset:	soft
Address:	72150h-72153h
Name:	Cursor Watermarks
ShortName:	CUR_WM_4_C
Reset:	soft
Address:	72154h-72157h
Name:	Cursor Watermarks
ShortName:	CUR_WM_5_C
Reset:	soft
Address:	72158h-7215Bh
Name:	Cursor Watermarks
ShortName:	CUR_WM_6_C
Reset:	soft
Address:	7215Ch-7215Fh

PLANE_WM	
Name:	Cursor Watermarks
ShortName:	CUR_WM_7_C
Reset:	soft
Address:	72168h-7216Bh
Name:	Cursor Transition Watermark
ShortName:	CUR_WM_TRANS_C
Reset:	soft
Address:	73140h-73143h
Name:	Cursor Watermarks
ShortName:	CUR_WM_0_D
Reset:	soft
Address:	73144h-73147h
Name:	Cursor Watermarks
ShortName:	CUR_WM_1_D
Reset:	soft
Address:	73148h-7314Bh
Name:	Cursor Watermarks
ShortName:	CUR_WM_2_D
Reset:	soft
Address:	7314Ch-7314Fh
Name:	Cursor Watermarks
ShortName:	CUR_WM_3_D
Reset:	soft
Address:	73150h-73153h
Name:	Cursor Watermarks
ShortName:	CUR_WM_4_D
Reset:	soft
Address:	73154h-73157h
Name:	Cursor Watermarks
ShortName:	CUR_WM_5_D
Reset:	soft
Address:	73158h-7315Bh
Name:	Cursor Watermarks
ShortName:	CUR_WM_6_D
Reset:	soft
Address:	7315Ch-7315Fh

PLANE_WM	
Name:	Cursor Watermarks
ShortName:	CUR_WM_7_D
Reset:	soft
Address:	73168h-7316Bh
Name:	Cursor Transition Watermark
ShortName:	CUR_WM_TRANS_D
Reset:	soft
Address:	70540h-70543h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_4_A
Reset:	soft
Address:	70544h-70547h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_4_A
Reset:	soft
Address:	70548h-7054Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_4_A
Reset:	soft
Address:	7054Ch-7054Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_4_A
Reset:	soft
Address:	70550h-70553h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_4_A
Reset:	soft
Address:	70554h-70557h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_4_A
Reset:	soft
Address:	70558h-7055Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_4_A
Reset:	soft
Address:	7055Ch-7055Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_4_A
Reset:	soft
Address:	70568h-7056Bh
Name:	Plane Transition Watermark
ShortName:	PLANE_WM_TRANS_4_A
Reset:	soft
Address:	70640h-70643h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_5_A
Reset:	soft
Address:	70644h-70647h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_5_A
Reset:	soft
Address:	70648h-7064Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_5_A
Reset:	soft
Address:	7064Ch-7064Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_5_A
Reset:	soft
Address:	70650h-70653h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_5_A
Reset:	soft
Address:	70654h-70657h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_5_A
Reset:	soft
Address:	70658h-7065Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_5_A
Reset:	soft
Address:	7065Ch-7065Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_5_A
Reset:	soft
Address:	70668h-7066Bh
Name:	Plane Transition Watermark
ShortName:	PLANE_WM_TRANS_5_A
Reset:	soft
Address:	70740h-70743h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_6_A
Reset:	soft
Address:	70744h-70747h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_6_A
Reset:	soft
Address:	70748h-7074Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_6_A
Reset:	soft
Address:	7074Ch-7074Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_6_A
Reset:	soft
Address:	70750h-70753h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_6_A
Reset:	soft
Address:	70754h-70757h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_6_A
Reset:	soft
Address:	70758h-7075Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_6_A
Reset:	soft
Address:	7075Ch-7075Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_6_A
Reset:	soft
Address:	70768h-7076Bh
Name:	Plane Transition Watermark
ShortName:	PLANE_WM_TRANS_6_A
Reset:	soft
Address:	70840h-70843h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_7_A
Reset:	soft
Address:	70844h-70847h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_7_A
Reset:	soft
Address:	70848h-7084Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_7_A
Reset:	soft
Address:	7084Ch-7084Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_7_A
Reset:	soft
Address:	70850h-70853h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_7_A
Reset:	soft
Address:	70854h-70857h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_7_A
Reset:	soft
Address:	70858h-7085Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_7_A
Reset:	soft
Address:	7085Ch-7085Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_7_A
Reset:	soft
Address:	70868h-7086Bh
Name:	Plane Transition Watermark
ShortName:	PLANE_WM_TRANS_7_A
Reset:	soft
Address:	71540h-71543h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_4_B
Reset:	soft
Address:	71544h-71547h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_4_B
Reset:	soft
Address:	71548h-7154Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_4_B
Reset:	soft
Address:	7154Ch-7154Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_4_B
Reset:	soft
Address:	71550h-71553h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_4_B
Reset:	soft
Address:	71554h-71557h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_4_B
Reset:	soft
Address:	71558h-7155Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_4_B
Reset:	soft
Address:	7155Ch-7155Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_4_B
Reset:	soft
Address:	71568h-7156Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_4_B
Reset:	soft
Address:	71640h-71643h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_5_B
Reset:	soft
Address:	71644h-71647h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_5_B
Reset:	soft
Address:	71648h-7164Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_5_B
Reset:	soft
Address:	7164Ch-7164Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_5_B
Reset:	soft
Address:	71650h-71653h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_5_B
Reset:	soft
Address:	71654h-71657h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_5_B
Reset:	soft
Address:	71658h-7165Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_5_B
Reset:	soft
Address:	7165Ch-7165Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_5_B
Reset:	soft
Address:	71668h-7166Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_5_B
Reset:	soft
Address:	71740h-71743h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_6_B
Reset:	soft
Address:	71744h-71747h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_6_B
Reset:	soft
Address:	71748h-7174Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_6_B
Reset:	soft
Address:	7174Ch-7174Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_6_B
Reset:	soft
Address:	71750h-71753h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_6_B
Reset:	soft
Address:	71754h-71757h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_6_B
Reset:	soft
Address:	71758h-7175Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_6_B
Reset:	soft
Address:	7175Ch-7175Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_6_B
Reset:	soft
Address:	71768h-7176Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_6_B
Reset:	soft
Address:	71840h-71843h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_7_B
Reset:	soft
Address:	71844h-71847h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_7_B
Reset:	soft
Address:	71848h-7184Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_7_B
Reset:	soft
Address:	7184Ch-7184Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_7_B
Reset:	soft
Address:	71850h-71853h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_7_B
Reset:	soft
Address:	71854h-71857h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_7_B
Reset:	soft
Address:	71858h-7185Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_7_B
Reset:	soft
Address:	7185Ch-7185Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_7_B
Reset:	soft
Address:	71868h-7186Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_7_B
Reset:	soft
Address:	72540h-72543h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_4_C
Reset:	soft
Address:	72544h-72547h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_4_C
Reset:	soft
Address:	72548h-7254Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_4_C
Reset:	soft
Address:	7254Ch-7254Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_4_C
Reset:	soft
Address:	72550h-72553h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_4_C
Reset:	soft
Address:	72554h-72557h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_4_C
Reset:	soft
Address:	72558h-7255Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_4_C
Reset:	soft
Address:	7255Ch-7255Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_4_C
Reset:	soft
Address:	72568h-7256Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_4_C
Reset:	soft
Address:	72640h-72643h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_5_C
Reset:	soft
Address:	72644h-72647h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_5_C
Reset:	soft
Address:	72648h-7264Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_5_C
Reset:	soft
Address:	7264Ch-7264Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_5_C
Reset:	soft
Address:	72650h-72653h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_5_C
Reset:	soft
Address:	72654h-72657h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_5_C
Reset:	soft
Address:	72658h-7265Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_5_C
Reset:	soft
Address:	7265Ch-7265Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_5_C
Reset:	soft
Address:	72668h-7266Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_5_C
Reset:	soft
Address:	72740h-72743h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_6_C
Reset:	soft
Address:	72744h-72747h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_6_C
Reset:	soft
Address:	72748h-7274Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_6_C
Reset:	soft
Address:	7274Ch-7274Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_6_C
Reset:	soft
Address:	72750h-72753h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_6_C
Reset:	soft
Address:	72754h-72757h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_6_C
Reset:	soft
Address:	72758h-7275Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_6_C
Reset:	soft
Address:	7275Ch-7275Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_6_C
Reset:	soft
Address:	72768h-7276Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_6_C
Reset:	soft
Address:	72840h-72843h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_7_C
Reset:	soft
Address:	72844h-72847h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_7_C
Reset:	soft
Address:	72848h-7284Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_7_C
Reset:	soft
Address:	7284Ch-7284Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_7_C
Reset:	soft
Address:	72850h-72853h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_7_C
Reset:	soft
Address:	72854h-72857h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_7_C
Reset:	soft
Address:	72858h-7285Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_7_C
Reset:	soft
Address:	7285Ch-7285Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_7_C
Reset:	soft
Address:	72868h-7286Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_7_C
Reset:	soft
Address:	73540h-73543h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_4_D
Reset:	soft
Address:	73544h-73547h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_4_D
Reset:	soft
Address:	73548h-7354Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_4_D
Reset:	soft
Address:	7354Ch-7354Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_4_D
Reset:	soft
Address:	73550h-73553h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_4_D
Reset:	soft
Address:	73554h-73557h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_4_D
Reset:	soft
Address:	73558h-7355Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_4_D
Reset:	soft
Address:	7355Ch-7355Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_4_D
Reset:	soft
Address:	73568h-7356Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_4_D
Reset:	soft
Address:	73640h-73643h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_5_D
Reset:	soft
Address:	73644h-73647h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_5_D
Reset:	soft
Address:	73648h-7364Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_5_D
Reset:	soft
Address:	7364Ch-7364Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_5_D
Reset:	soft
Address:	73650h-73653h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_5_D
Reset:	soft
Address:	73654h-73657h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_5_D
Reset:	soft
Address:	73658h-7365Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_5_D
Reset:	soft
Address:	7365Ch-7365Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_5_D
Reset:	soft
Address:	73668h-7366Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_5_D
Reset:	soft
Address:	73740h-73743h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_6_D
Reset:	soft
Address:	73744h-73747h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_6_D
Reset:	soft
Address:	73748h-7374Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_6_D
Reset:	soft
Address:	7374Ch-7374Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_6_D
Reset:	soft
Address:	73750h-73753h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_6_D
Reset:	soft
Address:	73754h-73757h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_6_D
Reset:	soft
Address:	73758h-7375Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_6_D
Reset:	soft
Address:	7375Ch-7375Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_6_D
Reset:	soft
Address:	73768h-7376Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_6_D
Reset:	soft
Address:	73840h-73843h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_7_D
Reset:	soft
Address:	73844h-73847h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_7_D
Reset:	soft
Address:	73848h-7384Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_7_D
Reset:	soft
Address:	7384Ch-7384Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_7_D
Reset:	soft
Address:	73850h-73853h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_7_D
Reset:	soft
Address:	73854h-73857h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_7_D
Reset:	soft
Address:	73858h-7385Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_7_D
Reset:	soft
Address:	7385Ch-7385Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_7_D
Reset:	soft
Address:	73868h-7386Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_7_D
Reset:	soft
Address:	70240h-70243h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_1_A
Reset:	soft
Address:	70244h-70247h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_1_A
Reset:	soft
Address:	70248h-7024Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_1_A
Reset:	soft
Address:	7024Ch-7024Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_1_A
Reset:	soft
Address:	70250h-70253h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_1_A
Reset:	soft
Address:	70254h-70257h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_1_A
Reset:	soft
Address:	70258h-7025Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_1_A
Reset:	soft
Address:	7025Ch-7025Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_1_A
Reset:	soft
Address:	70268h-7026Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_1_A
Reset:	soft
Address:	70340h-70343h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_2_A
Reset:	soft
Address:	70344h-70347h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_2_A
Reset:	soft
Address:	70348h-7034Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_2_A
Reset:	soft
Address:	7034Ch-7034Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_2_A
Reset:	soft
Address:	70350h-70353h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_2_A
Reset:	soft
Address:	70354h-70357h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_2_A
Reset:	soft
Address:	70358h-7035Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_2_A
Reset:	soft
Address:	7035Ch-7035Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_2_A
Reset:	soft
Address:	70368h-7036Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_2_A
Reset:	soft
Address:	70440h-70443h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_3_A
Reset:	soft
Address:	70444h-70447h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_3_A
Reset:	soft
Address:	70448h-7044Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_3_A
Reset:	soft
Address:	7044Ch-7044Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_3_A
Reset:	soft
Address:	70450h-70453h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_3_A
Reset:	soft
Address:	70454h-70457h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_3_A
Reset:	soft
Address:	70458h-7045Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_3_A
Reset:	soft
Address:	7045Ch-7045Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_3_A
Reset:	soft
Address:	70468h-7046Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_3_A
Reset:	soft
Address:	71240h-71243h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_1_B
Reset:	soft
Address:	71244h-71247h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_1_B
Reset:	soft
Address:	71248h-7124Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_1_B
Reset:	soft
Address:	7124Ch-7124Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_1_B
Reset:	soft
Address:	71250h-71253h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_1_B
Reset:	soft
Address:	71254h-71257h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_1_B
Reset:	soft
Address:	71258h-7125Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_1_B
Reset:	soft
Address:	7125Ch-7125Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_1_B
Reset:	soft
Address:	71268h-7126Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_1_B
Reset:	soft
Address:	71340h-71343h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_2_B
Reset:	soft
Address:	71344h-71347h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_2_B
Reset:	soft
Address:	71348h-7134Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_2_B
Reset:	soft
Address:	7134Ch-7134Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_2_B
Reset:	soft
Address:	71350h-71353h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_2_B
Reset:	soft
Address:	71354h-71357h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_2_B
Reset:	soft
Address:	71358h-7135Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_2_B
Reset:	soft
Address:	7135Ch-7135Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_2_B
Reset:	soft
Address:	71368h-7136Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_2_B
Reset:	soft
Address:	71440h-71443h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_3_B
Reset:	soft
Address:	71444h-71447h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_3_B
Reset:	soft
Address:	71448h-7144Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_3_B
Reset:	soft
Address:	7144Ch-7144Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_3_B
Reset:	soft
Address:	71450h-71453h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_3_B
Reset:	soft
Address:	71454h-71457h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_3_B
Reset:	soft
Address:	71458h-7145Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_3_B
Reset:	soft
Address:	7145Ch-7145Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_3_B
Reset:	soft
Address:	71468h-7146Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_3_B
Reset:	soft
Address:	72240h-72243h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_1_C
Reset:	soft
Address:	72244h-72247h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_1_C
Reset:	soft
Address:	72248h-7224Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_1_C
Reset:	soft
Address:	7224Ch-7224Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_1_C
Reset:	soft
Address:	72250h-72253h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_1_C
Reset:	soft
Address:	72254h-72257h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_1_C
Reset:	soft
Address:	72258h-7225Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_1_C
Reset:	soft
Address:	7225Ch-7225Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_1_C
Reset:	soft
Address:	72268h-7226Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_1_C
Reset:	soft
Address:	72340h-72343h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_2_C
Reset:	soft
Address:	72344h-72347h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_2_C
Reset:	soft
Address:	72348h-7234Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_2_C
Reset:	soft
Address:	7234Ch-7234Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_2_C
Reset:	soft
Address:	72350h-72353h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_2_C
Reset:	soft
Address:	72354h-72357h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_2_C
Reset:	soft
Address:	72358h-7235Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_2_C
Reset:	soft
Address:	7235Ch-7235Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_2_C
Reset:	soft
Address:	72368h-7236Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_2_C
Reset:	soft
Address:	72440h-72443h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_3_C
Reset:	soft
Address:	72444h-72447h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_3_C
Reset:	soft
Address:	72448h-7244Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_3_C
Reset:	soft
Address:	7244Ch-7244Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_3_C
Reset:	soft
Address:	72450h-72453h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_3_C
Reset:	soft
Address:	72454h-72457h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_3_C
Reset:	soft
Address:	72458h-7245Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_3_C
Reset:	soft
Address:	7245Ch-7245Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_3_C
Reset:	soft
Address:	72468h-7246Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_3_C
Reset:	soft
Address:	73240h-73243h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_1_D
Reset:	soft
Address:	73244h-73247h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_1_D
Reset:	soft
Address:	73248h-7324Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_1_D
Reset:	soft
Address:	7324Ch-7324Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_1_D
Reset:	soft
Address:	73250h-73253h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_1_D
Reset:	soft
Address:	73254h-73257h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_1_D
Reset:	soft
Address:	73258h-7325Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_1_D
Reset:	soft
Address:	7325Ch-7325Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_1_D
Reset:	soft
Address:	73268h-7326Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_1_D
Reset:	soft
Address:	73340h-73343h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_2_D
Reset:	soft
Address:	73344h-73347h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_2_D
Reset:	soft
Address:	73348h-7334Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_2_D
Reset:	soft
Address:	7334Ch-7334Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_2_D
Reset:	soft
Address:	73350h-73353h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_2_D
Reset:	soft
Address:	73354h-73357h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_2_D
Reset:	soft
Address:	73358h-7335Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_2_D
Reset:	soft
Address:	7335Ch-7335Fh

PLANE_WM	
Name:	Plane Watermarks
ShortName:	PLANE_WM_7_2_D
Reset:	soft
Address:	73368h-7336Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_2_D
Reset:	soft
Address:	73440h-73443h
Name:	Plane Watermarks
ShortName:	PLANE_WM_0_3_D
Reset:	soft
Address:	73444h-73447h
Name:	Plane Watermarks
ShortName:	PLANE_WM_1_3_D
Reset:	soft
Address:	73448h-7344Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_2_3_D
Reset:	soft
Address:	7344Ch-7344Fh
Name:	Plane Watermarks
ShortName:	PLANE_WM_3_3_D
Reset:	soft
Address:	73450h-73453h
Name:	Plane Watermarks
ShortName:	PLANE_WM_4_3_D
Reset:	soft
Address:	73454h-73457h
Name:	Plane Watermarks
ShortName:	PLANE_WM_5_3_D
Reset:	soft
Address:	73458h-7345Bh
Name:	Plane Watermarks
ShortName:	PLANE_WM_6_3_D
Reset:	soft
Address:	7345Ch-7345Fh

PLANE_WM

Name:	Plane Watermarks
ShortName:	PLANE_WM_7_3_D
Reset:	soft

Address:	73468h-7346Bh
Name:	Plane Transition Watermarks
ShortName:	PLANE_WM_TRANS_3_D
Reset:	soft

Programming Notes

There are eight regular watermarks and a transition watermark per plane/cursor. For YUV planar source formats, only the Y surface watermark value should be programmed. Watermark programming instructions are documented separately.

Restriction

For minimum watermark requirements refer to Display Watermark Programming section.

DWord	Bit	Description
0	31	**Enable** This field enables this watermark. All the watermarks at this level for all enabled planes must be enabled before the level will be used. Value: 1b — Name: Enable Value: 0b — Name: Disable
	30	**Reserved**
	29:27	**Reserved** Access: RO Format: MBZ
	26:19	**Reserved** Access: RO Format: MBZ
	18:14	**Lines** Default Value: 01h This field contains the watermark value in lines. Hardware ignores the lines for the transition watermark.
	13:12	**Reserved** Access: RO Format: MBZ

PLANE_WM		
	11	**Reserved** Access: RO Format: MBZ
	10:0	**Blocks** Default Value: 007h This field contains the watermark value in blocks of 8 cachelines.

POISON DATA HANDLING ENABLE

POISON_DATA_HANDLING_ENABLE - POISON DATA HANDLING ENABLE

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	100120h

This register holds the enable for Poison data detection and corking behavior for Gunit.

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**ERROR CONTROL** Access: R/W _Custom_GTIReset: BUS Enable Poison Data detection and Corking behavior for Gunit Value / Name 0b **[Default]** 1b

POISON DATA STATUS

POISON_DATA_STATUS - POISON DATA STATUS

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	100124h

This register holds the sticky bit which when set will indicate a Poisoned data has been received on IOSF-Primary.

DWord	Bit	Description
0	31:9	**Reserved** Access: RO Format: MBZ
	8	**DEDICATED PATH ERROR STATUS** Access: R/W One Clear _Custom_GTIReset: DEV Set to 1 when Poisoned data has been received from IOSF-Primary on Dedicated memory path. This sticky bit must survive Platform/SoC level reset. 0b **[Default]** 1b
	7:1	**Reserved** Access: RO Format: MBZ
	0	**ERROR STATUS** Access: R/W One Clear _Custom_GTIReset: DEV Set to 1 when Poisoned data has been received from IOSF-Primary. This sticky bit must survive Platform/SoC level reset. 0b **[Default]** 1b

PORT_CL_DW5

PORT_CL_DW5		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	162014h-162017h	
Name:	PORT_CL_DW5	
ShortName:	PORT_CL_DW5_A	
Reset:	global	
Address:	6C014h-6C017h	
Name:	PORT_CL_DW5	
ShortName:	PORT_CL_DW5_B	
Reset:	global	
Address:	160014h-160017h	
Name:	PORT_CL_DW5	
ShortName:	PORT_CL_DW5_C	
Reset:	global	

DWord	Bit	Description
0	31:24	**Force** Default Value: 00010010b
	23	**Reserved** Access: RO Format: MBZ
	22	**Fusevalid Reset**
	21	**Fusevalid Override**
	20	**Fuse Repull**
	19:16	**CRI Clock Count Max** Default Value: 0100b
	15	**Reserved** Access: RO Format: MBZ
	14:13	**IOSF PD Count**
	12	**Reserved** Access: RO Format: MBZ

PORT_CL_DW5

	11:9	**IOSF ClkDiv Sel** Default Value: 010b
	8	**DL Broadcast Enable** This field causes all Tx's to get programmed when writing to a group access offset for a single Tx.
	7	**Reserved** Access: RO Format: MBZ
	6	**Enable Port Staggering** Default Value: 1b
	5	**PG Staggering Control Disable** Default Value: 1b
	4	**CL Power Down Enable** Value: 0b — Name: Disable Value: 1b — Name: Enable
	3	**CRI Clock Select** Default Value: 1b
	2	**Phy Power Ack Override**
	1:0	**SUS Clock Config**

PORT_CL_DW10

PORT_CL_DW10	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	162028h-16202Bh
Name:	PORT_CL_DW10
ShortName:	PORT_CL_DW10_A
Reset:	global
Address:	6C028h-6C02Bh
Name:	PORT_CL_DW10
ShortName:	PORT_CL_DW10_B
Reset:	global
Address:	160028h-16002Bh
Name:	PORT_CL_DW10
ShortName:	PORT_CL_DW10_C
Reset:	global

DWord	Bit	Description
0	31:27	**Reserved** Access: RO Format: MBZ
	26:25	**PG Seq Delay Override** PG Sequential Delay Override
	24	**PG Seq Delay Override Enable** PG Sequential Delay Override Enable Value: 0b — Name: Disable Value: 1b — Name: Enable
	23	**ohvpg_ctrl_mipia** MIPI A HVPG Control
	22	**spare 22** Spare 22
	21:16	**ospare_cri_ret** Default Value: 000011b ospare_cri_ret[6:0]

PORT_CL_DW10

15:12 Reserved

Access:	RO
Format:	MBZ

11 Spare 11
Spare 11

10 Spare 10
Spare 10

9 Spare 9
Spare 9

8 Spare 8
Spare 8

7:4 Static Power Down
This field powers down individual lanes for the DDI that is accessed through this instance of the register. To save power, unused lanes should be powered down after link training is complete.

Value	Name	Description
0000b	Power up all lanes	Enable x4
1100b	Power down lanes 3,2	Enable x2
1110b	Power down lanes 3,2,1	Enable x1
0011b	Power down lanes 1,0	Enable x2 Reversed
0111b	Power down lanes 2,1,0	Enable x1 Reversed
1011b	Power down lanes 3,1,0	Enable DSI x1
1010b	Power down lanes 3,1	Enable DSI x2
1000b	Power down lane 3	Enable DSI x3

3 o_edp4k2k_mode_ovrd_en
eDP power optimized setting

Value	Name
1b	Enable **[Default]**
0b	Disable

2 o_edp4k2k_mode_ovrd_val
eDP power optimized setting. Valid only when corresponding enable bit (o_edp4k2k_mode_ovrd_en) is set and specific voltage swing programming is used.

Value	Name
1b	Optimized
0b	Non-optimized

1 o_rterm100en_h_ovrd_en
rterm override enable

Value	Name
1b	**[Default]**

PORT_CL_DW10			
		0b	
	0	**o_rterm100en_h_ovrd_val** rterm override val valid only when corresponding enable bit (o_rterm100en_h_ovrd_en) is set.	
		Value	**Name**
		0b	150 Ohms
		1b	100 Ohms

PORT_CL_DW12

PORT_CL_DW12

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	162030h-162033h
Name:	PORT_CL_DW12
ShortName:	PORT_CL_DW12_A
Reset:	global
Address:	6C030h-6C033h
Name:	PORT_CL_DW12
ShortName:	PORT_CL_DW12_B
Reset:	global
Address:	160030h-160033h
Name:	PORT_CL_DW12
ShortName:	PORT_CL_DW12_C
Reset:	global

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29	**MIPI Lane Enable** 0b – Disable 1b – Enable
	28	**Reserved** Access: RO Format: MBZ
	27	**MIPI Mode Override Enable** 0b 1b
	26	**MIPI Mode Override** 0b 1b

PORT_CL_DW12

	25:18	**Reserved** Access: RO Format: MBZ
	17:16	**Reserved** Access: RO Format: MBZ
	15:12	**Reserved** Access: RO Format: MBZ
	11	**Pwr Req Override AUX** Value: 0b — Name: Value: 1b — Name:
	10	**Pwr Req Override Enable AUX** Value: 0b — Name: Disable Value: 1b — Name: Enable
	9:7	**Reserved** Access: RO Format: MBZ
	6	**Phy Status AUX** Access: RO
	5	**Reserved** Access: RO Format: MBZ
	4	**Power Ack AUX** Access: RO
	3:1	**Reserved** Access: RO Format: MBZ
	0	**Lane Enable AUX** Value: 0b — Name: Disable Value: 1b — Name: Enable

PORT_CL_DW15

PORT_CL_DW15

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	16203Ch-16203Fh
Name:	PORT_CL_DW15
ShortName:	PORT_CL_DW15_A
Reset:	global
Address:	6C03Ch-6C03Fh
Name:	PORT_CL_DW15
ShortName:	PORT_CL_DW15_B
Reset:	global
Address:	16003Ch-16003Fh
Name:	PORT_CL_DW15
ShortName:	PORT_CL_DW15_C
Reset:	global

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29	**HVPG Power Ack** Access: RO Value / Name 0b 1b
	28	**HVPG Enable Status** Access: RO Value / Name 0b 1b
	27	**Power Ack MIPI** Access: RO Value / Name

PORT_CL_DW15

	0b	
	1b	

26:22 Reserved

Access:	RO
Format:	MBZ

21 Power Req AUX

Access:	RO

Value	Name
0b	
1b	

20:18 Reserved

Access:	RO
Format:	MBZ

17 Power Ack AUX

Access:	RO

Value	Name
0b	
1b	

16:0 Reserved

Access:	RO
Format:	MBZ

PORT_CL_DW16

PORT_CL_DW16	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	162040h-162043h
Name:	PORT_CL_DW16
ShortName:	PORT_CL_DW16_A
Reset:	global
Address:	6C040h-6C043h
Name:	PORT_CL_DW16
ShortName:	PORT_CL_DW16_B
Reset:	global
Address:	160040h-160043h
Name:	PORT_CL_DW16
ShortName:	PORT_CL_DW16_C
Reset:	global

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15	**o_hd_ddib_sel_ovrden** DDI B HD Port select override enable
	14	**o_hd_ddib_sel_ovrd** DDI B HD Port select override
	13	**o_hd_ddic_sel_ovrden** DDI C HD Port select override enable
	12	**o_hd_ddic_sel_ovrd** DDI C HD Port select override
	11	**o_hd_ddid_sel_ovrden** DDI D HD Port select override enable
	10	**o_hd_ddid_sel_ovrd** DDI D HD Port select override
	9:8	**Reserved** Access: RO Format: MBZ
	7:4	**ospare_cri[3:0]** reserved

PORT_CL_DW16		
	3	**o_comp_pwrdown_ovrd** Comp Power Down Override
	2	**o_comp_pwrdown_ovrden** Comp Power Down Override Enable
	1	**o_cri_wake_ovrd** CRI Wake Override
	0	**o_cri_wake_ovrden** CRI Wake Override Enable

PORT_COMP_DW0

PORT_COMP_DW0		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	162100h-162103h	
Name:	PORT_COMP_DW0	
ShortName:	PORT_COMP_DW0_A	
Reset:	global	
Address:	6C100h-6C103h	
Name:	PORT_COMP_DW0	
ShortName:	PORT_COMP_DW0_B	
Reset:	global	
Address:	160100h-160103h	
Name:	PORT_COMP_DW0	
ShortName:	PORT_COMP_DW0_C	
Reset:	global	

DWord	Bit	Description
0	31	**Comp Init**
	30:29	**Tx Slew Ctl**
	28:27	**Tx Drvsw On**
	26	**Tx Drvsw Ctl**
	25:24	**Comp Spare**
	23	**Procmon clock Sel**
	22:20	**Reserved** Access: RO Format: MBZ
	19:8	**Periodic Comp Counter** Periodic comp programmable counter. Value: 05Fh — Name: 1.25ms **[Default]** — Description: Period of ~ 1.25ms w.r.t sus clock frequency of 19.2MHz.
	7:0	**Reserved** Access: RO Format: MBZ

PORT_COMP_DW1

PORT_COMP_DW1	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	162104h-162107h
Name:	PORT_COMP_DW1
ShortName:	PORT_COMP_DW1_A
Reset:	global
Address:	6C104h-6C107h
Name:	PORT_COMP_DW1
ShortName:	PORT_COMP_DW1_B
Reset:	global
Address:	160104h-160107h
Name:	PORT_COMP_DW1
ShortName:	PORT_COMP_DW1_C
Reset:	global

DWord	Bit	Description
0	31	**ldo_bypass** Default Value: 1b
	30	**fcomp_ovrd_en**
	29	**fcomp_capratio**
	28	**fcomp_bias_sel**
	27:26	**fcomp_inputsel_ovrd**
	25	**fcomp_polaritysel**
	24	**rcomp_en** Default Value: 1b
	23:22	**p_ref_highval[9:8]** Default Value: 01b
	21:20	**p_ref_lowval[9:8]**
	19:18	**n_ref_highval[9:8]** Default Value: 01b
	17:16	**n_ref_lowval[9:8]**
	15:14	**phvt_ref_highval[9:8]**
	13:12	**phvt_ref_lowval[9:8]**

PORT_COMP_DW1		
	11:10	**nhvt_ref_highval[9:8]** Default Value: 01h
	9:8	**nhvt_ref_lowval[9:8]**
	7:6	**plvt_ref_highval[9:8]** Default Value: 01b
	5:4	**plvt_ref_lowval[9:8]**
	3:2	**nlvt_ref_highval[9:8]** Default Value: 01b
	1:0	**nlvt_ref_lowval[9:8]**

PORT_COMP_DW3

PORT_COMP_DW3

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32

Address:	16210Ch-16210Fh
Name:	PORT_COMP_DW3
ShortName:	PORT_COMP_DW3_A
Reset:	global

Address:	6C10Ch-6C10Fh
Name:	PORT_COMP_DW3
ShortName:	PORT_COMP_DW3_B
Reset:	global

Address:	16010Ch-16010Fh
Name:	PORT_COMP_DW3
ShortName:	PORT_COMP_DW3_C
Reset:	global

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:26	**Process Info** 000b: dot-0 001b: dot-1 010b: dot-4
	25:24	**Voltage Info** 00b: 0.85V 01b: 0.95V 10b: 1.05
	23	**PLL DDI Pwr Ack**
	22	**First Comp Done**
	21	**Procmon Done**
	20	**Icomp Code Maxout**
	19	**Icomp Code Minout**

PORT_COMP_DW3		
	18:15	**Reserved** Access: RO Format: MBZ
	14:8	**Icomp Code**
	7	**Lpdn Code Maxout**
	6	**Lpdn Code Minout**
	5:0	**MIPI Lpdn Code**

PORT_COMP_DW8

PORT_COMP_DW8

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	162120h-162123h
Name:	PORT_COMP_DW8
ShortName:	PORT_COMP_DW8_A
Reset:	global
Address:	6C120h-6C123h
Name:	PORT_COMP_DW8
ShortName:	PORT_COMP_DW8_B
Reset:	global
Address:	160120h-160123h
Name:	PORT_COMP_DW8
ShortName:	PORT_COMP_DW8_C
Reset:	global

DWord	Bit	Description
0	31:25	**Reserved** Access: RO Format: MBZ
	24	**irefgen** 1b – Enable 0b – Disable
	23:15	**Reserved** Access: RO Format: MBZ
	14	**prdic_icomp_dis** disable periodic icomp 0b – Enable 1b – Disable
	13:0	**Reserved** Access: RO Format: MBZ

PORT_COMP_DW9

PORT_COMP_DW9		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	162124h-162127h	
Name:	PORT_COMP_DW9	
ShortName:	PORT_COMP_DW9_A	
Reset:	global	
Address:	6C124h-6C127h	
Name:	PORT_COMP_DW9	
ShortName:	PORT_COMP_DW9_B	
Reset:	global	
Address:	160124h-160127h	
Name:	PORT_COMP_DW9	
ShortName:	PORT_COMP_DW9_C	
Reset:	global	

DWord	Bit	Description	
0	31:24	**n_ref_lowval[7:0]**	
		Default Value:	11011010b
	23:16	**n_ref_highval[7:0]**	
		Default Value:	10001100b
	15:8	**p_ref_lowval[7:0]**	
		Default Value:	11011100b
	7:0	**p_ref_highval[7:0]**	
		Default Value:	10100101b

PORT_COMP_DW10

PORT_COMP_DW10		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	162128h-16212Bh	
Name:	PORT_COMP_DW10	
ShortName:	PORT_COMP_DW10_A	
Reset:	global	
Address:	6C128h-6C12Bh	
Name:	PORT_COMP_DW10	
ShortName:	PORT_COMP_DW10_B	
Reset:	global	
Address:	160128h-16012Bh	
Name:	PORT_COMP_DW10	
ShortName:	PORT_COMP_DW10_C	
Reset:	global	

DWord	Bit	Description	
0	31:24	**nlvt_ref_lowval[7:0]**	
		Default Value:	10101010b
	23:16	**nlvt_ref_highval[7:0]**	
		Default Value:	00111101b
	15:8	**plvt_ref_lowval[7:0]**	
		Default Value:	10101000b
	7:0	**plvt_ref_highval[7:0]**	
		Default Value:	01010011b

PORT_PCS_DW1

PORT_PCS_DW1	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	162304h-162307h
Name:	PORT_PCS_DW1
ShortName:	PORT_PCS_DW1_AUX_A
Reset:	global
Address:	162604h-162607h
Name:	PORT_PCS_DW1
ShortName:	PORT_PCS_DW1_GRP_A
Reset:	global
Address:	162804h-162807h
Name:	PORT_PCS_DW1
ShortName:	PORT_PCS_DW1_LN0_A
Reset:	global
Address:	162904h-162907h
Name:	PORT_PCS_DW1
ShortName:	PORT_PCS_DW1_LN1_A
Reset:	global
Address:	162A04h-162A07h
Name:	PORT_PCS_DW1
ShortName:	PORT_PCS_DW1_LN2_A
Reset:	global
Address:	162B04h-162B07h
Name:	PORT_PCS_DW1
ShortName:	PORT_PCS_DW1_LN3_A
Reset:	global
Address:	6C304h-6C307h
Name:	PORT_PCS_DW1
ShortName:	PORT_PCS_DW1_AUX_B
Reset:	global
Address:	6C604h-6C607h
Name:	PORT_PCS_DW1
ShortName:	PORT_PCS_DW1_GRP_B

PORT_PCS_DW1	
Reset:	global
Address:	6C804h-6C807h
Name:	PORT_PCS_DW1
ShortName:	PORT_PCS_DW1_LN0_B
Reset:	global
Address:	6C904h-6C907h
Name:	PORT_PCS_DW1
ShortName:	PORT_PCS_DW1_LN1_B
Reset:	global
Address:	6CA04h-6CA07h
Name:	PORT_PCS_DW1
ShortName:	PORT_PCS_DW1_LN2_B
Reset:	global
Address:	6CB04h-6CB07h
Name:	PORT_PCS_DW1
ShortName:	PORT_PCS_DW1_LN3_B
Reset:	global
Address:	160304h-160307h
Name:	PORT_PCS_DW1
ShortName:	PORT_PCS_DW1_AUX_C
Reset:	global
Address:	160604h-160607h
Name:	PORT_PCS_DW1
ShortName:	PORT_PCS_DW1_GRP_C
Reset:	global
Address:	160804h-160807h
Name:	PORT_PCS_DW1
ShortName:	PORT_PCS_DW1_LN0_C
Reset:	global
Address:	160904h-160907h
Name:	PORT_PCS_DW1
ShortName:	PORT_PCS_DW1_LN1_C
Reset:	global
Address:	160A04h-160A07h
Name:	PORT_PCS_DW1
ShortName:	PORT_PCS_DW1_LN2_C

PORT_PCS_DW1

Reset:	global

Address:	160B04h-160B07h
Name:	PORT_PCS_DW1
ShortName:	PORT_PCS_DW1_LN3_C
Reset:	global

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28	**cmnkeeper_enable_in_pg** 1b: enable **[Default]** 0b: disable
	27	**pg_pwrdownen** PG Power Down Enable 1b: enable **[Default]** 0b: disable
	26	**cmnkeeper_enable** Common Keeper Enable 1b: enable **[Default]** 0b: disable
	25:24	**cmnkeep_biasctr** Common Keeper Bias Control
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**DCC Mode Select** 00b: Run DCC once **[Default]** 01b: Run DCC every 100us 10b: Run DCC every 1ms 11b: Run DCC continuosly
	19	**reg_dcc_bypass** Default Value: 0b Setting this bit will bypass the DCC calibration and will also bypass the DFx RX calibration since

PORT_PCS_DW1

		both are triggered using same signal in PCS
	18	**reg_dcc_calib_wake_en** Default Value: 0b Asserting this bit will run DCC again during DL wake up from powerdown
	17	**tx_dcc_calib_enable** Default Value: 0b Force Tx DCC Calibration Enable (Should be used only after initial boot is done)
	16	**Reserved**
	15:14	**Reserved** Access: RO Format: MBZ
	13:12	**txhigh**
	11:10	**Reserved** Access: RO Format: MBZ
	9:8	**clkreq**
	7	**tbc_as_symbclk** Select tbc clock to be used as symbol clock internal to the data lane
	6	**txfifo_rst_main_ovrden** Override enable for Tx main resets
	5	**txfifo_rst_main_ovrd** Reset Main Override for Tx
	4	**txdeemp** Deemphasis Value
	3:2	**latencyoptim** Default Value: 01b Latency Optim
	1	**softreset_enable** Allow soft_reset_n to reset the lanes Value: 1b — Name: enable Value: 0b — Name: disable
	0	**soft_reset_n** Default Value: 1b Active low soft reset override

PORT_PCS_DW9

PORT_PCS_DW9	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	162324h-162327h
Name:	PORT_PCS_DW9
ShortName:	PORT_PCS_DW9_AUX_A
Reset:	global
Address:	162624h-162627h
Name:	PORT_PCS_DW9
ShortName:	PORT_PCS_DW9_GRP_A
Reset:	global
Address:	162824h-162827h
Name:	PORT_PCS_DW9
ShortName:	PORT_PCS_DW9_LN0_A
Reset:	global
Address:	162924h-162927h
Name:	PORT_PCS_DW9
ShortName:	PORT_PCS_DW9_LN1_A
Reset:	global
Address:	162A24h-162A27h
Name:	PORT_PCS_DW9
ShortName:	PORT_PCS_DW9_LN2_A
Reset:	global
Address:	162B24h-162B27h
Name:	PORT_PCS_DW9
ShortName:	PORT_PCS_DW9_LN3_A
Reset:	global
Address:	6C324h-6C327h
Name:	PORT_PCS_DW9
ShortName:	PORT_PCS_DW9_AUX_B
Reset:	global
Address:	6C624h-6C627h
Name:	PORT_PCS_DW9
ShortName:	PORT_PCS_DW9_GRP_B

PORT_PCS_DW9	
Reset:	global
Address:	6C824h-6C827h
Name:	PORT_PCS_DW9
ShortName:	PORT_PCS_DW9_LN0_B
Reset:	global
Address:	6C924h-6C927h
Name:	PORT_PCS_DW9
ShortName:	PORT_PCS_DW9_LN1_B
Reset:	global
Address:	6CA24h-6CA27h
Name:	PORT_PCS_DW9
ShortName:	PORT_PCS_DW9_LN2_B
Reset:	global
Address:	6CB24h-6CB27h
Name:	PORT_PCS_DW9
ShortName:	PORT_PCS_DW9_LN3_B
Reset:	global
Address:	160324h-160327h
Name:	PORT_PCS_DW9
ShortName:	PORT_PCS_DW9_AUX_C
Reset:	global
Address:	160624h-160627h
Name:	PORT_PCS_DW9
ShortName:	PORT_PCS_DW9_GRP_C
Reset:	global
Address:	160824h-160827h
Name:	PORT_PCS_DW9
ShortName:	PORT_PCS_DW9_LN0_C
Reset:	global
Address:	160924h-160927h
Name:	PORT_PCS_DW9
ShortName:	PORT_PCS_DW9_LN1_C
Reset:	global
Address:	160A24h-160A27h
Name:	PORT_PCS_DW9
ShortName:	PORT_PCS_DW9_LN2_C

PORT_PCS_DW9

Reset:	global

Address:	160B24h-160B27h
Name:	PORT_PCS_DW9
ShortName:	PORT_PCS_DW9_LN3_C
Reset:	global

DWord	Bit	Description
0	31:28	**Reserved** Access: RO Format: MBZ
	27:16	**Strong CM Count Ovrd** Default Value: 301h
	15:11	**Reserved** Access: RO Format: MBZ
	10:8	**Stagger Mult** Default Value: 001b
	7:6	**Reserved** Access: RO Format: MBZ
	5	**Stagger Override**
	4:0	**Stagger**

PORT_TX_DFLEXDPCSSS

PORT_TX_DFLEXDPCSSS		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	163894h-163897h	
Name:	PORT_TX_DFLEXDPCSSS	
ShortName:	PORT_TX_DFLEXDPCSSS_FIA1	
Reset:	global	
Address:	16E894h-16E897h	
Name:	PORT_TX_DFLEXDPCSSS	
ShortName:	PORT_TX_DFLEXDPCSSS_FIA2	
Reset:	global	
Address:	16F894h-16F897h	
Name:	PORT_TX_DFLEXDPCSSS	
ShortName:	PORT_TX_DFLEXDPCSSS_FIA3	
Reset:	global	

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**Displayport Phy Mode Status for Type-C Connector 7** Similar to register DFLEXDPCSSS.DPPMSTC0 but this register is for Type-C Connector 7.
	6	**Displayport Phy Mode Status for Type-C Connector 6** Similar to register DFLEXDPCSSS.DPPMSTC0 but this register is for Type-C Connector 6.
	5	**Displayport Phy Mode Status for Type-C Connector 5** Similar to register DFLEXDPCSSS.DPPMSTC0 but this register is for Type-C Connector 5.
	4	**Displayport Phy Mode Status for Type-C Connector 4** Similar to register DFLEXDPCSSS.DPPMSTC0 but this register is for Type-C Connector 4.
	3	**Displayport Phy Mode Status for Type-C Connector 3** Similar to register DFLEXDPCSSS.DPPMSTC0 but this register is for Type-C Connector 3.
	2	**Displayport Phy Mode Status for Type-C Connector 2** Similar to register DFLEXDPCSSS.DPPMSTC0 but this register is for Type-C Connector 2.
	1	**Displayport Phy Mode Status for Type-C Connector 1** Similar to register DFLEXDPCSSS.DPPMSTC0 but this register is for Type-C Connector 1.
	0	**Displayport Phy Mode Status for Type-C Connector 0** Displayport Phy Mode Status for Type-C Connector 0 (DPPMSTC0):

PORT_TX_DFLEXDPCSSS			
		Value	Name
		1b	DP Controller is not in safe state
		0b	DP controller is in safe state

PORT_TX_DFLEXDPMLE1

PORT_TX_DFLEXDPMLE1 - PORT_TX_DFLEXDPMLE1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Reset:	soft

Address:	1638C0h-1638C3h
Name:	PORT_TX_DFLEXDPMLE1
ShortName:	PORT_TX_DFLEXDPMLE1_FIA1
Reset:	soft

Address:	16E8C0h-16E8C3h
Name:	PORT_TX_DFLEXDPMLE1
ShortName:	PORT_TX_DFLEXDPMLE1_FIA2
Reset:	soft

Address:	16F8C0h-16F8C3h
Name:	PORT_TX_DFLEXDPMLE1
ShortName:	PORT_TX_DFLEXDPMLE1_FIA3
Reset:	soft

Display Driver writes to these bits to tell FIA hardware which Main Links of the Display Port are enabled on Type-C Connector 0. FIA hardware uses this information for PHY to Controller signal mapping.
For example, in DP Pin Assignment C, the register DFLEXDPSP1.DPX4TXLATC0 tells Display Driver that all the 4 TX Lanes in PHY can be used. However, Display Driver may choose to use only x1, i.e. for ML0. Then Display Driver will program 0001b to this register. For x2 and x4, Display Driver will program 0011b and 1111b, respectively.
Note that display driver should not use its internal lane reversal feature with Type-C ALT connections.
Display Driver is expected to write to this register when the DDI Interface between DP Controller and FIA is in the Safe Mode, e.g. pllen=pwrreq=lane_enable=0. Display Driver writes to this register and then only it brings up the DP Controller, i.e. to bring the DDI interface out from Safe Mode.
A mode set is required to switch the number of DP lanes.
This register is applicable in both Type-C connector's Alternate mode and also DP connector mode.

DWord	Bit	Description
0	31:28	**Displayport Main Link Enable for Type-C Connector 7** Similar to register DFLEXDPMLE1.DPMLETC0 but this register is for Type-C Connector 7.

Value	Name	Description
0001b	ML0	
0011b	ML[1:0]	
1100b	ML[3:2]	This setting should not be used with Type-C ALT connections.
1111b	ML[3:0]	

PORT_TX_DFLEXDPMLE1 - PORT_TX_DFLEXDPMLE1

27:24 **Displayport Main Link Enable for Type-C Connector 6**

Similar to register DFLEXDPMLE1.DPMLETC0 but this register is for Type-C Connector 6.

Value	Name	Description
0001b	ML0	
0011b	ML[1:0]	
1100b	ML[3:2]	This setting should not be used with Type-C ALT connections.
1111b	ML[3:0]	

23:20 **Displayport Main Link Enable for Type-C Connector 5**

Similar to register DFLEXDPMLE1.DPMLETC0 but this register is for Type-C Connector 5.

Value	Name	Description
0001b	ML0	
0011b	ML[1:0]	
1100b	ML[3:2]	This setting should not be used with Type-C ALT connections.
1111b	ML[3:0]	

19:16 **Displayport Main Link Enable for Type-C Connector 4**

Similar to register DFLEXDPMLE1.DPMLETC0 but this register is for Type-C Connector 4.

Value	Name	Description
0001b	ML0	
0011b	ML[1:0]	
1100b	ML[3:2]	This setting should not be used with Type-C ALT connections.
1111b	ML[3:0]	

15:12 **Displayport Main Link Enable for Type-C Connector 3**

Similar to register DFLEXDPMLE1.DPMLETC0 but this register is for Type-C Connector 3.

Value	Name	Description
0001b	ML0	
0011b	ML[1:0]	
1100b	ML[3:2]	This setting should not be used with Type-C ALT connections.
1111b	ML[3:0]	

11:8 **Displayport Main Link Enable for Type-C Connector 2**

Similar to register DFLEXDPMLE1.DPMLETC0 but this register is for Type-C Connector 2.

Value	Name	Description
0001b	ML0	
0011b	ML[1:0]	
1100b	ML[3:2]	This setting should not be used with Type-C ALT connections.
1111b	ML[3:0]	

7:4 **Displayport Main Link Enable for Type-C Connector 1**

Similar to register DFLEXDPMLE1.DPMLETC0 but this register is for Type-C Connector 1.

PORT_TX_DFLEXDPMLE1 - PORT_TX_DFLEXDPMLE1

Value	Name	Description
0001b	ML0	
0011b	ML[1:0]	
1100b	ML[3:2]	This setting should not be used with Type-C ALT connections.
1111b	ML[3:0]	

3:0 **Displayport Main Link Enable for Type-C Connector 0**

Display Port Main Link Enable for Type-C Connector 0 (DPMLETC0):
4 bits correspond to 4 Main Link in DP Controller. Bit [0] is ML0,bit [1] is ML1 and so on.
The Type-C Connector number is logical number. Its not physical lane numbers. Refer to the SOC block diagram for the mapping of Type-C Connector number to the actual physical lane number of the PHY.
Display Driver writes to these bits to tell FIA hardware which Main Links of the Display Port are enabled on Type-C Connector 0. FIA hardware use this information for PHY to Controller signal mapping.
For example, in DP Pin Assignment C, the register DFLEXDPSP1.DPX4TXLATC0 tells Display Driver that all the 4 TX Lane in PHY can be used. However, Display Driver may choose to use only x1, i.e. for ML0. Then Display Driver will program 0001b to this register. For x2 and x4, Display Driver will program 0011b and 1111b, respectively.

Value	Name	Description
0001b	ML0	
0011b	ML[1:0]	
1100b	ML[3:2]	This setting should not be used with Type-C ALT connections.
1111b	ML[3:0]	

PORT_TX_DFLEXDPPMS

PORT_TX_DFLEXDPPMS - PORT_TX_DFLEXDPPMS

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Reset:	soft

Address:	163890h-163893h
Name:	PORT_TX_DFLEXDPPMS
ShortName:	PORT_TX_DFLEXDPPMS_FIA1
Reset:	soft

Address:	16E890h-16E893h
Name:	PORT_TX_DFLEXDPPMS
ShortName:	PORT_TX_DFLEXDPPMS_FIA2
Reset:	soft

Address:	16F890h-16F893h
Name:	PORT_TX_DFLEXDPPMS
ShortName:	PORT_TX_DFLEXDPPMS_FIA3
Reset:	soft

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15	**Display Port PHY Mode status for Type-C connector 15** Similar to register DFLEXDPPMS.DPPMSTC0 but this register is for Type-C Connector 15.
	14	**Display Port PHY Mode status for Type-C connector 14** Similar to register DFLEXDPPMS.DPPMSTC0 but this register is for Type-C Connector 14.
	13	**Display Port PHY Mode status for Type-C connector 13** Similar to register DFLEXDPPMS.DPPMSTC0 but this register is for Type-C Connector 13.
	12	**Display Port PHY Mode status for Type-C connector 12** Similar to register DFLEXDPPMS.DPPMSTC0 but this register is for Type-C Connector 12.
	11	**Display Port PHY Mode status for Type-C connector 11** Similar to register DFLEXDPPMS.DPPMSTC0 but this register is for Type-C Connector 11.
	10	**Display Port PHY Mode status for Type-C connector 10** Similar to register DFLEXDPPMS.DPPMSTC0 but this register is for Type-C Connector 10.
	9	**Display Port PHY Mode status for Type-C connector 9** Similar to register DFLEXDPPMS.DPPMSTC0 but this register is for Type-C Connector 9.
	8	**Display Port PHY Mode status for Type-C connector 8** Similar to register DFLEXDPPMS.DPPMSTC0 but this register is for Type-C Connector 8.

PORT_TX_DFLEXDPPMS - PORT_TX_DFLEXDPPMS		
	7	**Display Port PHY Mode status for Type-C connector 7** Similar to register DFLEXDPPMS.DPPMSTC0 but this register is for Type-C Connector 7.
	6	**Display Port PHY Mode status for Type-C connector 6** Similar to register DFLEXDPPMS.DPPMSTC0 but this register is for Type-C Connector 6.
	5	**Display Port PHY Mode status for Type-C connector 5** Similar to register DFLEXDPPMS.DPPMSTC0 but this register is for Type-C Connector 5.
	4	**Display Port PHY Mode status for Type-C connector 4** Similar to register DFLEXDPPMS.DPPMSTC0 but this register is for Type-C Connector 4.
	3	**Display Port PHY Mode status for Type-C connector 3** Similar to register DFLEXDPPMS.DPPMSTC0 but this register is for Type-C Connector 3.
	2	**Display Port PHY Mode status for Type-C connector 2** Similar to register DFLEXDPPMS.DPPMSTC0 but this register is for Type-C Connector 2.
	1	**Display Port PHY Mode status for Type-C connector 1** Similar to register DFLEXDPPMS.DPPMSTC0 but this register is for Type-C Connector 1.
	0	**Display Port PHY Mode status for Type-C connector 0** DFLEXDPPMS.DPPMSTC0 PD FW writes 1 to this bit to tell DP Driver that PHY is ready. PD FW writes '0' to this bit to tell DP Driver that PHY is not ready.

Value	Name
0b	Not Completed
1b	Completed

PORT_TX_DFLEXDPSP

PORT_TX_DFLEXDPSP - PORT_TX_DFLEXDPSP	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Reset:	soft
Address:	1638A0h-1638A3h
Name:	PORT_TX_DFLEXDPSP1
ShortName:	PORT_TX_DFLEXDPSP1_FIA1
Reset:	soft
Address:	1638A4h-1638A7h
Name:	PORT_TX_DFLEXDPSP2
ShortName:	PORT_TX_DFLEXDPSP2_FIA1
Reset:	soft
Address:	1638A8h-1638ABh
Name:	PORT_TX_DFLEXDPSP3
ShortName:	PORT_TX_DFLEXDPSP3_FIA1
Reset:	soft
Address:	1638ACh-1638AFh
Name:	PORT_TX_DFLEXDPSP4
ShortName:	PORT_TX_DFLEXDPSP4_FIA1
Reset:	soft
Address:	16E8A0h-16E8A3h
Name:	PORT_TX_DFLEXDPSP1
ShortName:	PORT_TX_DFLEXDPSP1_FIA2
Reset:	soft
Address:	16E8A4h-16E8A7h
Name:	PORT_TX_DFLEXDPSP2
ShortName:	PORT_TX_DFLEXDPSP2_FIA2
Reset:	soft
Address:	16E8A8h-16E8ABh
Name:	PORT_TX_DFLEXDPSP3
ShortName:	PORT_TX_DFLEXDPSP3_FIA2
Reset:	soft
Address:	16E8ACh-16E8AFh
Name:	PORT_TX_DFLEXDPSP4

PORT_TX_DFLEXDPSP - PORT_TX_DFLEXDPSP

ShortName:	PORT_TX_DFLEXDPSP4_FIA2
Reset:	soft

Address:	16F8A0h-16F8A3h
Name:	PORT_TX_DFLEXDPSP1
ShortName:	PORT_TX_DFLEXDPSP1_FIA3
Reset:	soft

Address:	16F8A4h-16F8A7h
Name:	PORT_TX_DFLEXDPSP2
ShortName:	PORT_TX_DFLEXDPSP2_FIA3
Reset:	soft

Address:	16F8A8h-16F8ABh
Name:	PORT_TX_DFLEXDPSP3
ShortName:	PORT_TX_DFLEXDPSP3_FIA3
Reset:	soft

Address:	16F8ACh-16F8AFh
Name:	PORT_TX_DFLEXDPSP4
ShortName:	PORT_TX_DFLEXDPSP4_FIA3
Reset:	soft

Dynamic FlexIO DP Scratch Pad (Type-C)

See the TypeC Programming section for information on how the connector number here maps to the port instance.

DWord	Bit	Description
0	31:29	**TC3 Live State** 000b - No HPD - No HPD connect for TypeC (DP alternate) or TBT 001b - TypeC HPD - HPD connect for TypeC (DP alternate) 010b - TBT HPD - HPD connect for TBT 011b - Invalid - Invalid
	28	**Reserved** Access: RO Format: MBZ
	27:24	**Display Port x4 TX Lane Assignment for Type-C Connector 3** Same definition as DFLEXDPSP1.DPX4TXLATC0, but this register is for Type-C Connector 3. 0001b - PHY TX[0]

0010b	PHY TX[1]
0011b	PHY TX[1:0]
0100b	PHY TX[2]
0101b	PHY TX[2] TX[0]
1000b	PHY TX[3]
1100b	PHY TX[3:2]
1111b	PHY TX[3:0]

23:21 **TC2 Live State**

Value	Name	Description
000b	No HPD	No HPD connect for TypeC (DP alternate) or TBT
001b	TypeC HPD	HPD connect for TypeC (DP alternate)
010b	TBT HPD	HPD connect for TBT
011b	Invalid	Invalid

20 **Reserved**

Access:	RO
Format:	MBZ

19:16 **Display Port x4 TX Lane Assignment for Type-C Connector 2**

Same definition as DFLEXDPSP1.DPX4TXLATC0, but this register is for Type-C Connector 2.

Value	Name
0001b	PHY TX[0]
0010b	PHY TX[1]
0011b	PHY TX[1:0]
0100b	PHY TX[2]
0101b	PHY TX[2] TX[0]
1000b	PHY TX[3]
1100b	PHY TX[3:2]
1111b	PHY TX[3:0]

15:13 **TC1 Live State**

Value	Name	Description
000b	No HPD	No HPD connect for TypeC (DP alternate) or TBT
001b	TypeC HPD	HPD connect for TypeC (DP alternate)
010b	TBT HPD	HPD connect for TBT
011b	Invalid	Invalid

12 **IOM FW version**

This field identifies the IOM firmware that is used.

PORT_TX_DFLEXDPSP - PORT_TX_DFLEXDPSP

Value	Name
0b	old IOM FW
1b	IOM FW with MFD Gen2 support

11:8 **Display Port x4 TX Lane Assignment for Type-C Connector 1**

Same definition as DFLEXDPSP1.DPX4TXLATC0, but this register is for Type-C Connector 1.

Value	Name
0001b	PHY TX[0]
0010b	PHY TX[1]
0011b	PHY TX[1:0]
0100b	PHY TX[2]
0101b	PHY TX[2] TX[0]
1000b	PHY TX[3]
1100b	PHY TX[3:2]
1111b	PHY TX[3:0]

7:5 **TC0 Live state**

Value	Name	Description
000b	No HPD	No HPD connect for TypeC (DP alternate) or TBT
001b	TypeC HPD	HPD connect for TypeC (DP alternate)
010b	TBT HPD	HPD connect for TBT
011b	Invalid	Invalid

4 **Modular FIA (MF)**

Description
This bit is set by IOM FW and read by Display Driver. It tells the Display Driver if Modular FIA is used in the SOC. If Modular FIA is used in the SOC, then Display Driver will access the additional instances of FIA based on pre-assigned offset in GTTMADDR space. Each Modular FIA instance houses only 2 Type-C Ports. If Modular FIA is not used in the SOC, then a single monolithic FIA is used to house all the Type-C Ports which has only one IOSF Sideband Port ID.
Modular FIA is always used. IOM FW will program the MF bit in all FIA instances.

Value	Name
0b	Monolithic FIA
1b	Modular FIA

3:0 **Display Port x4 TX Lane Assignment for Type-C Connector 0**

DPX4TXLATC0

SOC FW writes to these bits to tell display software the Lane Assignment, which it generates

PORT_TX_DFLEXDPSP - PORT_TX_DFLEXDPSP

based on the DP Pin Assignment and the Connector Orientation. Display software uses this value to determine the number of lanes that can be enabled, and along with other registers, to determine the DP mode programming. See the Typec PHY DDI Buffer page for DP mode programming.
The 4 bits correspond to 4 TX, i.e. TX[3:0] Lane in PHY.
Lower 2 bits correspond to the 2 lower TX lane on the PHY of Type-C connector.
Upper 2 bits correspond to the upper 2 TX lane on the PHY of Type-C connector.
For example, in DP Pin Assignment D (Multi function) and Flip case, the x2 TX lane are on the upper TypeC Lane, hence the value written into this register will be 1100b.
Another example, in DP Pin Assignment B (Multi function) Active Gen2 cable and Flip case, the x1 TX lane is on the 1st TX of upper TypeC Lane, hence the value written into this register will be 0100b.

Value	Name
0001b	PHY TX[0]
0010b	PHY TX[1]
0011b	PHY TX[1:0]
0100b	PHY TX[2]
0101b	PHY TX[2] TX[0]
1000b	PHY TX[3]
1100b	PHY TX[3:2]
1111b	PHY TX[3:0]

PORT_TX_DFLEXNPCPMS

PORT_TX_DFLEXNPCPMS - PORT_TX_DFLEXNPCPMS	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Reset:	soft
Address:	163480h-163483h
Name:	PORT_TX_DFLEXNPCPMS
ShortName:	PORT_TX_DFLEXNPCPMS_FIA1
Reset:	soft
Address:	16E480h-16E483h
Name:	PORT_TX_DFLEXNPCPMS
ShortName:	PORT_TX_DFLEXNPCPMS_FIA2
Reset:	soft
Address:	16F480h-16F483h
Name:	PORT_TX_DFLEXNPCPMS
ShortName:	PORT_TX_DFLEXNPCPMS_FIA3
Reset:	soft

SW writes to these bits to control the Combo Ports mode. This register governs the Phy status tracking handling that could be different for different controllers.

DWord	Bit	Description
0	31:28	**Combo Port 7 Next Phy Combo Port Mode Select (CP7NPCPMS):** Similar to register DFLEXNPCPMS1.CP0NPCPMS but this register is for Combo Port 7.
	27:24	**Combo Port 6 Next Phy Combo Port Mode Select (CP6NPCPMS):** Similar to register DFLEXNPCPMS1.CP0NPCPMS but this register is for Combo Port 6.
	23:20	**Combo Port 5 Next Phy Combo Port Mode Select (CP5NPCPMS):** Similar to register DFLEXNPCPMS1.CP0NPCPMS but this register is for Combo Port 5.
	19:16	**Combo Port 4 Next Phy Combo Port Mode Select (CP4NPCPMS):** Similar to register DFLEXNPCPMS1.CP0NPCPMS but this register is for Combo Port 4.
	15:12	**Combo Port 3 Next Phy Combo Port Mode Select (CP3NPCPMS):** Similar to register DFLEXNPCPMS1.CP0NPCPMS but this register is for Combo Port 3.
	11:8	**Combo Port 2 Next Phy Combo Port Mode Select (CP2NPCPMS):** Similar to register DFLEXNPCPMS1.CP0NPCPMS but this register is for Combo Port 2.
	7:4	**Combo Port 1 Next Phy Combo Port Mode Select (CP1NPCPMS):** Similar to register DFLEXNPCPMS1.CP0NPCPMS but this register is for Combo Port 1.
	3:0	**Combo Port 0 Next Phy Combo Port Mode Select (CP0NPCPMS):** The Combo Port number is logical. Its not physical lane numbers. 0h: Port Mode is NoOwner_USB3. FIA will keep the Lane in Reset state. PhyMode=USB3 1h: Port Mode is Owner 1 (default owner)

PORT_TX_DFLEXNPCPMS - PORT_TX_DFLEXNPCPMS		
		2h: Port Mode is Owner 2 3h: Port Mode is Owner 3 4h: Port Mode is Owner 4 5h: Port Mode is Owner 5 6h-Fh: Reserved Others: Reserved SW mode: SW writes to these bits to control the Combo Ports mode. This register governs the Phy status tracking handling that could be different for different controller. HW mode: These bits have no impact to HW when written. The reset default is dependent on the internal HW mux select bits though its being indicated as soft-straps. The mux selects bits retain their context across Sx and will be reloaded back into this field.

PORT_TX_DFLEXPA1

PORT_TX_DFLEXPA1	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	163880h-163883h
Name:	PORT_TX_DFLEXPA1
ShortName:	PORT_TX_DFLEXPA1_FIA1
Reset:	global
Address:	16E880h-16E883h
Name:	PORT_TX_DFLEXPA1
ShortName:	PORT_TX_DFLEXPA1_FIA2
Reset:	global
Address:	16F880h-16F883h
Name:	PORT_TX_DFLEXPA1
ShortName:	PORT_TX_DFLEXPA1_FIA3
Reset:	global

FIA has per Connector register to govern the Pin Assignment of each Type-C Connector. For example, DFLEXPA1.DPPATC0 is used to govern the Pin Assignment of Type-C Connector 0. The Type-C Connector number (e.g. "0" in register DPPATC0) is logical number.

DWord	Bit	Description
0	31:28	**Displayport Pin Assignment for Type-C Connector 7** Similar to register DFLEXPA1.DPPATC0 but this register is for Type-C Connector 7.
	27:24	**Displayport Pin Assignment for Type-C Connector 6** Similar to register DFLEXPA1.DPPATC0 but this register is for Type-C Connector 6.
	23:20	**Displayport Pin Assignment for Type-C Connector 5** Similar to register DFLEXPA1.DPPATC0 but this register is for Type-C Connector 5.
	19:16	**Displayport Pin Assignment for Type-C Connector 4** Similar to register DFLEXPA1.DPPATC0 but this register is for Type-C Connector 4.
	15:12	**Displayport Pin Assignment for Type-C Connector 3** Similar to register DFLEXPA1.DPPATC0 but this register is for Type-C Connector 3.
	11:8	**Displayport Pin Assignment for Type-C Connector 2** Similar to register DFLEXPA1.DPPATC0 but this register is for Type-C Connector 2.
	7:4	**Displayport Pin Assignment for Type-C Connector 1** Similar to register DFLEXPA1.DPPATC0 but this register is for Type-C Connector 1.
	3:0	**Displayport Pin Assignment for Type-C Connector 0** Display Port Pin Assignment for Type-C Connector 0 (DPPATC0): Assignments Cand E have 4 lanes for DP alternate mode. Assignments D has2 lanes for DP alternate mode.

PORT_TX_DFLEXPA1			
		Value	Name
		0000b	No Pin Assignment (For Non Type-C/static/fixed/legacy connections)
		0011b	Pin Assignment C
		0100b	Pin Assignment D
		0101b	Pin Assignment E

PORT_TX_DFLEXPA2

PORT_TX_DFLEXPA2 - PORT_TX_DFLEXPA2	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Reset:	soft
Address:	163884h-163887h
Name:	PORT_TX_DFLEXPA2
ShortName:	PORT_TX_DFLEXPA2_FIA1
Reset:	soft
Address:	16E884h-16E887h
Name:	PORT_TX_DFLEXPA2
ShortName:	PORT_TX_DFLEXPA2_FIA2
Reset:	soft
Address:	16F884h-16F887h
Name:	PORT_TX_DFLEXPA2
ShortName:	PORT_TX_DFLEXPA2_FIA3
Reset:	soft

FIA has per Connector register to govern the Pin Assignment of each Type-C Connector. For example, DFLEXPA1.DPPATC0 is used to govern the Pin Assignment of Type-C Connector 0.The Type-C Connector number (e.g. 0 in register DPPATC0) is logical number.

DWord	Bit	Description
0	31:28	**Displayport Pin Assignment for Type-C Connector 15** Similar to register DFLEXPA2.DPPATC8 but this register is for Type-C Connector 15.
	27:24	**Displayport Pin Assignment for Type-C Connector 14** Similar to register DFLEXPA2.DPPATC8 but this register is for Type-C Connector 14.
	23:20	**Displayport Pin Assignment for Type-C Connector 13** Similar to register DFLEXPA2.DPPATC8 but this register is for Type-C Connector 13.
	19:16	**Displayport Pin Assignment for Type-C Connector 12** Similar to register DFLEXPA2.DPPATC8 but this register is for Type-C Connector 12.
	15:12	**Displayport Pin Assignment for Type-C Connector 11** Similar to register DFLEXPA2.DPPATC8 but this register is for Type-C Connector 11.
	11:8	**Displayport Pin Assignment for Type-C Connector 10** Similar to register DFLEXPA2.DPPATC8 but this register is for Type-C Connector 10.
	7:4	**Displayport Pin Assignment for Type-C Connector 9** Similar to register DFLEXPA2.DPPATC8 but this register is for Type-C Connector 9.
	3:0	**Displayport Pin Assignment for Type-C Connector 8** Display Port Pin Assignment for Type-C Connector 8 (DPPATC8): 0000 : No Pin Assignment (For Non Type-C DP)

PORT_TX_DFLEXPA2 - PORT_TX_DFLEXPA2		
		0001 : Pin Assignment A 0010 : Pin Assignment B 0011 : Pin Assignment C 0100 : Pin Assignment D 0101 : Pin Assignment E 0110 : Pin Assignment F 0111-1111 : Reserved

PORT_TX_DW0

PORT_TX_DW0	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	162380h-162383h
Name:	PORT_TX_DW0
ShortName:	PORT_TX_DW0_AUX_A
Reset:	global
Address:	162680h-162683h
Name:	PORT_TX_DW0
ShortName:	PORT_TX_DW0_GRP_A
Reset:	global
Address:	162880h-162883h
Name:	PORT_TX_DW0
ShortName:	PORT_TX_DW0_LN0_A
Reset:	global
Address:	162980h-162983h
Name:	PORT_TX_DW0
ShortName:	PORT_TX_DW0_LN1_A
Reset:	global
Address:	162A80h-162A83h
Name:	PORT_TX_DW0
ShortName:	PORT_TX_DW0_LN2_A
Reset:	global
Address:	162B80h-162B83h
Name:	PORT_TX_DW0
ShortName:	PORT_TX_DW0_LN3_A
Reset:	global
Address:	6C380h-6C383h
Name:	PORT_TX_DW0
ShortName:	PORT_TX_DW0_AUX_B
Reset:	global
Address:	6C680h-6C683h
Name:	PORT_TX_DW0
ShortName:	PORT_TX_DW0_GRP_B

PORT_TX_DW0	
Reset:	global
Address:	6C880h-6C883h
Name:	PORT_TX_DW0
ShortName:	PORT_TX_DW0_LN0_B
Reset:	global
Address:	6C980h-6C983h
Name:	PORT_TX_DW0
ShortName:	PORT_TX_DW0_LN1_B
Reset:	global
Address:	6CA80h-6CA83h
Name:	PORT_TX_DW0
ShortName:	PORT_TX_DW0_LN2_B
Reset:	global
Address:	6CB80h-6CB83h
Name:	PORT_TX_DW0
ShortName:	PORT_TX_DW0_LN3_B
Reset:	global
Address:	160380h-160383h
Name:	PORT_TX_DW0
ShortName:	PORT_TX_DW0_AUX_C
Reset:	global
Address:	160680h-160683h
Name:	PORT_TX_DW0
ShortName:	PORT_TX_DW0_GRP_C
Reset:	global
Address:	160880h-160883h
Name:	PORT_TX_DW0
ShortName:	PORT_TX_DW0_LN0_C
Reset:	global
Address:	160980h-160983h
Name:	PORT_TX_DW0
ShortName:	PORT_TX_DW0_LN1_C
Reset:	global
Address:	160A80h-160A83h
Name:	PORT_TX_DW0
ShortName:	PORT_TX_DW0_LN2_C

PORT_TX_DW0

Reset:	global

Address:	160B80h-160B83h
Name:	PORT_TX_DW0
ShortName:	PORT_TX_DW0_LN3_C
Reset:	global

This register controls Tx Equalization within the Combo-PHY's AFE.

Restriction : Do not change the default Cursor Coefficient values within this register since that will change the equalization being applied by the PHY

DWord	Bit	Description
0	31	**MIPI EQ Select** This bit controls the Tx Equalization level of the PHY This bit is equivalent to the **TxEqLevelHS** PPI pin, but it is only used by the PHY when both the MIPI EQ Override Enable and MIPI EQ Enable bits are set. **Value** / **Name** 0b – Low Levlel Equalization (3.5 dB) 1b – High Level Equalization (7 dB)
	30	**MIPI EQ Enable** This bit represents the Tx Equalization active state (i.e. enable) This bit is equivalent to the **TxEqActiveHS** PPI pin, but it is only used by the PHY when the MIPI EQ Override Enable is set **Value** / **Name** 0b – Tx Equalization Disabled 1b – Tx Equalization Enabled
	29:24	**Post Cursor Coeff 0** Default Value: bh Default post cursor coeff Restriction : Do not change the default value
	23	**MIPI EQ Override Enable** This bit controls whether the PPI Tx Equalization pins are driving the equalization logic, or the override bits from this register **Value** / **Name** 0b – PPI inputs drive EQ logic 1b – This register drives EQ logic
	22:6	**Reserved** Access: RO Format: MBZ
	5:0	**Cursor Coeff 0** Default Value: 34h Default cursor coeff Restriction : Do not change the default value

PORT_TX_DW1

PORT_TX_DW1	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	162384h-162387h
Name:	PORT_TX_DW1
ShortName:	PORT_TX_DW1_AUX_A
Reset:	global
Address:	162684h-162687h
Name:	PORT_TX_DW1
ShortName:	PORT_TX_DW1_GRP_A
Reset:	global
Address:	162884h-162887h
Name:	PORT_TX_DW1
ShortName:	PORT_TX_DW1_LN0_A
Reset:	global
Address:	162984h-162987h
Name:	PORT_TX_DW1
ShortName:	PORT_TX_DW1_LN1_A
Reset:	global
Address:	162A84h-162A87h
Name:	PORT_TX_DW1
ShortName:	PORT_TX_DW1_LN2_A
Reset:	global
Address:	162B84h-162B87h
Name:	PORT_TX_DW1
ShortName:	PORT_TX_DW1_LN3_A
Reset:	global
Address:	6C384h-6C387h
Name:	PORT_TX_DW1
ShortName:	PORT_TX_DW1_AUX_B
Reset:	global
Address:	6C684h-6C687h
Name:	PORT_TX_DW1
ShortName:	PORT_TX_DW1_GRP_B

PORT_TX_DW1	
Reset:	global
Address:	6C884h-6C887h
Name:	PORT_TX_DW1
ShortName:	PORT_TX_DW1_LN0_B
Reset:	global
Address:	6C984h-6C987h
Name:	PORT_TX_DW1
ShortName:	PORT_TX_DW1_LN1_B
Reset:	global
Address:	6CA84h-6CA87h
Name:	PORT_TX_DW1
ShortName:	PORT_TX_DW1_LN2_B
Reset:	global
Address:	6CB84h-6CB87h
Name:	PORT_TX_DW1
ShortName:	PORT_TX_DW1_LN3_B
Reset:	global
Address:	160384h-160387h
Name:	PORT_TX_DW1
ShortName:	PORT_TX_DW1_AUX_C
Reset:	global
Address:	160684h-160687h
Name:	PORT_TX_DW1
ShortName:	PORT_TX_DW1_GRP_C
Reset:	global
Address:	160884h-160887h
Name:	PORT_TX_DW1
ShortName:	PORT_TX_DW1_LN0_C
Reset:	global
Address:	160984h-160987h
Name:	PORT_TX_DW1
ShortName:	PORT_TX_DW1_LN1_C
Reset:	global
Address:	160A84h-160A87h
Name:	PORT_TX_DW1
ShortName:	PORT_TX_DW1_LN2_C

PORT_TX_DW1

Reset:	global
Address:	160B84h-160B87h
Name:	PORT_TX_DW1
ShortName:	PORT_TX_DW1_LN3_C
Reset:	global

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**o_iref_config** ICOMP Config bit from COMP routed to Txana
	6:5	**o_iref_ctrl** control to change the ratio of tx iboost
	4:3	**o_tx_slew_ctrl** Used for MIPI HSTX Slew rate control config
	2	**o_vref_low_en** LDO Feedback path enable for low vref
	1	**o_vref_hi_en** LDO Feedback path enable for hi vref
	0	**o_vref_nom_en** LDO Feedback path enable for nominal vref

PORT_TX_DW2

PORT_TX_DW2	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	162388h-16238Bh
Name:	PORT_TX_DW2
ShortName:	PORT_TX_DW2_AUX_A
Reset:	global
Address:	162688h-16268Bh
Name:	PORT_TX_DW2
ShortName:	PORT_TX_DW2_GRP_A
Reset:	global
Address:	162888h-16288Bh
Name:	PORT_TX_DW2
ShortName:	PORT_TX_DW2_LN0_A
Reset:	global
Address:	162988h-16298Bh
Name:	PORT_TX_DW2
ShortName:	PORT_TX_DW2_LN1_A
Reset:	global
Address:	162A88h-162A8Bh
Name:	PORT_TX_DW2
ShortName:	PORT_TX_DW2_LN2_A
Reset:	global
Address:	162B88h-162B8Bh
Name:	PORT_TX_DW2
ShortName:	PORT_TX_DW2_LN3_A
Reset:	global
Address:	6C388h-6C38Bh
Name:	PORT_TX_DW2
ShortName:	PORT_TX_DW2_AUX_B
Reset:	global
Address:	6C688h-6C68Bh
Name:	PORT_TX_DW2
ShortName:	PORT_TX_DW2_GRP_B

PORT_TX_DW2	
Reset:	global
Address:	6C888h-6C88Bh
Name:	PORT_TX_DW2
ShortName:	PORT_TX_DW2_LN0_B
Reset:	global
Address:	6C988h-6C98Bh
Name:	PORT_TX_DW2
ShortName:	PORT_TX_DW2_LN1_B
Reset:	global
Address:	6CA88h-6CA8Bh
Name:	PORT_TX_DW2
ShortName:	PORT_TX_DW2_LN2_B
Reset:	global
Address:	6CB88h-6CB8Bh
Name:	PORT_TX_DW2
ShortName:	PORT_TX_DW2_LN3_B
Reset:	global
Address:	160388h-16038Bh
Name:	PORT_TX_DW2
ShortName:	PORT_TX_DW2_AUX_C
Reset:	global
Address:	160688h-16068Bh
Name:	PORT_TX_DW2
ShortName:	PORT_TX_DW2_GRP_C
Reset:	global
Address:	160888h-16088Bh
Name:	PORT_TX_DW2
ShortName:	PORT_TX_DW2_LN0_C
Reset:	global
Address:	160988h-16098Bh
Name:	PORT_TX_DW2
ShortName:	PORT_TX_DW2_LN1_C
Reset:	global
Address:	160A88h-160A8Bh
Name:	PORT_TX_DW2
ShortName:	PORT_TX_DW2_LN2_C

PORT_TX_DW2

Reset:	global

Address:	160B88h-160B8Bh
Name:	PORT_TX_DW2
ShortName:	PORT_TX_DW2_LN3_C
Reset:	global

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15	**swing_sel upper** Swing_sel bit 3
	14	**cmnmode_sel** Select one of the weak common modes.
	13:11	**swing_sel lower** Default Value: 010b Select the voltage swing level. Note that this field has swing_sel bits 2:0 and bit 3 is in swing_sel upper, which is not adjacent.
	10:8	**frclatencyoptim** Enables forcing the latency optimized value for the FIFO.
	7:0	**Rcomp scalar** Default Value: 10011000b Also called Swing Scalar. Scalar to be applied to comp code to get required termination.

PORT_TX_DW4

PORT_TX_DW4	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	162390h-162393h
Name:	PORT_TX_DW4
ShortName:	PORT_TX_DW4_AUX_A
Reset:	global
Address:	162690h-162693h
Name:	PORT_TX_DW4
ShortName:	PORT_TX_DW4_GRP_A
Reset:	global
Address:	162890h-162893h
Name:	PORT_TX_DW4
ShortName:	PORT_TX_DW4_LN0_A
Reset:	global
Address:	162990h-162993h
Name:	PORT_TX_DW4
ShortName:	PORT_TX_DW4_LN1_A
Reset:	global
Address:	162A90h-162A93h
Name:	PORT_TX_DW4
ShortName:	PORT_TX_DW4_LN2_A
Reset:	global
Address:	162B90h-162B93h
Name:	PORT_TX_DW4
ShortName:	PORT_TX_DW4_LN3_A
Reset:	global
Address:	6C390h-6C393h
Name:	PORT_TX_DW4
ShortName:	PORT_TX_DW4_AUX_B
Reset:	global
Address:	6C690h-6C693h
Name:	PORT_TX_DW4
ShortName:	PORT_TX_DW4_GRP_B

PORT_TX_DW4	
Reset:	global
Address:	6C890h-6C893h
Name:	PORT_TX_DW4
ShortName:	PORT_TX_DW4_LN0_B
Reset:	global
Address:	6C990h-6C993h
Name:	PORT_TX_DW4
ShortName:	PORT_TX_DW4_LN1_B
Reset:	global
Address:	6CA90h-6CA93h
Name:	PORT_TX_DW4
ShortName:	PORT_TX_DW4_LN2_B
Reset:	global
Address:	6CB90h-6CB93h
Name:	PORT_TX_DW4
ShortName:	PORT_TX_DW4_LN3_B
Reset:	global
Address:	160390h-160393h
Name:	PORT_TX_DW4
ShortName:	PORT_TX_DW4_AUX_C
Reset:	global
Address:	160690h-160693h
Name:	PORT_TX_DW4
ShortName:	PORT_TX_DW4_GRP_C
Reset:	global
Address:	160890h-160893h
Name:	PORT_TX_DW4
ShortName:	PORT_TX_DW4_LN0_C
Reset:	global
Address:	160990h-160993h
Name:	PORT_TX_DW4
ShortName:	PORT_TX_DW4_LN1_C
Reset:	global
Address:	160A90h-160A93h
Name:	PORT_TX_DW4
ShortName:	PORT_TX_DW4_LN2_C

PORT_TX_DW4

Reset:	global
Address:	160B90h-160B93h
Name:	PORT_TX_DW4
ShortName:	PORT_TX_DW4_LN3_C
Reset:	global

DWord	Bit	Description
0	31	**Loadgen Select**
	30:24	**Spare**
	23	**BS Comp Ovrd**
	22:18	**Rterm Limit** Default Value: 10000b
	17:12	**Post Cursor 1** o_txscaling_coeff[17:12]
	11:6	**Post Cursor 2** o_txscaling_coeff[11:6]
	5:0	**Cursor Coeff** Default Value: 011000b o_txscaling_coeff[5:0]

PORT_TX_DW5

PORT_TX_DW5	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	162394h-162397h
Name:	PORT_TX_DW5
ShortName:	PORT_TX_DW5_AUX_A
Reset:	global
Address:	162694h-162697h
Name:	PORT_TX_DW5
ShortName:	PORT_TX_DW5_GRP_A
Reset:	global
Address:	162894h-162897h
Name:	PORT_TX_DW5
ShortName:	PORT_TX_DW5_LN0_A
Reset:	global
Address:	162994h-162997h
Name:	PORT_TX_DW5
ShortName:	PORT_TX_DW5_LN1_A
Reset:	global
Address:	162A94h-162A97h
Name:	PORT_TX_DW5
ShortName:	PORT_TX_DW5_LN2_A
Reset:	global
Address:	162B94h-162B97h
Name:	PORT_TX_DW5
ShortName:	PORT_TX_DW5_LN3_A
Reset:	global
Address:	6C394h-6C397h
Name:	PORT_TX_DW5
ShortName:	PORT_TX_DW5_AUX_B
Reset:	global
Address:	6C694h-6C697h
Name:	PORT_TX_DW5
ShortName:	PORT_TX_DW5_GRP_B

PORT_TX_DW5	
Reset:	global
Address:	6C894h-6C897h
Name:	PORT_TX_DW5
ShortName:	PORT_TX_DW5_LN0_B
Reset:	global
Address:	6C994h-6C997h
Name:	PORT_TX_DW5
ShortName:	PORT_TX_DW5_LN1_B
Reset:	global
Address:	6CA94h-6CA97h
Name:	PORT_TX_DW5
ShortName:	PORT_TX_DW5_LN2_B
Reset:	global
Address:	6CB94h-6CB97h
Name:	PORT_TX_DW5
ShortName:	PORT_TX_DW5_LN3_B
Reset:	global
Address:	160394h-160397h
Name:	PORT_TX_DW5
ShortName:	PORT_TX_DW5_AUX_C
Reset:	global
Address:	160694h-160697h
Name:	PORT_TX_DW5
ShortName:	PORT_TX_DW5_GRP_C
Reset:	global
Address:	160894h-160897h
Name:	PORT_TX_DW5
ShortName:	PORT_TX_DW5_LN0_C
Reset:	global
Address:	160994h-160997h
Name:	PORT_TX_DW5
ShortName:	PORT_TX_DW5_LN1_C
Reset:	global
Address:	160A94h-160A97h
Name:	PORT_TX_DW5
ShortName:	PORT_TX_DW5_LN2_C

PORT_TX_DW5

Reset: global

Address: 160B94h-160B97h
Name: PORT_TX_DW5
ShortName: PORT_TX_DW5_LN3_C
Reset: global

DWord	Bit	Description
0	31	**TX Training Enable** Value: 1b — Name: enable Value: 0b — Name: disable
	30	**Disable 2tap** ospare2[5] Value: 0b — Name: Enable Value: 1b — Name: Disable
	29	**Disable 3tap** ospare2[5] Value: 0b — Name: Enable Value: 1b — Name: Disable
	28:27	**Spare 28 27** ospare2[4:3]
	26	**Cursor Program** ospare2[2] Value: 0b — Name: Enable Value: 1b — Name: Disable
	25	**Coeff Polarity** ospare2[1] Value: 0b — Name: Enable Value: 1b — Name: Disable
	24	**Spare 24** ospare2[0]
	23:21	**Reserved** Access: RO Format: MBZ

PORT_TX_DW5		
	20:18	**Scaling Mode Sel** Default Value: 010b
	17:16	**Decode Timer Sel** Default Value: 01b
	15:11	**CR Scaling Coef**
	10:6	**Spare 10 6** o_tx_vswing[10:6]
	5:3	**Rterm Select** o_tx_vswing[5:3]
	2:0	**Spare 2 0** o_tx_vswing[2:0]

PORT_TX_DW6

PORT_TX_DW6 - PORT_TX_DW6	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	162398h-16239Bh
Name:	PORT_TX_DW6
ShortName:	PORT_TX_DW6_AUX_A
Reset:	global
Address:	162698h-16269Bh
Name:	PORT_TX_DW6
ShortName:	PORT_TX_DW6_GRP_A
Reset:	global
Address:	162898h-16289Bh
Name:	PORT_TX_DW6
ShortName:	PORT_TX_DW6_LN0_A
Reset:	global
Address:	162998h-16299Bh
Name:	PORT_TX_DW6
ShortName:	PORT_TX_DW6_LN1_A
Reset:	global
Address:	162A98h-162A9Bh
Name:	PORT_TX_DW6
ShortName:	PORT_TX_DW6_LN2_A
Reset:	global
Address:	162B98h-162B9Bh
Name:	PORT_TX_DW6
ShortName:	PORT_TX_DW6_LN3_A
Reset:	global
Address:	6C398h-6C39Bh
Name:	PORT_TX_DW6
ShortName:	PORT_TX_DW6_AUX_B
Reset:	global
Address:	6C698h-6C69Bh
Name:	PORT_TX_DW6
ShortName:	PORT_TX_DW6_GRP_B

PORT_TX_DW6 - PORT_TX_DW6	
Reset:	global
Address:	6C898h-6C89Bh
Name:	PORT_TX_DW6
ShortName:	PORT_TX_DW6_LN0_B
Reset:	global
Address:	6C998h-6C99Bh
Name:	PORT_TX_DW6
ShortName:	PORT_TX_DW6_LN1_B
Reset:	global
Address:	6CA98h-6CA9Bh
Name:	PORT_TX_DW6
ShortName:	PORT_TX_DW6_LN2_B
Reset:	global
Address:	6CB98h-6CB9Bh
Name:	PORT_TX_DW6
ShortName:	PORT_TX_DW6_LN3_B
Reset:	global
Address:	160398h-16039Bh
Name:	PORT_TX_DW6
ShortName:	PORT_TX_DW6_AUX_C
Reset:	global
Address:	160698h-16069Bh
Name:	PORT_TX_DW6
ShortName:	PORT_TX_DW6_GRP_C
Reset:	global
Address:	160898h-16089Bh
Name:	PORT_TX_DW6
ShortName:	PORT_TX_DW6_LN0_C
Reset:	global
Address:	160998h-16099Bh
Name:	PORT_TX_DW6
ShortName:	PORT_TX_DW6_LN1_C
Reset:	global
Address:	160A98h-160A9Bh
Name:	PORT_TX_DW6
ShortName:	PORT_TX_DW6_LN2_C

PORT_TX_DW6 - PORT_TX_DW6

Reset:	global
Address:	160B98h-160B9Bh
Name:	PORT_TX_DW6
ShortName:	PORT_TX_DW6_LN3_C
Reset:	global

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**o_func_ovrd_en** ovrd enable signal
	6:1	**o_ldo_ref_sel_cri** ovrd for ldo_ref_sel
	0	**o_ldo_bypass_cri** ovrd for ldo bypass

PORT_TX_DW7

PORT_TX_DW7	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	16239Ch-16239Fh
Name:	PORT_TX_DW7
ShortName:	PORT_TX_DW7_AUX_A
Reset:	global
Address:	16269Ch-16269Fh
Name:	PORT_TX_DW7
ShortName:	PORT_TX_DW7_GRP_A
Reset:	global
Address:	16289Ch-16289Fh
Name:	PORT_TX_DW7
ShortName:	PORT_TX_DW7_LN0_A
Reset:	global
Address:	16299Ch-16299Fh
Name:	PORT_TX_DW7
ShortName:	PORT_TX_DW7_LN1_A
Reset:	global
Address:	162A9Ch-162A9Fh
Name:	PORT_TX_DW7
ShortName:	PORT_TX_DW7_LN2_A
Reset:	global
Address:	162B9Ch-162B9Fh
Name:	PORT_TX_DW7
ShortName:	PORT_TX_DW7_LN3_A
Reset:	global
Address:	6C39Ch-6C39Fh
Name:	PORT_TX_DW7
ShortName:	PORT_TX_DW7_AUX_B
Reset:	global
Address:	6C69Ch-6C69Fh
Name:	PORT_TX_DW7
ShortName:	PORT_TX_DW7_GRP_B

PORT_TX_DW7	
Reset:	global
Address:	6C89Ch-6C89Fh
Name:	PORT_TX_DW7
ShortName:	PORT_TX_DW7_LN0_B
Reset:	global
Address:	6C99Ch-6C99Fh
Name:	PORT_TX_DW7
ShortName:	PORT_TX_DW7_LN1_B
Reset:	global
Address:	6CA9Ch-6CA9Fh
Name:	PORT_TX_DW7
ShortName:	PORT_TX_DW7_LN2_B
Reset:	global
Address:	6CB9Ch-6CB9Fh
Name:	PORT_TX_DW7
ShortName:	PORT_TX_DW7_LN3_B
Reset:	global
Address:	16039Ch-16039Fh
Name:	PORT_TX_DW7
ShortName:	PORT_TX_DW7_AUX_C
Reset:	global
Address:	16069Ch-16069Fh
Name:	PORT_TX_DW7
ShortName:	PORT_TX_DW7_GRP_C
Reset:	global
Address:	16089Ch-16089Fh
Name:	PORT_TX_DW7
ShortName:	PORT_TX_DW7_LN0_C
Reset:	global
Address:	16099Ch-16099Fh
Name:	PORT_TX_DW7
ShortName:	PORT_TX_DW7_LN1_C
Reset:	global
Address:	160A9Ch-160A9Fh
Name:	PORT_TX_DW7
ShortName:	PORT_TX_DW7_LN2_C

PORT_TX_DW7

Reset:	global
Address:	160B9Ch-160B9Fh
Name:	PORT_TX_DW7
ShortName:	PORT_TX_DW7_LN3_C
Reset:	global

DWord	Bit	Description
0	31	**Spare 31**
	30:24	**N Scalar** Default Value: 7Fh
	23:0	**Spare 23 0**

PORT_TX_DW8

PORT_TX_DW8	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	1623A0h-1623A3h
Name:	PORT_TX_DW8
ShortName:	PORT_TX_DW8_AUX_A
Reset:	global
Address:	1626A0h-1626A3h
Name:	PORT_TX_DW8
ShortName:	PORT_TX_DW8_GRP_A
Reset:	global
Address:	1628A0h-1628A3h
Name:	PORT_TX_DW8
ShortName:	PORT_TX_DW8_LN0_A
Reset:	global
Address:	1629A0h-1629A3h
Name:	PORT_TX_DW8
ShortName:	PORT_TX_DW8_LN1_A
Reset:	global
Address:	162AA0h-162AA3h
Name:	PORT_TX_DW8
ShortName:	PORT_TX_DW8_LN2_A
Reset:	global
Address:	162BA0h-162BA3h
Name:	PORT_TX_DW8
ShortName:	PORT_TX_DW8_LN3_A
Reset:	global
Address:	6C3A0h-6C3A3h
Name:	PORT_TX_DW8
ShortName:	PORT_TX_DW8_AUX_B
Reset:	global
Address:	6C6A0h-6C6A3h
Name:	PORT_TX_DW8
ShortName:	PORT_TX_DW8_GRP_B

PORT_TX_DW8	
Reset:	global
Address:	6C8A0h-6C8A3h
Name:	PORT_TX_DW8
ShortName:	PORT_TX_DW8_LN0_B
Reset:	global
Address:	6C9A0h-6C9A3h
Name:	PORT_TX_DW8
ShortName:	PORT_TX_DW8_LN1_B
Reset:	global
Address:	6CAA0h-6CAA3h
Name:	PORT_TX_DW8
ShortName:	PORT_TX_DW8_LN2_B
Reset:	global
Address:	6CBA0h-6CBA3h
Name:	PORT_TX_DW8
ShortName:	PORT_TX_DW8_LN3_B
Reset:	global
Address:	1603A0h-1603A3h
Name:	PORT_TX_DW8
ShortName:	PORT_TX_DW8_AUX_C
Reset:	global
Address:	1606A0h-1606A3h
Name:	PORT_TX_DW8
ShortName:	PORT_TX_DW8_GRP_C
Reset:	global
Address:	1608A0h-1608A3h
Name:	PORT_TX_DW8
ShortName:	PORT_TX_DW8_LN0_C
Reset:	global
Address:	1609A0h-1609A3h
Name:	PORT_TX_DW8
ShortName:	PORT_TX_DW8_LN1_C
Reset:	global
Address:	160AA0h-160AA3h
Name:	PORT_TX_DW8
ShortName:	PORT_TX_DW8_LN2_C

PORT_TX_DW8

Reset: global

Address: 160BA0h-160BA3h
Name: PORT_TX_DW8
ShortName: PORT_TX_DW8_LN3_C
Reset: global

DWord	Bit	Description
0	31	**odcc_clksel** Default Value: 0b DCC clock select
	30:29	**odcc_clk_div_sel** Value / Name: 00b,01b — div2 10b — div4 **[Default]** 11b — div8
	28:24	**odcc_code_ovrd** Default Value: 10000b
	23	**odcc_code_ovrd_en**
	22	**odccfuse_en**
	21	**Reserved**
	20:16	**odcc_lower_limit** Default Value: 00011b
	15	**Reserved**
	14:13	**idcc_code_therm_4_3** Default Value: 10b
	12:8	**idcc_code** Default Value: 10000b
	7:5	**idcc_code_therm_2_0**
	4:0	**odcc_upper_limit** Default Value: 11100b

POSH LRCA

POSH_LRCA - POSH LRCA	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	021B0h-021B3h
Name:	POSH LRCA
ShortName:	POSH_LRCA_RCSUNIT
Address:	181B0h-181B3h
Name:	POSH LRCA
ShortName:	POSH_LRCA_POCSUNIT
Address:	221B0h-221B3h
Name:	POSH LRCA
ShortName:	POSH_LRCA_BCSUNIT
Address:	1C01B0h-1C01B3h
Name:	POSH LRCA
ShortName:	POSH_LRCA_VCSUNIT0
Address:	1C41B0h-1C41B3h
Name:	POSH LRCA
ShortName:	POSH_LRCA_VCSUNIT1
Address:	1C81B0h-1C81B3h
Name:	POSH LRCA
ShortName:	POSH_LRCA_VECSUNIT0
Address:	1D01B0h-1D01B3h
Name:	POSH LRCA
ShortName:	POSH_LRCA_VCSUNIT2
Address:	1D41B0h-1D41B3h
Name:	POSH LRCA
ShortName:	POSH_LRCA_VCSUNIT3
Address:	1D81B0h-1D81B3h
Name:	POSH LRCA
ShortName:	POSH_LRCA_VECSUNIT1
Address:	1E01B0h-1E01B3h
Name:	POSH LRCA
ShortName:	POSH_LRCA_VCSUNIT4

POSH_LRCA - POSH LRCA

Address:	1E41B0h-1E41B3h
Name:	POSH LRCA
ShortName:	POSH_LRCA_VCSUNIT5

Address:	1E81B0h-1E81B3h
Name:	POSH LRCA
ShortName:	POSH_LRCA_VECSUNIT2

Address:	1F01B0h-1F01B3h
Name:	POSH LRCA
ShortName:	POSH_LRCA_VCSUNIT6

Address:	1F41B0h-1F41B3h
Name:	POSH LRCA
ShortName:	POSH_LRCA_VCSUNIT7

Address:	1F81B0h-1F81B3h
Name:	POSH LRCA
ShortName:	POSH_LRCA_VECSUNIT3

Address:	1A1B0h-1A1B3h
Name:	POSH LRCA
ShortName:	POSH_LRCA_CCSUNIT0

This register contains the LRCA address for the POSH pipe to which POCS does context save/restore. LRCA address programmed in this register is only effective when "POSH Enable" field is set in CTX_SR_CTL register. This register is not functional and must not be programmed for VideoCS, VideoEnhacementCS, BlitterCS.

DWord	Bit	Description
0	31:12	**POSH Logical Ring Context Address**
	11:0	**Reserved** Access: RO Format: MBZ

Power Clock State Register

PWR_CLK_STATE - Power Clock State Register

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV

Address:	020C8h-020CBh
Name:	PWR_CLK_STATE
ShortName:	PWR_CLK_STATE_RCSUNIT_BE_COMPUTE

Address:	1A0C8h-1A0CBh
Name:	PWR_CLK_STATE
ShortName:	PWR_CLK_STATE_CCSUNIT_BE_COMPUTE0

This register contains the mode selection for configuring render engine to attain desired performance and power requirements for a given context. This register is render context save/restored. This register must be initialized correctly when the context is submitted for the first time. This register is context save/restored as part of Exec-List context image in both Exec-List and Ring-Buffer mode of scheduling. This register contents are valid only when "Enable" bit [31] of the register is set.

Programming Notes

This register is only functional for RenderCS. This register must not be exercised in VideoCS, BlitterCS and VideoEnhancementCS.

This register must not be programmed directly through CPU MMIO cycle.
Exec-List Scheduling Mode: Every context can have its own required render engine configuration by programming this register appropriately in the logical render context image in memory (LRCA) before submitting the context to the execlist submit port. This register must not be programmed using MI_LOAD_REGISTER_IMM command in ring buffer or in batch buffer.

DWord	Bit	Description
0	31:0	**Reserved** Format: PBC

Power Management Capabilities

PMCAP_0_2_0_PCI - Power Management Capabilities

Register Space: PCI: 0/2/0
Size (in bits): 16

Address: 000D2h

This register provides information on the capabilities of the function related to power management.

DWord	Bit	Description
0	15:11	**PME Support** Default Value: 00000b Access: RO _Custom_GTIReset: BUS This field indicates the power states in which the IGD may assert PME#. Hardwired to 0 to indicate that the IGD does not assert the PME# signal.
	10	**D2 Support** Default Value: 0b Access: RO _Custom_GTIReset: BUS Hardwired to 0 to indicate the D2 power management state is not supported.
	9	**D1 Support** Default Value: 0b Access: RO _Custom_GTIReset: BUS Hardwired to 0 to indicate that the D1 power management state is not supported.
	8:6	**Reserved** Access: RO Format: MBZ
	5	**Device Specific Initialization** Default Value: 1b Access: RO _Custom_GTIReset: BUS Hardwired to 1 to indicate that special initialization of the IGD is required before generic class device driver is to use it.
	4	**Reserved** Access: RO Format: MBZ

PMCAP_0_2_0_PCI - Power Management Capabilities

	3	**PME Clock** Default Value: 0b Access: RO _Custom_GTIReset: BUS Hardwired to 0 to indicate IGD does not support PME# generation.
	2:0	**Version** Access: RO _Custom_GTIReset: BUS **Description** Hardwired to 010b to indicate that there are 4 bytes of power management registers implemented and that this device complies with revision 1.1 of the PCI Power Management Interface Specification. **Value** / **Name** 010b / **[Default]**

Power Management Capabilities ID

PMCAPID_0_2_0_PCI - Power Management Capabilities ID

Register Space:	PCI: 0/2/0
Size (in bits):	16
Address:	000D0h

This register contains the PCI Power Management Capability ID and the next capability pointer.

DWord	Bit	Description
0	15:8	**Next Capability Pointer** Default Value: 00000000b Access: RO _Custom_GTIReset: BUS This is a hardwired pointer to the next item in the capabilities list.
	7:0	**Capability Identifier** Default Value: 00000001b Access: RO _Custom_GTIReset: BUS Hardwired to 01h for power management.

Power Management Control and Status

PMCS_0_2_0_PCI - Power Management Control and Status

Register Space: PCI: 0/2/0
Size (in bits): 16

Address: 000D4h

DWord	Bit	Description
0	15	**PME Status** Default Value: 0b Access: RO _Custom_GTIReset: BUS This bit is hardwired to 0 to indicate that IGD does not support PME# generation from D3 (cold).
	14:13	**Data Scale** Default Value: 00b Access: RO _Custom_GTIReset: BUS This field is hardwired to 00 to indicate IGD does not support data register.
	12:9	**Data Select** Default Value: 0000b Access: RO _Custom_GTIReset: BUS This field is hardwired to 0h to indicate IGD does not support data register.
	8	**PME Enable** Default Value: 0b Access: RO _Custom_GTIReset: BUS This bit is hardwired to 0 to indicate that PME# assertion from D3 (cold) is disabled.
	7:2	**Reserved** Access: RO Format: MBZ
	1:0	**Power State** Default Value: 00b Access: R/W Variant _Custom_GTIReset: BUS This field indicates the current power state of the IGD and can be used to set the IGD into a new power state. If software attempts to write an unsupported state to this field, write operation must complete normally on the bus, but the data is discarded and no state change occurs. Behavior of the graphics controller in supported states is detailed in the power management section Bits[1:0]Power state00:D0Default01:D1Not Supported10:D2Not Supported11:D3

PPPR

PPPR - PPPR	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	124824h

GT uses this register to post pending page requests to software, such as x86 page faults.
A write to this register triggers an MSI per the registers PRESTS, PRECTL, PREDATA, PREADR, and PREUADR.

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**POST PENDING PAGE REQUEST** Default Value: 0h Access: WO _Custom_GTIReset: BUS Post Pending Page Request

PPRO

PPRO - PPRO		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
SOC_Consumer:	BIOS	
Address:	124820h	
GT uses this register to post Page Request Queue overflow faults		
DWord	**Bit**	**Description**
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**POST PAGE REQUEST OVERFLOW FAULT** Default Value: 0h Access: WO _Custom_GTIReset: BUS Post Page Request overflow fault

PRE_CSC_GAMC_DATA

PRE_CSC_GAMC_DATA	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	4A488h-4A48Bh
Name:	Pipe Pre CSC Gamma Data
ShortName:	PRE_CSC_GAMC_DATA_A
Reset:	soft
Address:	4AC88h-4AC8Bh
Name:	Pipe Pre CSC Gamma Data
ShortName:	PRE_CSC_GAMC_DATA_B
Reset:	soft
Address:	4B488h-4B48Bh
Name:	Pipe Pre CSC Gamma Data
ShortName:	PRE_CSC_GAMC_DATA_C
Reset:	soft
Address:	4BC88h-4BC8Bh
Name:	Pipe Pre CSC Gamma Data
ShortName:	PRE_CSC_GAMC_DATA_D
Reset:	soft

Description

PRE_CSC_GAMC_INDEX and PRE_CSC_GAMC_DATA registers are used to program the values that determine the characteristics of the gamma correction for the pipe pixel data before Color Space Conversion. Additional gamma correction can be done after the Color Space Conversion Gamma if desired.
The gamma correction curve is represented by specifying a set of gamma entry reference points spaced equally along the curve for values between -1 and 1. For extended values there are extended gamma entry reference points. All input values are clamped to the greater than -7.0 and less than 7.0 range before the gamma calculation.
For input values greater than or equal to 0 and less than 1.0, the input value is used to linearly interpolate between two adjacent points of the first 33 gamma entries to create the result value. The first 32 entries are stored as16 bits per color in an unsigned 0.16 format with 0 integer and 16 fractional. The 33rd, 34th and 35th entries are stored as 19 bits per color in an unsigned 3.16 format with 3 integer and 16 fractional bits.
For input values greater than or equal to 1.0 and less than 3.0, the input value is used to linearly interpolate between the 33rd and 34th gamma entries to create the result value.
For input values greater than or equal to 3.0 and less than 7.0, the input value is used to linearly interpolate between the 34th and 35th gamma entries to create the result value.
For negative input values, gamma is mirrored along the X-axis, giving the same result as positive input values, except for a negative sign. When gamma input may be negative, the first gamma point should be programmed to a value of 0.0 in order to have a symmetric mirroring.

PRE_CSC_GAMC_DATA

Pre-CSC Gamma correction gets enabled or disabled based on the "Pipe CSC Enable" bit in the PLANE_COLOR_CTL register. The same set of values is used for gamma correction of the red, blue and green channels.
See Pipe Gamma for an example gamma curve diagram.

Programming Notes

To program the gamma correction entries, calculate the desired gamma curve for inputs from 0 to 7.0. For inputs of 0 to 1.0, multiply the input value by 32 to find the gamma entry number, then store the desired gamma result in that entry. For inputs greater than 1.0 and less than or equal to 3.0, store the result for an input of 3.0 in the 34th gamma entry. For inputs greater than 3.0 and less than or equal to 7.0, store the result for an input of 7.0 in the 35th gamma entry.
Recommended sRGB degamma programming:

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Values	0	9F	153	253	3AC	565	783	A0C	D06	1076	145F	18C8	1DB3	2325	2922	2FAE	36CB	3E7E	46CA	4FB1	5938	6361	6E2E	79A4	85C4	9292	A00F	AE3F	BD25	CCC2	DD19	EE2D	10000

Restriction

The gamma curve must be flat or increasing, never decreasing. The gamma correction registers should only be updated when the plane is off, otherwise screen artifacts may show temporarily.

DWord	Bit	Description
0	31:27	**Reserved** Access: RO Format: MBZ
	26:19	**Reserved** Access: RO Format: MBZ
	18:0	**Gamma Value** Default Value: 00000000000000000b Format: U3.16

PRE_CSC_GAMC_INDEX

PRE_CSC_GAMC_INDEX

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	4A484h-4A487h
Name:	Pipe Pre CSC Gamma Index
ShortName:	PRE_CSC_GAMC_INDEX_A
Reset:	soft
Address:	4AC84h-4AC87h
Name:	Pipe Pre CSC Gamma Index
ShortName:	PRE_CSC_GAMC_INDEX_B
Reset:	soft
Address:	4B484h-4B487h
Name:	Pipe Pre CSC Gamma Index
ShortName:	PRE_CSC_GAMC_INDEX_C
Reset:	soft
Address:	4BC84h-4BC87h
Name:	Pipe Pre CSC Gamma Index
ShortName:	PRE_CSC_GAMC_INDEX_D
Reset:	soft

DWord	Bit	Description
0	31:11	**Reserved** Access: RO Format: MBZ
	10	**Index Auto Increment** This field enables the index auto increment. Value 0b: No Increment — Do not automatically increment the index value. Value 1b: Auto Increment **[Default]** — Increment the index value with each read or write to the data register.
	9:8	**Reserved** Access: RO Format: MBZ
	7:6	**Reserved** Access: RO Format: MBZ

PRE_CSC_GAMC_INDEX

	5:0	**Index Value** Access: Write/Read Status This index controls access to the array of pipe pre color space conversion gamma values. This value can be automatically incremented by a read or a write to the data register if the index auto increment bit is set. When automatically incrementing, the index will roll over to 0 after writing or reading the entire allowed range. While in auto increment mode, after performing reads or writes to only part of the range, the auto increment bit must be cleared before resetting the index value. **Value** / **Name** [0,34]

Predicate Rendering Data Result

MI_PREDICATE_RESULT - Predicate Rendering Data Result		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
_Custom_GTIReset:	DEV	
DWord	**Bit**	**Description**
0	31:1	**Reserved** Format: PBC
	0	**MI_PREDICATE_RESULT** This bit is the result of the last MI_PREDICATE.

Predicate Rendering Data Result 1

MI_PREDICATE_RESULT_1 - Predicate Rendering Data Result 1	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0241Ch-0241Fh
Name:	Predicate Rendering Data Result 1
ShortName:	MI_PREDICATE_RESULT_1_RCSUNIT
Address:	1841Ch-1841Fh
Name:	Predicate Rendering Data Result 1
ShortName:	MI_PREDICATE_RESULT_1_POCSUNIT
Address:	2241Ch-2241Fh
Name:	Predicate Rendering Data Result 1
ShortName:	MI_PREDICATE_RESULT_1_BCSUNIT
Address:	1C041Ch-1C041Fh
Name:	Predicate Rendering Data Result 1
ShortName:	MI_PREDICATE_RESULT_1_VCSUNIT0
Address:	1C441Ch-1C441Fh
Name:	Predicate Rendering Data Result 1
ShortName:	MI_PREDICATE_RESULT_1_VCSUNIT1
Address:	1C841Ch-1C841Fh
Name:	Predicate Rendering Data Result 1
ShortName:	MI_PREDICATE_RESULT_1_VECSUNIT0
Address:	1D041Ch-1D041Fh
Name:	Predicate Rendering Data Result 1
ShortName:	MI_PREDICATE_RESULT_1_VCSUNIT2
Address:	1D441Ch-1D441Fh
Name:	Predicate Rendering Data Result 1
ShortName:	MI_PREDICATE_RESULT_1_VCSUNIT3
Address:	1D841Ch-1D841Fh
Name:	Predicate Rendering Data Result 1
ShortName:	MI_PREDICATE_RESULT_1_VECSUNIT1
Address:	1E041Ch-1E041Fh
Name:	Predicate Rendering Data Result 1
ShortName:	MI_PREDICATE_RESULT_1_VCSUNIT4

MI_PREDICATE_RESULT_1 - Predicate Rendering Data Result 1

Address:	1E441Ch-1E441Fh
Name:	Predicate Rendering Data Result 1
ShortName:	MI_PREDICATE_RESULT_1_VCSUNIT5

Address:	1E841Ch-1E841Fh
Name:	Predicate Rendering Data Result 1
ShortName:	MI_PREDICATE_RESULT_1_VECSUNIT2

Address:	1F041Ch-1F041Fh
Name:	Predicate Rendering Data Result 1
ShortName:	MI_PREDICATE_RESULT_1_VCSUNIT6

Address:	1F441Ch-1F441Fh
Name:	Predicate Rendering Data Result 1
ShortName:	MI_PREDICATE_RESULT_1_VCSUNIT7

Address:	1F841Ch-1F841Fh
Name:	Predicate Rendering Data Result 1
ShortName:	MI_PREDICATE_RESULT_1_VECSUNIT3

Address:	1A41Ch-1A41Fh
Name:	Predicate Rendering Data Result 1
ShortName:	MI_PREDICATE_RESULT_1_CCSUNIT0

DWord	Bit	Description
0	31:1	**Reserved** Format: PBC
	0	**MI_PREDICATE_RESULT_1** **Description** This bit is used to predicate MI_BATCH_BUFFER_START commands in the RCS command stream. Usage Model: MI_MATH command will be used to do some ALU operations over GPR followed by a MI_LOAD_REGISTER_REGISTER to move the result from GPR to MI_PREDICATE_RESULT_1.

Predicate Rendering Data Result 2

MI_PREDICATE_RESULT_2 - Predicate Rendering Data Result 2	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	023BCh-023BFh
Name:	Predicate Rendering Data Result 2
ShortName:	MI_PREDICATE_RESULT_2_RCSUNIT
Address:	183BCh-183BFh
Name:	Predicate Rendering Data Result 2
ShortName:	MI_PREDICATE_RESULT_2_POCSUNIT
Address:	223BCh-223BFh
Name:	Predicate Rendering Data Result 2
ShortName:	MI_PREDICATE_RESULT_2_BCSUNIT
Address:	1C03BCh-1C03BFh
Name:	Predicate Rendering Data Result 2
ShortName:	MI_PREDICATE_RESULT_2_VCSUNIT0
Address:	1C43BCh-1C43BFh
Name:	Predicate Rendering Data Result 2
ShortName:	MI_PREDICATE_RESULT_2_VCSUNIT1
Address:	1C83BCh-1C83BFh
Name:	Predicate Rendering Data Result 2
ShortName:	MI_PREDICATE_RESULT_2_VECSUNIT0
Address:	1D03BCh-1D03BFh
Name:	Predicate Rendering Data Result 2
ShortName:	MI_PREDICATE_RESULT_2_VCSUNIT2
Address:	1D43BCh-1D43BFh
Name:	Predicate Rendering Data Result 2
ShortName:	MI_PREDICATE_RESULT_2_VCSUNIT3
Address:	1D83BCh-1D83BFh
Name:	Predicate Rendering Data Result 2
ShortName:	MI_PREDICATE_RESULT_2_VECSUNIT1
Address:	1E03BCh-1E03BFh
Name:	Predicate Rendering Data Result 2
ShortName:	MI_PREDICATE_RESULT_2_VCSUNIT4

MI_PREDICATE_RESULT_2 - Predicate Rendering Data Result 2

Address:	1E43BCh-1E43BFh
Name:	Predicate Rendering Data Result 2
ShortName:	MI_PREDICATE_RESULT_2_VCSUNIT5

Address:	1E83BCh-1E83BFh
Name:	Predicate Rendering Data Result 2
ShortName:	MI_PREDICATE_RESULT_2_VECSUNIT2

Address:	1F03BCh-1F03BFh
Name:	Predicate Rendering Data Result 2
ShortName:	MI_PREDICATE_RESULT_2_VCSUNIT6

Address:	1F43BCh-1F43BFh
Name:	Predicate Rendering Data Result 2
ShortName:	MI_PREDICATE_RESULT_2_VCSUNIT7

Address:	1F83BCh-1F83BFh
Name:	Predicate Rendering Data Result 2
ShortName:	MI_PREDICATE_RESULT_2_VECSUNIT3

Address:	1A3BCh-1A3BFh
Name:	Predicate Rendering Data Result 2
ShortName:	MI_PREDICATE_RESULT_2_CCSUNIT0

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**MI_PREDICATE_RESULT_2** This bit must be loaded with by SW based on GT mode of operation. This register must be loaded appropriately before using MI_SET_PREDICATE command. 0h **[Default]** Indicates GT2 mode and lower slice is disabled. 1h Indicates GT3 mode and lower slice is enabled.

Predicate Rendering Data Storage

MI_PREDICATE_DATA - Predicate Rendering Data Storage		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	64	
_Custom_GTIReset:	DEV	
DWord	**Bit**	**Description**
0	63:32	**MI_PREDICATE_DATA_UDW** This register is used either as computed value based off the MI_PREDICATE_SRC0 and MI_PREDICATE_SRC1 or a temporary register. See Predicate Rendering section for more details.
	31:0	**MI_PREDICATE_DATA_LDW** This register is used either as computed value based off the MI_PREDICATE_SRC0 and MI_PREDICATE_SRC1 or a temporary register. See Predicate Rendering section for more details.

Predicate Rendering Temporary Register0

MI_PREDICATE_SRC0 - Predicate Rendering Temporary Register0		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	64	
_Custom_GTIReset:	DEV	
DWord	**Bit**	**Description**
0	63:0	**MI_PREDICATE_SRC0** This register is a temporary register for Predicate Rendering. See Predicate Rendering section for more details.

Predicate Rendering Temporary Register1

MI_PREDICATE_SRC1 - Predicate Rendering Temporary Register1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV

DWord	Bit	Description
0	63:0	**MI_PREDICATE_SRC1** This register is a temporary register for Predicate Rendering. See Predicate Rendering section for more details.

Preemption Status

PREEMPTION_STATUS - Preemption Status	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	025ACh-025AFh
Name:	PREEMPTION_STATUS
ShortName:	PREEMPTION_STATUS_RCSUNIT
Address:	185ACh-185AFh
Name:	PREEMPTION_STATUS
ShortName:	PREEMPTION_STATUS_POCSUNIT
Address:	225ACh-225AFh
Name:	PREEMPTION_STATUS
ShortName:	PREEMPTION_STATUS_BCSUNIT
Address:	1C05ACh-1C05AFh
Name:	PREEMPTION_STATUS
ShortName:	PREEMPTION_STATUS_VCSUNIT0
Address:	1C45ACh-1C45AFh
Name:	PREEMPTION_STATUS
ShortName:	PREEMPTION_STATUS_VCSUNIT1
Address:	1C85ACh-1C85AFh
Name:	PREEMPTION_STATUS
ShortName:	PREEMPTION_STATUS_VECSUNIT0
Address:	1D05ACh-1D05AFh
Name:	PREEMPTION_STATUS
ShortName:	PREEMPTION_STATUS_VCSUNIT2
Address:	1D45ACh-1D45AFh
Name:	PREEMPTION_STATUS
ShortName:	PREEMPTION_STATUS_VCSUNIT3
Address:	1D85ACh-1D85AFh
Name:	PREEMPTION_STATUS
ShortName:	PREEMPTION_STATUS_VECSUNIT1
Address:	1E05ACh-1E05AFh
Name:	PREEMPTION_STATUS
ShortName:	PREEMPTION_STATUS_VCSUNIT4

PREEMPTION_STATUS - Preemption Status

Address:	1E45ACh-1E45AFh
Name:	PREEMPTION_STATUS
ShortName:	PREEMPTION_STATUS_VCSUNIT5

Address:	1E85ACh-1E85AFh
Name:	PREEMPTION_STATUS
ShortName:	PREEMPTION_STATUS_VECSUNIT2

Address:	1F05ACh-1F05AFh
Name:	PREEMPTION_STATUS
ShortName:	PREEMPTION_STATUS_VCSUNIT6

Address:	1F45ACh-1F45AFh
Name:	PREEMPTION_STATUS
ShortName:	PREEMPTION_STATUS_VCSUNIT7

Address:	1F85ACh-1F85AFh
Name:	PREEMPTION_STATUS
ShortName:	PREEMPTION_STATUS_VECSUNIT3

Address:	1A5ACh-1A5AFh
Name:	PREEMPTION_STATUS
ShortName:	PREEMPTION_STATUS_CCSUNIT0

This register captures the context switch status on a context switch and gets saved as part of the context image. This register doesn't get restored on a context restore. For all functional purposes this register must be red or sampled form the context image in memory and must not be sampled live from reading to this register.
This register is for HW internal purpose and must not be programmed by SW.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:13	**CS Engine ID** This field holds the EngineID of the command streamer on which the context got recently executed.
	12:7	**CS Instance ID** This field holds the InstanceID of the command streamer on which the context got recently executed.
	6:4	**Reserved** Access: RO Format: MBZ
	3:2	**Context Switch Status** This field indicates the preemption boundary and must be inferred based on the value in "Pipeline Selection" field.

PREEMPTION_STATUS - Preemption Status

Value	Name	Description	Exists If
0h		**Command Boundary:** Context switched out on a clean command boundary with no work pending from the commands executed.	
1h		**Thread Group Boundary:** Context got switched out in middle of workload (Media/GPGPU Walker) execution in Media or GPGPU mode of operation. Workload got preempted on a thread group boundary. HW will resume from the point it got preempted on resubmission of the context.	//PipelineSelection != 0
1h		**Object Boundary:** Context got switch out in middle of workload (3DPRIMTIVE) execution in 3D mode of operation. Workload got preempted on an object level boundary. HW will resume from the point it got preempted on resubmission of the context.	//PipelineSelection == 0
2h		**Mid Thread Boundary:**Context got switched out in middle of workload (Media/GPGPU Walker) execution in Media or GPGPU mode of operation. Workload got preempted in middle of thread execution (mid-thread boundary). HW will resume from the point it got preempted on resubmission of the context.	//PipelineSelection != 0
3h	Reserved		

1:0 **Pipeline Selection**

Access:	RO
Format:	U2

This bit indicates the pipeline select mode (PIPELINE_SELECT) at the time of context switch.

Value	Name	Description
0h		3D mode of operation.
1h		Media mode of operation.
2h		GPGPU mode of operation.
3h	Reserved	

PRERESETMessagingRegister

MSG_RESET_PRE_GCP - PRERESETMessagingRegister

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 0866Ch

PRE RESET Messaging Register for Clocking Unit

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29	**Request to Prepare for GSC Reset** Access: R/W _Custom_GTIReset: BUS Prepare for cgscrst_b Domain Reset (GSC).
	28:21	**Reserved** Access: RO Format: MBZ
	20	**Request to Prepare for VEBox3 Reset** Access: R/W _Custom_GTIReset: BUS Prepare for cvrst_b Domain Reset (vecs3unit).
	19	**Request to Prepare for VEBox2 Reset** Access: R/W _Custom_GTIReset: BUS Prepare for cvrst_b Domain Reset (vecs2unit).
	18	**Request to Prepare for VEBox1 Reset** Access: R/W _Custom_GTIReset: BUS Prepare for cvrst_b Domain Reset (vecs1unit).
	17	**Request to Prepare for VEBox0 Reset** Access: R/W _Custom_GTIReset: BUS Prepare for cvrst_b Domain Reset (vecs0unit).
	16	**Request to Prepare for SFC3 Reset** Access: R/W _Custom_GTIReset: BUS Prepare for csfcrst_b Domain Reset (vecs0unit).

MSG_RESET_PRE_GCP - PRERESETMessagingRegister

	15	**Request to Prepare for SFC2 Reset** Access: R/W _Custom_GTIReset: BUS Prepare for csfcrst_b Domain Reset (vecs0unit).
	14	**Request to Prepare for SFC1 Reset** Access: R/W _Custom_GTIReset: BUS Prepare for csfcrst_b Domain Reset (vecs0unit).
	13	**Request to Prepare for SFC0 Reset** Access: R/W _Custom_GTIReset: BUS Prepare for csfcrst_b Domain Reset (vecs0unit).
	12	**Request to Prepare for Media7 Reset** Access: R/W _Custom_GTIReset: BUS Prepare for cmrst_b Domain Reset (vcs7unit).
	11	**Request to Prepare for Media6 Reset** Access: R/W _Custom_GTIReset: BUS Prepare for cmrst_b Domain Reset (vcs6unit).
	10	**Request to Prepare for Media5 Reset** Access: R/W _Custom_GTIReset: BUS Prepare for cmrst_b Domain Reset (vcs5unit).
	9	**Request to Prepare for Media4 Reset** Access: R/W _Custom_GTIReset: BUS Prepare for cmrst_b Domain Reset (vcs4unit).
	8	**Request to Prepare for Media3 Reset** Access: R/W _Custom_GTIReset: BUS Prepare for cmrst_b Domain Reset (vcs3unit).
	7	**Request to Prepare for Media2 Reset** Access: R/W _Custom_GTIReset: BUS Prepare for cmrst_b Domain Reset (vcs2unit).
	6	**Request to Prepare for Media1 Reset** Access: R/W

MSG_RESET_PRE_GCP - PRERESETMessagingRegister

		_Custom_GTIReset: BUS Prepare for cmrst_b Domain Reset (vcs1unit).
	5	**Request to Prepare for Media0 Reset** Access: R/W _Custom_GTIReset: BUS Prepare for cmrst_b Domain Reset (vcs0unit).
	4	**Request to Prepare for Render Reset** Access: R/W _Custom_GTIReset: BUS Prepare for crrst_b Domain Reset (csunit).
	3	**Reserved**
	2	**Request to Prepare for Blitter Reset** Access: R/W _Custom_GTIReset: BUS Prepare for crblitrst_b Domain Reset (bcsunit).
	1	**Request to Prepare for Full Reset** Access: R/W _Custom_GTIReset: BUS Prepare for all the soft Domain Reset (csunit).
	0	**Request to Prepare for FLR** Access: R/W _Custom_GTIReset: BUS Prepare for devrst_b Domain Reset (FLR) Note: All resets except busrst_b are asserted for an FLR.

Primary Latency Timer

MLT2_0_2_0_PCI - Primary Latency Timer		
Register Space:	PCI: 0/2/0	
Size (in bits):	8	
Address:	0000Dh	
The IGD does not support the programmability of the primary latency timer because it does not perform bursts.		
DWord	**Bit**	**Description**
0	7:0	**Primary Latency Timer Count Value** Default Value: 00000000b Access: RO _Custom_GTIReset: BUS Hardwired to 0s.

Primitives Generated By VF

IA_PRIMITIVES_COUNT - Primitives Generated By VF	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02318h-0231Fh
Name:	Primitives Generated By VF
ShortName:	IA_PRIMITIVES_COUNT_RCSUNIT_BE_GEOMETRY
Address:	18318h-1831Fh
Name:	Primitives Generated By VF
ShortName:	IA_PRIMITIVES_COUNT_POCSUNIT_BE_GEOMETRY

This register stores the count of primitives generated by VF. This register is part of the context save and restore.

DWord	Bit	Description
0	63:32	**IA Primitives Count Report UDW** Total number of primitives output by the Vertex Fetch (IA) stage. This count is updated for every primitive output by the VF stage, as long as Statistics Enable is set in VF_STATE (see the Vertex Fetch Chapter in the 3D Volume.)
	31:0	**IA Primitives Count Report LDW** Total number of primitives output by the Vertex Fetch (IA) stage. This count is updated for every primitive output by the VF stage, as long as Statistics Enable is set in VF_STATE (see the Vertex Fetch Chapter in the 3D Volume.)

PS_ADAPTIVE_CTRL

PS_ADAPTIVE_CTRL	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PS_WIN_SZ
_Custom_Display_DoubleBufferUpdatePoint:	Start of horizontal blank after armed
Address:	681A8h-681ABh
Name:	PS Adaptive Control Set 0 1
ShortName:	PS_ADAPTIVE_CTRL_SET_0_1_A
Reset:	soft
Address:	681ACh-681AFh
Name:	PS Adaptive Control Set 1 1
ShortName:	PS_ADAPTIVE_CTRL_SET_1_1_A
Reset:	soft
Address:	682A8h-682ABh
Name:	PS Adaptive Control Set 0 1
ShortName:	PS_ADAPTIVE_CTRL_SET_0_2_A
Reset:	soft
Address:	682ACh-682AFh
Name:	PS Adaptive Control Set 1 1
ShortName:	PS_ADAPTIVE_CTRL_SET_1_2_A
Reset:	soft
Address:	689A8h-689ABh
Name:	PS Adaptive Control Set 0 1
ShortName:	PS_ADAPTIVE_CTRL_SET_0_1_B
Reset:	soft
Address:	689ACh-689AFh
Name:	PS Adaptive Control Set 1 1
ShortName:	PS_ADAPTIVE_CTRL_SET_1_1_B
Reset:	soft
Address:	68AA8h-68AABh
Name:	PS Adaptive Control Set 0 1
ShortName:	PS_ADAPTIVE_CTRL_SET_0_2_B
Reset:	soft
Address:	68AACh-68AAFh

PS_ADAPTIVE_CTRL

Name:	PS Adaptive Control Set 1 1
ShortName:	PS_ADAPTIVE_CTRL_SET_1_2_B
Reset:	soft

Address:	691A8h-691ABh
Name:	PS Adaptive Control Set 0 1
ShortName:	PS_ADAPTIVE_CTRL_SET_0_1_C
Reset:	soft

Address:	691ACh-691AFh
Name:	PS Adaptive Control Set 1 1
ShortName:	PS_ADAPTIVE_CTRL_SET_1_1_C
Reset:	soft

Address:	692A8h-692ABh
Name:	PS Adaptive Control Set 0 1
ShortName:	PS_ADAPTIVE_CTRL_SET_0_2_C
Reset:	soft

Address:	692ACh-692AFh
Name:	PS Adaptive Control Set 1 1
ShortName:	PS_ADAPTIVE_CTRL_SET_1_2_C
Reset:	soft

Address:	699A8h-699ABh
Name:	PS Adaptive Control Set 0 1
ShortName:	PS_ADAPTIVE_CTRL_SET_0_1_D
Reset:	soft

Address:	699ACh-699AFh
Name:	PS Adaptive Control Set 1 1
ShortName:	PS_ADAPTIVE_CTRL_SET_1_1_D
Reset:	soft

Address:	69AA8h-69AABh
Name:	PS Adaptive Control Set 0 1
ShortName:	PS_ADAPTIVE_CTRL_SET_0_2_D
Reset:	soft

Address:	69AACh-69AAFh
Name:	PS Adaptive Control Set 1 1
ShortName:	PS_ADAPTIVE_CTRL_SET_1_2_D
Reset:	soft

Programming Notes

PS_ADAPTIVE_CTRL

Recommended threshold programming:
Threshold 1: 1Eh
Threshold 2: 2Dh
Threshold 3: 3Ch

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**Threshold 3** This field specifies the third threshold value used in adaptive filtering.
	15:8	**Threshold 2** This field specifies the second threshold value used in adaptive filtering.
	7:0	**Threshold 1** This field specifies the first threshold value used in adaptive filtering.

PS_COEF_DATA

PS_COEF_DATA	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PS_WIN_SZ
_Custom_Display_DoubleBufferUpdatePoint:	Start of horizontal blank after armed
Address:	6819Ch-6819Fh
Name:	PS Coeffecient Set 0 Data 1
ShortName:	PS_COEF_SET_0_DATA_1_A
Reset:	soft
Address:	681A4h-681A7h
Name:	PS Coeffecient Set 1 Data 1
ShortName:	PS_COEF_SET_1_DATA_1_A
Reset:	soft
Address:	6829Ch-6829Fh
Name:	PS Coeffecient Set 0 Data 1
ShortName:	PS_COEF_SET_0_DATA_2_A
Reset:	soft
Address:	682A4h-682A7h
Name:	PS Coeffecient Set 1 Data 1
ShortName:	PS_COEF_SET_1_DATA_2_A
Reset:	soft
Address:	6899Ch-6899Fh
Name:	PS Coeffecient Set 0 Data 1
ShortName:	PS_COEF_SET_0_DATA_1_B
Reset:	soft
Address:	689A4h-689A7h
Name:	PS Coeffecient Set 1 Data 1
ShortName:	PS_COEF_SET_1_DATA_1_B
Reset:	soft
Address:	68A9Ch-68A9Fh
Name:	PS Coeffecient Set 0 Data 1
ShortName:	PS_COEF_SET_0_DATA_2_B
Reset:	soft
Address:	68AA4h-68AA7h

PS_COEF_DATA	
Name:	PS Coeffecient Set 1 Data 1
ShortName:	PS_COEF_SET_1_DATA_2_B
Reset:	soft
Address:	6919Ch-6919Fh
Name:	PS Coeffecient Set 0 Data 1
ShortName:	PS_COEF_SET_0_DATA_1_C
Reset:	soft
Address:	691A4h-691A7h
Name:	PS Coeffecient Set 1 Data 1
ShortName:	PS_COEF_SET_1_DATA_1_C
Reset:	soft
Address:	6929Ch-6929Fh
Name:	PS Coeffecient Set 0 Data 1
ShortName:	PS_COEF_SET_0_DATA_2_C
Reset:	soft
Address:	692A4h-692A7h
Name:	PS Coeffecient Set 1 Data 1
ShortName:	PS_COEF_SET_1_DATA_2_C
Reset:	soft
Address:	6999Ch-6999Fh
Name:	PS Coeffecient Set 0 Data 1
ShortName:	PS_COEF_SET_0_DATA_1_D
Reset:	soft
Address:	699A4h-699A7h
Name:	PS Coeffecient Set 1 Data 1
ShortName:	PS_COEF_SET_1_DATA_1_D
Reset:	soft
Address:	69A9Ch-69A9Fh
Name:	PS Coeffecient Set 0 Data 1
ShortName:	PS_COEF_SET_0_DATA_2_D
Reset:	soft
Address:	69AA4h-69AA7h
Name:	PS Coeffecient Set 1 Data 1
ShortName:	PS_COEF_SET_1_DATA_2_D
Reset:	soft
These are the coefficient values for scaler.	

PS_COEF_DATA

The scaler coefficient Index indicates the coefficients array location to be accessed through this register.
The contents of the coefficient array is uninitialized until Software loads the array (i.e. the array is not resettable). Use of the coefficient array or reading from the coefficient array before Software has initialized it will result in non-deterministic behavior or read back data.

Restriction
This register must be written only as a full 32 bit dword. Byte or word writes are not supported.

DWord	Bit	Description
0	31:16	**Coefficient2** Format: SCALER_COEFFICIENT_FORMAT Specifies the value for the second coefficient stored in this dword.
	15:0	**Coefficient1** Format: SCALER_COEFFICIENT_FORMAT Specifies the value for the first coefficient stored in this dword.

PS_COEF_INDEX

PS_COEF_INDEX	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	68198h-6819Bh
Name:	PS Coeffecient Set 0 Index 1
ShortName:	PS_COEF_SET_0_INDEX_1_A
Reset:	soft
Address:	681A0h-681A3h
Name:	PS Coeffecient Set 1 Index 1
ShortName:	PS_COEF_SET_1_INDEX_1_A
Reset:	soft
Address:	68298h-6829Bh
Name:	PS Coeffecient Set 0 Index 1
ShortName:	PS_COEF_SET_0_INDEX_2_A
Reset:	soft
Address:	682A0h-682A3h
Name:	PS Coeffecient Set 1 Index 1
ShortName:	PS_COEF_SET_1_INDEX_2_A
Reset:	soft
Address:	68998h-6899Bh
Name:	PS Coeffecient Set 0 Index 1
ShortName:	PS_COEF_SET_0_INDEX_1_B
Reset:	soft
Address:	689A0h-689A3h
Name:	PS Coeffecient Set 1 Index 1
ShortName:	PS_COEF_SET_1_INDEX_1_B
Reset:	soft
Address:	68A98h-68A9Bh
Name:	PS Coeffecient Set 0 Index 1
ShortName:	PS_COEF_SET_0_INDEX_2_B
Reset:	soft
Address:	68AA0h-68AA3h
Name:	PS Coeffecient Set 1 Index 1
ShortName:	PS_COEF_SET_1_INDEX_2_B

PS_COEF_INDEX

Reset:	soft
Address:	69198h-6919Bh
Name:	PS Coeffecient Set 0 Index 1
ShortName:	PS_COEF_SET_0_INDEX_1_C
Reset:	soft
Address:	691A0h-691A3h
Name:	PS Coeffecient Set 1 Index 1
ShortName:	PS_COEF_SET_1_INDEX_1_C
Reset:	soft
Address:	69298h-6929Bh
Name:	PS Coeffecient Set 0 Index 1
ShortName:	PS_COEF_SET_0_INDEX_2_C
Reset:	soft
Address:	692A0h-692A3h
Name:	PS Coeffecient Set 1 Index 1
ShortName:	PS_COEF_SET_1_INDEX_2_C
Reset:	soft
Address:	69998h-6999Bh
Name:	PS Coeffecient Set 0 Index 1
ShortName:	PS_COEF_SET_0_INDEX_1_D
Reset:	soft
Address:	699A0h-699A3h
Name:	PS Coeffecient Set 1 Index 1
ShortName:	PS_COEF_SET_1_INDEX_1_D
Reset:	soft
Address:	69A98h-69A9Bh
Name:	PS Coeffecient Set 0 Index 1
ShortName:	PS_COEF_SET_0_INDEX_2_D
Reset:	soft
Address:	69AA0h-69AA3h
Name:	PS Coeffecient Set 1 Index 1
ShortName:	PS_COEF_SET_1_INDEX_2_D
Reset:	soft

DWord	Bit	Description	
0	31:11	**Reserved**	
		Access:	RO

PS_COEF_INDEX

	Format:	MBZ

10 Index Auto Increment

Access:	R/W

This field enables the index auto increment.

Value	Name	Description
0b	No Increment	Do not automatically increment the index value.
1b	Auto Increment **[Default]**	Increment the index value with each read or write to the data register.

9:6 Reserved

Access:	RO
Format:	MBZ

5:0 Index Value

Access:	R/W

This index controls access to the array of scaler coefficient values.

Value	Name
[0,59]	

PS_CTRL

PS_CTRL	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PS_WIN_SZ
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank after armed
Address:	68180h-68183h
Name:	PS Control 1
ShortName:	PS_CTRL_1_A
Reset:	soft
Address:	68280h-68283h
Name:	PS Control 1
ShortName:	PS_CTRL_2_A
Reset:	soft
Address:	68980h-68983h
Name:	PS Control 1
ShortName:	PS_CTRL_1_B
Reset:	soft
Address:	68A80h-68A83h
Name:	PS Control 1
ShortName:	PS_CTRL_2_B
Reset:	soft
Address:	69180h-69183h
Name:	PS Control 1
ShortName:	PS_CTRL_1_C
Reset:	soft
Address:	69280h-69283h
Name:	PS Control 1
ShortName:	PS_CTRL_2_C
Reset:	soft
Address:	69980h-69983h
Name:	PS Control 1
ShortName:	PS_CTRL_1_D
Reset:	soft
Address:	69A80h-69A83h

PS_CTRL

Name:	PS Control 1
ShortName:	PS_CTRL_2_D
Reset:	soft

Description

The pipe scalers are used to scale the output of a display pipe or of a display plane. All pipes have two scalers each.
The scaler preserves 8 bits of alpha and 10 bits of each color channel for plane scaling and 12 bits of each color channel for pipe scaling.
The scalers can be assigned to any plane (except cursor) output or the output of the display pipe (after blending and color correction, before dithering and color clamping).
Downscale usages have scale factor restrictions:

- All scaler modes support a downscale factor of less than 3.0 in each direction.
- Pipe YUV 420 encoding for port output supports Y downscale factor of less than 1.5 in each direction.

Beyond the restrictions of the Scaler output fitting within the destination window size, there are effectively no upscale restrictions except for the following:
(Scale Factor) * 2^{15} >= 1.0
Where the Scale Factor = (Source Size) / (Destination Size)

The scalers support horizontal source sizes up to 5120 and vertical source sizes up to 4096.

Programming Notes

The scalers must not be enabled when the horizontal source sizes are greater than 5120 and the vertical sizes greater than 4320.

Driver is responsible for making sure all the plane, pipe, and scaler size registers are programmed appropriately and gets applied atomically to the same frame since hardware does not ensure an atomic update of plane, scaler, and pipe source size registers.

When scaling a pipe, the scaler window size and position must fit within the pipe active size. If there is a seam present (i.e. PIPE_SEAM_EXCESS is non-zero), then the pipe's horizontal active size that the scaler sees is the horizontal active size defined within the TRANS_HTOTAL register plus the amount(s) specified within the PIPE_SEAM_EXCESS.
Pipe Horizontal Active = Horizontal Active + Left Excess Amount + Right Excess Amount

Refer to 'YUV 420 Support' page for scaler restrictions with YUV 420 pipe output.

Restriction

Down scaling (scaler input size is larger than scaler window size) can reduce the maximum supported pixel rate for a pipe as well as increase the watermark and data buffer requirements. Refer to the Display Resolution Support page and Watermark Calculations page for detailed calculations.

Scaler 1 and 2 must not be both scaling the same plane output.
When scaling a pipe, the scaler window size and position must fit within the pipe active size.
When scaling a plane, the plane position must be programmed to 0 and the scaler window size and position must fit within the pipe source size.
When scaling is enabled, the scaler input width should be a minimum of 8 pixels and the height should be minimum of 8 scanlines.

PS_CTRL

When the plane scaling is used with YUV 420 planar formats, the height should be a minimum of 16 scanlines. When using down scaling (scaler input size is larger than scaler output size) the maximum supported pixel rate will be reduced by the down scale amount.

DWord	Bit	Description
0	31	**Enable Scaler** This field enables the scaler. Value: 0b = Disable; 1b = Enable
	30	**Reserved**
	29	**Scaler Mode** The Planar mode must be used when non-HDR capable planes are using YUV 420 planar surface formats. In this configuration, the scaler takes care of both the chroma upsampling and scaling. The Planar mode must not be used for the HDR capable planes. They have dedicated chroma upsamplers and only use the scaler in Normal mode for scaling. Value: 0b = Normal; 1b = Planar
	28	**Adaptive Filtering** This field enables the scaler adaptive vertical and horizontal filtering. When adaptive filtering is enabled, the adaptive threshold values must be programmed in the PS_ADAPTIVE_CTRL register and the Filter Set Select bits should be programmed. Value: 0h = Disable; 1h = Enable
	27:25	**Scaler Binding** This field selects the where the scaling operation is done. When scaling a pipe, the pipe source size specifies the input size to the scaler. When scaling a plane, the PLANE_SIZE specifies the input size to the scaler. Any border around a scaled plane window will become transparent at the plane blender. Value: 000b = Pipe Scaler; 001b = Plane 1 Scaler; 010b = Plane 2 Scaler; 011b = Plane 3 Scaler; 100b = Plane 4 Scaler; 101b = Plane 5 Scaler; 110b = Plane 6 Scaler; 111b = Plane 7 Scaler

PS_CTRL

Programming Notes
When plane scaling is enabled on planes 1 through 3, make sure that the *PLANE_CUS_CTL.Plane Scaling Enabled* (bit 30) is programmed correctly.

Restriction
The scaler input size should be at least 8 scanlines. Plane/Pipe scaling is not compatible with interlaced fetch mode. Plane up and down scaling is not compatible with keying. Keying can be enabled with 1:1 plane scaling. Plane scaling is not compatible with the Indexed 8-bit, XR_BIAS, or any pixel values less than 0 or greater than 1.

24:23 **FILTER SELECT**

This field selects filter coefficients. The medium coefficients will provide an unfiltered image when the scale factor is 1:1.
In the programmed mode, the filter coefficients must be programmed using the PS_COEF_INDEX and PS_COEF_DATA registers and the Filter Set Select bits should be programmed.

Value	Name
00b	Medium
01b	Programmed
10b	Edge Enhance
11b	Bilinear

22 **ADAPTIVE FILTER SELECT**

This field selects the filter coefficients used for adaptive filtering. The field is ignored when adaptive filtering is not enabled.

Value	Name
0b	Medium
1b	Edge Enhance

21 **Pipe Scaler Location**

This field selects where the pipe scaling is done in the pipe.

Value	Name	Description
0b	After Output CSC	This is a non-linear tap point
1b	After CSC	This is a linear tap point

Restriction
The HDR Mode cannot be enabled for the Pipe(PIPE_MISC[23] = 0), when the Scaler is bound to the linear tap point

20 **Reserved**

PS_CTRL

19	**Reserved**
18	**Reserved** Access: RO Format: MBZ
17	**Reserved**
16	**Reserved** Access: RO Format: MBZ
15	**Reserved**
14	**Reserved** Access: RO Format: MBZ
13:12	**Reserved**
11:10	**Reserved** Access: RO Format: MBZ
9	**Allow Double Buffer Update Disable** Access: R/W This field controls whether double buffer updates are allowed to be disabled for this scaler. The DOUBLE_BUFFER_CTL register can be configured to globally disable double buffer updates for resources that allow them to be disabled.

Value	Name
0b	Not Allowed
1b	Allowed **[Default]**

8	**Reserved**
7:5	**Scaler Binding Y** This field selects where the planar YUV420 Y plane scaling operation is done. This field is ignored if planar YUV420 plane scaling is not used. This field is used only for planes 4-7 when the plane scaler is used for chroma upsampling. Planes 1-3 must use the dedicated chroma up sampler (programmed in PLANE_CUS_CTL)for YUV 444 up conversion.

Value	Name
110b	Plane 6 Scaler
111b	Plane 7 Scaler

Restriction
The scaler input size should be at least 16 scanlines.

PS_CTRL

	4	**Y Vert Filter Set Sel** This field selects the programmed coefficient set and/or the adaptive threshold set used by the Y component vertical filter when filtering YUV planar formats. This field is ignored with other formats. **Value** / **Name** 0b — Set 0 **[Default]** 1b — Set 1
	3	**Y Horz Filter Set Sel** This field selects the programmed coefficient set and/or the adaptive threshold set used by the Y component horizontal filter when filtering YUV hybrid planar formats. This field is ignored with other formats. **Value** / **Name** 0b — Set 0 **[Default]** 1b — Set 1
	2	**UV Vert Filter Set Sel** This field selects the programmed coefficient set and/or the adaptive threshold set used by the UV component vertical filter when filtering YUV hybrid planar formats. With other formats, this field selects the coefficient set and/or the adaptive threshold set used by the vertical filter. **Value** / **Name** 0b — Set 0 **[Default]** 1b — Set 1
	1	**UV Horz Filter Set Sel** This field selects the programmed coefficient set and/or the adaptive threshold set used by the UV component horizontal filter when filtering YUV hybrid planar formats. With other formats, this field selects the coefficient set and/or the adaptive threshold set used by the horizontal filter. **Value** / **Name** 0b — Set 0 **[Default]** 1b — Set 1
	0	**Reserved** Access: RO Format: MBZ

PS_ECC_STAT

PS_ECC_STAT	
Register Space:	MMIO: 0/2/0
Access:	R/WC
Size (in bits):	32
Address:	681D0h-681D3h
Name:	PS ECC Status 1
ShortName:	PS_ECC_STAT_1_A
Reset:	soft
Address:	682D0h-682D3h
Name:	PS ECC Status 1
ShortName:	PS_ECC_STAT_2_A
Reset:	soft
Address:	689D0h-689D3h
Name:	PS ECC Status 1
ShortName:	PS_ECC_STAT_1_B
Reset:	soft
Address:	68AD0h-68AD3h
Name:	PS ECC Status 1
ShortName:	PS_ECC_STAT_2_B
Reset:	soft
Address:	691D0h-691D3h
Name:	PS ECC Status 1
ShortName:	PS_ECC_STAT_1_C
Reset:	soft
Address:	692D0h-692D3h
Name:	PS ECC Status 1
ShortName:	PS_ECC_STAT_2_C
Reset:	soft
Address:	699D0h-699D3h
Name:	PS ECC Status 1
ShortName:	PS_ECC_STAT_1_D
Reset:	soft
Address:	69AD0h-69AD3h
Name:	PS ECC Status 1
ShortName:	PS_ECC_STAT_2_D

PS_ECC_STAT

Reset: soft

Each of these fields is a sticky bit that gives the ECC error status for a particular memory bank. A '1' in a bit indicates that ECC detected an error. A write of '1' to a bit will clear the bit. Single errors are corrected by ECC. Double errors are not correctable.

DWord	Bit	Description
0	31:17	**Reserved** Access: RO Format: MBZ
	16	**Double Error Detected**
	15:1	**Reserved** Access: RO Format: MBZ
	0	**Single Error Detected**

PS_HPHASE

PS_HPHASE	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PS_WIN_SZ
_Custom_Display_DoubleBufferUpdatePoint:	Start of horizontal blank after armed
Address:	68194h-68197h
Name:	PS Horizontal Phase 1
ShortName:	PS_HPHASE_1_A
Reset:	soft
Address:	68294h-68297h
Name:	PS Horizontal Phase 1
ShortName:	PS_HPHASE_2_A
Reset:	soft
Address:	68994h-68997h
Name:	PS Horizontal Phase 1
ShortName:	PS_HPHASE_1_B
Reset:	soft
Address:	68A94h-68A97h
Name:	PS Horizontal Phase 1
ShortName:	PS_HPHASE_2_B
Reset:	soft
Address:	69194h-69197h
Name:	PS Horizontal Phase 1
ShortName:	PS_HPHASE_1_C
Reset:	soft
Address:	69294h-69297h
Name:	PS Horizontal Phase 1
ShortName:	PS_HPHASE_2_C
Reset:	soft
Address:	69994h-69997h
Name:	PS Horizontal Phase 1
ShortName:	PS_HPHASE_1_D
Reset:	soft
Address:	69A94h-69A97h

PS_HPHASE

Name: PS Horizontal Phase 1

ShortName: PS_HPHASE_2_D

Reset: soft

Description
This register programs the scaler horizontal filtering initial phase. The initial phase within the -0.5 to 1.5 range is supported. Refer to PS_VPHASE for programming details.
The programming of this register is ignored by the pipe scaler when the pipe is in the Full Blend YUV420 mode. The scaler hardware is responsible for calculating and applying the appropriate horizontal phase when encoding the YUV420 format.

DWord	Bit	Description
0	31:30	**Y Initial HPhase Int** This field specifies the integer part of the Y horizontal filtering initial phase when the scaler is operating on YUV420 hybrid planar formats. This field is ignored for non-YUV420 pixel formats.
	29:17	**Y Initial HPhase Frac** This field specifies the most significant 13 bits of the fractional part of the Y horizontal filtering initial phase when the scaler is operating onYUV420 hybrid planar formats. This field should be programmed with the fractional portion of the initial phase multiplied by 2^13. This field is ignored for non-YUV420 pixel formats.
	16	**Y Initial HPhase Trip** This field specifies whether the initial trip, that may occur while applying the initial phase, is used in Y horizontal filtering. This field is ignored for non-YUV420 pixel formats. Value: 1b — Name: Enable Value: 0b — Name: Disable
	15:14	**UV or RGB Initial HPhase Int** This field specifies the integer part of the UV or RGB horizontal filtering initial phase.
	13:1	**UV or RGB Initial HPhase Frac** This field specifies the most significant 13 bits of the fractional part of the UV or RGB horizontal filtering initial phase. This field should be programmed with the fractional portion of the initial phase multiplied by 2^13.
	0	**UV or RGB Initial HPhase Trip** This field specifies whether the initial trip, that may occur while applying the initial phase, is used in UV or RGB horizontal filtering. Value: 1b — Name: Enable Value: 0b — Name: Disable

PS_HSCALE

PS_HSCALE	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	68190h-68193h
Name:	PS Horizontal Scale 1
ShortName:	PS_HSCALE_1_A
Reset:	soft
Address:	68290h-68293h
Name:	PS Horizontal Scale 1
ShortName:	PS_HSCALE_2_A
Reset:	soft
Address:	68990h-68993h
Name:	PS Horizontal Scale 1
ShortName:	PS_HSCALE_1_B
Reset:	soft
Address:	68A90h-68A93h
Name:	PS Horizontal Scale 1
ShortName:	PS_HSCALE_2_B
Reset:	soft
Address:	69190h-69193h
Name:	PS Horizontal Scale 1
ShortName:	PS_HSCALE_1_C
Reset:	soft
Address:	69290h-69293h
Name:	PS Horizontal Scale 1
ShortName:	PS_HSCALE_2_C
Reset:	soft
Address:	69990h-69993h
Name:	PS Horizontal Scale 1
ShortName:	PS_HSCALE_1_D
Reset:	soft
Address:	69A90h-69A93h
Name:	PS Horizontal Scale 1
ShortName:	PS_HSCALE_2_D

PS_HSCALE

Reset: soft

DWord	Bit	Description
0	31:18	**Reserved** Access: RO Format: MBZ
	17:15	**HScale Int** Access: RO This field gives the integer part of the horizontal scaling factor divided by the oversampling rate. HSCALE_INT = int(src width/dest width)
	14:0	**HScale Frac** Access: RO This field gives the fractional part of the horizontal scaling factor divided by the oversampling rate. HSCALE_FRAC = int((((src width/dest width)-HSCALE_INT) * 2^^15) + 0.5)

PS_PROG_HSCALE

PS_PROG_HSCALE	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PS_WIN_SZ
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank after armed
Address:	68168h-6816Bh
Name:	PS Programmed Horizontal Scale 1
ShortName:	PS_PROG_HSCALE_1_A
Reset:	soft
Address:	68268h-6826Bh
Name:	PS Programmed Horizontal Scale 1
ShortName:	PS_PROG_HSCALE_2_A
Reset:	soft
Address:	68968h-6896Bh
Name:	PS Programmed Horizontal Scale 1
ShortName:	PS_PROG_HSCALE_1_B
Reset:	soft
Address:	68A68h-68A6Bh
Name:	PS Programmed Horizontal Scale 1
ShortName:	PS_PROG_HSCALE_2_B
Reset:	soft
Address:	69168h-6916Bh
Name:	PS Programmed Horizontal Scale 1
ShortName:	PS_PROG_HSCALE_1_C
Reset:	soft
Address:	69268h-6926Bh
Name:	PS Programmed Horizontal Scale 1
ShortName:	PS_PROG_HSCALE_2_C
Reset:	soft
Address:	69968h-6996Bh
Name:	PS Programmed Horizontal Scale 1
ShortName:	PS_PROG_HSCALE_1_D
Reset:	soft
Address:	69A68h-69A6Bh

PS_PROG_HSCALE

Name:	PS Programmed Horizontal Scale 1
ShortName:	PS_PROG_HSCALE_2_D
Reset:	soft

This register is used to specify the horizontal scale factor when Programmable Scale Factor is enabled.

DWord	Bit	Description
0	31:18	**Reserved** Access: RO Format: MBZ
	17:15	**HScale Int** This field gives the integer part of the horizontal scaling factor divided by the oversampling rate. HSCALE_INT = int(src width/dest width)
	14:0	**HScale Frac** This field gives the fractional part of the horizontal scaling factor divided by the oversampling rate. HSCALE_FRAC = int(((src width/dest width)-HSCALE_INT)*2^^15)

PS_PROG_VSCALE

PS_PROG_VSCALE	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PS_WIN_SZ
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank after armed
Address:	68164h-68167h
Name:	PS Programmed Vertical Scale 1
ShortName:	PS_PROG_VSCALE_1_A
Reset:	soft
Address:	68264h-68267h
Name:	PS Programmed Vertical Scale 1
ShortName:	PS_PROG_VSCALE_2_A
Reset:	soft
Address:	68964h-68967h
Name:	PS Programmed Vertical Scale 1
ShortName:	PS_PROG_VSCALE_1_B
Reset:	soft
Address:	68A64h-68A67h
Name:	PS Programmed Vertical Scale 1
ShortName:	PS_PROG_VSCALE_2_B
Reset:	soft
Address:	69164h-69167h
Name:	PS Programmed Vertical Scale 1
ShortName:	PS_PROG_VSCALE_1_C
Reset:	soft
Address:	69264h-69267h
Name:	PS Programmed Vertical Scale 1
ShortName:	PS_PROG_VSCALE_2_C
Reset:	soft
Address:	69964h-69967h
Name:	PS Programmed Vertical Scale 1
ShortName:	PS_PROG_VSCALE_1_D
Reset:	soft
Address:	69A64h-69A67h

PS_PROG_VSCALE

Name: PS Programmed Vertical Scale 1

ShortName: PS_PROG_VSCALE_2_D

Reset: soft

This register is used to specify the vertical scale factor when Programmable Scale Factor is enabled.

DWord	Bit	Description
0	31:18	**Reserved** Access: RO Format: MBZ
	17:15	**VScale Int** This field gives the integer part of the vertical scale factor. VSCALE_INT = int(src height/(interlace x dest height)) Interlace = 1/2 in interlace modes, 1 in progressive modes.
	14:0	**VScale Frac** This field gives the fractional part of the vertical scale factor. VSCALE_FRAC = int((src height/(interlace x dest height)-VSCALE_INT)*2^^15) Interlace = 1/2 in interlace modes, 1 in progressive modes.

PS_PWR_GATE

PS_PWR_GATE	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PS_WIN_SZ
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank after armed
Address:	68160h-68163h
Name:	Power Gate Control 1
ShortName:	PS_PWR_GATE_1_A
Reset:	soft
Address:	68260h-68263h
Name:	Power Gate Control 1
ShortName:	PS_PWR_GATE_2_A
Reset:	soft
Address:	68960h-68963h
Name:	Power Gate Control 1
ShortName:	PS_PWR_GATE_1_B
Reset:	soft
Address:	68A60h-68A63h
Name:	Power Gate Control 1
ShortName:	PS_PWR_GATE_2_B
Reset:	soft
Address:	69160h-69163h
Name:	Power Gate Control 1
ShortName:	PS_PWR_GATE_1_C
Reset:	soft
Address:	69260h-69263h
Name:	Power Gate Control 1
ShortName:	PS_PWR_GATE_2_C
Reset:	soft
Address:	69960h-69963h
Name:	Power Gate Control 1
ShortName:	PS_PWR_GATE_1_D
Reset:	soft
Address:	69A60h-69A63h

PS_PWR_GATE

Name: Power Gate Control 1
ShortName: PS_PWR_GATE_2_D
Reset: soft

DWord	Bit	Description
0	31	**Reserved**
	30	**Reserved** Access: RO Format: MBZ
	29:6	**Reserved** Access: RO Format: MBZ
	5	**Dynamic Pwr Gate Disable** Disables the dynamic power gate of unused EBB's when processing low resolution source images. Value 0b: Do Not Disable **[Default]** Value 1b: Disable
	4:3	**Reserved**
	2	**Reserved** Access: RO Format: MBZ
	1:0	**Reserved**

PS_VPHASE

PS_VPHASE	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PS_WIN_SZ
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank after armed
Address:	68188h-6818Bh
Name:	PS Vertical Phase 1
ShortName:	PS_VPHASE_1_A
Reset:	soft
Address:	68288h-6828Bh
Name:	PS Vertical Phase 1
ShortName:	PS_VPHASE_2_A
Reset:	soft
Address:	68988h-6898Bh
Name:	PS Vertical Phase 1
ShortName:	PS_VPHASE_1_B
Reset:	soft
Address:	68A88h-68A8Bh
Name:	PS Vertical Phase 1
ShortName:	PS_VPHASE_2_B
Reset:	soft
Address:	69188h-6918Bh
Name:	PS Vertical Phase 1
ShortName:	PS_VPHASE_1_C
Reset:	soft
Address:	69288h-6928Bh
Name:	PS Vertical Phase 1
ShortName:	PS_VPHASE_2_C
Reset:	soft
Address:	69988h-6998Bh
Name:	PS Vertical Phase 1
ShortName:	PS_VPHASE_1_D
Reset:	soft
Address:	69A88h-69A8Bh

PS_VPHASE

Name: PS Vertical Phase 1
ShortName: PS_VPHASE_2_D
Reset: soft

Description

This register programs the scaler vertical filtering initial phase. The programming of this register is ignored in the pipe scaler PF/ID fetch mode, and the pipe scaler is responsible for applying the appropriate vertical phase to the proper frame when interlacing.
The initial phase within the -0.5 to 1.5 range is supported.
Programming +ve intital phase:

- Initial Phase Trip = 1b
- Initial Phase Int = Desired Initial Phase Int
- Initial Phase Frac = Desired Initial Phase Frac

Programming -ve initial phase:

- Initial Phase Trip = 0b
- Initial Phase Int = 00b
- Initial Phase Frac = 1 - Desired Initial Phase Frac

For example, -0.25 initial phase should be programmed as

- Initial Phase Trip = 0b
- Initial Phase Int = 00b
- Initial Phase Frac = 1 - 0.25 = 0.75

The following table shows phase programming for frequently used YUV420 to YUV444 chroma upsampling scenarios (chroma being filtered to the center of the pixel).

YUV 420 Chroma Siting	H Phase	V Phase	Programmed H Initial Phase	Programmed H Initial Trip	Programmed V Initial Phase	Programmed V Initial Trip
Top Left	0.25	0.25	0.25	1	0.25	1
Bottom Right (MPEG-1)	-0.25	-0.25	0.75	0	0.75	0
Bottom Center (MPEG-2)	0	-0.25	0	0	0.75	0

The programming of this register is ignored by a pipe scaler when the pipe is in the Full Blend YUV420 mode. The scaler hardware is responsible for calculating and applying the appropriate vertical phase when encoding the YUV420 format.

DWord	Bit	Description
0	31:30	**Y Initial VPhase Int** This field specifies the integer part of the Y vertical filtering initial phase when the scaler is operating on YUV420 hybrid planarformats. This field is ignored for non-YUV420 pixel formats.

PS_VPHASE

	29:17	**Y Initial VPhase Frac** This field specifies the most significant 13 bits of the fractional part of the Y vertical filtering initial phase when the scaler is operating on YUV420 hybrid planar formats. This field should be programmed with the fractional portion of the initial phase multiplied by 2^13. This field is ignored for non-YUV420 pixel formats.
	16	**Y Initial VPhase Trip** This field specifies the whether the initial trip, that may occur while applying the initial phase, is used in Y vertical filtering. This field is ignored for non-YUV420 pixel formats. **Value** / **Name** 1b – Used 0b – Not Used
	15:14	**UV or RGB Initial VPhase Int** This field specifies the integer part of the UV or RGB vertical filtering initial phase.
	13:1	**UV or RGB Initial VPhase Frac** This field specifies the most significant 13 bits of the fractional part of the UV or RGB vertical filtering initial phase. This field should be programmed with the fractional portion of the initial phase multiplied by 2^13.
	0	**UV or RGB Initial VPhase Trip** This field specifies whether the initial trip, that may occur while applying the initial phase, is used in UV or RGB vertical filtering. **Value** / **Name** 1b – Used 0b – Not Used

PS_VSCALE

PS_VSCALE	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	68184h-68187h
Name:	PS Vertical Scale 1
ShortName:	PS_VSCALE_1_A
Reset:	soft
Address:	68284h-68287h
Name:	PS Vertical Scale 1
ShortName:	PS_VSCALE_2_A
Reset:	soft
Address:	68984h-68987h
Name:	PS Vertical Scale 1
ShortName:	PS_VSCALE_1_B
Reset:	soft
Address:	68A84h-68A87h
Name:	PS Vertical Scale 1
ShortName:	PS_VSCALE_2_B
Reset:	soft
Address:	69184h-69187h
Name:	PS Vertical Scale 1
ShortName:	PS_VSCALE_1_C
Reset:	soft
Address:	69284h-69287h
Name:	PS Vertical Scale 1
ShortName:	PS_VSCALE_2_C
Reset:	soft
Address:	69984h-69987h
Name:	PS Vertical Scale 1
ShortName:	PS_VSCALE_1_D
Reset:	soft
Address:	69A84h-69A87h
Name:	PS Vertical Scale 1
ShortName:	PS_VSCALE_2_D

PS_VSCALE

Reset: soft

DWord	Bit	Description
0	31:18	**Reserved** Access: RO Format: MBZ
	17:15	**VScale Int** Access: RO This field gives the integer part of the vertical scale factor. VSCALE_INT = int(src height/(interlace x dest height)) Interlace = 1/2 in interlace modes, 1 in progressive modes.
	14:0	**VScale Frac** Access: RO This field gives the fractional part of the vertical scale factor. VSCALE_FRAC = int(((src height/(interlace x dest height)-VSCALE_INT) * 2^^15) + 0.5) Interlace = 1/2 in interlace modes, 1 in progressive modes.

PS_WIN_POS

PS_WIN_POS	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PS_WIN_SZ
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank after armed
Address:	68170h-68173h
Name:	PS Window Position 1
ShortName:	PS_WIN_POS_1_A
Reset:	soft
Address:	68270h-68273h
Name:	PS Window Position 1
ShortName:	PS_WIN_POS_2_A
Reset:	soft
Address:	68970h-68973h
Name:	PS Window Position 1
ShortName:	PS_WIN_POS_1_B
Reset:	soft
Address:	68A70h-68A73h
Name:	PS Window Position 1
ShortName:	PS_WIN_POS_2_B
Reset:	soft
Address:	69170h-69173h
Name:	PS Window Position 1
ShortName:	PS_WIN_POS_1_C
Reset:	soft
Address:	69270h-69273h
Name:	PS Window Position 1
ShortName:	PS_WIN_POS_2_C
Reset:	soft
Address:	69970h-69973h
Name:	PS Window Position 1
ShortName:	PS_WIN_POS_1_D
Reset:	soft
Address:	69A70h-69A73h

PS_WIN_POS

Name: PS Window Position 1

ShortName: PS_WIN_POS_2_D

Reset: soft

Coordinates are determined with a value of (0,0) being the upper left corner of the display device (rotation does not affect this).

Restriction

When scaling a pipe, the scaled output must fit inside the pipe active area, so Pipe active size >= PS window position + PS window size. When scaling a plane, the scaled output must fit inside the pipe source area, so Pipe source size >= PS window position + PS window size.

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**XPOS** This field specifies the horizontal coordinate in pixels of the upper left most pixel of the scaled output window. Restriction : This field must be even when the scaler is delivering YUV420 format for HDMI output.
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**YPOS** This field specifies the vertical coordinate in lines of the upper left most pixel of the scaled output window. **Restriction** Bit 0 must be zero for interlaced modes. This field must be even when the scaler is delivering YUV420 format for HDMI output.

PS_WIN_SZ

PS_WIN_SZ	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank
Address:	68174h-68177h
Name:	PS Window Size 1
ShortName:	PS_WIN_SZ_1_A
Reset:	soft
Address:	68274h-68277h
Name:	PS Window Size 1
ShortName:	PS_WIN_SZ_2_A
Reset:	soft
Address:	68974h-68977h
Name:	PS Window Size 1
ShortName:	PS_WIN_SZ_1_B
Reset:	soft
Address:	68A74h-68A77h
Name:	PS Window Size 1
ShortName:	PS_WIN_SZ_2_B
Reset:	soft
Address:	69174h-69177h
Name:	PS Window Size 1
ShortName:	PS_WIN_SZ_1_C
Reset:	soft
Address:	69274h-69277h
Name:	PS Window Size 1
ShortName:	PS_WIN_SZ_2_C
Reset:	soft
Address:	69974h-69977h
Name:	PS Window Size 1
ShortName:	PS_WIN_SZ_1_D
Reset:	soft
Address:	69A74h-69A77h
Name:	PS Window Size 1

PS_WIN_SZ

ShortName: PS_WIN_SZ_2_D

Reset: soft

This register specifies the size in pixels of the scaled output window. A programmed value of (100, 100) will result in scaled output window of size 100x100 pixels.
Writes to this register arm PS registers on this pipe. After arming, any write to other PS registers will disarm all PS registers. Subsequent write to this register will arm them again.

Restriction

When scaling a pipe, the scaled output must fit inside the pipe active area, so Pipe active size >= PS window position + PS window size. When scaling a plane, the scaled output must fit inside the pipe source area, so Pipe source size >= PS window position + PS window size.

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**XSIZE** This field specifies the horizontal size in pixels of the scaled output window. Restriction : When the pipe scalar is configured to output YUV 420, the X size must be even.
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**YSIZE** This field specifies the vertical size in scan lines of the scaled output window. Restriction : Bit 0 must be zero for interlaced modes. **Restriction** When the pipe scalar is configured to output YUV 420, the Y size must be even.

PS Depth Count

PS_DEPTH_COUNT - PS Depth Count	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02350h

This register stores the value of the count of samples that have passed the depth test. This register is part of the context save and restore. Note that the value of this register can be obtained in a pipeline-synchronous fashion without a pipeline flush by using the 3DCONTROL command. See 3D Overview in the 3D volume.

DWord	Bit	Description
0..1	63:32	**Depth Count UDW** This register reflects the total number of samples that have passed the depth test (i.e., will be visible). All samples are counted when Statistics Enable is set in the Windower State. See the Windower chapter of the 3D volume for details. Samples that pass the depth test but fail the stencil test will not be counted.
	31:0	**Depth Count LDW** This register reflects the total number of samples that have passed the depth test (i.e., will be visible). All samples are counted when Statistics Enable is set in the Windower State. See the Windower chapter of the 3D volume for details. Samples that pass the depth test but fail the stencil test will not be counted.

PS Depth Count for Slice0

PS_DEPTH_COUNT_SLICE0 - PS Depth Count for Slice0	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	022D8h
Name:	PS Depth Count for Slice0
ShortName:	PS_DEPTH_COUNT_SLICE0

This register stores the value of the count of pixels that have passed the depth test in Slice0. This register is part of the context save and restore. This register should not be programmed by SW. Note: In GT3 mode there are two slices Slice0 (Upper Slice) and Slice1 (Lower slice).

DWord	Bit	Description
0..1	63:32	**Depth Count UDW** This register reflects the total number of pixels that have passed the depth test in Slice0(i.e., will be visible). All pixels are counted when Statistics Enable is set in the Windower State. See the Windower chapter of the 3D volume for details. Pixels that pass the depth test but fail the stencil test will not be counted.
	31:0	**Depth Count LDW** This register reflects the total number of pixels that have passed the depth test in Slice0(i.e., will be visible). All pixels are counted when Statistics Enable is set in the Windower State. See the Windower chapter of the 3D volume for details. Pixels that pass the depth test but fail the stencil test will not be counted.

PS Depth Count for Slice1

PS_DEPTH_COUNT_SLICE1 - PS Depth Count for Slice1	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	022F8h
Name:	PS Depth Count for Slice1
ShortName:	PS_DEPTH_COUNT_SLICE1

This register stores the value of the count of pixels that have passed the depth test in Slice1. This register is part of the context save and restore. This register should not be programmed by SW. Note: In GT3 mode there are two slices Slice0 (Upper Slice) and Slice1 (Lower slice).

DWord	Bit	Description
0..1	63:32	**Depth Count UDW** This register reflects the total number of pixels that have passed the depth test in Slice1(i.e., will be visible). All pixels are counted when Statistics Enable is set in the Windower State. See the Windower chapter of the 3D volume for details. Pixels that pass the depth test but fail the stencil test will not be counted.
	31:0	**Depth Count LDW** This register reflects the total number of pixels that have passed the depth test in Slice1(i.e., will be visible). All pixels are counted when Statistics Enable is set in the Windower State. See the Windower chapter of the 3D volume for details. Pixels that pass the depth test but fail the stencil test will not be counted.

PS Depth Count for Slice2

PS_DEPTH_COUNT_SLICE2 - PS Depth Count for Slice2	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02450h
Name:	PS Depth Count for Slice2
ShortName:	PS_DEPTH_COUNT_SLICE2

This register stores the value of the count of pixels that have passed the depth test in Slice2. This register is part of the render context save and restore. This register should not be programmed by SW.

DWord	Bit	Description
0..1	63:32	**Depth Count UDW** This register reflects the depth test in slice2 (i.e., will be visible). All pixels are counted when Statistics Enable is set in the Windower State. See the Windower chapter of the 3D volume for details. Pixels that pass the depth test but fail the stencil test will not be counted.
	31:0	**Depth Count LDW** This register reflects the depth test in slice2 (i.e., will be visible). All pixels are counted when Statistics Enable is set in the Windower State. See the Windower chapter of the 3D volume for details. Pixels that pass the depth test but fail the stencil test will not be counted.

PS Depth Count for Slice3

PS_DEPTH_COUNT_SLICE3 - PS Depth Count for Slice3	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02460h
Name:	PS Depth Count for Slice3
ShortName:	PS_DEPTH_COUNT_SLICE3

This register stores the value of the count of pixels that have passed the depth test in Slice3. This register is part of the render context save and restore. This register should not be programmed by SW.

DWord	Bit	Description
0..1	63:32	**Depth Count UDW** This register reflects the depth test in slice3 (i.e., will be visible). All pixels are counted when Statistics Enable is set in the Windower State. See the Windower chapter of the 3D volume for details. Pixels that pass the depth test but fail the stencil test will not be counted.
	31:0	**Depth Count LDW** This register reflects the depth test in slice3 (i.e., will be visible). All pixels are counted when Statistics Enable is set in the Windower State. See the Windower chapter of the 3D volume for details. Pixels that pass the depth test but fail the stencil test will not be counted.

PS Depth Count for Slice4

PS_DEPTH_COUNT_SLICE4 - PS Depth Count for Slice4	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02470h
Name:	PS Depth Count for Slice4
ShortName:	PS_DEPTH_COUNT_SLICE4

This register stores the value of the count of pixels that have passed the depth test in Slice4. This register is part of the render context save and restore. This register should not be programmed by SW.

DWord	Bit	Description
0..1	63:32	**Depth Count UDW** This register reflects the depth test in slice4 (i.e., will be visible). All pixels are counted when Statistics Enable is set in the Windower State. See the Windower chapter of the 3D volume for details. Pixels that pass the depth test but fail the stencil test will not be counted.
	31:0	**Depth Count LDW** This register reflects the depth test in slice4 (i.e., will be visible). All pixels are counted when Statistics Enable is set in the Windower State. See the Windower chapter of the 3D volume for details. Pixels that pass the depth test but fail the stencil test will not be counted.

PS Depth Count for Slice5

PS_DEPTH_COUNT_SLICE5 - PS Depth Count for Slice5	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	024A8h
Name:	PS Depth Count for Slice5
ShortName:	PS_DEPTH_COUNT_SLICE5

This register stores the value of the count of pixels that have passed the depth test in Slice5. This register is part of the render context save and restore. This register should not be programmed by SW.

DWord	Bit	Description
0..1	63:32	**Depth Count UDW** This register reflects the depth test in slice5 (i.e., will be visible). All pixels are counted when Statistics Enable is set in the Windower State. See the Windower chapter of the 3D volume for details. Pixels that pass the depth test but fail the stencil test will not be counted.
	31:0	**Depth Count LDW** This register reflects the depth test in slice5 (i.e., will be visible). All pixels are counted when Statistics Enable is set in the Windower State. See the Windower chapter of the 3D volume for details. Pixels that pass the depth test but fail the stencil test will not be counted.

PS Depth Count for Slice6

PS_DEPTH_COUNT_SLICE6 - PS Depth Count for Slice6	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	025B0h
Name:	PS Depth Count for Slice6
ShortName:	PS_DEPTH_COUNT_SLICE6

This register stores the value of the count of pixels that have passed the depth test in Slice6. This register is part of the render context save and restore. This register should not be programmed by SW.

DWord	Bit	Description
0..1	63:32	**Depth Count UDW** This register reflects the depth test in slice5 (i.e., will be visible). All pixels are counted when Statistics Enable is set in the Windower State. See the Windower chapter of the 3D volume for details. Pixels that pass the depth test but fail the stencil test will not be counted.
	31:0	**Depth Count LDW** This register reflects the depth test in slice5 (i.e., will be visible). All pixels are counted when Statistics Enable is set in the Windower State. See the Windower chapter of the 3D volume for details. Pixels that pass the depth test but fail the stencil test will not be counted.

PS Depth Count for Slice7

PS_DEPTH_COUNT_SLICE7 - PS Depth Count for Slice7	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	025B8h
Name:	PS Depth Count for Slice7
ShortName:	PS_DEPTH_COUNT_SLICE7

This register stores the value of the count of pixels that have passed the depth test in Slice7. This register is part of the render context save and restore. This register should not be programmed by SW.

DWord	Bit	Description
0..1	63:32	**Depth Count UDW** This register reflects the depth test in slice5 (i.e., will be visible). All pixels are counted when Statistics Enable is set in the Windower State. See the Windower chapter of the 3D volume for details. Pixels that pass the depth test but fail the stencil test will not be counted.
	31:0	**Depth Count LDW** This register reflects the depth test in slice5 (i.e., will be visible). All pixels are counted when Statistics Enable is set in the Windower State. See the Windower chapter of the 3D volume for details. Pixels that pass the depth test but fail the stencil test will not be counted.

PS Invocation Count

PS_INVOCATION_COUNT - PS Invocation Count		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	64	
_Custom_GTIReset:	DEV	
Address:	02348h	
DWord	**Bit**	**Description**
0..1	63:32	**PS Invocation Count UDW** Reflects a count of the total number of pixels (including unlit "helper pixels" within a subspan that need to go through the PS shader to provide 2x2 gradients) that are dispatched to pixel shader invocations while Statistics Enable is set in the Windower. See the Windower chapter of the 3D volume for details. This count will generally be much greater than the actual count of PS threads since a single thread may process up to 32 pixels.
	31:0	**PS Invocation Count LDW** Reflects a count of the total number of pixels (including unlit "helper pixels" within a subspan that need to go through the PS shader to provide 2x2 gradients) that are dispatched to pixel shader invocations while Statistics Enable is set in the Windower. See the Windower chapter of the 3D volume for details. This count will generally be much greater than the actual count of PS threads since a single thread may process up to 32 pixels.

PS Invocation Count for Slice0

PS_INVOCATION_COUNT_SLICE0 - PS Invocation Count for Slice0	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	022C8h
Name:	PS Invocation Count for Slice0
ShortName:	PS_INVOCATION_COUNT_SLICE0

This register stores the value of the count of pixels that get shaded in Slice0. This register is part of the context save and restore. This register should not be programmed by SW. Note: In GT3 mode there are two slices Slice0 (Upper Slice) and Slice1 (Lower slice).

DWord	Bit	Description
0..1	63:32	**PS Invocation Count UDW** Reflects a count of the total number of pixels (including unlit "helper pixels" within a subspan that need to go through the PS shader to provide 2x2 gradients) that are dispatched to pixel shader invocations while Statistics Enable is set in the Windower in Slice0. See the Windower chapter of the 3D volume for details. This count will generally be much greater than the actual count of PS threads since a single thread may process up to 32 pixels.
	31:0	**PS Invocation Count LDW** Reflects a count of the total number of pixels (including unlit "helper pixels" within a subspan that need to go through the PS shader to provide 2x2 gradients) that are dispatched to pixel shader invocations while Statistics Enable is set in the Windower in Slice0. See the Windower chapter of the 3D volume for details. This count will generally be much greater than the actual count of PS threads since a single thread may process up to 32 pixels.

PS Invocation Count for Slice1

PS_INVOCATION_COUNT_SLICE1 - PS Invocation Count for Slice1	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	022F0h
Name:	PS Invocation Count for Slice1
ShortName:	PS_INVOCATION_COUNT_SLICE1

This register stores the value of the count of pixels that get shaded in Slice0. This register is part of the context save and restore. This register should not be programmed by SW. Note: In GT3 mode there are two slices Slice0 (Upper Slice) and Slice1 (Lower slice).

DWord	Bit	Description
0..1	63:32	**PS Invocation Count UDW** Reflects a count of the total number of pixels (including unlit "helper pixels" within a subspan that need to go through the PS shader to provide 2x2 gradients) that are dispatched to pixel shader invocations while Statistics Enable is set in the Windower in Slice1. See the Windower chapter of the 3D volume for details. This count will generally be much greater than the actual count of PS threads since a single thread may process up to 32 pixels.
	31:0	**PS Invocation Count LDW** Reflects a count of the total number of pixels (including unlit "helper pixels" within a subspan that need to go through the PS shader to provide 2x2 gradients) that are dispatched to pixel shader invocations while Statistics Enable is set in the Windower in Slice1. See the Windower chapter of the 3D volume for details. This count will generally be much greater than the actual count of PS threads since a single thread may process up to 32 pixels.

PS Invocation Count for Slice2

PS_INVOCATION_COUNT_SLICE2 - PS Invocation Count for Slice2	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02448h
Name:	PS Invocation Count for Slice2
ShortName:	PS_INVOCATION_COUNT_SLICE2

This register stores the value of the count of pixels that get shaded in Slice2. This register is part of the render context save and restore. This register should not be programmed by SW.

DWord	Bit	Description
0..1	63:32	**PS Invocation Count UDW** Reflects a count of the total number of pixels (including unlit "helper pixels" within a subspan that need to go through the PS shader to provide 2x2 gradients) that are dispatched to pixel shader invocations while Statistics Enable is set in the Windower in Slice2. See the Windower chapter of the 3D volume for details. This count will generally be much greater than the actual count of PS threads since a single thread may process up to 32 pixels.
	31:0	**PS Invocation Count LDW** Reflects a count of the total number of pixels (including unlit "helper pixels" within a subspan that need to go through the PS shader to provide 2x2 gradients) that are dispatched to pixel shader invocations while Statistics Enable is set in the Windower in Slice2. See the Windower chapter of the 3D volume for details. This count will generally be much greater than the actual count of PS threads since a single thread may process up to 32 pixels.

PS Invocation Count for Slice3

PS_INVOCATION_COUNT_SLICE3 - PS Invocation Count for Slice3	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02458h
Name:	PS Invocation Count for Slice3
ShortName:	PS_INVOCATION_COUNT_SLICE3

This register stores the value of the count of pixels that get shaded in Slice3. This register is part of the render context save and restore. This register should not be programmed by SW.

DWord	Bit	Description
0..1	63:32	**PS Invocation Count UDW** Reflects a count of the total number of pixels (including unlit "helper pixels" within a subspan that need to go through the PS shader to provide 2x2 gradients) that are dispatched to pixel shader invocations while Statistics Enable is set in the Windower in Slice3. See the Windower chapter of the 3D volume for details. This count will generally be much greater than the actual count of PS threads since a single thread may process up to 32 pixels.
	31:0	**PS Invocation Count LDW** Reflects a count of the total number of pixels (including unlit "helper pixels" within a subspan that need to go through the PS shader to provide 2x2 gradients) that are dispatched to pixel shader invocations while Statistics Enable is set in the Windower in Slice3. See the Windower chapter of the 3D volume for details. This count will generally be much greater than the actual count of PS threads since a single thread may process up to 32 pixels.

PS Invocation Count for Slice4

PS_INVOCATION_COUNT_SLICE4 - PS Invocation Count for Slice4	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02468h
Name:	PS Invocation Count for Slice4
ShortName:	PS_INVOCATION_COUNT_SLICE4

This register stores the value of the count of pixels that get shaded in Slice4. This register is part of the render context save and restore. This register should not be programmed by SW.

DWord	Bit	Description
0..1	63:32	**PS Invocation Count UDW** Reflects a count of the total number of pixels (including unlit "helper pixels" within a subspan that need to go through the PS shader to provide 2x2 gradients) that are dispatched to pixel shader invocations while Statistics Enable is set in the Windower in Slice4. See the Windower chapter of the 3D volume for details. This count will generally be much greater than the actual count of PS threads since a single thread may process up to 32 pixels.
	31:0	**PS Invocation Count LDW** Reflects a count of the total number of pixels (including unlit "helper pixels" within a subspan that need to go through the PS shader to provide 2x2 gradients) that are dispatched to pixel shader invocations while Statistics Enable is set in the Windower in Slice4. See the Windower chapter of the 3D volume for details. This count will generally be much greater than the actual count of PS threads since a single thread may process up to 32 pixels.

PS Invocation Count for Slice5

PS_INVOCATION_COUNT_SLICE5 - PS Invocation Count for Slice5	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	024A0h
Name:	PS Invocation Count for Slice5
ShortName:	PS_INVOCATION_COUNT_SLICE5

This register stores the value of the count of pixels that get shaded in Slice5. This register is part of the render context save and restore. This register should not be programmed by SW.

DWord	Bit	Description
0..1	63:32	**PS Invocation Count UDW** Reflects a count of the total number of pixels (including unlit "helper pixels" within a subspan that need to go through the PS shader to provide 2x2 gradients) that are dispatched to pixel shader invocations while Statistics Enable is set in the Windower in Slice4. See the Windower chapter of the 3D volume for details. This count will generally be much greater than the actual count of PS threads since a single thread may process up to 32 pixels.
	31:0	**PS Invocation Count LDW** Reflects a count of the total number of pixels (including unlit "helper pixels" within a subspan that need to go through the PS shader to provide 2x2 gradients) that are dispatched to pixel shader invocations while Statistics Enable is set in the Windower in Slice4. See the Windower chapter of the 3D volume for details. This count will generally be much greater than the actual count of PS threads since a single thread may process up to 32 pixels.

PS Invocation Count for Slice6

PS_INVOCATION_COUNT_SLICE6 - PS Invocation Count for Slice6	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	025D0h
Name:	PS Invocation Count for Slice6
ShortName:	PS_INVOCATION_COUNT_SLICE6

This register stores the value of the count of pixels that get shaded in Slice6. This register is part of the render context save and restore. This register should not be programmed by SW.

DWord	Bit	Description
0..1	63:32	**PS Invocation Count UDW** Reflects a count of the total number of pixels (including unlit "helper pixels" within a subspan that need to go through the PS shader to provide 2x2 gradients) that are dispatched to pixel shader invocations while Statistics Enable is set in the Windower in Slice4. See the Windower chapter of the 3D volume for details. This count will generally be much greater than the actual count of PS threads since a single thread may process up to 32 pixels.
	31:0	**PS Invocation Count LDW** Reflects a count of the total number of pixels (including unlit "helper pixels" within a subspan that need to go through the PS shader to provide 2x2 gradients) that are dispatched to pixel shader invocations while Statistics Enable is set in the Windower in Slice4. See the Windower chapter of the 3D volume for details. This count will generally be much greater than the actual count of PS threads since a single thread may process up to 32 pixels.

PS Invocation Count for Slice7

PS_INVOCATION_COUNT_SLICE7 - PS Invocation Count for Slice7	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	025D8h
Name:	PS Invocation Count for Slice7
ShortName:	PS_INVOCATION_COUNT_SLICE7

This register stores the value of the count of pixels that get shaded in Slice7. This register is part of the render context save and restore. This register should not be programmed by SW.

DWord	Bit	Description
0..1	63:32	**PS Invocation Count UDW** Reflects a count of the total number of pixels (including unlit "helper pixels" within a subspan that need to go through the PS shader to provide 2x2 gradients) that are dispatched to pixel shader invocations while Statistics Enable is set in the Windower in Slice4. See the Windower chapter of the 3D volume for details. This count will generally be much greater than the actual count of PS threads since a single thread may process up to 32 pixels.
	31:0	**PS Invocation Count LDW** Reflects a count of the total number of pixels (including unlit "helper pixels" within a subspan that need to go through the PS shader to provide 2x2 gradients) that are dispatched to pixel shader invocations while Statistics Enable is set in the Windower in Slice4. See the Windower chapter of the 3D volume for details. This count will generally be much greater than the actual count of PS threads since a single thread may process up to 32 pixels.

PSR_EVENT

PSR_EVENT	
Register Space:	MMIO: 0/2/0
Access:	R/WC
Size (in bits):	32
Address:	60848h-6084Bh
Name:	Transcoder PSR Event
ShortName:	PSR_EVENT_A
Reset:	soft
Address:	61848h-6184Bh
Name:	Transcoder PSR Event
ShortName:	PSR_EVENT_B
Reset:	soft
Address:	62848h-6284Bh
Name:	Transcoder PSR Event
ShortName:	PSR_EVENT_C
Reset:	soft
Address:	63848h-6384Bh
Name:	Transcoder PSR Event
ShortName:	PSR_EVENT_D
Reset:	soft

This register captures the event that caused an exit from PSR or PSR2. The exit events will be set by hardware. Software will need to clear these events.

DWord	Bit	Description
0	31:18	**Reserved** Access: RO Format: MBZ
	17	**PSR2 watch dog timer expire** Access: R/WC This is a sticky bit which is set when the PSR2 watch dog timer expires, causing PSR exit. Clear by writing with a 1. Value / Name: 0b – Condition Not Detected 1b – Condition Detected
	16	**PSR2 Disable** Access: R/WC This is a sticky bit which is set when the PSR2 is disabled, causing PSR exit. Clear by writing with

PSR_EVENT

a 1.

Value	Name
0b	Condition Not Detected
1b	Condition Detected

15 Selective Update Dirty FIFO Underrun

Access:	R/WC

This is a sticky bit which is set when the selective update dirty/clean FIFO Underruns, causing PSR exit. Clear by writing with a 1.

Value	Name
0b	Condition Not Detected
1b	Condition Detected

14 Selective Update CRC FIFO Underrun

Access:	R/WC

This is a sticky bit which is set when the selective update CRC FIFO Underruns, causing PSR exit. Clear by writing with a 1.

Value	Name
0b	Condition Not Detected
1b	Condition Detected

13 Reserved

Access:	RO
Format:	MBZ

12 Graphics Reset

Access:	R/WC

This is a sticky bit which is set when a graphics reset causes PSR exit. Clear by writing with a 1.

Value	Name
0b	Condition Not Detected
1b	Condition Detected

11 PCH Interrupt

Access:	R/WC

This is a sticky bit which is set when a PCH Interrupt causes PSR exit. Clear by writing with a 1.

Value	Name
0b	Condition Not Detected
1b	Condition Detected

10 Memory Up

Access:	R/WC

This is a sticky bit which is set when a PCU memup up event causes PSR exit. Clear by writing with a 1.

PSR_EVENT

Value	Name
0b	Condition Not Detected
1b	Condition Detected

9 **Front Buffer Modify**

Access:	R/WC

This is a sticky bit which is set when a front buffer modify causes PSR exit. Clear by writing with a 1.

Value	Name
0b	Condition Not Detected
1b	Condition Detected

8 **Watch dog timer expire**

Access:	R/WC

This is a sticky bit which is set when the PSR watch dog timer expires, causing PSR exit. Clear by writing with a 1.

Value	Name
0b	Condition Not Detected
1b	Condition Detected

7 **Reserved**

Access:	RO
Format:	MBZ

6 **Pipe Registers Update**

Access:	R/WC

This is a sticky bit which is set when a display pipe register update causes PSR exit. Clear by writing with a 1.

5 **Reserved**

Access:	RO
Format:	MBZ

4 **Reserved**

3 **KVMR session enable**

Access:	R/WC

This is a sticky bit which is set when a KVMR session is enabled, causing PSR exit. Clear by writing with a 1.

Value	Name
0b	Condition Not Detected
1b	Condition Detected

2 **VBI enable**

Access:	R/WC

This is a sticky bit which is set when vblank or vsync interrupt is enabled, causing PSR exit. Clear

PSR_EVENT

by writing with a 1.

Value	Name
0b	Condition Not Detected
1b	Condition Detected

1 Reserved

Access:	RO
Format:	MBZ

0 SRD disable

Access:	R/WC

This is a sticky bit which is set when SRD enable is cleared, causing PSR exit. Clear by writing with a 1.

Value	Name
0b	Condition Not Detected
1b	Condition Detected

PSR_IIR

PSR_IIR	
Register Space:	MMIO: 0/2/0
Access:	R/WC
Size (in bits):	32
Address:	60818h-6081Bh
Name:	Transcoder PSR Interrupt Identity
ShortName:	PSR_IIR_A
Reset:	soft
Address:	61818h-6181Bh
Name:	Transcoder PSR Interrupt Identity
ShortName:	PSR_IIR_B
Reset:	soft
Address:	62818h-6281Bh
Name:	Transcoder PSR Interrupt Identity
ShortName:	PSR_IIR_C
Reset:	soft
Address:	63818h-6381Bh
Name:	Transcoder PSR Interrupt Identity
ShortName:	PSR_IIR_D
Reset:	soft

This register holds the persistent values of the PSR interrupt bits which are unmasked by PSR_IMR. Bits set in this register will propagate to the PSR/SRD interrupt in the Display Engine Miscellaneous Interrupts.

DWord	Bit	Description
0	31:4	**Reserved** Access: RO Format: MBZ
	3	**Push Done** Access: R/WC This is a sticky bit which is set after double buffer update or Push done. After this interrupt, Logic is ready to receive another push frame indication. Clear by writing with a 1. 0b – Condition Not Detected 1b – Condition Detected
	2	**PSR Aux Error** Access: R/WC This is a sticky bit which is set on the rising edge of the PSR Aux error (receive error or timeout)

indication. Clear by writing with a 1.

Value	Name
0b	Condition Not Detected
1b	Condition Detected

1 **PSR Exit**

Access:	R/WC

This is a sticky bit which is set on the first blank start after PSR exit. Clear by writing with a 1.

Value	Name
0b	Condition Not Detected
1b	Condition Detected

0 **PSR PreWarn**

Access:	R/WC

This is a sticky bit which is set two display frames prior to entering PSR. Clear by writing with a 1.

Value	Name
0b	Not Detected
1b	Condition Detected

PSR_IMR

PSR_IMR

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60814h-60817h
Name:	Transcoder PSR Interrupt Mask
ShortName:	PSR_IMR_A
Reset:	soft
Address:	61814h-61817h
Name:	Transcoder PSR Interrupt Mask
ShortName:	PSR_IMR_B
Reset:	soft
Address:	62814h-62817h
Name:	Transcoder PSR Interrupt Mask
ShortName:	PSR_IMR_C
Reset:	soft
Address:	63814h-63817h
Name:	Transcoder PSR Interrupt Mask
ShortName:	PSR_IMR_D
Reset:	soft

This register contains a bit mask which selects the PSR events that will be reported in the PSR_IIR.

DWord	Bit	Description
0	31:4	**Reserved** Access: RO Format: MBZ
	3	**Mask Push Done** 0b: Not Masked 1b: Masked **[Default]**
	2	**Mask PSR Aux Error** 0b: Not Masked 1b: Masked **[Default]**

PSR_IMR

	1	**Mask PSR Exit** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**
	0	**Mask PSR PreWarn** Value: 0b — Name: Not Masked Value: 1b — Name: Masked **[Default]**

PSR_MASK

PSR_MASK	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60860h-60863h
Name:	Transcoder PSR Event Mask
ShortName:	PSR_MASK_A
Reset:	soft
Address:	61860h-61863h
Name:	Transcoder PSR Event Mask
ShortName:	PSR_MASK_B
Reset:	soft
Address:	62860h-62863h
Name:	Transcoder PSR Event Mask
ShortName:	PSR_MASK_C
Reset:	soft
Address:	63860h-63863h
Name:	Transcoder PSR Event Mask
ShortName:	PSR_MASK_D
Reset:	soft

Some of the masking is controlled here and some in the PIPE_MISC register.

Restriction
Only bit 30 (Idle Frame Override) can be changed while PSR or PSR2 is enabled. The other fields must not be changed while PSR or PSR2 is enabled.

DWord	Bit	Description
0	31:30	**Idle Frame Override** This field overrides the entry/exit conditions to force PSR or PSR2 Deep Sleep entry/exit.

Value	Name	Description
00b,01b	No Override	Do not override. Use regular entry and exit conditions.
10b	Force Idle Frame	Force Idle Frames to force PSR entry or PSR2 Deep Sleep
11b	Force Non-Idle Frame	Force Non-Idle Frames to force PSR exit or exit from PSR2 Deep Sleep

DWord	Bit	Description
0	29	**Reserved**

Access:	RO
Format:	MBZ

PSR_MASK

28 Mask Max Sleep

This field controls the mask for the max sleep time event.

Value	Name
0b	Not Masked
1b	Masked

27 Mask LPSP

This field controls the mask for the low power single pipe event. This field is ignored by transcoder A/B/C.

Value	Name
0b	Not masked
1b	Masked

26 Mask Memup

This field controls the mask for the memory up event.

Value	Name	Description
0b	Not Masked	
1b	Masked **[Default]**	Masked - will not be considered in PSR idleness tracking (default)

25 Mask Hotplug

This field controls the mask for the hotplug event. Not used in PSR2 Deep Sleep entry/exit.

Value	Name
0b	Not Masked
1b	Masked

24 Mask FBC Modify

This field controls the mask for the FBC front buffer modify event.

Value	Name
0b	Not Masked
1b	Masked

23:16 Reserved

Access:	RO
Format:	MBZ

15 Exit on Pixel Underrun

This field controls the mask for exit on pixel underrun.

Value	Name
0b	Not Masked
1b	Masked **[Default]**

14:1 Reserved

Access:	RO
Format:	MBZ

PSR_MASK		
	0	**Global Mask** This field is no longer used. The global mask function moved to 0x42084 bit 0. **Value** / **Name** 0b – Not Masked 1b – Masked

PSR2_CTL

PSR2_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60900h-60903h
Name:	PSR2 Control
ShortName:	PSR2_CTL_A
Reset:	soft

Programming Notes

To use FBC modification tracking for idleness calculations when FBC is disabled, program FBC_CTL CPU Fence Enable, FBC_CONTROL_SA_REGISTER, FBC_CPU_FENCE_OFFSET_REGISTER, FBC_RT_BASE_ADDR_REGISTER, and BLITTER_TRACKING_REGISTER as they are programmed when FBC is enabled.

Restriction

- PSR needs to be enabled only when at least one plane is enabled.
- PSR2 is limited to 30bpp 10:10:10, even when using the manual tracking mode.
- Only the PSR2 Enable can be changed while PSR2 is enabled. The other fields must not be changed while PSR2 is enabled. Selective Update Tracking Enable must be set before or along with PSR2 enable
- PSR2 is supported for pipe active sizes up to 5120 pixels wide and 3200 lines tall.

DWord	Bit	Description
0	31	**PSR2 Enable** This bit enables Revision 2.0 of the Panel Self Refresh function. Updates will take place at the start of the next vertical blank. The port will send PSR2 VDMs while enabled.

Value	Name
0b	Disable
1b	Enable

Programming Notes

Clear the register field SRD_CTL [TP2 TP3 Select] before enabling this bit.
Do not set the register field SRD_CTL [TP2 TP3 Select] while PSR2 is enabled.

Restriction

- PSR2 must not be enabled when the PSR Setup time from DPCD 00071h is greater than the time for vertical blank minus one line.
- PSR2 must not be enabled together with Interlacing, Black Frame Insertion (BFI), Compression Mode, or S3D.
- Disable FBC when PSR2 is enabled.

PSR2_CTL

30 Selective Update Tracking Enable

Access:	Double Buffered
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank OR transcoder disabled

This field enables the Selective Update Tracking Mechanism. Updates to this field will take effect at the next vertical blank.

Value	Name
0b	Disable
1b	Enable

Restriction
This field must be enabled anytime before PSR2 Enable. It must be disabled anytime after PSR2 disable.

29 Context restore to PSR2 Deep Sleep State

This field restores PSR2 into Deep Sleep State

Value	Name
0b	Disable
1b	Enable

Restriction
This bit should only be used with context save restore.

28 Block count number

This field selects block count number before SU turn on sequence

Value	Name
0b	2 blocks
1b	3 blocks

27 Aux Frame Sync Enable

This field selects whether the frame sync will be sent on Aux channel.

Value	Name
1b	Enable
0b	Disable

Restriction
Must be programmed to match the panel's requirements.

26 Y-coordinate valid

This field selects whether PSR2 Y-coordinate valid behaves as per eDP 1.4a

Value	Name
0b	Include Y-coordinate valid eDP1.4a
1b	Do not include Y-coordinate valid eDP 1.4

PSR2_CTL

25 — Y-coordinate enable

This field selects whether PSR2 VSC packet will include vertical line count.

Value	Name
0b	Do not include count
1b	Include count

24:20 — Max SU Disable Time

Default Value:	00000b Disabled

This field is the maximum time to spend in PSR2 Selective update without fetching a full frame. It is programmed in increments of sixty frames. Programming all 1s gives 31x60 frames time.

Restriction
Programming all 0s disable the forced fetch of a full frame in SU.

19 — Reserved

18 — PSR2 RAM power state

Access:	RO

17:16 — Reserved

15:13 — IO buffer Wake

This field selects the number of lines before the Selective Update Region to wake the IO Buffers.

Value	Name
000b	5 lines
001b	6 lines
010b	7 lines **[Default]**
011b	8 lines
100b	9 lines
101b	10 lines
110b	11 lines
111b	12 lines

Restriction
To program line 9 to 12, block count number bit [28] must be set.

12:10 — Fast Wake

This field selects the number of lines before the Selective Update Region to send the Fast Wake.

Value	Name
000b	5 lines
001b	6 lines
010b	7 lines **[Default]**
011b	8 lines

PSR2_CTL

		100b: 9 lines 101b: 10 lines 110b: 11 lines 111b: 12 lines **Restriction** To program line 9 to 12, block count number bit [28] must be set.
	9:8	**TP2 Time** This field selects the TP2 time when training the link on exit from PSR2 DeepSleep (waking). **Value** / **Name** 00b: 500us 01b: 100us 10b: 2.5ms 11b: 50us
	7:4	**Frames Before SU Entry** Default Value: 0001b 1 Frames Before SU Entry This field is the number of frames it takes to enter into Selective Update when PSR2 is enabled. Note: HW takes a minimum of 2frames, so'0' and '1' are not valid entries for this field.
	3:0	**Idle Frames** This field is the number of idle frames required before entering PSR2 Deep Sleep. Write to this field doesn't cause a PSR2 exit and frame update. **Value** / **Name** 0000b: Deep Sleep Disabled 0001b: 1 idle frame **[Default]**

PSR2_MAN_TRK_CTL

PSR2_MAN_TRK_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60910h-60913h
Name:	PSR2 Manual Tracking Control
ShortName:	PSR2_MAN_TRK_CTL_A
Reset:	soft

Programming Notes

The frame is divided into blocks of four scan lines each. The blocks are addressed starting from 1 for the first block of the frame and ending with ROUNDUP[(TRANS_VTOTAL Vertical Active + 1) / 4]for the last block of the frame.
Software must provide the starting and ending block address of the selective update region.
The SU Region Start Address is programmed to the first block of the selective update region.
The SU Region End Address is programmed to the final block of the selective update region + 1.
There can be only one selective update region in a frame.
To disable selective update, set the selective update region to the full frame by programming SU Region Start Address to the startof the frame and SU Region End Address to the end of the frame.

DWord	Bit	Description
0	31	**PSR2 Manual Tracking Enable** This bit enables the manual tracking mode for PSR2 Selective Update. Restriction : This field cannot be modified while PSR2_CTL bit [30] is "1'. SW must make sure PSR2_CTL bit [30] is disabled before this bit is disabled. Restriction : This bit should not be modified without a modeset. Value: 0b = Disable; 1b = Enable
	30:21	**SU Region Start Address** This field indicates the starting block address of the selective update region.
	20:11	**SU Region End Address** This field indicates the ending block address of the selective update region.
	10:4	**Reserved** Access: RO Format: MBZ

PSR2_MAN_TRK_CTL

	3	**SF Single full frame** Access: R/W Set This bit will select for a single a full frame fetch, using the regular plane and cursor registers (not SEL_FETCH registers), and update when Selective Fetch is enabled. It is cleared on vblank. Do not set this bit to 1 if Selective Fetch is not enabled.
	2	**SF Continuous full frame** This bit will select for full frame fetches, using the regular plane and cursor registers (not SEL_FETCH registers), and updates continuously when Selective Fetch is enabled, until disabled by software. Do not set this bit to 1 if Selective Fetch is not enabled.
	1	**SF Partial Frame Enable** This field enables the planes to use the SEL_FETCH registers for selective fetch on selective update frames.

Value	Name
0b	Disable
1b	Enable

	0	**Allow Double Buffer Update Disable** This field controls whether double buffer updates are allowed to be disabled for this register.

Value	Name
0b	Not Allowed
1b	Allowed

PSR2_STATUS

PSR2_STATUS

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	60940h-60943h
Name:	PSR2 Status
ShortName:	PSR2_STATUS_A
Reset:	soft

DWord	Bit	Description
0	31:28	**PSR2 State** Access: RO This field indicates the live state of PSR2

Value	Name	Description
0000b	IDLE	Reset state
0001b	CAPTURE	Send capture frame
0010b	CPTURE_FS	Fast sleep after capture frame is sent
0011b	SLEEP	Selective Update
0100b	BUFON_FW	Turn Buffer on and Send Fast wake
0101b	ML_UP	Turn Main link up and send SR
0110b	SU_STANDBY	Selective update or Standby state
0111b	FAST_SLEEP	Send Fast sleep
1000b	DEEP_SLEEP	Enter Deep sleep
1001b	BUF_ON	Turn ON IO Buffer
1010b	TG_ON	Turn ON Timing Generator
1011b	BUFON_FW_2	Turn Buffer on and Send Fast wake for 3 Block case
Others	Reserved	Reserved

	27:26	**Link Status** Access: RO This field indicates the live status of the link.

Value	Name	Description
00b	Full Off	Link is fully off
01b	Full On	Link is fully on
11b	Reserved	Reserved

	25	**Reserved** Access: RO Format: MBZ

PSR2_STATUS

	24:20	**Max Sleep Time Counter** Access: RO This field provides the live status of the sleep time counter.
	19:16	**PSR2 Deep Sleep Entry Count** Access: RO The value in this register represents the number of times PSR2 Deep Sleep has been entered. The count will increment with each entry. After reaching the maximum count value the counter will rollover and continue from 0.
	15:12	**Reserved** Access: RO Format: MBZ
	11:10	**Reserved** Access: RO Format: MBZ
	9	**PSR2 idle frame indication** Access: RO This bit gets set when DP link goes to sleep state and gets reset on PSR2 exit.
	8	**Sending TP2** Access: RO

Description
This field indicates if TP2 is currently being sent.
Recommendation is to not use this status bit.

Value	Name	Description
0b	Not Sending	Not sending TP2
1b	Sending	Sending TP2

	7	**Reserved**
	6	**Reserved**
	5	**PSR2 deep Sleep Entry Completion** Access: R/WC This is a sticky bit which is set on PSR2 deep sleep entry completion. Clear this bit by writing a 1b to it.

Value	Name
0b	Not complete
1b	Complete

PSR2_STATUS

	4	**PSR2 SU Entry Completion** Access: R/WC This is a sticky bit which is set on PSR2 SU entry completion. Clear this bit by writing a 1b to it. **Value** / **Name** 0b: Not complete 1b: Complete
	3:0	**Idle Frame Counter** Access: RO This field provides the live status of the idle frame counter.

PSR2_SU_ECC_STAT

PSR2_SU_ECC_STAT

Register Space:	MMIO: 0/2/0
Access:	R/WC
Size (in bits):	32
Address:	60A64h-60A67h
Name:	PSR2 Selective Update ECC Status
ShortName:	PSR2_SU_ECC_STAT_A
Reset:	soft

Each of these fields is a sticky bit that gives the ECC error status for any PSR2 memory bank.

A '1' in a bit indicates that ECC detected an error. A write of '1' to a bit will clear the bit. Single errors are corrected by ECC. Double errors are not correctable.

DWord	Bit	Description
0	31:17	**Reserved** Access: RO Format: MBZ
	16	**Double Error Any Bank** Description: Errors in banks 8-13 are missed and not reported.
	15:1	**Reserved** Access: RO Format: MBZ
	0	**Single Error Any Bank** Description: Errors in banks 8-13 are missed and not reported.

PSR2_SU_STATUS

PSR2_SU_STATUS	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	96
Address:	60914h-6091Fh
Name:	PSR2 Selective Update Status
ShortName:	PSR2_SU_STATUS_A
Reset:	soft

A frame is divided into selective update blocks of four scan lines each. This register provides the count of the number of selective update blocks per frame, for the last eight frames

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29:20	**Number of SU blocks in frame N - 2** This field indicates the number of selective update blocks in frame N - 1.
	19:10	**Number of SU blocks in frame N - 1** This field indicates the number of selective update blocks in frame N - 1.
	9:0	**Number of SU blocks in frame N** This field indicates the number of selective update blocks in frame N.
1	31:30	**Reserved** Access: RO Format: MBZ
	29:20	**Number of SU blocks in frame N - 5** This field indicates the number of selective update blocks in frame N - 1.
	19:10	**Number of SU blocks in frame N - 4** This field indicates the number of selective update blocks in frame N - 1.
	9:0	**Number of SU blocks in frame N - 3** This field indicates the number of selective update blocks in frame N.
2	31:20	**Reserved** Access: RO Format: MBZ
	19:10	**Number of SU blocks in frame N - 7** This field indicates the number of selective update blocks in frame N - 1.
	9:0	**Number of SU blocks in frame N - 6** This field indicates the number of selective update blocks in frame N.

PTBR Number Of Pages Recorded

PTBR_NUM_PAGES_RECORDED_REGISTER - PTBR Number Of Pages Recorded

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	18594h
Name:	Register Template Address
ShortName:	PTBR_NUM_PAGES_RECORDED_REGISTER

This is a running count of number of pages allocated by the OVR unit for the visibility data. This includes pages that could have been allocated but were not allocated because the OVR unit early terminated the pages with out of memory marker. This register is engine context save/restored.

DWord	Bit	Description
0	31:17	**Reserved** Access: RO Format: MBZ
	16:0	**PTBR_NUM_PAGES_RECORDED** Default Value: 0h Access: R/W Format: U17 _Custom_GTIReset: DEV This is a running count of number of pages allocated by the OVR unit for the visibility data. This includes pages that could have been allocated but were not allocated because the OVR unit early terminated the pages with out of memory marker. **Programming Notes** SW must not write to this register.

PTBR Page Pool Size Register

PTBR_PAGE_POOL_SIZE_REGISTER - PTBR Page Pool Size Register

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	18590h
Name:	Register Template Address
ShortName:	PTBR_PAGE_POOL_SIZE_REGISTER

Indicates the size of the PTBR Page Pool Size allocated by SW. The page pool size is with respect to the PTBR_PAGE_POOL_BASE_ADDRESS programmed through 3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS command. SW can do multiple writes to this register with the increased page pool size as it allocates more pages to the PTBR page pool.
Coming out of reset or on executing PTBR_PAGE_POOL_RESTART by 3DSTATE_PTRB_PAGE_POOL_BASE_ADDRESS command, HW initializes PTBR_PAGE_POOL_SIZE to 0x0. HW on detecting a write to this register compares its current PTBR_PAGE_POOL_SIZE with that of the value programmed in this register to add more pages to the Free-List and updates itself with the latest value.
This is an non-privileged register and engine context save/restored by HW.

Programming Notes

- SW must not write a value to this register less than the existing value. SW must use the Restart field of the 3DSTATE_PTBR_PAGE_POOL_ADDRESS command to decrease the size of the pool.
- SW must always program PTBR_PAGE_POOL_SIZE_REGISTER through MI_LOAD_REGISTER_IMM command or any other MI command to load register in the command sequence.
- SW must ensure following state are set in HW prior to programming this register: 3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS 3DSTATE_PTBR_FEE_LIST_BASE_ADDRESS

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**PTBR_PAGE_POOL_SIZE** Default Value: 0h Access: R/W Format: U16-1 _Custom_GTIReset: DEV Indicates the PTBR page pool size (4KB granularity). A value of '0x0' indicates a single page and a value of 99h indicates 154 pages are available in the page pool for use by HW. Valid Range [0..65534].

PTE SW Fault Repair High

PTESWC_H - PTE SW Fault Repair High		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	DEV	
Address:	0CEF0h	
DWord	**Bit**	**Description**
0	31:0	**Repair Address High** Default Value: 00000000h Access: R/W Fixed PTE entry is written by SW here.

PTE SW Fault Repair Low

PTESWC_L - PTE SW Fault Repair Low		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	DEV	
Address:	0CEF4h	
DWord	**Bit**	**Description**
0	31:0	**Repair Address Low** Access: R/W Fixed PTE entry is written by SW here.

PWR_WELL_CTL

PWR_WELL_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	45400h-45403h
Name:	Power Well Control 1
ShortName:	PWR_WELL_CTL1
Reset:	soft
CrashLogSaved:	true
CrashLogPriority:	2
CrashLogVisibility:	public
ExternalLongName:	DE Power Well Control 1
ExternalDescription:	Display engine power well control
Address:	45404h-45407h
Name:	Power Well Control 2
ShortName:	PWR_WELL_CTL2
Reset:	soft

This register is used for display power control. There are multiple instances of this register format to allow software components to have parallel control of the display power.
PWR_WELL_CTL1 is generally used for BIOS to control power.
PWR_WELL_CTL2 is generally used for driver to control power.
The power enable requests from all sources are logically ORd together to enable the power, so the power will only disable after all sources have requested the power to disable.
When a power well is disabled (powered down), access to any registers in the power well will complete, but write data will be dropped and read data will be all zeroes.
The display connections diagram indicates which functional blocks are contained in each power well. The display MMIO register specification has a field for each register to indicate which power well it is in.

Restriction
The power request field must not be changed for a resource while a power enable/disable for that resource is currently in progress, as indicated by power well state for that resource.
Power wells must be enabled and disabled following the display initialization and mode set sequences.

DWord	Bit	Description
0	31:22	**Reserved** Access: RO Format: MBZ
	21:10	**Reserved** Access: RO Format: MBZ

PWR_WELL_CTL

	9	**Power Well 5 Request** Access: R/W This field requests power well to enable or disable. Value: 0b – Disable Value: 1b – Enable
	8	**Power Well 5 State** Access: RO This field indicates the status of power well. Value: 0b – Disabled Value: 1b – Enabled
	7	**Power Well 4 Request** Access: R/W This field requests power well to enable or disable. Value: 0b – Disable Value: 1b – Enable
	6	**Power Well 4 State** Access: RO This field indicates the status of power well. Value: 0b – Disabled Value: 1b – Enabled
	5	**Power Well 3 Request** Access: R/W This field requests power well to enable or disable. Value: 0b – Disable Value: 1b – Enable
	4	**Power Well 3 State** Access: RO This field indicates the status of power well. Value: 0b – Disabled Value: 1b – Enabled

PWR_WELL_CTL

	3	**Power Well 2 Request** Access: R/W This field requests power well to enable or disable. Value: 0b — Name: Disable Value: 1b — Name: Enable
	2	**Power Well 2 State** Access: RO This field indicates the status of power well. Value: 0b — Name: Disabled Value: 1b — Name: Enabled
	1	**Power Well 1 Request** Access: R/W This field requests power well to enable or disable. Value: 0b — Name: Disable Value: 1b — Name: Enable
	0	**Power Well 1 State** Access: RO This field indicates the status of power well. Value: 0b — Name: Disabled Value: 1b — Name: Enabled

PWR_WELL_CTL_AUX

PWR_WELL_CTL_AUX

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	45440h-45443h
Name:	Power Well Control AUX 1
ShortName:	PWR_WELL_CTL_AUX1
Reset:	soft
Address:	45444h-45447h
Name:	Power Well Control AUX 2
ShortName:	PWR_WELL_CTL_AUX2
Reset:	soft

This register is used for display power control. There are multiple instances of this register format to allow software components to have parallel control of the display power.
PWR_WELL_CTL_AUX1 is generally used for BIOS to control power.
PWR_WELL_CTL_AUX2 is generally used for driver to control power.
The power enable requests from all sources are logically ORd together to enable the power, so the power will only disable after all sources have requested the power to disable.
When a power well is disabled (powered down), access to any registers in the power well will complete, but write data will be dropped and read data will be all zeroes.
The display connections diagram indicates which functional blocks are contained in each power well. The display MMIO register specification has a field for each register to indicate which power well it is in.

Restriction

The power request field must not be changed for a resource while a power enable/disable for that resource is currently in progress, as indicated by power well state for that resource.

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29	**AUX TBT6 IO Power Request** Access: R/W This field requests power for this Thunderbolt Aux IO to enable or disable. This is used for typeC ports that are in thunderbolt mode. 0b – Disable 1b – Enable

PWR_WELL_CTL_AUX

28	**AUX TBT6 IO Power State** Access: RO This field indicates the status of power for this Thunderbolt Aux IO. 0b: Disable 1b: Enable
27	**AUX TBT5 IO Power Request** Access: R/W This field requests power for this Thunderbolt Aux IO to enable or disable. This is used for typeC ports that are in thunderbolt mode. 0b: Disable 1b: Enable
26	**AUX TBT5 IO Power State** Access: RO This field indicates the status of power for this Thunderbolt Aux IO. 0b: Disable 1b: Enable
25	**AUX TBT4 IO Power Request** Access: R/W This field requests power for this Thunderbolt Aux IO to enable or disable. This is used for typeC ports that are in thunderbolt mode. 0b: Disable 1b: Enable
24	**AUX TBT4 IO Power State** Access: RO This field indicates the status of power for this Thunderbolt Aux IO. 0b: Disable 1b: Enable
23	**AUX TBT3 IO Power Request** Access: R/W This field requests power for this Thunderbolt Aux IO to enable or disable. This is used for typeC ports that are in thunderbolt mode. 0b: Disable

PWR_WELL_CTL_AUX

	1b	Enable

22 AUX TBT3 IO Power State

Access:	RO

This field indicates the status of power for this Thunderbolt Aux IO.

Value	Name
0b	Disable
1b	Enable

21 AUX TBT2 IO Power Request

Access:	R/W

This field requests power for this Thunderbolt Aux IO to enable or disable. This is used for typeC ports that are in thunderbolt mode.

Value	Name
0b	Disable
1b	Enable

20 AUX TBT2 IO Power State

Access:	RO

This field indicates the status of power for this Thunderbolt Aux IO.

Value	Name
0b	Disable
1b	Enable

19 AUX TBT1 IO Power Request

Access:	R/W

This field requests power for this Thunderbolt Aux IO to enable or disable. This is used for typeC ports that are in thunderbolt mode.

Value	Name
0b	Disable
1b	Enable

18 AUX TBT1 IO Power State

Access:	RO

This field indicates the status of power for this Thunderbolt Aux IO.

Value	Name
0b	Disable
1b	Enable

17 USBC6 IO Power Request

Access:	R/W

This field requests power for this USBC Aux IO to enable or disable. This is used for typeC ports that are not in thunderbolt mode.

PWR_WELL_CTL_AUX

Value	Name
0b	Disable
1b	Enable

16 **USBC6 IO Power State**

Access:	RO

This field indicates the status of power for this USBC Aux IO.

Value	Name
0b	Disable
1b	Enable

15 **USBC5 IO Power Request**

Access:	R/W

This field requests power for this USBC Aux IO to enable or disable. This is used for typeC ports that are not in thunderbolt mode.

Value	Name
0b	Disable
1b	Enable

14 **USBC5 IO Power State**

Access:	RO

This field indicates the status of power for this USBC Aux IO.

Value	Name
0b	Disable
1b	Enable

13 **USBC4 IO Power Request**

Access:	R/W

This field requests power for this USBC Aux IO to enable or disable. This is used for typeC ports that are not in thunderbolt mode.

Value	Name
0b	Disable
1b	Enable

12 **USBC4 IO Power State**

Access:	RO

This field indicates the status of power for this USBC Aux IO.

Value	Name
0b	Disable
1b	Enable

PWR_WELL_CTL_AUX

11 USBC3 IO Power Request

Access:	R/W

This field requests power for this USBC Aux IO to enable or disable. This is used for typeC ports that are not in thunderbolt mode.

Value	Name
0b	Disable
1b	Enable

10 USBC3 IO Power State

Access:	RO

This field indicates the status of power for this USBC Aux IO.

Value	Name
0b	Disable
1b	Enable

9 USBC2 IO Power Request

Access:	R/W

This field requests power for this USBC Aux IO to enable or disable. This is used for typeC ports that are not in thunderbolt mode.

Value	Name
0b	Disable
1b	Enable

8 USBC2 IO Power State

Access:	RO

This field indicates the status of power for this USBC Aux IO.

Value	Name
0b	Disable
1b	Enable

7 USBC1 IO Power Request

Access:	R/W

This field requests power for this USBC Aux IO to enable or disable. This is used for typeC ports that are not in thunderbolt mode.

Value	Name
0b	Disable
1b	Enable

6 USBC1 IO Power State

Access:	RO

This field indicates the status of power for this USBC Aux IO.

Value	Name
0b	Disabled

PWR_WELL_CTL_AUX

	1b	Enabled

5 AUX C IO Power Request

Access:	R/W

This field requests power for this Aux IO to enable or disable.

Value	Name
0b	Disable
1b	Enable

4 AUX C IO Power State

Access:	RO

This field indicates the status of power for this Aux IO.

Value	Name
0b	Disabled
1b	Enabled

3 AUX B IO Power Request

Access:	R/W

This field requests power for this Aux IO to enable or disable.

Value	Name
0b	Disable
1b	Enable

2 AUX B IO Power State

Access:	RO

This field requests power for this Aux IO to enable or disable.

Value	Name
0b	Disabled
1b	Enabled

1 AUX A IO Power Request

Access:	R/W

This field requests power for this Aux IO to enable or disable.

Value	Name
0b	Disable
1b	Enable

0 AUX A IO Power State

Access:	RO

This field indicates the status of power for this Aux IO.

Value	Name
0b	Disabled
1b	Enabled

PWR_WELL_CTL_DDI

PWR_WELL_CTL_DDI

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	45450h-45453h
Name:	Power Well Control DDI 1
ShortName:	PWR_WELL_CTL_DDI1
Reset:	soft
Address:	45454h-45457h
Name:	Power Well Control DDI 2
ShortName:	PWR_WELL_CTL_DDI2
Reset:	soft

This register is used for display power control. There are multiple instances of this register format to allow software components to have parallel control of the display power.
PWR_WELL_CTL _DDI1 is generally used for BIOS to control power.
PWR_WELL_CTL_DDI2 is generally used for driver to control power.
The power enable requests from all sources are logically ORd together to enable the power, so the power will only disable after all sources have requested the power to disable.
When a power well is disabled (powered down), access to any registers in the power well will complete, but write data will be dropped and read data will be all zeroes.
The display connections diagram indicates which functional blocks are contained in each power well. The display MMIO register specification has a field for each register to indicate which power well it is in.

Restriction

The power request field must not be changed for a resource while a power enable/disable for that resource is currently in progress, as indicated by power well state for that resource.

DWord	Bit	Description
0	31:23	**Reserved** Access: RO Format: MBZ
	22:18	**Reserved** Access: RO Format: MBZ
	17	**USBC6 IO Power Request** Access: R/W This field requests power for USBC6 IO to enable or disable. Value: 0b — Name: Disable

PWR_WELL_CTL_DDI

	1b	Enable

16 USBC6 IO Power State

Access:	RO

This field indicates the status of power for USBC6 IO.

Value	Name
0b	Disable
1b	Enable

15 USBC5 IO Power Request

Access:	R/W

This field requests power for USBC5 IO to enable or disable.

Value	Name
0b	Disable
1b	Enable

14 USBC5 IO Power State

Access:	RO

This field indicates the status of power for USBC5 IO.

Value	Name
0b	Disable
1b	Enable

13 USBC4 IO Power Request

Access:	R/W

This field requests power for USBC4 IO to enable or disable.

Value	Name
0b	Disable
1b	Enable

12 USBC4 IO Power State

Access:	RO

This field indicates the status of power for USBC4 IO.

Value	Name
0b	Disable
1b	Enable

11 USBC3 IO Power Request

Access:	R/W

This field requests power for USBC3 IO to enable or disable.

Value	Name
0b	Disable
1b	Enable

PWR_WELL_CTL_DDI

10	**USBC3 IO Power State** Access: RO This field indicates the status of power for USBC3 IO. 0b – Disable 1b – Enable
9	**USBC2 IO Power Request** Access: R/W This field requests power for USBC2 IO to enable or disable. 0b – Disable 1b – Enable
8	**USBC2 IO Power State** Access: RO This field indicates the status of power for USBC2 IO. 0b – Disabled 1b – Enabled
7	**USBC1 IO Power Request** Access: R/W This field requests power for USBC1 IO to enable or disable. 0b – Disable 1b – Enable
6	**USBC1 IO Power State** Access: RO This field indicates the status of power for USBC1 IO. 0b – Disabled 1b – Enabled
5	**DDI C IO Power Request** Access: R/W This field requests power for DDI C IO to enable or disable. 0b – Disable 1b – Enable

PWR_WELL_CTL_DDI

4 — DDI C IO Power State

Access:	RO

This field indicates the status of power for DDI C IO.

Value	Name
0b	Disabled
1b	Enabled

3 — DDI B IO Power Request

Access:	R/W

This field requests power for DDI B IO to enable or disable.

Value	Name
0b	Disable
1b	Enable

2 — DDI B IO Power State

Access:	RO

This field indicates the status of power for DDI B IO.

Value	Name
0b	Disabled
1b	Enabled

1 — DDI A IO Power Request

Access:	R/W

This field requests power for DDI A IO to enable or disable.

Value	Name
0b	Disable
1b	Enable

0 — DDI A IO Power State

Access:	RO

This field indicates the status of power for DDI A IO.

Value	Name
0b	Disabled
1b	Enabled

PWRCTXSAVE Message Register for Boot Controller Unit

MSG_PWRCTXSAVE_MBC - PWRCTXSAVE Message Register for Boot Controller Unit

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	0850Ch

Message registers have bit-wise masking applied for writes. The register consists of 16 bits of data in [15:0], and 16 bits of corresponding masks in [31:16].
To **set** bit0, for example, the data would be 0x0001_0001.
To **clear** bit0, for example, the data would be 0x0001_0000.
Note that mask bit is the data bit offset + 16.
Message registers are protected from non-GT writes via the Message Channel.

DWord	Bit	Description
0	31:10	**Reserved** Access: RO Format: MBZ
	9	**Power Context Save Request** Access: R/W _Custom_GTIReset: BUS Power Context Save Request 1'b0: Power context save is not being requested (default). 1'b1: Power context save is being requested. Unit needs to self-clear this bit upon sampling.
	8:0	**QWord Credits for Power Context Save Request** Access: R/W _Custom_GTIReset: BUS QWord Credits for Power Context Save Request. Minimum Credits = 1: Unit may send 1 QWord pair (enough for first LRI at least). Maximum Credits = 511: Unit may send 511 QWord pairs. A QWord pair is defined as a 32-bit register address and the corresponding 32-bitsof register data. Note that the LRI header and END commands are 64-bits each(32-bit command followed by 32-bit NOOP) and consume one QWord credit. Only valid with PWRCTXSAVE_REQ (Bit9).

RC6 Context Base

RC6CTXBASE - RC6 Context Base		
Register Space:	MMIO: 0/2/0	
Size (in bits):	64	
Address:	00D48h	
RC6 Location		
DWord	**Bit**	**Description**
0	31:12	**RC6 Memory Base Low** Access: R/W Lock _Custom_GTIReset: BUS This field is used to set the base of memory where the RC6 power context will be saved This value MUST be above the base and below the top of stolen memory This register is locked (becomes read-only) when RC6MEMLOCK is 1
	11:1	**Reserved** Access: RO Format: MBZ
	0	**RC6Context Base Register Lock** Access: R/W Lock _Custom_GTIReset: BUS 1'b0 : All fields of this register are writable (default) 1'b1 : This register is Read Only BIOS must set this bit to prevent further changes
1	31:0	**RC6 Memory Base High** Access: R/W Lock _Custom_GTIReset: BUS This field is used to set the base of memory where the RC6 power context will be saved. This value MUST be above the base and below the top of stolen memory. This High Dword must be written before the low word is written with RC6MEMLOCK of 1. This register is locked (becomes read-only) when RC6MEMLOCK is 1

RC6 LOCATION

RC6LOCATION - RC6 LOCATION

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00D40h

RC6 Location

DWord	Bit	Description
0	31	**RC6Context Location Lock** Access: R/W Lock _Custom_GTIReset: BUS 1'b0 : All fields of this register are writable (default) 1'b1 : This register is Read Only
	30:1	**Reserved** Access: RO Format: MBZ
	0	**RC6Context Location** Access: RO _Custom_GTIReset: BUS 1'b1 : Send context data to DRAM location specified in RC6MEMBASE (default) This will be tied to 1 with as there is no option to save context to a SRAM

Register_HCPBitstreamOutputCABACInsertionCount

HCP_CABAC_INSERTION_COUNT - Register_HCPBitstreamOutputCABACInsertionCount

Register Space:	MMIO: 0/2/0
Source:	VideoCS1
Access:	RO
Size (in bits):	32
_Custom_GTIReset:	BUS

This register stores the count in bytes of **CABAC ZERO_WORD insertion. It is primarily provided for statistical data gathering.**

DWord	Bit	Description
0	31:0	**HCP Cabac Insertion Count** Access: RO Format: U32 Total number of bytes in the bitstream output before for the CABAC zero word insertion. This count is updated each time when the insertion count is incremented.

Register_HCPSFCLOCKRequest

HCP_SFC_LOCK_REQUEST - Register_HCPSFCLOCKRequest	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	1C2910h
ShortName:	HCP_SFC_LOCK_REQUEST_VCS0
Description:	For VDBox0
Address:	1D2910h
ShortName:	HCP_SFC_LOCK_REQUEST_VCS2
Description:	For VDBox2
Address:	1E2910h
ShortName:	HCP_SFC_LOCK_REQUEST_VCS4
Description:	For VDBox4
Address:	1F2910h
ShortName:	HCP_SFC_LOCK_REQUEST_VCS6
Description:	For VDBox6

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**HCP_SFC_Forced_Lock** Format: U1 This bit can only be set by driver and it has to be clear by driver as well. Driver should set this bit before issuing the software (watchdog timer) reset. It tells HCP that a software reset is going to happen. HCP then issues a forced lock to SFC. If SFC is currently locked to HCP, SFC should not unlock itself from HCP. If SFC is NOT currently locked to HCP, SFC should not accept the lock request from HCP. Driver needs to clear this bit after the software reset sequence is complete.

Register_HCPSFCLOCKStatus

HCP_SFC_LOCK_STATUS - Register_HCPSFCLOCKStatus

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1C2914h
ShortName:	HCP_SFC_LOCK_STATUS_VCS0
Description:	For VDbox0
Address:	1D2914h
ShortName:	HCP_SFC_LOCK_STATUS_VCS2
Description:	For VDBox2
Address:	1E2914h
ShortName:	HCP_SFC_LOCK_STATUS_VCS4
Description:	For VDBox4
Address:	1F2914h
ShortName:	HCP_SFC_LOCK_STATUS_VCS6
Description:	For VDBox6

DWord	Bit	Description
0	31:2	**Reserved** Access: RO Format: MBZ
	1	**HCP_SFC_Forced_Act** Format: U1 This bit can only be set by hardware and it has to be clear by hardware as well. This bit is going to be polled by driver. This bit indicates that HCP has received HCP_SFC_Forced_Lock from driver and it has sent that signal to SFC. Once this bit is set, it indicates SFC status (lock or unlock) will not be changed anymore. Driver will be safe to start the reset process after this bit is set. Hardware has to de-assert this bit after driver de-assert HCP_SFC_Forced_Lock as well.
	0	**HCP_SFC_Usage** Format: U1 This bit can only be set by hardware and it has to be clear by hardware as well. This bit indicates SFC is currently locked to HCP. This bit should be set after SFC accepts the lock request from HCP. This bit should be clear once SFC finishes the workload and unlocked from HCP. In case a reset happens on HCP, this bit must be reset once a new workload is received

Render and Media Context Restore Needed

RENDER_MEDIA_NEED_RESTORE - Render and Media Context Restore Needed		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	080F0h	
DWord	**Bit**	**Description**
0	31:2	**Reserved** Access: RO
	1	**Render Context Restore is needed** Access: RO _Custom_GTIReset: BUS Render Context Restore Needed Flag to support Render Powergating Feature This flag will let PM know if Media context need to be restored on C6 exit 1'b0: Do not proceed if Wait for VCR1 is set (default). 1'b1: Proceed to unblocking FIFO.
	0	**Media Context Restore is needed** Access: RO _Custom_GTIReset: BUS Render Context Restore Needed Flag to support Render Powergating Feature This flag will let PM know if Media context need to be restored on C6 exit 1'b0: Do not proceed if Wait for VCR1 is set (default). 1'b1: Proceed to unblocking FIFO.

Render Control Unit Mode Register

RCU_MODE - Render Control Unit Mode Register

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	14800h

Mode register for Render Control Unit (RCU).

DWord	Bit	Description
0	31:16	**Mask** Access: WO Format: Mask Must be set to modify corresponding bit in Bits 15:0. (All implemented bits)
	15:5	**Reserved** Format: PBC
	4:3	**Reserved** Format: PBC
	2	**Disable Dual Context Limiting to Dual Queue** This bit enables two contexts of different address space to run in parallel. By default, HW enables RenderCS and ComputeCS to run contexts concurrently only when the executing contexts are from the same address space otherwise the context execution gets serialized. Two contexts are said to be from the same address space when they have the identical Virtual Function Number and the Page Directory Pointer-0. This both controls disabling of the default HW behavior. **Value** / **Name** / **Description** 0 / Force Run Alone when different context **[Default]** / HW allows concurrent execution only when both contexts are from the same address space. 1 / Enable Dual Context / Concurrent execution of context on RenderCS and ComputeCS is allowed with no address space comparison.
	1	**Reserved** Format: PBC
	0	**Compute Engine Enable** **Description** This bit indicates if Compute Engine (a.k.a Dual Context or Multi Context) is enabled or not. This bit must be treated as global control for enabling and disabling of compute engine. Hardware allocates required resources for the compute engine based on this bit. • Mid Thread Preemption for GPGPU workloads is not supported when compute engine is enabled. Hardware implicitly demotes the GPGPU preemption granularity to Thread Group or lower irrespective of the preemtion granularity programmed through GPGPU_PREEMPTION_CONTROL

mode bits.

- HW reserves 4KB of URB space per bank for Compute Engine out of the total storage available in L3. SW must consider that 4KB of storage per bank will be reduced from what is programmed for the URB space in L3 for Render Engine executed workloads. Example: When URB space programmed is 64KB (per bank) for Render Engine, the actual URB space available for operation is only 60KB (per bank). Similarly, when URB space programmed is 128KB (per bank) for render engine, the actual URB space available for operation is only 124KB (per bank). More detailed description available in "L3 Cache" section of the PRM.

In Dual Context mode of operation Compute Engine and Render Engine address space separation is based on the memory access address bit [47]. Each of the engine's addressable space is limited to 128 TB ([46:0]).

Value	Name	Description
0	**[Default]**	Compute engine is disabled.
1		Compute engine is enabled.

Programming Notes
This bit must be programmed when render engine is idle with no contexts getting executed.

Render Control Unit Power Clock State Register

RCU_PWR_CLK_STATE - Render Control Unit Power Clock State Register

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	148C8h

This register provides a mechanism to override the PWR_CLK_STATE requested by the RenderCS or ComputeCS. Both ComputeCS and RenderCS make their R_PWR_CLK_STATE request to Render Control Unit (RCU), RCU computes the required PWR_CLK_STATE and interfaces with power management to get the desired state.

DWord	Bit	Description
0	31	**Power Clock State Enable** Format: U1 When enabled (set) RCU will force the PWR_CLK_STATE programmed in the RCU_PWR_CLK_STATE to Power Management instead of the PWR_CLK_STATE received from the RenderCS or ComputeCS. RCU will enforce the updated value of RCU_PWR_CLK_STATE on an engines PWR_CLK_STATE request when the requesting engine is the only active engine in the GPU, i.e RCU will ensure the other engines are not active and are not impacted on changing the PWR_CLK_STATE. (Example: Updated value in RCU_PWR_CLK_STATE will be used to override the RenderCS PWR_CLK_STATE request when ComputeCS is idle or Vice-Versa).
	30:0	**Render Power Clock State** Format: U31

Value	Name	Description
0h	Power Clock State Disabled	No specific power state set, bits[30:0] are ignored.
1h	Power Clock State Enabled	Power state is set and bit[30:0] are valid and have the desired state. RCU will use the powe state to overide the values recevied from RenderCS or ComputeCS.

Reported BitRateControl Convergence Status

MFX_VP8_BRC_CONVERGENCE_STATUS - Reported BitRateControl Convergence Status

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	1C928h
ShortName:	MFX_VP8_BRC_CONVERGENCE_STATUS_VD2

This register stores BitRateControl Convergence Status.

DWord	Bit	Description
0	31	**Reserved**
	30:28	**Reserved**
	27	**Reserved**
	26:24	**Reserved**
	23	**Reserved**
	22:20	**Reserved**
	19	**Reserved**
	18:16	**Reserved**
	15:12	**Reserved** Access: RO Format: MBZ
	11:8	**Total Num of Pass** Format: U4 This bit indicates the number of Multipass including current frame. Note that Initial Pass is not counted.
	7:2	**Reserved** Access: RO Format: MBZ
	1	**Overflow OR Underflow Flag** Format: U1 This bit indicates the current frame has BRC overflow OR underflow.
	0	**MB Max. Conformance Flag** Format: U1 This contains flag that indicate Inter MB or Intra MB Max. Conformance is not met. This is legacy support and this feature is not validated.

Reported BitRateControl CumulativeDeltaLoopFilter and LoopFilter 01

MFX_VP8_BRC_CUMULATIVE_D_LOOP_FILTER01 - Reported BitRateControl CumulativeDeltaLoopFilter and LoopFilter 01

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	1C920h

This register stores per segment Bit Rate Control DeltaLoopFilter.

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30:24	**Segment1 CumulativeDeltaLoopFilter** Format: S6 This contains Segment1 CumulativeDeltaLoopFilter in Bit Rate Control. It accumulates all Segment1 DeltaQindices per pass in Multipass. This register is valid after a BRC pass is done.
	23:22	**Reserved** Access: RO Format: MBZ
	21:16	**Segment1 LoopFilter** Format: U6 This contains Segment1 LoopFilter used in current frame. This register is valid after a BRC pass is done.
	15	**Reserved** Access: RO Format: MBZ
	14:8	**Segment0 CumulativeDeltaLoopFilter** Format: S6 This contains Segment0 CumulativeDeltaLoopFilter in Bit Rate Control. It accumulates all Segment0 DeltaQindices per pass in Multipass. This register is valid after a BRC pass is done. If Segmentation is not enabled, this field reflects CumulativeDeltaLoopFilter. If Segmentation is enabled, this field reflects Segment0 CumulativeDeltaLoopFilter.
	7:6	**Reserved** Access: RO Format: MBZ

MFX_VP8_BRC_CUMULATIVE_D_LOOP_FILTER01 - Reported BitRateControl CumulativeDeltaLoopFilter and LoopFilter 01		
	5:0	**Segment0 LoopFilter** Format: U6 This contains Segment0 LoopFilter used in current frame. This register is valid after a BRC pass is done. If Segmentation is not enabled, this field reflects LoopFilter. If Segmentation is enabled, this field reflects Segment0 LoopFilter.

Reported BitRateControl CumulativeDeltaLoopFilter and LoopFilter 23

MFX_VP8_BRC_CUMULATIVE_D_LOOP_FILTER23 - Reported BitRateControl CumulativeDeltaLoopFilter and LoopFilter 23

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	1C924h

This register stores per segment Bit Rate Control DeltaLoopFilter.

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30:24	**Segment3 CumulativeDeltaLoopFilter** Format: S6 This contains Segment3 CumulativeDeltaLoopFilter in Bit Rate Control. It accumulates all Segment3 DeltaQindices per pass in Multipass. This register is valid after a BRC pass is done.
	23:22	**Reserved** Access: RO Format: MBZ
	21:16	**Segment3 LoopFilter** Format: U6 This contains Segment3 LoopFilter used in current frame. This register is valid after a BRC pass is done.
	15	**Reserved** Access: RO Format: MBZ
	14:8	**Segment2 CumulativeDeltaLoopFilter** Format: S6 This contains Segment2 CumulativeDeltaLoopFilter in Bit Rate Control. It accumulates all Segment2 DeltaQindices per pass in Multipass. This register is valid after a BRC pass is done.
	7:6	**Reserved** Access: RO Format: MBZ

MFX_VP8_BRC_CUMULATIVE_D_LOOP_FILTER23 - Reported BitRateControl CumulativeDeltaLoopFilter and LoopFilter 23		
	5:0	**Segment2 LoopFilter** Format: U6 This contains Segment2 LoopFilter used in current frame. This register is valid after a BRC pass is done.

Reported BitRateControl CumulativeDeltaQindex and Qindex 01

MFX_VP8_BRC_CUMULATIVE_DQ_INDEX01 - Reported BitRateControl CumulativeDeltaQindex and Qindex 01

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	12918h
ShortName:	MFX_VP8_BRC_CUMULATIVE_DQ_INDEX01_VB0
Address:	1C918h
ShortName:	MFX_VP8_BRC_CUMULATIVE_DQ_INDEX01_VB1

This register stores per segment Bit Rate Control CumulativeDeltaQindex.

DWord	Bit	Description
0	31:24	**Segment1 CumulativeDeltaQindex** Format: S7 This contains Segment1 CumulativeDeltaQindex in Bit Rate Control. It accumulates all Segment1 DeltaQindices per pass in Multipass. This register is valid after a BRC pass is done.
	23	**Reserved** Access: RO Format: MBZ
	22:16	**Segment1 Qindex** Format: U7 This contains Segment1 Qindex used in current frame. This register is valid after a BRC pass is done.
	15:8	**Segment0 CumulativeDeltaQindex** Format: S7 This contains Segment0 CumulativeDeltaQindex in Bit Rate Control. It accumulates all Segment0 DeltaQindices per pass in Multipass. This register is valid after a BRC pass is done. If Segmentation is not enabled, this field reflects CumulativeDeltaQindex. If Segmentation is enabled, this field reflects Segment0 CumulativeDeltaQindex.
	7	**Reserved** Access: RO Format: MBZ
	6:0	**Segment0 Qindex** Format: U7 This contains Segment0 Qindex used in current frame. This register is valid after a BRC pass is done. If Segmentation is not enabled, this field reflects Qindex. If Segmentation is enabled, this field reflects Segment0 Qindex.

Reported BitRateControl CumulativeDeltaQindex and Qindex 23

MFX_VP8_BRC_CUMULATIVE_DQ_INDEX23 - Reported BitRateControl CumulativeDeltaQindex and Qindex 23

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	1291Ch
ShortName:	MFX_VP8_BRC_CUMULATIVE_DQ_INDEX23_VB0
Address:	1C91Ch
ShortName:	MFX_VP8_BRC_CUMULATIVE_DQ_INDEX23_VB1

This register stores per segment Bit Rate Control CumulativeDeltaQindex and Qindex.

DWord	Bit	Description
0	31:24	**Segment3 CumulativeDeltaQindex** Format: S7 This contains Segment3 CumulativeDeltaQindex in Bit Rate Control. It accumulates all Segment3 DeltaQindices per pass in Multipass. This register is valid after a BRC pass is done.
	23	**Reserved** Access: RO Format: MBZ
	22:16	**Segment3 Qindex** Format: U7 This contains Segment3 Qindex used in current frame. This register is valid after a BRC pass is done.
	15:8	**Segment2 CumulativeDeltaQindex** Format: S7 This contains Segment2 CumulativeDeltaQindex in Bit Rate Control. It accumulates all Segment2 DeltaQindices per pass in Multipass. This register is valid after a BRC pass is done.
	7	**Reserved** Access: RO Format: MBZ
	6:0	**Segment2 Qindex** Format: U7 This contains Segment2 Qindex used in current frame. This register is valid after a BRC pass is done.

Reported BitRateControl DeltaLoopFilter

MFX_VP8_BRC_D_LOOP_FILTER - Reported BitRateControl DeltaLoopFilter

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	12914h
ShortName:	MFX_VP8_BRC_D_LOOP_FILTER_VB0
Address:	1C914h
ShortName:	MFX_VP8_BRC_D_LOOP_FILTER_VB1

This register stores per segment Bit Rate Control DeltaLoopFilter.

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30:24	**Segment3 DeltaLoopFilter** Format: S6 This contains Segment3 DeltaLoopFilter in Bit Rate Control. This register is valid after a BRC pass is done
	23	**Reserved** Access: RO Format: MBZ
	22:16	**Segment2 DeltaLoopFilter** Format: S6 This contains Segment2 DeltaLoopFilter in Bit Rate Control. This register is valid after a BRC pass is done
	15	**Reserved** Access: RO Format: MBZ
	14:8	**Segment1 DeltaLoopFilter** Format: S6 This contains Segment1 DeltaLoopFilter in Bit Rate Control. This register is valid after a BRC pass is done.
	7	**Reserved** Access: RO Format: MBZ

MFX_VP8_BRC_D_LOOP_FILTER - Reported BitRateControl DeltaLoopFilter		
	6:0	**Segment0 DeltaLoopFilter** Format: S6 This contains Segment0 DeltaLoopFilter in Bit Rate Control. This register is valid after a BRC pass is done. If Segmentation is not enabled, this field reflects DeltaLoopFilter. If Segmentation is enabled, this field reflects Segment0 DeltaLoopFilter.

Reported BitRateControl DeltaQindex

MFX_VP8_BRC_DQ_INDEX - Reported BitRateControl DeltaQindex

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	12910h
ShortName:	MFX_VP8_BRC_DQ_INDEX_VB0
Address:	1C910h
ShortName:	MFX_VP8_BRC_DQ_INDEX_VB1

This register stores per segment Bit Rate Control DeltaQindex.

DWord	Bit	Description
0	31:24	**Segment3 DeltaQindex** Format: S7 This contains Segment3 DeltaQindex in Bit Rate Control. This register is valid after a BRC pass is done
	23:16	**Segment2 DeltaQindex** Format: S7 This contains Segment2 DeltaQindex in Bit Rate Control. This register is valid after a BRC pass is done
	15:8	**Segment1 DeltaQindex** Format: S7 This contains Segment1 DeltaQindex in Bit Rate Control. This register is valid after a BRC pass is done.
	7:0	**Segment0 DeltaQindex** Format: S7 This contains Segment0 DeltaQindex in Bit Rate Control. This register is valid after a BRC pass is done. If Segmentation is not enabled, this field reflects DeltaQindex. If Segmentation is enabled, this field reflects Segment0 DeltaQindex.

Reported BitRateControl parameter Mask

MFX_VP8_CNTRL_MASK - Reported BitRateControl parameter Mask

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	12900h
ShortName:	MFX_VP8_CNTRL_MASK_VB0
Address:	1C900h
ShortName:	MFX_VP8_CNTRL_MASK_VB1

This register stores the count of bytes of the bitstream output per frame

DWord	Bit	Description
0	31:6	**Reserved** Access: RO Format: MBZ
	5	**Final Bitstream Buffer Overrun Mask** Format: U1 This is same bit reflected in MFX_VP8_ENC_CFG DW2 bit5. This denotes Final bitstream buffer overrun feature is enabled.
	4	**Intermediate Bitstream Buffer Overrun Mask** Format: U1 This is same bit reflected in MFX_VP8_ENC_CFG DW2 bit4. This denotes intermediate bitstream buffer overrun feature is enabled.
	3	**Intra MB Bit Count Conformance Mask** Format: U1 This is legacy support as AVC for Intra MB Bit Count conformance. This is same bit reflected in MFX_VP8_ENC_CFG DW2 bit3. This feature is not validated.
	2	**Inter MB Bit Count Conformance Mask** Format: U1 This is legacy support as AVC for Inter MB Bit Count conformance. This is same bit reflected in MFX_VP8_ENC_CFG DW2 bit2. This feature is not validated.
	1	**Frame Bit Rate Overflow Mask** Format: U1 This is same bit reflected in MFX_VP8_ENC_CFG DW2 bit0. It denotes if Frame Bit Rate Overflow is enabled for Bit Rate Control

MFX_VP8_CNTRL_MASK - Reported BitRateControl parameter Mask		
	0	**Frame Bit Rate Underflow Mask** Format: U1 This is same bit reflected in MFX_VP8_ENC_CFG DW2 bit1. It denotes if Frame Bit Rate Underflow is enabled for Bit Rate Control

Reported BitRateControl parameter Status

MFX_VP8_CNTRL_STATUS - Reported BitRateControl parameter Status

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	12904h
ShortName:	MFX_VP8_CNTRL_STATUS_VB0
Address:	1C904h
ShortName:	MFX_VP8_CNTRL_STATUS_VB1

This register stores the count of bytes of the bitstream output per frame

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**QindexClampHigh Status** Format: U1 This denotes if Qindex is clamped by QindexClampHigh value programmed in MFX_VP8_PIC_STATE.DW7.
	6	**QindexClampLow Status** Format: U1 This denotes if Qindex is clamped by QindexClampLow value programmed in MFX_VP8_PIC_STATE.DW7.
	5	**Final Bitstream Buffer Overrun Status** Format: U1 This denotes if Final bitstream buffer overrun.
	4	**Intermediate Bitstream Buffer Overrun Status** Format: U1 This denotes if any of the Intermediate bitstream buffer overrun. (including FrameHeader, Partition1 to Partition8)
	3	**Intra MB Bit Count Conformance Status** Format: U1 This is legacy support as AVC for Intra MB Bit Count conformance. It denotes if Intra MB Bit Count meets conformance size. This feature is not validated.
	2	**Inter MB Bit Count Conformance Status** Format: U1 This is legacy support as AVC for Inter MB Bit Count conformance. It denotes if Inter MB Bit Count meets conformance size. This feature is not validated.

MFX_VP8_CNTRL_STATUS - Reported BitRateControl parameter Status		
	1	**Frame Bit Rate Overflow Status** Format: U1 It denotes if Frame Bit Rate Overflow in current frame
	0	**Frame Bit Rate Underflow Status** Format: U1 It denotes if Frame Bit Rate Underflow in current frame

Reported Bitstream Output Bit Count for Syntax Elements Only

HCP_BITSTREAMSE_BITCOUNT_FRAME - Reported Bitstream Output Bit Count for Syntax Elements Only

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1E9A8h

This register stores the count of number of bits in the bitstream due to syntax elements only. This excludes header/ byte alignment /tail/EMU/CABAC-0word/padding bits but includes the stop-one-bit. This register is part of the context save and restore.

DWord	Bit	Description
0	31:0	**HCP Bitstream Syntax Element Only Bit Count** Format: U32 Total number of bits in the bitstream output due to syntax elements only. It includes the data bytes only. This count is updated for every time the internal bitstream counter is incremented and its reset at image start.

Reported Bitstream Output Bit Count for Syntax Elements Only Register

MFC_BITSTREAM_SE_BITCOUNT_FRAME - Reported Bitstream Output Bit Count for Syntax Elements Only Register	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	128A4h

This register stores the count of number of bits in the bitstream due to syntax elements only. This excludes header/ byte alignment /tail/EMU/CABAC-0word/padding bits but includes the stop-one-bit. This register is part of the context save and restore.

DWord	Bit	Description
0	31:0	**MFC Bitstream Syntax Element Only Bit Count** Total number of bits in the bitstream output due to syntax elements only. It includes the data bytes only. This count is updated for every time the internal bitstream counter is incremented and its reset at image start.

Reported Bitstream Output Byte Count per Frame Register

MFC_BITSTREAM_BYTECOUNT_FRAME - Reported Bitstream Output Byte Count per Frame Register		
Register Space:	MMIO: 0/2/0	
Access:	RO	
Size (in bits):	32	
Address:	128A0h	
This register stores the count of bytes of the bitstream output per frame		
DWord	**Bit**	**Description**
0	31:0	**MFC Bitstream Byte Count per Frame** Total number of bytes in the bitstream output per frame from the encoder. This includes header/tail/byte alignment/data bytes/EMU (emulation) bytes/cabac-zero word insertion/padding insertion. This count is updated for every time the internal bitstream counter is incremented and its reset at image start.

Reported Bitstream Output Byte Count per Tile

HCP_BITSTREAM_BYTECOUNT_TILE - Reported Bitstream Output Byte Count per Tile

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1E9CCh

This register stores the count of bytes of the bitstream output per tile.

DWord	Bit	Description
0	31:0	**HCP Bitstream Byte Count per Tile** Format: U32 Total number of bytes in the bitstream output per Tile from the encoder. This includes header/byte alignment/data bytes/EMU (emulation) bytes/. This count is updated for every time the internal bitstream counter is incremented and its reset at tile start.

Reported Bitstream Output CABAC Bin Count Register

MFC_AVC_CABAC_BIN_COUNT_FRAME - Reported Bitstream Output CABAC Bin Count Register		
Register Space:	MMIO: 0/2/0	
Access:	RO	
Size (in bits):	32	
Address:	128A8h	
This register stores the count of number of bins per frame.		
DWord	**Bit**	**Description**
0	31:0	**MFC AVC Cabac Bin Count** Total number of BINs in the bitstream output per frame from the encoder. This count is updated for every time the bin counter is incremented and its reset at image start.

Reported Final Bitstream Byte Count

MFX_VP8_FRM_BYTE_CNT - Reported Final Bitstream Byte Count		
Register Space:	MMIO: 0/2/0	
Access:	RO	
Size (in bits):	32	
Address:	12908h	
ShortName:	MFX_VP8_FRM_BYTE_CNT_01	
Address:	1C908h	
ShortName:	MFX_VP8_FRM_BYTE_CNT_02	
This register stores the count of bytes of the bitstream output per frame		
DWord	**Bit**	**Description**
0	31:0	**Final BitStream Byte Count** Format: U32 This register contains Final Bitstream byte count

Reported Frame Zero Padding Byte Count

MFX_VP8_FRM_ZERO_PAD - Reported Frame Zero Padding Byte Count

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1290Ch
ShortName:	MFX_VP8_FRM_ZERO_PAD_VB0
Address:	1C90Ch
ShortName:	MFX_VP8_FRM_ZERO_PAD_VB1

This register stores Frame Zero Padding Byte Count

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Frame Zero Padding Byte Count** Format: U16 This register contains Frame Zero Padding byte count This is legacy support. This feature is not validated.

Reported Timestamp Count

TIMESTAMP - Reported Timestamp Count	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02358h-0235Fh
Name:	Reported Timestamp Count
ShortName:	TIMESTAMP_RCSUNIT
Address:	18358h-1835Fh
Name:	Reported Timestamp Count
ShortName:	TIMESTAMP_POCSUNIT
Address:	22358h-2235Fh
Name:	Reported Timestamp Count
ShortName:	TIMESTAMP_BCSUNIT
Address:	1C0358h-1C035Fh
Name:	Reported Timestamp Count
ShortName:	TIMESTAMP_VCSUNIT0
Address:	1C4358h-1C435Fh
Name:	Reported Timestamp Count
ShortName:	TIMESTAMP_VCSUNIT1
Address:	1C8358h-1C835Fh
Name:	Reported Timestamp Count
ShortName:	TIMESTAMP_VECSUNIT0
Address:	1D0358h-1D035Fh
Name:	Reported Timestamp Count
ShortName:	TIMESTAMP_VCSUNIT2
Address:	1D4358h-1D435Fh
Name:	Reported Timestamp Count
ShortName:	TIMESTAMP_VCSUNIT3
Address:	1D8358h-1D835Fh
Name:	Reported Timestamp Count
ShortName:	TIMESTAMP_VECSUNIT1
Address:	1E0358h-1E035Fh
Name:	Reported Timestamp Count
ShortName:	TIMESTAMP_VCSUNIT4

TIMESTAMP - Reported Timestamp Count

Address:	1E4358h-1E435Fh
Name:	Reported Timestamp Count
ShortName:	TIMESTAMP_VCSUNIT5

Address:	1E8358h-1E835Fh
Name:	Reported Timestamp Count
ShortName:	TIMESTAMP_VECSUNIT2

Address:	1F0358h-1F035Fh
Name:	Reported Timestamp Count
ShortName:	TIMESTAMP_VCSUNIT6

Address:	1F4358h-1F435Fh
Name:	Reported Timestamp Count
ShortName:	TIMESTAMP_VCSUNIT7

Address:	1F8358h-1F835Fh
Name:	Reported Timestamp Count
ShortName:	TIMESTAMP_VECSUNIT3

Address:	1A358h-1A35Fh
Name:	Reported Timestamp Count
ShortName:	TIMESTAMP_CCSUNIT0

This register provides an elapsed real-time value that can be used as a timestamp for GPU events over short periods of time. Note that the value of this register can be obtained in a 3D pipeline-synchronous fashion without a pipeline flush by using the PIPE_CONTROL command. See 3D Geometry Pipeline in the "3D and Media" volume. This register (effectively) counts at a constant frequency by adjusting the increment amount according to the actual reference clock frequency. SW therefore does not need to know the reference clock frequency.

This register provides an elapsed real-time value that can be used as a timestamp. The accumulated value in this register is of the timestamp stamp granularity (base unit) defined in the Time Stamp Bases subsection in Power Management chapter.

This register is *not* reset by a graphics reset. It will maintain its value unless a full chipset reset is performed.

DWord	Bit	Description
0..1	63:36	**Reserved** Access: RO Format: MBZ
	35:32	**Timestamp Value UN** Format: U4 This register increments for every timestamp base unit. The granularity of the time stamp base unit is defined in the Time Stamp Bases subsection in Power Management chapter. Note: This is the Upper Nibble of the Timesamp Value, a 36-bit signal.

TIMESTAMP - Reported Timestamp Count		
	31:0	**Timestamp Value LDW** Format: U32 This register increments for every timestamp base unit. The granularity of the time stamp base unit is defined in the Time Stamp Bases subsection in Power Management chapter.

Report Queue CFG HI

RPTQCFGHI - Report Queue CFG HI		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	098ACh	
Config to MCI and DFT Ring		
DWord	**Bit**	**Description**
0	31	**Report fifo cfg hi valid** Access: RO _Custom_GTIReset: BUS
	30:18	**RSVD_30_18** Access: RO _Custom_GTIReset: BUS
	17:16	**Report fifo cfg hi bits 9_8** Access: RO _Custom_GTIReset: BUS
	15:8	**RSVD_15_8** Access: RO _Custom_GTIReset: BUS
	7:0	**Report fifo cfg hi bits 0_7** Access: RO _Custom_GTIReset: BUS

Report Queue CFG LO

RPTQCFGLO - Report Queue CFG LO		
Register Space: MMIO: 0/2/0 Size (in bits): 32		
Address: 098A8h		
Config to MCI and DFT Ring		
DWord	**Bit**	**Description**
0	31:0	**Report fifo cfg low** Access: RO _Custom_GTIReset: BUS

Reset Control Register

RESET_CTRL - Reset Control Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	020D0h-020D3h
Name:	Reset Control Register
ShortName:	RESET_CTRL_RCSUNIT
Address:	180D0h-180D3h
Name:	Reset Control Register
ShortName:	RESET_CTRL_POCSUNIT
Address:	220D0h-220D3h
Name:	Reset Control Register
ShortName:	RESET_CTRL_BCSUNIT
Address:	1C00D0h-1C00D3h
Name:	Reset Control Register
ShortName:	RESET_CTRL_VCSUNIT0
Address:	1C40D0h-1C40D3h
Name:	Reset Control Register
ShortName:	RESET_CTRL_VCSUNIT1
Address:	1C80D0h-1C80D3h
Name:	Reset Control Register
ShortName:	RESET_CTRL_VECSUNIT0
Address:	1D00D0h-1D00D3h
Name:	Reset Control Register
ShortName:	RESET_CTRL_VCSUNIT2
Address:	1D40D0h-1D40D3h
Name:	Reset Control Register
ShortName:	RESET_CTRL_VCSUNIT3
Address:	1D80D0h-1D80D3h
Name:	Reset Control Register
ShortName:	RESET_CTRL_VECSUNIT1
Address:	1E00D0h-1E00D3h
Name:	Reset Control Register
ShortName:	RESET_CTRL_VCSUNIT4

RESET_CTRL - Reset Control Register

Address:	1E40D0h-1E40D3h
Name:	Reset Control Register
ShortName:	RESET_CTRL_VCSUNIT5

Address:	1E80D0h-1E80D3h
Name:	Reset Control Register
ShortName:	RESET_CTRL_VECSUNIT2

Address:	1F00D0h-1F00D3h
Name:	Reset Control Register
ShortName:	RESET_CTRL_VCSUNIT6

Address:	1F40D0h-1F40D3h
Name:	Reset Control Register
ShortName:	RESET_CTRL_VCSUNIT7

Address:	1F80D0h-1F80D3h
Name:	Reset Control Register
ShortName:	RESET_CTRL_VECSUNIT3

Address:	1A0D0h-1A0D3h
Name:	Reset Control Register
ShortName:	RESET_CTRL_CCSUNIT0

Soft reset flow for an engine (Render, Blitter, Video, Video Enhancement) is asynchronous to the context execution in HW. SW needs a deterministic way to ensure it resets the context it intends to. One way to achieve this is to ensure HW doesn't switch out the context while SW is doing a soft reset. This is achieved by having an explicit interface between HW-SW to prepare the engine prior to the soft reset. SW sets the Request Reset in RESET_CTRL register of an engine indicating SW wants to initiate a soft reset flow for the corresponding engine. In response to Request Reset bit set, HW sets Ready for Reset bit of RESET_CTRL register indicating engine readiness for reset. As part of the reset readiness HW will not allow any context switch to take place and also ensure any ongoing context switch is paused on a clean context boundary (context save in progress is completed, Context Switch Status Buffer updates are allowed to complete).
SW polls for Ready for Reset bit to be set before it does soft reset for the corresponding engine. Reading EXECLIST_STATUS register at this point provides the active context in HW that will get reset. On engine reset Request Reset bit will get reset with rest of the engine logic.
Upon polling EXECLIST_STATUS register for active context SW might decide not to reset the engine and can reset the Request Reset in RESET_CTRL register. On Request Reset getting reset by SW, HW must continue with execution.
SW setting Ready for Reset bit in RESET_CTRL register of an engine need not be followed by the corresponding engine reset.
SW writing to Request Reset bit in RESET_CTRL register is preparing the engine for reset whereas SW writing to GDRST triggers the actual reset flow in HW.

Programming Notes
SW must not do Reset Readiness Handshake as part of the reset recovery on a CAT error.

DWord	Bit	Description

RESET_CTRL - Reset Control Register

0	31:16	**Mask** Access: WO Format: Mask Must be set to modify corresponding bit in Bits 15:0. (All implemented bits)
	15:4	**Reserved** Access: RO Format: MBZ
	3	**Reserved** Format: PBC
	2	**Reserved**
	1	**Ready for Reset** Format: U1 When set indicates render engine is ready for reset. This bit gets cleared on engine reset or when Soft Reset In progress is cleared.
	0	**Request Reset** Format: U1 "Request Reset" bit must be read as "Readyness for Reset". When set indicates SW wishes to reset the render engine. On seeing this bit set Command Streamer will take appropriate action and set Ready For Reset status bit. This bit gets cleared on engine reset. This bit can also be cleared by writing "0" to this bit.

Reset Flow Control Messages 0

RSTFCTLMSG0 - Reset Flow Control Messages 0

Register Space:	MMIO: 0/2/0
Size (in bits):	32
CrashLogSaved:	true
CrashLogVisibility:	cspec
Address:	08108h

Soft-Reset and FLR Flow Control Message Registers

DWord	Bit	Description
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask In order to write to bits 15:0, the corresponding message mask bits must be written. For example, for bit 14 to be set, bit 30 needs to be 1 : 40004000
	15:3	**Reserved** Access: RO Format: MBZ
	2	**FLR Done ack from Pmunit** Access: R/W Set _Custom_GTIReset: BUS FLR Done ack from Pmunit: 1: PM unit sets this bit to acknowledge the FLR done message has been forwarded to SA through GAM interface. 0: Default Value. If the bit was set by PM then Cpunit hardware clears it once FLR is completed.
	1	**Global Resource Arbitration Acknowledgement Messages** Access: R/W _Custom_GTIReset: BUS Global Resource Arbitration Acknowledgement Message from PM: '1' : CP_ARB_REQ_ACK - Acknowledgement for CPunit's global resource arbitration request '0' : CP_ARB_RELEASE_ACK - Acknowledgement to CPunit's release of global resources
	0	**CP Busy / Idle Status Acknowledgement Messages** Access: R/W _Custom_GTIReset: BUS CP Busy / Idle Status Acknowledgement Message from PM: '0' : CP_NOT_BUSY_ACK - Acknowledgement that the CPunit is idle. '1' : CP_BUSY_ACK - Acknowledgement that the CPunit is busy.

Revision Identification and Class Code register

RID2_CC_0_2_0_PCI - Revision Identification and Class Code register

Register Space:	PCI: 0/2/0
Size (in bits):	32
Address:	00008h

This register contains the revision number for Device #2 Functions 0 and contains the device programming interface information related to the Sub-Class Code and Base Class Code definition for the IGD. This register also contains the Base Class Code and the function sub-class in relation to the Base Class Code.

DWord	Bit	Description
0	31:24	**Base Class Code** Default Value: 00000011b Access: RO Variant _Custom_GTIReset: BUS This is an 8-bit value that indicates the base class code. When MGGC0[VAMEN] is 0 this code has the value 03h, indicating a Display Controller. When MGGC0[VAMEN] is 1 this code has the value 03h, indicating a Display Controller Device.
	23:16	**Sub-Class Code** Default Value: 00000000b Access: RO Variant _Custom_GTIReset: BUS When MGGC0[VAMEN] is 0, this value is 00h. When MGGC0[VAMEN] is 1, this value is 80h, indicating other display device.
	15:8	**Programming Interface** Default Value: 00000000b Access: RO _Custom_GTIReset: BUS When MGGC0[VAMEN] is 0 this value is 00h, indicating a Display Controller. When MGGC0[VAMEN] is 1 this value is 00h, indicating a NOP.
	7:0	**Revision Identification Number** Default Value: 00h Access: R/W Variant _Custom_GTIReset: BUS All 8 bits of Revision ID is acquired through fuse pull as per Chassis 2.1 updates

RING_BUFFER_HEAD_PREEMPT_REG

RING_BUFFER_HEAD_PREEMPT_REG - RING_BUFFER_HEAD_PREEMPT_REG	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0214Ch-0214Fh
Name:	RING_BUFFER_HEAD_PREEMPT_REG
ShortName:	RING_BUFFER_HEAD_PREEMPT_REG_RCSUNIT
Address:	1814Ch-1814Fh
Name:	RING_BUFFER_HEAD_PREEMPT_REG
ShortName:	RING_BUFFER_HEAD_PREEMPT_REG_POCSUNIT
Address:	2214Ch-2214Fh
Name:	RING_BUFFER_HEAD_PREEMPT_REG
ShortName:	RING_BUFFER_HEAD_PREEMPT_REG_BCSUNIT
Address:	1C014Ch-1C014Fh
Name:	RING_BUFFER_HEAD_PREEMPT_REG
ShortName:	RING_BUFFER_HEAD_PREEMPT_REG_VCSUNIT0
Address:	1C414Ch-1C414Fh
Name:	RING_BUFFER_HEAD_PREEMPT_REG
ShortName:	RING_BUFFER_HEAD_PREEMPT_REG_VCSUNIT1
Address:	1C814Ch-1C814Fh
Name:	RING_BUFFER_HEAD_PREEMPT_REG
ShortName:	RING_BUFFER_HEAD_PREEMPT_REG_VECSUNIT0
Address:	1D014Ch-1D014Fh
Name:	RING_BUFFER_HEAD_PREEMPT_REG
ShortName:	RING_BUFFER_HEAD_PREEMPT_REG_VCSUNIT2
Address:	1D414Ch-1D414Fh
Name:	RING_BUFFER_HEAD_PREEMPT_REG
ShortName:	RING_BUFFER_HEAD_PREEMPT_REG_VCSUNIT3
Address:	1D814Ch-1D814Fh
Name:	RING_BUFFER_HEAD_PREEMPT_REG
ShortName:	RING_BUFFER_HEAD_PREEMPT_REG_VECSUNIT1
Address:	1E014Ch-1E014Fh

RING_BUFFER_HEAD_PREEMPT_REG - RING_BUFFER_HEAD_PREEMPT_REG

Name:	RING_BUFFER_HEAD_PREEMPT_REG
ShortName:	RING_BUFFER_HEAD_PREEMPT_REG_VCSUNIT4
Address:	1E414Ch-1E414Fh
Name:	RING_BUFFER_HEAD_PREEMPT_REG
ShortName:	RING_BUFFER_HEAD_PREEMPT_REG_VCSUNIT5
Address:	1E814Ch-1E814Fh
Name:	RING_BUFFER_HEAD_PREEMPT_REG
ShortName:	RING_BUFFER_HEAD_PREEMPT_REG_VECSUNIT2
Address:	1F014Ch-1F014Fh
Name:	RING_BUFFER_HEAD_PREEMPT_REG
ShortName:	RING_BUFFER_HEAD_PREEMPT_REG_VCSUNIT6
Address:	1F414Ch-1F414Fh
Name:	RING_BUFFER_HEAD_PREEMPT_REG
ShortName:	RING_BUFFER_HEAD_PREEMPT_REG_VCSUNIT7
Address:	1F814Ch-1F814Fh
Name:	RING_BUFFER_HEAD_PREEMPT_REG
ShortName:	RING_BUFFER_HEAD_PREEMPT_REG_VECSUNIT3
Address:	1A14Ch-1A14Fh
Name:	RING_BUFFER_HEAD_PREEMPT_REG
ShortName:	RING_BUFFER_HEAD_PREEMPT_REG_CCSUNIT0

This register contains the Head pointer offset in the ring when the last PREEMPTABLE command was executed and caused the head pointer to move due to the UHPTR register being valid. If the PREEMPTABLE command is executed as part of the batch buffer then the value of the register will be the offset in the ring of the command past the batch buffer start that contained the preemptable command.
This is a global register and context save/restored as part of power context image.

Preemptable Commands	Source
• MI_ARB_CHECK • 3D_PRIMITIVE • GPGPU_WALKER • MEDIA_STATE_FLUSH • PIPE_CONTROL (Only in GPGPU mode of pipeline selection) • MI_ATOMIC (Post Sync Operation set in GPGPU mode of pipeline selection) • MI_SEMAPHORE_SIGNAL (Post Sync Operation set in GPGPU mode of pipeline selection)	RenderCS

Preemptable Commands	Source

RING_BUFFER_HEAD_PREEMPT_REG - RING_BUFFER_HEAD_PREEMPT_REG

MI_ARB_CHECK	BlitterCS, VideoCS, VideoCS2, VideoEnhancementCS

Programming Notes

Programming Restriction:
This register should NEVER be programmed by driver. This is for HW internal use only.

DWord	Bit	Description
0	31:21	**Last Wrap Count**
	20:2	**Preempted Head Offset** Format: U19 This field contains the Head pointer offset in the ring when the last MI_ARB_CHECK command was executed and caused the head pointer to move due to the UHPTR register being valid.
	1:0	**Ring/Batch Indicator** Value / Name / Description 0h / Ring / Preemptable command was executed in ring and caused head pointer to be updated. 1h / Batch / Preemptable command was executed in batch and caused head pointer to be updated. 2h / 2nd level batch / Preemptable command was executed in second level batch and caused head pointer to be updated.

Ring Buffer Control

RING_BUFFER_CTL - Ring Buffer Control	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0203Ch-0203Fh
Name:	Ring Buffer Control
ShortName:	RING_BUFFER_CTL_RCSUNIT
Address:	1803Ch-1803Fh
Name:	Ring Buffer Control
ShortName:	RING_BUFFER_CTL_POCSUNIT
Address:	2203Ch-2203Fh
Name:	Ring Buffer Control
ShortName:	RING_BUFFER_CTL_BCSUNIT
Address:	1C003Ch-1C003Fh
Name:	Ring Buffer Control
ShortName:	RING_BUFFER_CTL_VCSUNIT0
Address:	1C403Ch-1C403Fh
Name:	Ring Buffer Control
ShortName:	RING_BUFFER_CTL_VCSUNIT1
Address:	1C803Ch-1C803Fh
Name:	Ring Buffer Control
ShortName:	RING_BUFFER_CTL_VECSUNIT0
Address:	1D003Ch-1D003Fh
Name:	Ring Buffer Control
ShortName:	RING_BUFFER_CTL_VCSUNIT2
Address:	1D403Ch-1D403Fh
Name:	Ring Buffer Control
ShortName:	RING_BUFFER_CTL_VCSUNIT3
Address:	1D803Ch-1D803Fh
Name:	Ring Buffer Control
ShortName:	RING_BUFFER_CTL_VECSUNIT1
Address:	1E003Ch-1E003Fh
Name:	Ring Buffer Control
ShortName:	RING_BUFFER_CTL_VCSUNIT4

RING_BUFFER_CTL - Ring Buffer Control

Address:	1E403Ch-1E403Fh
Name:	Ring Buffer Control
ShortName:	RING_BUFFER_CTL_VCSUNIT5

Address:	1E803Ch-1E803Fh
Name:	Ring Buffer Control
ShortName:	RING_BUFFER_CTL_VECSUNIT2

Address:	1F003Ch-1F003Fh
Name:	Ring Buffer Control
ShortName:	RING_BUFFER_CTL_VCSUNIT6

Address:	1F403Ch-1F403Fh
Name:	Ring Buffer Control
ShortName:	RING_BUFFER_CTL_VCSUNIT7

Address:	1F803Ch-1F803Fh
Name:	Ring Buffer Control
ShortName:	RING_BUFFER_CTL_VECSUNIT3

Address:	1A03Ch-1A03Fh
Name:	Ring Buffer Control
ShortName:	RING_BUFFER_CTL_CCSUNIT0

These registers are used to define and operate the "ring buffer" mechanism which can be used to pass instructions to the command interface. The buffer itself is located in a linear memory region. The ring buffer is defined by a 4 Dword register set that includes starting address, length, head offset, tail offset, and control information. These registers can only be updated through a restore of a context thru execution list submission.

DWord	Bit	Description
0	31:21	**Reserved** Access: RO Format: MBZ
	20:12	**Buffer Length** Format: U9-1 This field is written by SW to specify the length of the ring buffer in 4 KB Pages. Range = [0 = 1 page = 4 KB, 1FFh = 512 pages = 2 MB] Value / Name / Description: 0 — 1 page = 4 KB 1FFh — 512 pages = 2 MB
	11	**RBWait** Indicates that this ring has executed a WAIT_FOR_EVENT instruction and is currently waiting. Software can write a "1" to clear this bit, write of "0" has no effect. When the RB is waiting for an event and this bit is cleared, the wait will be terminated and the RB will be returned to arbitration.

RING_BUFFER_CTL - Ring Buffer Control

10	**Semaphore Wait** Indicates that this ring has executed a MI_SEMAPHORE_WAIT instruction and is currently waiting for wait condition to satisfy. Software can write a "1" to clear this bit, write of "0" has no effect. **Programming Notes** Writing a value of 1 will unconditionally cancel the semaphore wait on the next memory comparison. Memory comparison is triggered in signal mode on receiving a semaphore signal and in poll mode on wait timer getting expired.
9:3	**Reserved** Access: RO Format: MBZ
2:1	**Automatic Report Head Pointer** This field is written by software to control the automatic reporting (write) of this ring buffer's Head Pointer register (register DWord 1) to the corresponding location within the Hardware Status Page. Automatic reporting can either be disabled or enabled at 4KB, 64KB or 128KB boundaries within the ring buffer. When **Execlist Enable** bit is set the head pointer will be reported to the head pointer location in the Per-Process Hardware Status Page.

Value	Name	Description
0	MI_AUTOREPORT_OFF	Automatic reporting disabled
1	MI_AUTOREPORT_64KB	Report every 16 pages (64KB)
2	MI_AUTOREPORT_4KB	Report every page (4KB) This mode must not be enabled in Ring Buffer mode of scheduling to minimize the auto reports.
3	MI_AUTO_REPORT_128KB	Report every 32 pages (128KB).

0	**Ring Buffer Enable** Format: Enable This field is used to enable or disable this ring buffer. It can be enabled or disabled regardless of whether there are valid instructions pending. If disabled and the ring head equals ring tail, all state currently loaded in hardware is considered invalid. **Programming Notes** Ring Buffer Mode of Scheduling: SW must follow the below programming notes during SW initialization phase or while enabling render engine's ring buffer for the first time, this would be coming out of boot, standby, hibernate or reset. This flow must be also followed during ring replay when ring buffer is disabled and enabled. • SW must set the Force Wakeup bit to prevent GT from entering C6. • SW must dispatch workload (dummy context) to initialize render engine with default state such that any context switches that occur subsequently (Power Save) will save and restore coherent device state. Indirect pointers used in 3D states must point to valid graphics surface existing in memory. PP_DCLV followed by PP_DIR_BASE register should

		be programmed as part of initialization workload if PPGTT is enabled in GFX_MODE register. • SW must ensure all the register (MMIO) initialization/programming through CPU happens in this block or latter, this ensures the MMIO state is save/restored on subsequent context switches (Power Sequences). • Once the render engine is programmed with valid state and the configuration, Force Wakeup bit should be reset to enable C6 entry.

Ring Buffer Head

RING_BUFFER_HEAD - Ring Buffer Head	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02034h-02037h
Name:	Ring Buffer Head
ShortName:	RING_BUFFER_HEAD_RCSUNIT
Address:	18034h-18037h
Name:	Ring Buffer Head
ShortName:	RING_BUFFER_HEAD_POCSUNIT
Address:	22034h-22037h
Name:	Ring Buffer Head
ShortName:	RING_BUFFER_HEAD_BCSUNIT
Address:	1C0034h-1C0037h
Name:	Ring Buffer Head
ShortName:	RING_BUFFER_HEAD_VCSUNIT0
Address:	1C4034h-1C4037h
Name:	Ring Buffer Head
ShortName:	RING_BUFFER_HEAD_VCSUNIT1
Address:	1C8034h-1C8037h
Name:	Ring Buffer Head
ShortName:	RING_BUFFER_HEAD_VECSUNIT0
Address:	1D0034h-1D0037h
Name:	Ring Buffer Head
ShortName:	RING_BUFFER_HEAD_VCSUNIT2
Address:	1D4034h-1D4037h
Name:	Ring Buffer Head
ShortName:	RING_BUFFER_HEAD_VCSUNIT3
Address:	1D8034h-1D8037h
Name:	Ring Buffer Head
ShortName:	RING_BUFFER_HEAD_VECSUNIT1
Address:	1E0034h-1E0037h
Name:	Ring Buffer Head
ShortName:	RING_BUFFER_HEAD_VCSUNIT4

RING_BUFFER_HEAD - Ring Buffer Head

Address:	1E4034h-1E4037h
Name:	Ring Buffer Head
ShortName:	RING_BUFFER_HEAD_VCSUNIT5

Address:	1E8034h-1E8037h
Name:	Ring Buffer Head
ShortName:	RING_BUFFER_HEAD_VECSUNIT2

Address:	1F0034h-1F0037h
Name:	Ring Buffer Head
ShortName:	RING_BUFFER_HEAD_VCSUNIT6

Address:	1F4034h-1F4037h
Name:	Ring Buffer Head
ShortName:	RING_BUFFER_HEAD_VCSUNIT7

Address:	1F8034h-1F8037h
Name:	Ring Buffer Head
ShortName:	RING_BUFFER_HEAD_VECSUNIT3

Address:	1A034h-1A037h
Name:	Ring Buffer Head
ShortName:	RING_BUFFER_HEAD_CCSUNIT0

These registers are used to define and operate the "ring buffer" mechanism which can be used to pass instructions to the command interface. The buffer itself is located in a linear memory region. The ring buffer is defined by a 4 Dword register set that includes starting address, length, head offset, tail offset, and control information. These registers can only be updated through a restore of a context thru execution list submission.

DWord	Bit	Description
0	31:21	**Wrap Count** Format: U11 This field is incremented by 1 whenever the **Head Offset** wraps from the end of the buffer back to the start (i.e., whenever it wraps back to 0). Appending this field to the **Head Offset** field effectively creates a virtual 4GB Head "Pointer" which can be used as a tag associated with instructions placed in a ring buffer. The Wrap Count itself will wrap to 0 upon overflow.
	20:2	**Head Offset** Format: GraphicsAddress[20:2] This field indicates the offset of the *next* instruction DWord to be parsed. Software will initialize this field to select the first DWord to be parsed once the RB is enabled. (Writing the Head Offset while the RB is enabled is UNDEFINED). Subsequently, the device will increment this offset as it executes instructions - until it reaches the QWord specified by the **Tail Offset**. At this point the ring buffer is considered "empty". **Programming Notes** An RB can be enabled empty or containing some number of valid instructions.

RING_BUFFER_HEAD - Ring Buffer Head		
	1:0	**Reserved** Access: RO Format: MBZ

Ring Buffer Start

RING_BUFFER_START - Ring Buffer Start	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02038h-0203Bh
Name:	Ring Buffer Start
ShortName:	RING_BUFFER_START_RCSUNIT
Address:	18038h-1803Bh
Name:	Ring Buffer Start
ShortName:	RING_BUFFER_START_POCSUNIT
Address:	22038h-2203Bh
Name:	Ring Buffer Start
ShortName:	RING_BUFFER_START_BCSUNIT
Address:	1C0038h-1C003Bh
Name:	Ring Buffer Start
ShortName:	RING_BUFFER_START_VCSUNIT0
Address:	1C4038h-1C403Bh
Name:	Ring Buffer Start
ShortName:	RING_BUFFER_START_VCSUNIT1
Address:	1C8038h-1C803Bh
Name:	Ring Buffer Start
ShortName:	RING_BUFFER_START_VECSUNIT0
Address:	1D0038h-1D003Bh
Name:	Ring Buffer Start
ShortName:	RING_BUFFER_START_VCSUNIT2
Address:	1D4038h-1D403Bh
Name:	Ring Buffer Start
ShortName:	RING_BUFFER_START_VCSUNIT3
Address:	1D8038h-1D803Bh
Name:	Ring Buffer Start
ShortName:	RING_BUFFER_START_VECSUNIT1
Address:	1E0038h-1E003Bh
Name:	Ring Buffer Start
ShortName:	RING_BUFFER_START_VCSUNIT4

RING_BUFFER_START - Ring Buffer Start

Address:	1E4038h-1E403Bh
Name:	Ring Buffer Start
ShortName:	RING_BUFFER_START_VCSUNIT5

Address:	1E8038h-1E803Bh
Name:	Ring Buffer Start
ShortName:	RING_BUFFER_START_VECSUNIT2

Address:	1F0038h-1F003Bh
Name:	Ring Buffer Start
ShortName:	RING_BUFFER_START_VCSUNIT6

Address:	1F4038h-1F403Bh
Name:	Ring Buffer Start
ShortName:	RING_BUFFER_START_VCSUNIT7

Address:	1F8038h-1F803Bh
Name:	Ring Buffer Start
ShortName:	RING_BUFFER_START_VECSUNIT3

Address:	1A038h-1A03Bh
Name:	Ring Buffer Start
ShortName:	RING_BUFFER_START_CCSUNIT0

These registers are used to define and operate the "ring buffer" mechanism which can be used to pass instructions to the command interface. The buffer itself is located in a linear memory region. The ring buffer is defined by a 4 Dword register set that includes starting address, length, head offset, tail offset, and control information. These registers can only be updated through a restore of a context thru execution list submission.

DWord	Bit	Description
0	31:12	**Starting Address** Format: GraphicsAddress[31:12] This field specifies Bits 31:12 of the 4KB-aligned starting Graphics Address of the ring buffer. Address bits 31 down to 29 must be zero. All ring buffer pages must map to Main Memory (uncached) pages. Ring Buffer addresses are always translated through the global GTT.
	11:0	**Reserved** Access: RO Format: MBZ

Ring Buffer Tail

RING_BUFFER_TAIL - Ring Buffer Tail	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02030h-02033h
Name:	Ring Buffer Tail
ShortName:	RING_BUFFER_TAIL_RCSUNIT
Address:	18030h-18033h
Name:	Ring Buffer Tail
ShortName:	RING_BUFFER_TAIL_POCSUNIT
Address:	22030h-22033h
Name:	Ring Buffer Tail
ShortName:	RING_BUFFER_TAIL_BCSUNIT
Address:	1C0030h-1C0033h
Name:	Ring Buffer Tail
ShortName:	RING_BUFFER_TAIL_VCSUNIT0
Address:	1C4030h-1C4033h
Name:	Ring Buffer Tail
ShortName:	RING_BUFFER_TAIL_VCSUNIT1
Address:	1C8030h-1C8033h
Name:	Ring Buffer Tail
ShortName:	RING_BUFFER_TAIL_VECSUNIT0
Address:	1D0030h-1D0033h
Name:	Ring Buffer Tail
ShortName:	RING_BUFFER_TAIL_VCSUNIT2
Address:	1D4030h-1D4033h
Name:	Ring Buffer Tail
ShortName:	RING_BUFFER_TAIL_VCSUNIT3
Address:	1D8030h-1D8033h
Name:	Ring Buffer Tail
ShortName:	RING_BUFFER_TAIL_VECSUNIT1
Address:	1E0030h-1E0033h
Name:	Ring Buffer Tail
ShortName:	RING_BUFFER_TAIL_VCSUNIT4

RING_BUFFER_TAIL - Ring Buffer Tail

Address:	1E4030h-1E4033h
Name:	Ring Buffer Tail
ShortName:	RING_BUFFER_TAIL_VCSUNIT5

Address:	1E8030h-1E8033h
Name:	Ring Buffer Tail
ShortName:	RING_BUFFER_TAIL_VECSUNIT2

Address:	1F0030h-1F0033h
Name:	Ring Buffer Tail
ShortName:	RING_BUFFER_TAIL_VCSUNIT6

Address:	1F4030h-1F4033h
Name:	Ring Buffer Tail
ShortName:	RING_BUFFER_TAIL_VCSUNIT7

Address:	1F8030h-1F8033h
Name:	Ring Buffer Tail
ShortName:	RING_BUFFER_TAIL_VECSUNIT3

Address:	1A030h-1A033h
Name:	Ring Buffer Tail
ShortName:	RING_BUFFER_TAIL_CCSUNIT0

These registers are used to define and operate the "ring buffer" mechanism which can be used to pass instructions to the command interface. The buffer itself is located in a linear memory region. The ring buffer is defined by a 4 Dword register set that includes starting address, length, head offset, tail offset, and control information. These registers can only be updated through a restore of a context thru execution list submission.

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30:21	**Reserved** Access: RO Format: MBZ
	20:3	**Tail Offset** Format: GraphicsAddress[20:3] This field is written by software to specify where the valid instructions placed in the ring buffer end. The value written points to the QWord past the last valid QWord of instructions. In other words, it can be defined as the next QWord that software will write instructions into. Software must write subsequent instructions to QWords following the Tail Offset, possibly wrapping around to the top of the buffer (i.e., software can't skip around within the buffer).Note that all DWords prior to the location indicated by the **Tail Offset** must contain valid instruction data - which may require instruction padding by software. See **Head Offset** for more information.

RING_BUFFER_TAIL - Ring Buffer Tail		
	2:0	**Reserved** Access: RO Format: MBZ

Ring Context Address Register 0 for GTI Doorbell Unit

DRBCTXADDR0 - Ring Context Address Register 0 for GTI Doorbell Unit

Register Space:	MMIO: 0/2/0
Size (in bits):	64
Address:	01950h

DWord	Bit	Description
0..1	63:32	**CTX base address 63:32** Access: R/W _Custom_GTIReset: BUS Bits [63:32] of HPA base address which MDRB will use as the offset for its context save / restore
	31:6	**CTX base address 31:6** Access: R/W _Custom_GTIReset: BUS Bits [31:6] of HPA base address which MDRB will use as the offset for its context save / restore
	5:0	**Reserved** Access: RO Format: MBZ

Ring Context Start Register for GTI Doorbell Unit

DRBCTXSTART - Ring Context Start Register for GTI Doorbell Unit

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	0194Ch

Message registers have bit-wise masking applied for writes. The register consists of 16 bits of data in [15:0], and 16 bits of corresponding masks in [31:16].
In order to set bit0, for example, the data would be 0x0001_0001. In order to clear bit0, for example, the data would be 0x0001_0000.
Note that mask bit is the data bit offset + 16. Message registers are protected from non-GT writes via the Message Channel.

DWord	Bit	Description
0	31:16	**Context Save Mask** Access: WO _Custom_GTIReset: BUS This registers isn't context save/restored.
	15:2	**Reserved** Access: RO Format: MBZ
	1:0	**Context start** Access: R/W Hardware Clear _Custom_GTIReset: BUS RPM will program these bits to notify MDRB to initiate either a context save or a context restore. MDRB will clear these bits once the context procedure is completed and the ACK is sent back to RPM. 2'b11 = Reserved 2'b10 = Context restore start 2'b01 = Context save start 2'b00 = Normal operation - DEFAULT Programming Note - It is assumed that this value will be static once programmed and not changed until MDRB clears the bits after ACKing the context procedure back to RPM

RMTIMEOUTREG_CAPTURE

RMTIMEOUTREG_CAPTURE		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	420E0h-420E3h	
Name:	RM TIMEOUT REGISTER OFFSET	
ShortName:	RMTIMEOUTREG_CAPTURE	
Reset:	soft	

DWord	Bit	Description
0	31:0	**REGISTER OFFSET VALUE** Access: RO Offset of the Register that caused RM TIMEOUT.

RSA for uOS/Soft Scratch

UOS_RSA_SCRATCH - RSA for uOS/Soft Scratch

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32

Description

There are 64 Dword registers (2048 bits) for RSA check for uOS code, which is performed by the on chip Boot ROM code. This RSA check is done to ensure that the full HASH check value used for Hash operation of the uOS by Shim, itself is correct. These registers can also be used as Soft Scratch registers defined for use after the RSA operation at Boot. Note that these registers contents are not context saved/restored for C6 purposes. Usage: The Host programs these 64 registers, before programming the DMA. The Boot ROM then on being initiated reads these 64 registers and starts the RSA unwrap.

DWord	Bit	Description
0	31:0	**RSA/Scratch**

RTADDR_LSB

RTADDR_LSB - RTADDR_LSB

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	124830h

Register providing the base address of root-entry table.

DWord	Bit	Description
0	31:12	**RTA** Default Value: 0000000h Access: R/W _Custom_GTIReset: BUS This register points to base of page aligned, 4KB-sized root-entry table in system memory. Hardware ignores and not implements bits 63:HAW, where HAW is the host address width. Software specifies the base address of the root-entry table through this register, and programs it in hardware through the SRTP field in the Global Command register. Reads of this register returns value that was last programmed to it.
	11	**RTT** Default Value: 0h Access: R/W _Custom_GTIReset: BUS This field specifies the type of root-table referenced by the Root Table Address (RTA) field; 0: Root Table1: Extended Root Table
	10:0	**Reserved** Access: RO Format: MBZ

RTADDR_MSB

RTADDR_MSB - RTADDR_MSB

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	124834h

Register providing the base address of root-entry table.

DWord	Bit	Description
0	31:14	**Reserved** Access: RO Format: MBZ
	13:7	**Reserved** Access: RO Format: MBZ
	6:0	**RTA** Default Value: 00h Access: R/W _Custom_GTIReset: BUS This register points to base of page aligned, 4KB-sized root-entry table in system memory. Hardware ignores and not implements bits 63:HAW, where HAW is the host address width. Software specifies the base address of the root-entry table through this register, and programs it in hardware through the SRTP field in the Global Command register. Reads of this register returns value that was last programmed to it.

RTT_CR_SPARE

RTT_CR_SPARE		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	168B90h-168B93h	
Name:	RTT_CR_SPARE	
ShortName:	RTT_CR_SPARE	
Reset:	global	
Spare registers for RTT Lane		
DWord	**Bit**	**Description**
0	31:0	**cfg_rtt_cr_spare** Default Value: 0041000h cfg_rtt_cr_spare_defaultreset Access: R/W PCS_Glue::RTT_CR_SPARE::rtt_cr_spare

Sampler control register

SAMPLER_CTL - Sampler control register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0E140h

DWord	Bit	Description
0	31:16	**ECO Reserved 1** Reserved: MBZ
	15:8	**Reserved**
	7	**ECO Reseved 7**
	6	**ECO Reseved 6**
	5	**ECO Reseved 5**
	4	**ECO Reseved 4**
	3	**ECO Reseved 3**
	2	**ECO Reseved 2**
	1	**ECO Reseved 1**
	0	**ECO Reseved 0**

Sampler Dummy Register

SMP_DUMMY - Sampler Dummy Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0E000h
Name:	Sampler Dummy Address
ShortName:	Sampler_Dummy_Address

This register is defined so that a non-posted MMIO cycle to this destination would ensure all cycles are flushed on the message channel between the source and destination. This register is used in the engine context to ensure all state is delivered. The value programmed in this register must not change the behavior of the GPU.

DWord	Bit	Description
0	31:0	**Reserved** Access: RO Format: MBZ

SAMPLER Mode Register

SAMPLER_MODE - SAMPLER Mode Register

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV

Address:	0E18Ch
Name:	SAMPLER Mode Register
ShortName:	SAMPLER_MODE

This register has bit-wise masking applied for writes. The register consists of 16 bits of data in [15:0], and 16 bits of corresponding masks in [31:16]. To set bit0, for example, the data would be 0x0001_0001. To clear bit0, for example, the data would be 0x0001_0000. Note that mask bit is the data bit offset + 16.

DWord	Bit	Description
0	31:16	**Mask** Access: WO Format: Mask
	15	**enble smallPL** Format: enable **Programming Notes** This bit MUST be set to ensure optimal power in 3D Sampler. Must not be enabled if **cache_flush** message is sent to sampler.
	14	**L1 Cache Set Selection** This field controls how Set0 and Set1 of the Sampler L1 Cache are selected. **Value** / **Name** / **Description** 0h / Extended_Tag **[Default]** / When this bit is set to 0h, Sampler will use Bit 8 of the Extended Tag To Control Set selection. 1h / XOR_UVQ / When this bit is set to 1h, Sampler will use an XOR of Various u, v, and q texel coordinate bits to select set
	13:12	**Sampler Cache Set XOR selection** Format: U2 These bits have an impact only when the Sampler cache is configured in 16 way set associative mode. If the cache is being used for immediate data or for blitter data these bits have no effect. **Value** / **Name** / **Description** 00b / None / No XOR. 01b / Scheme 1 / New_set_mask[3:0] = Tiled_address[16:13]. New_set[3:0] less than or = New_set_mask[3:0] ^Old_set[3:0]. Rationale: These bits can distinguish among 16 different equivalent classes of virtual pages. These bits also represent the lsb for tile rows

SAMPLER_MODE - SAMPLER Mode Register

		ranging from a pitch of 1 tile to 16 tiles.
10b	Scheme 2	New_set_mask[3] = Tiled_address[17] ^ Tiled_address[16]. New_set_mask[2] = Tiled_address[16] ^ Tiled_address[15]. New_set_mask[1] = Tiled_address[15] ^ Tiled_address[14]. New_set_mask[0] = Tiled_address[14] ^ Tiled_address[13]. New_set[3:0] less than or = New_set_mask[3:0] ^ Old_set[3:0]. Rationale: More bits on each XOR can give better statistical uniformity on sets and since two lsbs are taken for each tile row size, it reduces the chance of aliasing on sets.
11b	Scheme 3 **[Default]**	New_set_mask[3] = Tiled_address[22] ^ Tiled_address[21] ^ Tiled_address[20] ^ Tiled_address[19]. New_set_mask[2] = Tiled_address[18] ^ Tiled_address[17] ^ Tiled_address[16]. New_set_mask[1] = Tiled_address[15] ^ Tiled_address[14]. New_set_mask[0] = Tiled_address[13]. New_set[3:0] less than or = New_set_mask[3:0] ^ Old_set[3:0]. Rationale: More bits on each XOR can give better statistical uniformity on sets and since each XOR has different bits, it reduces the chance of aliasing on sets even more.

Programming Notes

This field should be programmed as "00b" corresponding to NO XOR option when 3D map performance fix in MT is enabled using the field "Sampler Set Remapping for 3D Disable" in the SAMPLER Mode Register.

11:10	**ECO Reserved 2b** Format: MBZ
9	**ECO_SCRATCH_9** Format: MBZ
8	**Sampler L2 Disable** Format: Disable Will disable the L2 cache and force all access to be misses
7	**ECO Reserved 3** Format: MBZ
6	**Compressed Overfill disable** Format: disable
5	**Headerless Message for Pre-emptable Contexts** Format: Enable When set to 1h, this bit enables sampler to support headerless messages for pre-emptable GPGPU contexts. When set to 0h, it reverts to the previous behavior where pre-emptable GPGPU contexts must have headers on all sampler messages.

SAMPLER_MODE - SAMPLER Mode Register

Value	Name	Description
0h		Headers must be used on all sampler messages if this bit is programmed to 0h.
1h	**[Default]**	Headerless sampler messages may be used for pre-emptable contexts (this is the default behavior all contexts)

4 **ECO Reserved**

3 **ECO Reserved 5**

Format:	MBZ

2 **ECO Reserved 4-2**

1 **PL float to fix precision fix disable**

Format:	disable

If disabled use legacy behavior

0 **Indirect State Base Addr Override**

Format:	Enable

This bit is used to control whether Indirect State (Border Color) to be relative to same base address as SAMPLER_STATE or relative to the DYNAMIC_STATE_BASE_ADDR

This bit must be set to 1. Border Color must be in the Dynamic State heap in order to allow for resource copy of sampler state to work correctly.

Value	Name	Description
0h	OVERRIDE_DISABLE **[Default]**	When set to 0h, this bit allows Indirect State (.e.g Border Color) to be stored in the Bindless Sampler State heap. This bit should not be set to 0h by SW as this will prevent resource copies of sampler state from being done correct when border color is required.
1h	OVERRIDE_ENABLE	When set to 1h, this bit allows Indirect State (e.g. Border Color) to be stored in the Dynamic State heap. This bit must be set to 1h by SW to ensure resource copies of sampler state are done correctly when border color is required.

SAMPLER READ DATA

SAMPLER_RDATA - SAMPLER READ DATA		
Register Space:	MMIO: 0/2/0	
Access:	RO Variant	
Size (in bits):	32	
Address:	0E144h	
DWord	**Bit**	**Description**
0	31:0	**Reserved**

SCRATCH 1 from LPFCunit

SCRATCH_LPFC1 - SCRATCH 1 from LPFCunit		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
_Custom_GTIReset:	DEV	
Address:	0B474h	
DWord	**Bit**	**Description**
0	31:0	**SCRATCH bits from LPFCunit** Access: R/W _Custom_GTIReset: DEV

SCRATCH 2 for LNCFunit

SCRATCH_LNCF2 - SCRATCH 2 for LNCFunit

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 0B0A0h

DWord	Bit	Description
0	31:23	**Reserved** Access: RO Format: MBZ
	22:3	**SCRATCH 2 field for LNCFunit** Access: R/W _Custom_GTIReset: DEV
	2	**Colored GSYNC Disable** Access: R/W Format: Disable _Custom_GTIReset: DEV
	1	**LNI Cross-Slice Tracking Queue Disable** Access: R/W _Custom_GTIReset: DEV Value / Name / Description 0 / **[Default]** / LNI will only use the fifth tracking queue, allow two tracking queues for cross-slice cycles 1 / / LNI will only allocate one tracking queue for cross-slice cycles
	0	**Reserved** Access: RO Format: MBZ

SCRATCH2 Register

SCRATCH2 - SCRATCH2 Register

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B140h

DWord	Bit	Description
0	31:30	**SCRATCH31**
	29:28	**SCRATCH29**
	27	**Reserved** Access: RO Format: MBZ
	26	**SBFT Physical Override** This bit-field will enable DFT to control the physical(coherent) attribute of cacheline being loaded into L3 during SBFT loading. This bit-field can be set to 1 only during SBFT loading/testing. DFX will handle making sure these get set properly (ie take care of race conditions between loading and setting the bits, etc) L3 can make the assumption that no cycles will be in-flight when the values are changing. DFX will only change the value when transactions are idle. Value: 0 **[Default]** Physical Vs. Virtual indication sent by LSQC will be used Value: 1 Coherent lines will be loaded in L3 during SBFT loading. L3 is in SBFT loading/testing phase. Can be set to 1 only during SBFT loading/testing
	25:21	**SCRATCH25**
	20:15	**Reserved** Access: RO Format: MBZ
	14:9	**Reserved** Access: RO Format: MBZ
	8:4	**LTCD EBB Conflict L3-Read Aging count** This bit-field will determine the number of EBB conflicts experienced by the L3-read in order to age in the L3 FIFO. Value: 10 **[Default]** Value: [1,31]

SCRATCH2 - SCRATCH2 Register

	3	**LTCD EBB Conflict L3-Read Aging enable** This bit-field will enable L3-Read aging in LTCD dataunit due to EBB conflicts. During EBB conflicts, L3-fill will be preferred over L3-reads, and this aggressive fill selection can hold up the reads in L3 FIFO depending on the duration of EBB conflicts. L3-read aging will ensure fairness to an L3-read cycle that has been held back due to EBB conflicts by making the L3-fills ineligible once the read has aged.

Value	Name	Description
1	**[Default]**	LTCD EBB conflict L3-Read aging is enabled
0		LTCD EBB conflict L3-Read aging is disabled

	2	**SCRATCH2**
	1	**LTCD L3 FIFO OOO Disable** This bit-field will disable L3 FIFO OOO implementation in LTCD-dataunit and revert to implementation where only one L3 FIFO was used.

Value	Name	Description
0	**[Default]**	LTCD L3 FIFO OOO implementation is enabled
1		LTCD L3 FIFO OOO implementation is disabled

	0	**Reserved**

Access:	RO
Format:	MBZ

SCRATCH 3 for LNCFunit

SCRATCH_LNCF3 - SCRATCH 3 for LNCFunit		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	0B0A8h	
DWord	**Bit**	**Description**
0	31:0	**SCRATCH 3 field for LNCFunit** Access: R/W _Custom_GTIReset: DEV Bit[0] : LSN VR MARK Performance fix Disable. 0 : (Default) Performance fix is enabled. 1 : Disables performance fix.

SCRATCH3 Register

SCRATCH3 - SCRATCH3 Register

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B154h

DWord	Bit	Description
0	31:13	**SPARE FIELDS**
	12	**Merge Disable for non-128B Compressible Writes** Default Value: 0 Access: R/W Format: Disable _Custom_GTIReset: DEV By default, if compression partial write merging support is enabled, all compressible writes can serve as mergeable parent cycles. If this bit is set, L3 will limit merging for compressible surfaces to only apply when receiving a 128B write (marked as such by the client).
	11	**Force Uncacheable Big Hammer** Value 0 (**[Default]**): All the POR rules for Cacheable vs Uncacheable are applied Value 1: • All the POR rules for C/UC will be overridden and every L3 cycle is made as L3 Uncacheable. • Note: This does not apply for URB related transactions.
	10:8	**LTCD TAG ROINV FSM WAIT TIME** Used to derive the number of clocks for which the ROINV FSM will wait, in a WAIT state, prior to invalidating the state arrays. Valid values start from 4 to account for time it takes ltcc to receive roinv in progress and stall its pipeline towards LTCD-TAGUNIT, to ensure that any functional cycles that sneaks in TAG for those clocks are serviced gracefully. 4: default **[Default]** [4,7]: range
	7:4	**LTCD DATA INIT FSM WAIT TIME** Used to derive the number of 2x clocks for which the DATA-INIT FSM will wait, in a WAIT state, prior to initializing the data arrays. Final-wait time = 2**this-reg-fields-value. 4: default **[Default]** [4,12]: range

SCRATCH3 - SCRATCH3 Register		
	3:0	**Reserved** Access: RO Format: MBZ

Second Level Batch Buffer Head Pointer Preemption Register

SBB_PREEMPT_ADDR - Second Level Batch Buffer Head Pointer Preemption Register	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0213Ch-0213Fh
Name:	Second Level Batch Buffer Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_RCSUNIT
Address:	1813Ch-1813Fh
Name:	Second Level Batch Buffer Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_POCSUNIT
Address:	2213Ch-2213Fh
Name:	Second Level Batch Buffer Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_BCSUNIT
Address:	1C013Ch-1C013Fh
Name:	Second Level Batch Buffer Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_VCSUNIT0
Address:	1C413Ch-1C413Fh
Name:	Second Level Batch Buffer Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_VCSUNIT1
Address:	1C813Ch-1C813Fh
Name:	Second Level Batch Buffer Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_VECSUNIT0
Address:	1D013Ch-1D013Fh
Name:	Second Level Batch Buffer Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_VCSUNIT2
Address:	1D413Ch-1D413Fh
Name:	Second Level Batch Buffer Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_VCSUNIT3
Address:	1D813Ch-1D813Fh
Name:	Second Level Batch Buffer Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_VECSUNIT1
Address:	1E013Ch-1E013Fh

SBB_PREEMPT_ADDR - Second Level Batch Buffer Head Pointer Preemption Register

Name:	Second Level Batch Buffer Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_VCSUNIT4

Address:	1E413Ch-1E413Fh
Name:	Second Level Batch Buffer Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_VCSUNIT5

Address:	1E813Ch-1E813Fh
Name:	Second Level Batch Buffer Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_VECSUNIT2

Address:	1F013Ch-1F013Fh
Name:	Second Level Batch Buffer Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_VCSUNIT6

Address:	1F413Ch-1F413Fh
Name:	Second Level Batch Buffer Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_VCSUNIT7

Address:	1F813Ch-1F813Fh
Name:	Second Level Batch Buffer Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_VECSUNIT3

Address:	1A13Ch-1A13Fh
Name:	Second Level Batch Buffer Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_CCSUNIT0

Description

This register gets updated with the DWord-aligned graphics memory address of the PREEMPTABLE command in the second level batch buffer on which preemption has occurred.
This register value should be looked at only when the preemption has occurred in the second level batch buffer. This is indicated by "Ring/Batch Indicator" in "RING_BUFFER_HEAD_PREEMPT_REG". This register value retains its previous value and doesn't change when the preemption occurs on a preemptable command in ring buffer or in batch buffer.
Preemption is triggered by valid UHPTR in ring buffer mode of scheduling and by a pending execlistin Exec-List mode of scheduling.
This is a global register and context save/restored as part of power context image.

Refer to **Preemption > ExecList Scheduling** for a list of preemptible commands.

Programming Notes

Programming Restriction: This register should NEVER be programmed by driver, this is for HW internal use only.

DWord	Bit	Description

SBB_PREEMPT_ADDR - Second Level Batch Buffer Head Pointer Preemption Register

0	31:2	**Second Level Batch Buffer Head Pointer** Format: GraphicsAddress[31:2] This field specifies the DWord-aligned Graphics Memory Address of the PREEMPTABLE command in a batch buffer where the Preemption has occurred.
	1:0	**Reserved** Access: RO Format: MBZ

Second Level Batch Buffer Head Pointer Register

SBB_ADDR - Second Level Batch Buffer Head Pointer Register	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02114h-02117h
Name:	Second Level Batch Buffer Head Pointer Register
ShortName:	SBB_ADDR_RCSUNIT
Address:	18114h-18117h
Name:	Second Level Batch Buffer Head Pointer Register
ShortName:	SBB_ADDR_POCSUNIT
Address:	22114h-22117h
Name:	Second Level Batch Buffer Head Pointer Register
ShortName:	SBB_ADDR_BCSUNIT
Address:	1C0114h-1C0117h
Name:	Second Level Batch Buffer Head Pointer Register
ShortName:	SBB_ADDR_VCSUNIT0
Address:	1C4114h-1C4117h
Name:	Second Level Batch Buffer Head Pointer Register
ShortName:	SBB_ADDR_VCSUNIT1
Address:	1C8114h-1C8117h
Name:	Second Level Batch Buffer Head Pointer Register
ShortName:	SBB_ADDR_VECSUNIT0
Address:	1D0114h-1D0117h
Name:	Second Level Batch Buffer Head Pointer Register
ShortName:	SBB_ADDR_VCSUNIT2
Address:	1D4114h-1D4117h
Name:	Second Level Batch Buffer Head Pointer Register
ShortName:	SBB_ADDR_VCSUNIT3
Address:	1D8114h-1D8117h
Name:	Second Level Batch Buffer Head Pointer Register
ShortName:	SBB_ADDR_VECSUNIT1
Address:	1E0114h-1E0117h
Name:	Second Level Batch Buffer Head Pointer Register
ShortName:	SBB_ADDR_VCSUNIT4

SBB_ADDR - Second Level Batch Buffer Head Pointer Register

Address:	1E4114h-1E4117h
Name:	Second Level Batch Buffer Head Pointer Register
ShortName:	SBB_ADDR_VCSUNIT5

Address:	1E8114h-1E8117h
Name:	Second Level Batch Buffer Head Pointer Register
ShortName:	SBB_ADDR_VECSUNIT2

Address:	1F0114h-1F0117h
Name:	Second Level Batch Buffer Head Pointer Register
ShortName:	SBB_ADDR_VCSUNIT6

Address:	1F4114h-1F4117h
Name:	Second Level Batch Buffer Head Pointer Register
ShortName:	SBB_ADDR_VCSUNIT7

Address:	1F8114h-1F8117h
Name:	Second Level Batch Buffer Head Pointer Register
ShortName:	SBB_ADDR_VECSUNIT3

Address:	1A114h-1A117h
Name:	Second Level Batch Buffer Head Pointer Register
ShortName:	SBB_ADDR_CCSUNIT0

This register contains the current DWord Graphics Memory Address of the last-initiated batch buffer.

Programming Notes

This register should NEVER be programmed by driver, this is for HW internal use only. This register should not be written by software. These fields should only get written by a context restore. Software should always set these fields via the MI_BATCH_BUFFER_START command when initiating a batch buffer. This register is saved and restored with context.

DWord	Bit	Description
0	31:2	**Second Level Batch Buffer Head Pointer** Format: GraphicsAddress[31:2] This field specifies the DWord-aligned Graphics Memory Address where the last initiated Second Level Batch Buffer is currently fetching commands. This field is meaningful only when Valid field is set to "1".
	1	**Reserved** Access: RO Format: MBZ
	0	**Valid** Format: U1

SBB_ADDR - Second Level Batch Buffer Head Pointer Register

Value	Name	Description
0h	Invalid **[Default]**	Second Level Batch buffer Invalid
1h	Valid	Second Batch buffer Valid.

Second Level Batch Buffer State Register

SBB_STATE - Second Level Batch Buffer State Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02118h-0211Bh
Name:	Second Level Batch Buffer State Register
ShortName:	SBB_STATE_RCSUNIT
Address:	18118h-1811Bh
Name:	Second Level Batch Buffer State Register
ShortName:	SBB_STATE_POCSUNIT
Address:	22118h-2211Bh
Name:	Second Level Batch Buffer State Register
ShortName:	SBB_STATE_BCSUNIT
Address:	1C0118h-1C011Bh
Name:	Second Level Batch Buffer State Register
ShortName:	SBB_STATE_VCSUNIT0
Address:	1C4118h-1C411Bh
Name:	Second Level Batch Buffer State Register
ShortName:	SBB_STATE_VCSUNIT1
Address:	1C8118h-1C811Bh
Name:	Second Level Batch Buffer State Register
ShortName:	SBB_STATE_VECSUNIT0
Address:	1D0118h-1D011Bh
Name:	Second Level Batch Buffer State Register
ShortName:	SBB_STATE_VCSUNIT2
Address:	1D4118h-1D411Bh
Name:	Second Level Batch Buffer State Register
ShortName:	SBB_STATE_VCSUNIT3
Address:	1D8118h-1D811Bh
Name:	Second Level Batch Buffer State Register
ShortName:	SBB_STATE_VECSUNIT1
Address:	1E0118h-1E011Bh
Name:	Second Level Batch Buffer State Register
ShortName:	SBB_STATE_VCSUNIT4

SBB_STATE - Second Level Batch Buffer State Register

Address:	1E4118h-1E411Bh
Name:	Second Level Batch Buffer State Register
ShortName:	SBB_STATE_VCSUNIT5

Address:	1E8118h-1E811Bh
Name:	Second Level Batch Buffer State Register
ShortName:	SBB_STATE_VECSUNIT2

Address:	1F0118h-1F011Bh
Name:	Second Level Batch Buffer State Register
ShortName:	SBB_STATE_VCSUNIT6

Address:	1F4118h-1F411Bh
Name:	Second Level Batch Buffer State Register
ShortName:	SBB_STATE_VCSUNIT7

Address:	1F8118h-1F811Bh
Name:	Second Level Batch Buffer State Register
ShortName:	SBB_STATE_VECSUNIT3

Address:	1A118h-1A11Bh
Name:	Second Level Batch Buffer State Register
ShortName:	SBB_STATE_CCSUNIT0

This register contains the attributes of the second level batch buffer initiated from the batch Buffer.

This register should not be written by software. These fields should only get written by a context restore. Software should always set these fields via the MI_BATCH_BUFFER_START command when initiating a batch buffer. This register is saved and restored with context.

DWord	Bit	Description
0	31:10	**Reserved** Access: RO Format: MBZ
	9	**POSH Start** Exists If: //RCS, POCS This bit reflects the POSH Start value programmed by the active first level MI_BATCH_BUFFER_START command.
	8	**POSH Enable** Exists If: //RCS, POCS This bit reflects the POSH Enable value programmed by the active first level MI_BATCH_BUFFER_START command.
	7:6	**Reserved** Access: RO Format: MBZ

SBB_STATE - Second Level Batch Buffer State Register

	5	**Address Space Indicator** Note: This field reflects the effective address space indicator security level and may not be the same as the Address Space Indicator written using MI_BATCH_BUFFER_START.

Value	Name	Description
0h	GGTT **[Default]**	This second level batch buffer is located in GGTT memory and is privileged
1h	PPGTT	This second level batch buffer is located in PPGTT memory and is non-privileged.

	4	**Reserved**
	3:0	**Reserved**

Access:	RO
Format:	MBZ

Second Level Batch Buffer Upper Head Pointer Preemption Register

SBB_PREEMPT_ADDR_UDW - Second Level Batch Buffer Upper Head Pointer Preemption Register	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02138h-0213Bh
Name:	Second Level Batch Buffer Upper Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_UDW_RCSUNIT
Address:	18138h-1813Bh
Name:	Second Level Batch Buffer Upper Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_UDW_POCSUNIT
Address:	22138h-2213Bh
Name:	Second Level Batch Buffer Upper Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_UDW_BCSUNIT
Address:	1C0138h-1C013Bh
Name:	Second Level Batch Buffer Upper Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_UDW_VCSUNIT0
Address:	1C4138h-1C413Bh
Name:	Second Level Batch Buffer Upper Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_UDW_VCSUNIT1
Address:	1C8138h-1C813Bh
Name:	Second Level Batch Buffer Upper Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_UDW_VECSUNIT0
Address:	1D0138h-1D013Bh
Name:	Second Level Batch Buffer Upper Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_UDW_VCSUNIT2
Address:	1D4138h-1D413Bh
Name:	Second Level Batch Buffer Upper Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_UDW_VCSUNIT3
Address:	1D8138h-1D813Bh
Name:	Second Level Batch Buffer Upper Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_UDW_VECSUNIT1
Address:	1E0138h-1E013Bh

SBB_PREEMPT_ADDR_UDW - Second Level Batch Buffer Upper Head Pointer Preemption Register

Name:	Second Level Batch Buffer Upper Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_UDW_VCSUNIT4
Address:	1E4138h-1E413Bh
Name:	Second Level Batch Buffer Upper Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_UDW_VCSUNIT5
Address:	1E8138h-1E813Bh
Name:	Second Level Batch Buffer Upper Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_UDW_VECSUNIT2
Address:	1F0138h-1F013Bh
Name:	Second Level Batch Buffer Upper Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_UDW_VCSUNIT6
Address:	1F4138h-1F413Bh
Name:	Second Level Batch Buffer Upper Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_UDW_VCSUNIT7
Address:	1F8138h-1F813Bh
Name:	Second Level Batch Buffer Upper Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_UDW_VECSUNIT3
Address:	1A138h-1A13Bh
Name:	Second Level Batch Buffer Upper Head Pointer Preemption Register
ShortName:	SBB_PREEMPT_ADDR_UDW_CCSUNIT0

Upper 32 bits of the 4GB aligned base address within the host's 64-bit virtual address space of the last preempted second level batch buffer. This register follows the same rules as the SBB_PREEMPT_ADDR register.

Programming Notes

Programming Restriction: This register should NEVER be programmed by driver, this is for HW internal use only.

DWord	Bit	Description
0	31:25	**Reserved** Access: RO Format: MBZ
	24:16	**Reserved** Access: RO Format: MBZ
	15:0	**Second Level Batch Buffer Head Pointer Upper DWORD** Format: GraphicsAddress[47:32]

Second Level Batch Buffer Upper Head Pointer Register

SBB_ADDR_UDW - Second Level Batch Buffer Upper Head Pointer Register	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0211Ch-0211Fh
Name:	Second Level Batch Buffer Upper Head Pointer Register
ShortName:	SBB_ADDR_UDW_RCSUNIT
Address:	1811Ch-1811Fh
Name:	Second Level Batch Buffer Upper Head Pointer Register
ShortName:	SBB_ADDR_UDW_POCSUNIT
Address:	2211Ch-2211Fh
Name:	Second Level Batch Buffer Upper Head Pointer Register
ShortName:	SBB_ADDR_UDW_BCSUNIT
Address:	1C011Ch-1C011Fh
Name:	Second Level Batch Buffer Upper Head Pointer Register
ShortName:	SBB_ADDR_UDW_VCSUNIT0
Address:	1C411Ch-1C411Fh
Name:	Second Level Batch Buffer Upper Head Pointer Register
ShortName:	SBB_ADDR_UDW_VCSUNIT1
Address:	1C811Ch-1C811Fh
Name:	Second Level Batch Buffer Upper Head Pointer Register
ShortName:	SBB_ADDR_UDW_VECSUNIT0
Address:	1D011Ch-1D011Fh
Name:	Second Level Batch Buffer Upper Head Pointer Register
ShortName:	SBB_ADDR_UDW_VCSUNIT2
Address:	1D411Ch-1D411Fh
Name:	Second Level Batch Buffer Upper Head Pointer Register
ShortName:	SBB_ADDR_UDW_VCSUNIT3
Address:	1D811Ch-1D811Fh
Name:	Second Level Batch Buffer Upper Head Pointer Register
ShortName:	SBB_ADDR_UDW_VECSUNIT1
Address:	1E011Ch-1E011Fh

SBB_ADDR_UDW - Second Level Batch Buffer Upper Head Pointer Register

Name:	Second Level Batch Buffer Upper Head Pointer Register
ShortName:	SBB_ADDR_UDW_VCSUNIT4
Address:	1E411Ch-1E411Fh
Name:	Second Level Batch Buffer Upper Head Pointer Register
ShortName:	SBB_ADDR_UDW_VCSUNIT5
Address:	1E811Ch-1E811Fh
Name:	Second Level Batch Buffer Upper Head Pointer Register
ShortName:	SBB_ADDR_UDW_VECSUNIT2
Address:	1F011Ch-1F011Fh
Name:	Second Level Batch Buffer Upper Head Pointer Register
ShortName:	SBB_ADDR_UDW_VCSUNIT6
Address:	1F411Ch-1F411Fh
Name:	Second Level Batch Buffer Upper Head Pointer Register
ShortName:	SBB_ADDR_UDW_VCSUNIT7
Address:	1F811Ch-1F811Fh
Name:	Second Level Batch Buffer Upper Head Pointer Register
ShortName:	SBB_ADDR_UDW_VECSUNIT3
Address:	1A11Ch-1A11Fh
Name:	Second Level Batch Buffer Upper Head Pointer Register
ShortName:	SBB_ADDR_UDW_CCSUNIT0

Upper 32 bits of the 4GB aligned base address within the host's 64-bit virtual address space, where the last initiated Batch Buffer is currently fetching commands. If no batch buffer is currently active, the Valid bit in BB_ADDR will be 0 and this field is meaningless.

Programming Notes

This register should NEVER be programmed by driver. This is for HW internal use only.

DWord	Bit	Description
0	31:25	**Reserved** Access: RO Format: MBZ
	24:16	**Reserved** Access: RO Format: MBZ
	15:0	**Batch Buffer Head Pointer Upper DWORD** Format: GraphicsAddress[47:32]

SEL_FETCH_PLANE_CTL

SEL_FETCH_PLANE_CTL	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to PLANE_SURF or plane not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank or pipe not enabled; after armed
Address:	70890h-70893h
Name:	Selective Fetch Plane Control
ShortName:	SEL_FETCH_PLANE_CTL_1_A
Reset:	soft
Address:	708B0h-708B3h
Name:	Selective Fetch Plane Control
ShortName:	SEL_FETCH_PLANE_CTL_2_A
Reset:	soft
Address:	708D0h-708D3h
Name:	Selective Fetch Plane Control
ShortName:	SEL_FETCH_PLANE_CTL_3_A
Reset:	soft
Address:	708F0h-708F3h
Name:	Selective Fetch Plane Control
ShortName:	SEL_FETCH_PLANE_CTL_4_A
Reset:	soft
Address:	70920h-70923h
Name:	Selective Fetch Plane Control
ShortName:	SEL_FETCH_PLANE_CTL_5_A
Reset:	soft
Address:	70940h-70943h
Name:	Selective Fetch Plane Control
ShortName:	SEL_FETCH_PLANE_CTL_6_A
Reset:	soft
Address:	70960h-70963h
Name:	Selective Fetch Plane Control
ShortName:	SEL_FETCH_PLANE_CTL_7_A
Reset:	soft
Restriction	

SEL_FETCH_PLANE_CTL

Refer to 'Plane Capability and Interoperability' page for plane capabilities and restrictions.

DWord	Bit	Description
0	31	**Selective Fetch Plane Enable** When this bit is set, Plane is enabled for selective fetch update. Value: 0b — Name: Disable Value: 1b — Name: Enable
	30:0	**Spares**

Semaphore Interrupt Register

GUC_SEM_INTR - Semaphore Interrupt Register		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Bits are set based on the Semaphore Token number sent by Command streamer Read and cleared by the MinIA uKernel or host CPU.		
DWord	**Bit**	**Description**
0	31:0	**Semaphore Token ID** Default Value: 0000h Access: R/WC _Custom_GTIReset: BUS

Semaphore Polling Interval on Wait

SEMA_WAIT_POLL - Semaphore Polling Interval on Wait	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0224Ch-0224Fh
Name:	Semaphore Polling Interval on Wait
ShortName:	SEMA_WAIT_POLL_RCSUNIT
Address:	1824Ch-1824Fh
Name:	Semaphore Polling Interval on Wait
ShortName:	SEMA_WAIT_POLL_POCSUNIT
Address:	2224Ch-2224Fh
Name:	Semaphore Polling Interval on Wait
ShortName:	SEMA_WAIT_POLL_BCSUNIT
Address:	1C024Ch-1C024Fh
Name:	Semaphore Polling Interval on Wait
ShortName:	SEMA_WAIT_POLL_VCSUNIT0
Address:	1C424Ch-1C424Fh
Name:	Semaphore Polling Interval on Wait
ShortName:	SEMA_WAIT_POLL_VCSUNIT1
Address:	1C824Ch-1C824Fh
Name:	Semaphore Polling Interval on Wait
ShortName:	SEMA_WAIT_POLL_VECSUNIT0
Address:	1D024Ch-1D024Fh
Name:	Semaphore Polling Interval on Wait
ShortName:	SEMA_WAIT_POLL_VCSUNIT2
Address:	1D424Ch-1D424Fh
Name:	Semaphore Polling Interval on Wait
ShortName:	SEMA_WAIT_POLL_VCSUNIT3
Address:	1D824Ch-1D824Fh
Name:	Semaphore Polling Interval on Wait
ShortName:	SEMA_WAIT_POLL_VECSUNIT1
Address:	1E024Ch-1E024Fh
Name:	Semaphore Polling Interval on Wait
ShortName:	SEMA_WAIT_POLL_VCSUNIT4

SEMA_WAIT_POLL - Semaphore Polling Interval on Wait

Address:	1E424Ch-1E424Fh
Name:	Semaphore Polling Interval on Wait
ShortName:	SEMA_WAIT_POLL_VCSUNIT5

Address:	1E824Ch-1E824Fh
Name:	Semaphore Polling Interval on Wait
ShortName:	SEMA_WAIT_POLL_VECSUNIT2

Address:	1F024Ch-1F024Fh
Name:	Semaphore Polling Interval on Wait
ShortName:	SEMA_WAIT_POLL_VCSUNIT6

Address:	1F424Ch-1F424Fh
Name:	Semaphore Polling Interval on Wait
ShortName:	SEMA_WAIT_POLL_VCSUNIT7

Address:	1F824Ch-1F824Fh
Name:	Semaphore Polling Interval on Wait
ShortName:	SEMA_WAIT_POLL_VECSUNIT3

Address:	1A24Ch-1A24Fh
Name:	Semaphore Polling Interval on Wait
ShortName:	SEMA_WAIT_POLL_CCSUNIT0

The SEMA_WAIT_POLL register contains Poll Interval field which specifies the minimum number of microseconds allowed for command streamer to wait before re-fetching the data from the address mentioned in the MI_SEMAPHORE_WAIT command on WAIT Mode set to POLL until the condition is satisfied while the context is not switched out. When a value of 0 is written the poll interval will be equal to the memory latency of the read completion.

DWord	Bit	Description
0	31:21	**Reserved** Access: RO Format: MBZ
	20:0	**Poll Interval** Minimum number of micro-seconds allowed

Semaphore Signal Port

SEMAPHORE_SIGNAL_PORT - Semaphore Signal Port	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02020h-02023h
Name:	SEMAPHORE_SIGNAL_PORT
ShortName:	SEMAPHORE_SIGNAL_PORT_RCSUNIT
Address:	18020h-18023h
Name:	SEMAPHORE_SIGNAL_PORT
ShortName:	SEMAPHORE_SIGNAL_PORT_POCSUNIT
Address:	22020h-22023h
Name:	SEMAPHORE_SIGNAL_PORT
ShortName:	SEMAPHORE_SIGNAL_PORT_BCSUNIT
Address:	1C0020h-1C0023h
Name:	SEMAPHORE_SIGNAL_PORT
ShortName:	SEMAPHORE_SIGNAL_PORT_VCSUNIT0
Address:	1C4020h-1C4023h
Name:	SEMAPHORE_SIGNAL_PORT
ShortName:	SEMAPHORE_SIGNAL_PORT_VCSUNIT1
Address:	1C8020h-1C8023h
Name:	SEMAPHORE_SIGNAL_PORT
ShortName:	SEMAPHORE_SIGNAL_PORT_VECSUNIT0
Address:	1D0020h-1D0023h
Name:	SEMAPHORE_SIGNAL_PORT
ShortName:	SEMAPHORE_SIGNAL_PORT_VCSUNIT2
Address:	1D4020h-1D4023h
Name:	SEMAPHORE_SIGNAL_PORT
ShortName:	SEMAPHORE_SIGNAL_PORT_VCSUNIT3
Address:	1D8020h-1D8023h
Name:	SEMAPHORE_SIGNAL_PORT
ShortName:	SEMAPHORE_SIGNAL_PORT_VECSUNIT1
Address:	1E0020h-1E0023h
Name:	SEMAPHORE_SIGNAL_PORT
ShortName:	SEMAPHORE_SIGNAL_PORT_VCSUNIT4

SEMAPHORE_SIGNAL_PORT - Semaphore Signal Port

Address:	1E4020h-1E4023h
Name:	SEMAPHORE_SIGNAL_PORT
ShortName:	SEMAPHORE_SIGNAL_PORT_VCSUNIT5

Address:	1E8020h-1E8023h
Name:	SEMAPHORE_SIGNAL_PORT
ShortName:	SEMAPHORE_SIGNAL_PORT_VECSUNIT2

Address:	1F0020h-1F0023h
Name:	SEMAPHORE_SIGNAL_PORT
ShortName:	SEMAPHORE_SIGNAL_PORT_VCSUNIT6

Address:	1F4020h-1F4023h
Name:	SEMAPHORE_SIGNAL_PORT
ShortName:	SEMAPHORE_SIGNAL_PORT_VCSUNIT7

Address:	1F8020h-1F8023h
Name:	SEMAPHORE_SIGNAL_PORT
ShortName:	SEMAPHORE_SIGNAL_PORT_VECSUNIT3

Address:	1A020h-1A023h
Name:	SEMAPHORE_SIGNAL_PORT
ShortName:	SEMAPHORE_SIGNAL_PORT_CCSUNIT0

Each engine implements SEMAPHORE_SIGNAL_PORT register for receiving semaphore signal from the scheduler (SW or FW). A write to the SEMAPHORE_SIGNAL_PORT with data as 0xFFFF_FFFF is decoded as semaphore signal by an engine. An engine waiting on un-successful MI_SEMAPHORE_WAIT (signal mode) command will reacquire the semaphore data from memory and re-evaluate the semaphore comparison on receiving the semaphore signal. SEMAPHORE_SIGNAL_PORT register is privileged. Writing to the SEMAPHORE_SIGNAL_PORT of an idle engine (no context) does not trigger any action in HW and is of no use.

DWord	Bit	Description
0	31:0	**Semaphore Signal Value** Format: U32

Semaphore Token

SEMAPHORE_TOKEN - Semaphore Token	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	022B4h-022B7h
Name:	SEMAPHORE_TOKEN
ShortName:	SEMAPHORE_TOKEN_RCSUNIT
Address:	182B4h-182B7h
Name:	SEMAPHORE_TOKEN
ShortName:	SEMAPHORE_TOKEN_POCSUNIT
Address:	222B4h-222B7h
Name:	SEMAPHORE_TOKEN
ShortName:	SEMAPHORE_TOKEN_BCSUNIT
Address:	1C02B4h-1C02B7h
Name:	SEMAPHORE_TOKEN
ShortName:	SEMAPHORE_TOKEN_VCSUNIT0
Address:	1C42B4h-1C42B7h
Name:	SEMAPHORE_TOKEN
ShortName:	SEMAPHORE_TOKEN_VCSUNIT1
Address:	1C82B4h-1C82B7h
Name:	SEMAPHORE_TOKEN
ShortName:	SEMAPHORE_TOKEN_VECSUNIT0
Address:	1D02B4h-1D02B7h
Name:	SEMAPHORE_TOKEN
ShortName:	SEMAPHORE_TOKEN_VCSUNIT2
Address:	1D42B4h-1D42B7h
Name:	SEMAPHORE_TOKEN
ShortName:	SEMAPHORE_TOKEN_VCSUNIT3
Address:	1D82B4h-1D82B7h
Name:	SEMAPHORE_TOKEN
ShortName:	SEMAPHORE_TOKEN_VECSUNIT1
Address:	1E02B4h-1E02B7h
Name:	SEMAPHORE_TOKEN
ShortName:	SEMAPHORE_TOKEN_VCSUNIT4

SEMAPHORE_TOKEN - Semaphore Token

Address:	1E42B4h-1E42B7h
Name:	SEMAPHORE_TOKEN
ShortName:	SEMAPHORE_TOKEN_VCSUNIT5

Address:	1E82B4h-1E82B7h
Name:	SEMAPHORE_TOKEN
ShortName:	SEMAPHORE_TOKEN_VECSUNIT2

Address:	1F02B4h-1F02B7h
Name:	SEMAPHORE_TOKEN
ShortName:	SEMAPHORE_TOKEN_VCSUNIT6

Address:	1F42B4h-1F42B7h
Name:	SEMAPHORE_TOKEN
ShortName:	SEMAPHORE_TOKEN_VCSUNIT7

Address:	1F82B4h-1F82B7h
Name:	SEMAPHORE_TOKEN
ShortName:	SEMAPHORE_TOKEN_VECSUNIT3

Address:	1A2B4h-1A2B7h
Name:	SEMAPHORE_TOKEN
ShortName:	SEMAPHORE_TOKEN_CCSUNIT0

This register holds the Producer Token Number programmed by the SW. Producer Token Number is reported to the GUC by an engine on executing MI_SEMAPHORE_SIGNAL command. SEMAPHORE_TOKEN register is privileged and context save/restored.

DWord	Bit	Description
0	31:27	**Reserved** Access: RO Format: MBZ
	26:0	**Producer Token Number** Format: U27 **Description** This field specifies the token value (bit vector) that is sent to GuC from the CS

Slice 0 BONUS1 Reg

SL0SPCBONUS1 - Slice 0 BONUS1 Reg		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	24194h	
Clock Gating Messages Register		
DWord	**Bit**	**Description**
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**BONUS1 BIT 7** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	6	**BONUS1 BIT 6** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	5	**BONUS1 BIT 5** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	4	**BONUS1 BIT 4** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	3	**BONUS1 BIT 3** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe)

SL0SPCBONUS1 - Slice 0 BONUS1 Reg

		'1' : Initiate power up sequence (clk/rst/fwe)
	2	**BONUS1 BIT 2** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	1	**BONUS1 BIT 1** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**BONUS1 BIT 0** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

Slice 0 BONUS2 Reg

SL0SPCBONUS2 - Slice 0 BONUS2 Reg		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	24198h	
Clock Gating Messages Register		

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**BONUS2 BIT 7** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	6	**BONUS2 BIT 6** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	5	**BONUS2 BIT 5** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	4	**BONUS2 BIT 4** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	3	**BONUS2 BIT 3** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe)

SL0SPCBONUS2 - Slice 0 BONUS2 Reg

		'1' : Initiate power up sequence (clk/rst/fwe)
	2	**BONUS2 BIT 2** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	1	**BONUS2 BIT 1** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**BONUS2 BIT 0** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

Slice 0 PGFET control register with lock

SL0SPCPFETCTL - Slice 0 PGFET control register with lock

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 24188h

DWord	Bit	Description
0	31	**PFET Control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of Slice 0 PGFETCTL register are R/W 1 = All bits of Slice 0 PGFETCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:24	**Reserved** Access: RO Format: MBZ
	23	**Power Well Status** Access: RO _Custom_GTIReset: BUS 0 = Well is powered Down 1 = Well is powered up Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	22	**Reserved** Access: RO Format: MBZ
	21:19	**Delay from enabling secondary PFETs to power good.** Default Value: 111b Access: R/W Lock _Custom_GTIReset: BUS Delay from enabling secondary PFETs to power good 3'b000: 40ns 3'b001: 80ns 3'b010: 160ns 3'b011: 240ns 3'b100: 320ns 3'b101: 480ns 3'b110: 640ns 3'b111: 1280ns

SL0SPCPFETCTL - Slice 0 PGFET control register with lock

18:16 **Strobe pulse period**

Access:	R/W Lock
_Custom_GTIReset:	BUS

Time period b/w two adjacent strobes to the primary FETs
3'b000: 10ns (or 1 bclk)
3'b001: 20ns (or 2 bclk)
3'b010: 30ns (or 3 bclk)
3'b111: 80ns (or 8 bclk)

Value	Name
001b	**[Default]**

15:0 **PFET Ladder Step Sequence**

Access:	R/W Lock
_Custom_GTIReset:	BUS

PFET Ladder STEP sequence
The PFET ladder has 16 steps, each represented by its corresponding bit in pfetlddrstepseq[15:0]. If the bit location is set, it means we must step on that ladder stage before we go to the next stage.
The SPC controller will step through all the steps with asserted bits periodically with period equal to strbpulsprd[2:0]
Programming Rule: pfetldrstepseq[15] and pfetldrstepseq[0] must be '1; A '0 setting for these bits is illegal.
15'FFFFh: Ladder step (ladder_sel) goes 0, 1, 2, ?.15.
15'FFF1h: Ladder step goes 0, 4, 5, 6,?15; Steps 1, 2, 3 are skipped.
15'D555h: Ladder step goes 0, 2, 4, 6, 8, 10, 12, 14, 15; Steps 1, 3, 5, 7, 9, 11, 13 are skipped.
15'80001h: Ladder step goes 0, 15; Steps 1, 2, 3, 4, ?14 are skipped.

Value	Name
1000011111111001b	**[Default]**

Slice 0 Power Context Save request

SL0PGCTXREQ - Slice 0 Power Context Save request		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	24184h	
DWord	**Bit**	**Description**
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask bots for lower 16 bits
	15:10	**Reserved** Access: RO Format: MBZ
	9	**Power context save request** Access: R/W Set _Custom_GTIReset: BUS Power Context Save Request 1'b0 : Power context save is not being requested 1'b1 : Power context save is being requested CPUnit self-clears this bit upon sampling.
	8:0	**Power Context Save request crdit count** Access: R/W _Custom_GTIReset: BUS QWord Credits for Power Context Save Request Minimum Credits = 1 : Unit may send 1 QWord pair (enough for first LRI at least) Maximum Credits = 511 : Unit may send 511 QWord pairs A QWord pair is defined as a 32-bit register address and the corresponding 32-bits of register data. Note that the LRI header and END commands are 64-bits each (32-bit command followed by 32-bit NOOP) and will consume one QWord credit. Only valid with PWRCTX_SAVE_REQ (Bit9).

Slice 0 Power Down FSM control register with lock

SL0SPCPOWERDNFSMCTL - Slice 0 Power Down FSM control register with lock

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24190h

DWord	Bit	Description
0	31	**power down control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of Slice 0 POWERDNFSMCTL register are R/W 1 = All bits of Slice 0 POWERDNFSMCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:13	**Reserved** Access: RO Format: MBZ
	12	**Leave firewall disabled** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not firewall the gated domain for a power down flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e firewall gated domain to ungated domain crossing during power down flows 1 = Leave firewall disabled, i.e don't firewall the gated domain, but complete logical flow
	11	**Leave reset de-asserted** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not assert reset for power off flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e assert resets during power down flows 1 = Leave reset de-asserted mode, i.e don't assert reset, but complete logical flow
	10	**Leave CLKs ON** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not gate clks for power off flow. But it will pretend to complete the flow with PM Encodings:

SL0SPCPOWERDNFSMCTL - Slice 0 Power Down FSM control register with lock

		0 = Default mode, i.e gate clocks during power down flows 1 = Leave CLKS ON mode, i.e don't clock gate, but complete logical flow
	9	**Leave FET On** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not turn off the PFET even though it will complete the flow with PM Encodings: 0 = Default mode, i.e power off fets during power down flows 1 = Leave ON mode, i.e dont power off pfet, but complete logical flow
	8:0	**Reserved** Access: RO Format: MBZ

Slice 0 Power Gate Control Request

SL0PGCTLREQ - Slice 0 Power Gate Control Request

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24180h
Clock Gating Messages Register	

DWord	Bit	Description
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask In order to write to bits 15:0, the corresponding message mask bits must be written. For example, for bit 14 to be set, bit 30 needs to be 1 : 40004000
	15:2	**Reserved** Access: RO Format: MBZ
	1	**CLK RST FWE Request** Access: R/W _Custom_GTIReset: BUS SLICE 0 CLK RST FWE request: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**Power Gate Request** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

Slice 0 Power on FSM control register with lock

SL0SPCPOWERUPFSMCTL - Slice 0 Power on FSM control register with lock

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	2418Ch

DWord	Bit	Description
0	31	**power up control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of Slice 0 POWERUPFSMCTL register are R/W 1 = All bits of Slice 0 POWERUPFSMCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:9	**Reserved** Access: RO Format: MBZ
	8:0	**Reserved** Access: RO Format: MBZ

Slice 0 SubSlice 0 PGFET control register with lock

SSM0SPCPFETCTL - Slice 0 SubSlice 0 PGFET control register with lock

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24408h

DWord	Bit	Description
0	31	**PFET Control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of PGFETCTL register are R/W 1 = All bits of PGFETCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:24	**Reserved** Access: RO Format: MBZ
	23	**Power Well Status** Access: RO _Custom_GTIReset: BUS 0 = Well is powered Down 1 = Well is powered up Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	22	**Powergood timer error** Access: RO _Custom_GTIReset: BUS 0 = Well is powered Down 1 = Well is powered up Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	21:19	**Delay from enabling secondary PFETs to power good.** Default Value: 101b Access: R/W Lock _Custom_GTIReset: BUS Delay from enabling secondary PFETs to power good 3'b000: 40ns 3'b001: 80ns 3'b010: 160ns 3'b011: 240ns

SSM0SPCPFETCTL - Slice 0 SubSlice 0 PGFET control register with lock

		3'b100: 320ns 3'b101: 480ns 3'b110: 640ns 3'b111: 1280ns
	18:16	**Strobe pulse period** Default Value: 011b Access: R/W Lock _Custom_GTIReset: BUS Time period b/w two adjacent strobes to the primary FETs 3'b000: 10ns (or 1 bclk) 3'b001: 20ns (or 2 bclk) 3'b010: 30ns (or 3 bclk) 3'b111: 80ns (or 8 bclk)
	15:0	**PFET Ladder Step Sequence** Default Value: 1111111111111111b Access: R/W Lock _Custom_GTIReset: BUS PFET Ladder STEP sequence The PFET ladder has 16 steps, each represented by its corresponding bit in pfetlddrstepseq[15:0]. If the bit location is set, it means we must step on that ladder stage before we go to the next stage. The SPC controller will step through all the steps with asserted bits periodically with period equal to strbpulsprd[2:0] Programming Rule: pfetldrstepseq[15] and pfetldrstepseq[0] must be '1; A '0 setting for these bits is illegal. 15'FFFFh: Ladder step (ladder_sel) goes 0, 1, 2, ?.15. 15'FFF1h: Ladder step goes 0, 4, 5, 6,?15; Steps 1, 2, 3 are skipped. 15'D555h: Ladder step goes 0, 2, 4, 6, 8, 10, 12, 14, 15; Steps 1, 3, 5, 7, 9, 11, 13 are skipped. 15'80001h: Ladder step goes 0, 15; Steps 1, 2, 3, 4, ?14 are skipped.

Slice 0 SubSlice 1 PGFET control register with lock

SSM1SPCPFETCTL - Slice 0 SubSlice 1 PGFET control register with lock

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24488h

DWord	Bit	Description
0	31	**PFET Control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of PGFETCTL register are R/W 1 = All bits of PGFETCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:24	**Reserved** Access: RO Format: MBZ
	23	**Power Well Status** Access: RO _Custom_GTIReset: BUS 0 = Well is powered Down 1 = Well is powered up Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	22	**Powergood timer error** Access: RO _Custom_GTIReset: BUS 0 = Well is powered Down 1 = Well is powered up Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	21:19	**Delay from enabling secondary PFETs to power good.** Default Value: 101b Access: R/W Lock _Custom_GTIReset: BUS Delay from enabling secondary PFETs to power good 3'b000: 40ns 3'b001: 80ns 3'b010: 160ns 3'b011: 240ns

SSM1SPCPFETCTL - Slice 0 SubSlice 1 PGFET control register with lock

		3'b100: 320ns 3'b101: 480ns 3'b110: 640ns 3'b111: 1280ns
	18:16	**Strobe pulse period** Default Value: 011b Access: R/W Lock _Custom_GTIReset: BUS Time period b/w two adjacent strobes to the primary FETs 3'b000: 10ns (or 1 bclk) 3'b001: 20ns (or 2 bclk) 3'b010: 30ns (or 3 bclk) 3'b111: 80ns (or 8 bclk)
	15:0	**PFET Ladder Step Sequence** Default Value: 1111111111111111b Access: R/W Lock _Custom_GTIReset: BUS PFET Ladder STEP sequence The PFET ladder has 16 steps, each represented by its corresponding bit in pfetlddrstepseq[15:0]. If the bit location is set, it means we must step on that ladder stage before we go to the next stage. The SPC controller will step through all the steps with asserted bits periodically with period equal to strbpulsprd[2:0] Programming Rule: pfetldrstepseq[15] and pfetldrstepseq[0] must be '1; A '0 setting for these bits is illegal. 15'FFFFh: Ladder step (ladder_sel) goes 0, 1, 2, ?.15. 15'FFF1h: Ladder step goes 0, 4, 5, 6,?15; Steps 1, 2, 3 are skipped. 15'D555h: Ladder step goes 0, 2, 4, 6, 8, 10, 12, 14, 15; Steps 1, 3, 5, 7, 9, 11, 13 are skipped. 15'80001h: Ladder step goes 0, 15; Steps 1, 2, 3, 4, ?14 are skipped.

Slice 0 SubSlice 2 PGFET control register with lock

SSM2SPCPFETCTL - Slice 0 SubSlice 2 PGFET control register with lock

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24508h

DWord	Bit	Description
0	31	**PFET Control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of PGFETCTL register are R/W 1 = All bits of PGFETCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:24	**Reserved** Access: RO Format: MBZ
	23	**Power Well Status** Access: RO _Custom_GTIReset: BUS 0 = Well is powered Down 1 = Well is powered up Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	22	**Powergood timer error** Access: RO _Custom_GTIReset: BUS 0 = Well is powered Down 1 = Well is powered up Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	21:19	**Delay from enabling secondary PFETs to power good.** Default Value: 101b Access: R/W Lock _Custom_GTIReset: BUS Delay from enabling secondary PFETs to power good 3'b000: 40ns 3'b001: 80ns 3'b010: 160ns 3'b011: 240ns

SSM2SPCPFETCTL - Slice 0 SubSlice 2 PGFET control register with lock

		3'b100: 320ns 3'b101: 480ns 3'b110: 640ns 3'b111: 1280ns
	18:16	**Strobe pulse period** Default Value: 011b Access: R/W Lock _Custom_GTIReset: BUS Time period b/w two adjacent strobes to the primary FETs 3'b000: 10ns (or 1 bclk) 3'b001: 23ns (or 2 bclk) 3'b010: 30ns (or 3 bclk) 3'b111: 80ns (or 8 bclk)
	15:0	**PFET Ladder Step Sequence** Default Value: 1111111111111111b Access: R/W Lock _Custom_GTIReset: BUS PFET Ladder STEP sequence The PFET ladder has 16 steps, each represented by its corresponding bit in pfetlddrstepseq[15:0]. If the bit location is set, it means we must step on that ladder stage before we go to the next stage. The SPC controller will step through all the steps with asserted bits periodically with period equal to strbpulsprd[2:0] Programming Rule: pfetldrstepseq[15] and pfetldrstepseq[0] must be '1; A '0 setting for these bits is illegal. 15'FFFFh: Ladder step (ladder_sel) goes 0, 1, 2, ?.15. 15'FFF1h: Ladder step goes 0, 4, 5, 6,?15; Steps 1, 2, 3 are skipped. 15'D555h: Ladder step goes 0, 2, 4, 6, 8, 10, 12, 14, 15; Steps 1, 3, 5, 7, 9, 11, 13 are skipped. 15'80001h: Ladder step goes 0, 15; Steps 1, 2, 3, 4, ?14 are skipped.

Slice 1 - 5 BONUS1 Reg

SL15SPCBONUS1 - Slice 1 - 5 BONUS1 Reg

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 24214h

Clock Gating Messages Register

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**BONUS1 BIT 7** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	6	**BONUS1 BIT 6** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	5	**BONUS1 BIT 5** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	4	**BONUS1 BIT 4** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	3	**BONUS1 BIT 3** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe)

SL15SPCBONUS1 - Slice 1 - 5 BONUS1 Reg

		'1' : Initiate power up sequence (clk/rst/fwe)
	2	**BONUS1 BIT 2** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	1	**BONUS1 BIT 1** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**BONUS1 BIT 0** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

Slice 1 - 5 BONUS2 Reg

SL15SPCBONUS2 - Slice 1 - 5 BONUS2 Reg

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24218h
Clock Gating Messages Register	

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**BONUS2 BIT 7** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	6	**BONUS2 BIT 6** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	5	**BONUS2 BIT 5** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	4	**BONUS2 BIT 4** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	3	**BONUS2 BIT 3** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe)

SL15SPCBONUS2 - Slice 1 - 5 BONUS2 Reg

		'1' : Initiate power up sequence (clk/rst/fwe)
	2	**BONUS2 BIT 2** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	1	**BONUS2 BIT 1** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**BONUS2 BIT 0** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

Slice 1 - 5 PGFET control register with lock

SL15SPCPFETCTL - Slice 1 - 5 PGFET control register with lock

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 24208h

DWord	Bit	Description
0	31	**PFET Control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of Slice 0 PGFETCTL register are R/W 1 = All bits of Slice 0 PGFETCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:24	**Reserved** Access: RO Format: MBZ
	23	**Power Well Status** Access: R/WC _Custom_GTIReset: BUS strbpulsprdwered Down 1 = Well is powered up Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	22	**Powergood timer error** Access: R/WC _Custom_GTIReset: BUS 0 = Well is powered Down 1 = Well is powered up Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	21:19	**Delay from enabling secondary PFETs to power good.** Default Value: 111b Access: R/W Lock _Custom_GTIReset: BUS Delay from enabling secondary PFETs to power good 3'b000: 40ns 3'b001: 80ns 3'b010: 160ns 3'b011: 240ns 3'b100: 320ns

SL15SPCPFETCTL - Slice 1 - 5 PGFET control register with lock

		3'b101: 480ns 3'b110: 640ns 3'b111: 1280ns
	18:16	**Strobe pulse period** Default Value: 011b Access: R/W Lock _Custom_GTIReset: BUS Time period b/w two adjacent strobes to the primary FETs 3'b000: 10ns (or 1 bclk) 3'b001: 20ns (or 2 bclk) 3'b010: 30ns (or 3 bclk) 3'b111: 80ns (or 8 bclk)
	15:0	**PFET Ladder Step Sequence** Default Value: 1111111111111111b Access: R/W Lock _Custom_GTIReset: BUS PFET Ladder STEP sequence The PFET ladder has 16 steps, each represented by its corresponding bit in pfetlddrstepseq[15:0]. If the bit location is set, it means we must step on that ladder stage before we go to the next stage. The SPC controller will step through all the steps with asserted bits periodically with period equal to strbpulsprd[2:0] Programming Rule: pfetldrstepseq[15] and pfetldrstepseq[0] must be '1; A '0 setting for these bits is illegal. 15'FFFFh: Ladder step (ladder_sel) goes 0, 1, 2, ?.15. 15'FFF1h: Ladder step goes 0, 4, 5, 6,?15; Steps 1, 2, 3 are skipped. 15'D555h: Ladder step goes 0, 2, 4, 6, 8, 10, 12, 14, 15; Steps 1, 3, 5, 7, 9, 11, 13 are skipped. 15'80001h: Ladder step goes 0, 15; Steps 1, 2, 3, 4, ?14 are skipped.

Slice 1 - 5 Power Down FSM control register with lock

SL15SPCPOWERDNFSMCTL - Slice 1 - 5 Power Down FSM control register with lock

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24210h

DWord	Bit	Description
0	31	**power down control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of Slice 1 POWERDNFSMCTL register are R/W 1 = All bits of Slice 1 POWERDNFSMCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:13	**Reserved** Access: RO Format: MBZ
	12	**Leave firewall disabled** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not firewall the gated domain for a power down flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e firewall gated domain to ungated domain crossing during power down flows 1 = Leave firewall disabled, i.e don't firewall the gated domain, but complete logical flow
	11	**Leave reset de-asserted** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not assert reset for power off flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e assert resets during power down flows 1 = Leave reset de-asserted mode, i.e don't assert reset, but complete logical flow
	10	**Leave CLKs ON** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not gate clks for power off flow. But it will pretend to complete the flow with PM Encodings:

SL15SPCPOWERDNFSMCTL - Slice 1 - 5 Power Down FSM control register with lock

		0 = Default mode, i.e gate clocks during power down flows 1 = Leave CLKS ON mode, i.e don't clock gate, but complete logical flow
	9	**Leave FET On** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not turn off the PFET even though it will complete the flow with PM Encodings: 0 = Default mode, i.e power off fets during power down flows 1 = Leave ON mode, i.e don't power off pfet, but complete logical flow
	8:6	**Power Down state 3** Default Value: 010b Access: R/W Lock _Custom_GTIReset: BUS This will be the 3rd state before power is turned OFF in the well Encodings: 000 = Assert Reset 001 = Firewall ON 010 = Gate clocks 1xx = Rsvd for future Default : Gate Clocks
	5:3	**Power Down state 2** Default Value: 001b Access: R/W Lock _Custom_GTIReset: BUS This will be the 2nd state before power is turned OFF in the well Encodings: 000 = Assert Reset 001 = Firewall ON 010 = Gate clocks 1xx = Rsvd for future Default :Firewall ON

SL15SPCPOWERDNFSMCTL - Slice 1 - 5 Power Down FSM control register with lock

	2:0	**Power Down state 1** Default Value: 000b Access: R/W Lock _Custom_GTIReset: BUS This will be the 1st state before power is turned OFF in the well Encodings: 000 = Assert Reset 001 = Firewall ON 010 = Gate clocks 1xx = Rsvd for future Default : Assert Reset

Slice 1 - 5 Power Gate Control Request

SL15PGCTLREQ - Slice 1 - 5 Power Gate Control Request

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24200h
Clock Gating Messages Register	

DWord	Bit	Description
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask In order to write to bits 15:0, the corresponding message mask bits must be written. For example, for bit 14 to be set, bit 30 needs to be 1 : 40004000
	15:2	**Reserved** Access: RO Format: MBZ
	1	**CLK RST FWE Request** Access: R/W _Custom_GTIReset: BUS SLICE 0 CLK RST FWE request: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**Power Gate Request** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

Slice 1 -5 Power on FSM control register with lock

SL15SPCPOWERUPFSMCTL - Slice 1 -5 Power on FSM control register with lock

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	2420Ch

DWord	Bit	Description
0	31	**power up control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of Slice 1 POWERUPFSMCTL register are R/W 1 = All bits of Slice 1 POWERUPFSMCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:9	**Reserved** Access: RO Format: MBZ
	8:6	**Power UP state 3** Default Value: 010b Access: R/W Lock _Custom_GTIReset: BUS This will be the 3rd state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - De-assert resets 3'b000: 10ns (or 1 bclk)
	5:3	**Power UP state 2** Default Value: 001b Access: R/W Lock _Custom_GTIReset: BUS This will be the 2nd state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - Firewall OFF

SL15SPCPOWERUPFSMCTL - Slice 1 -5 Power on FSM control register with lock

	2:0	**Power UP state 1** Default Value: 000b Access: R/W Lock _Custom_GTIReset: BUS This will be the 1st state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - Clock Ungate

Slice unit Level Clock Gating Control 94D0

SCCGCTL94D0 - Slice unit Level Clock Gating Control 94D0

Register Space: MMIO: 0/2/0
Size (in bits): 32
CustomGTIIs_ContextSaved: true

Address: 094D0h

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31:2	**Reserved** Access: RO Format: MBZ
	1	**GCPunit Clock Gating Disable** Default Value: 1b Access: R/W _Custom_GTIReset: BUS GCPunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	0	**SMCRunit Clock Gating Disable** Default Value: 1b Access: R/W _Custom_GTIReset: BUS Graphics Documentation Intel Confidential [Build: 112015] My Dashboard Reports Filters Help SMCRunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

Slice unit Level Clock Gating Control 94D4

SCCGCTL94D4 - Slice unit Level Clock Gating Control 94D4

Register Space: MMIO: 0/2/0
Size (in bits): 32
CustomGTIIs_ContextSaved: true

Address: 094D4h

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31	**SPARE Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SPARE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	30	**lsqcunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS lsqcunit Clock Gating Disable Control '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	29	**ccunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS ccunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	28	**DAPunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS DAPunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	27	**GACBunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GACBunit Clock Gating Disable Control:

SCCGCTL94D4 - Slice unit Level Clock Gating Control 94D4

		'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	26	**GAFSRRB Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GAFSRRB Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	25	**GAHSunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GAHSunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	24	**GAPCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GAPCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	23	**GAPL3unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GAPL3unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	22	**GAPSunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GAPSunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	21	**GAPZunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS

SCCGCTL94D4 - Slice unit Level Clock Gating Control 94D4

		GAPZunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	20	**Reserved** Access: RO Format: MBZ
	19	**HIZunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HIZunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	18	**IZunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS IZunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	17	**L3 Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS L3 Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always) **Programming Notes** As a WA - Disable L3 clock gating by setting bit 16 & 17 of this register
	16	**L3 Clock Gating Disable cr2x** Access: R/W _Custom_GTIReset: BUS L3 Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always) **Programming Notes** As a WA - Disable L3 clock gating by setting bit 16 & 17 of this register

SCCGCTL94D4 - Slice unit Level Clock Gating Control 94D4

	15	**L3Bank Clock Gating Disable cr** Access: R/W _Custom_GTIReset: BUS L3 Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	14	**L3Bank Clock Gating Disable cr2x** Access: R/W _Custom_GTIReset: BUS L3 Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	13	**L3BANK Clock Gating Disable cu** Access: R/W _Custom_GTIReset: BUS l3bank L3BANK Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	12	**L3BANK Clock Gating Disable cu2x** Access: R/W _Custom_GTIReset: BUS L3BANK Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	11	**lsnunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS lsnunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	10	**MSCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS MSCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for

SCCGCTL94D4 - Slice unit Level Clock Gating Control 94D4

		functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	9	**OAADDRunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS OAADDRunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	8	**OASCREP Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS OASCREP Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	7	**RCCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS RCCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	6	**RCZunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS RCZunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	5	**Sarbunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS Sarbunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	4	**SBEunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SBEunit Clock Gating Disable Control:

SCCGCTL94D4 - Slice unit Level Clock Gating Control 94D4

		'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	3	**STCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS STCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	2	**SVLunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SVLunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	1	**WMBE Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS WMBE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	0	**WMFEunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS WMFEunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

Slice unit Level Clock Gating Control 94D8

SCCGCTL94D8 - Slice unit Level Clock Gating Control 94D8

Register Space: MMIO: 0/2/0
Size (in bits): 32
CustomGTIIs_ContextSaved: true

Address: 094D8h

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31	**SFR unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SFR unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	30	**SF unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SF unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	29	**OARUNIT Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS OARUNIT Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	28	**RCU unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS RCU unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	27	**POCS unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS POCS unit Clock Gating Disable Control:

SCCGCTL94D8 - Slice unit Level Clock Gating Control 94D8

		'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	26	**OVR unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS OVR unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	25	**HS unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HS unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	24	**GS unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GS unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	23	**GAFSWRBLK unit Clock Gating Disable wmf** Access: R/W _Custom_GTIReset: BUS GAFSWRBLK unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	22	**GAFSWRBLK unit Clock Gating Disable vf** Access: R/W _Custom_GTIReset: BUS GAFSWRBLK unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	21	**GAFSWRBLK unit Clock Gating Disable 2x** Access: R/W _Custom_GTIReset: BUS

SCCGCTL94D8 - Slice unit Level Clock Gating Control 94D8

		GAFSWRBLK unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	20	**GAFSFFROB unit Clock Gating Disable 2xrdrtn** Access: R/W _Custom_GTIReset: BUS GAFSFFROB unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	19	**GAFSFFROB unit Clock Gating Disable 2x** Access: R/W _Custom_GTIReset: BUS GAFSFFROB unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	18	**GAFD unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GAFD unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	17	**GAFS unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GAFS unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	16	**GAFM unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GAFM unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

SCCGCTL94D8 - Slice unit Level Clock Gating Control 94D8

15	**CS unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS CS unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
14	**CLR unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS CLR unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
13	**CL unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS CL unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
12	**AMFS unit Clock Gating Disable f** Access: R/W _Custom_GTIReset: BUS AMFS unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
11	**AMFS unit Clock Gating Disable c** Access: R/W _Custom_GTIReset: BUS AMFS unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
10	**AMFS unit Clock Gating Disable d** Access: R/W _Custom_GTIReset: BUS AMFS unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)

SCCGCTL94D8 - Slice unit Level Clock Gating Control 94D8

		'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	9	**GADSS unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GADSS unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	8	**SLMBE unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SLMBE unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	7	**SFBEunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SFBEunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	6	**LNEunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS LNEunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	5	**LNIunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS LNIunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	4	**RCPBunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS RCPBunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for

SCCGCTL94D8 - Slice unit Level Clock Gating Control 94D8

		functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	3	**RCPBEunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS RCPBEunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	2	**TDCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS TDCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	1	**RAMdft Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS RAMdft Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	0	**Reserved**

Slice unit Level Clock Gating Control 94DC

SCCGCTL94DC - Slice unit Level Clock Gating Control 94DC

Register Space: MMIO: 0/2/0
Size (in bits): 32
CustomGTIIsContextSaved: true

Address: 094DCh

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31:15	**Reserved** Access: RO _Custom_GTIReset: BUS Reserved
	14	**Reserved** Access: RO Format: MBZ
	13:0	**Reserved** Access: RO _Custom_GTIReset: BUS Reserved

Slice unit Level Clock Gating Control 94E0

SCCGCTL94E0 - Slice unit Level Clock Gating Control 94E0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	094E0h	
Unit Level Clock Gating Disable bits		
DWord	**Bit**	**Description**
0	31	**Reserved** Access: RO Format: MBZ
	30	**PSS Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS PSS Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	29	**VSR Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VSR Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	28	**VS Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VS Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	27	**VFR Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VFR Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

SCCGCTL94E0 - Slice unit Level Clock Gating Control 94E0

26	**VF Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VF Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
25	**TDS Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS TDS Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
24	**SVGR Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SVGR Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
23	**SVG Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SVG Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
22	**SOL Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SOL Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
21:20	**Reserved** Access: RO Format: MBZ
19	**OVR Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS

SCCGCTL94E0 - Slice unit Level Clock Gating Control 94E0

		OVR Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	18	**HS Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HS Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	17	**GS Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GS Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	16	**CS Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS CS Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	15	**SPARE Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SPARE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	14	**CPSSunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS CPSSunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

SCCGCTL94E0 - Slice unit Level Clock Gating Control 94E0

	13	**PSDunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS PSDunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	12	**BCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS BCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	11	**MSCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS MSCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	10	**RCPFEunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS RCPFEunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	9	**AVSunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS AVSunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	8	**Reserved** Access: RO Format: MBZ

SCCGCTL94E0 - Slice unit Level Clock Gating Control 94E0

	7	**HIZunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HIZunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	6	**IZunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS IZunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	5	**RCCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS RCCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	4	**RCZunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS RCZunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	3	**SBEunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SBEunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	2	**STCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS STCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)

SCCGCTL94E0 - Slice unit Level Clock Gating Control 94E0

		'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	1	**SVLunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SVLunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	0	**WMFEunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS WMFEunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

Slice unit Level Clock Gating Control 94E4

SCCGCTL94E4 - Slice unit Level Clock Gating Control 94E4

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	094E4h

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29	**CCS Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS CCS Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	28	**sdfifo Clock Gating Disable svg** Access: R/W Format: Disable _Custom_GTIReset: BUS sdfifo Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	27	**sdfifo Clock Gating Disable svl** Access: R/W Format: Disable _Custom_GTIReset: BUS sdfifo Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	26	**PSS unit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS PSS Clock Gating Disable Control:

SCCGCTL94E4 - Slice unit Level Clock Gating Control 94E4

		'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	25	**CARB unit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS CARB Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	24	**ZPBE unit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS ZPBE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	23	**ZSCBANK unit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS ZSCBANK Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	22	**ZSCCOMPUTE unit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS ZSCCOMPUTE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	21	**GAFARB unit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS GAFARB Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for

SCCGCTL94E4 - Slice unit Level Clock Gating Control 94E4

		functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	20	**VSR unit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS VSR Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	19	**VS unit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS VS Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	18	**VFR unit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS VFR Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	17	**VFE Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS VFE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	16	**VF unit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS VF Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)

SCCGCTL94E4 - Slice unit Level Clock Gating Control 94E4

	'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
15	**URBM unit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS URBM Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
14	**TSG unit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS TSG Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
13	**TETG unit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS TETG Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
12	**TE unit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS TE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
11	**TDS unit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS TDS Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

SCCGCTL94E4 - Slice unit Level Clock Gating Control 94E4

10	**TDG unit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS TDG Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
9	**SVGR unit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS SVGR Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
8	**SVG unit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS SVG Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
7	**SOL unit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS SOL Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
6	**gwunit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS gwunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

SCCGCTL94E4 - Slice unit Level Clock Gating Control 94E4

5	**psdunit Clock Gating Disable**

Access:	R/W
Format:	Disable
_Custom_GTIReset:	BUS

psdunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

4	**hdcunit Clock Gating Disable**

Access:	R/W
Format:	Disable
_Custom_GTIReset:	BUS

hdcunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

3	**cpssunit Clock Gating Disable**

Access:	R/W
Format:	Disable
_Custom_GTIReset:	BUS

cpssunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

2	**besbufunit Clock Gating Disable**

Access:	R/W
Format:	Disable
_Custom_GTIReset:	BUS

besbufunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

1	**BC Clock Gating Disable**

Access:	R/W
Format:	Disable
_Custom_GTIReset:	BUS

BC Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

SCCGCTL94E4 - Slice unit Level Clock Gating Control 94E4		
	0	**tdpunit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS CCS Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

Slice unit Level Clock Gating override during rstflow 94F0

SCMISCCP94F0 - Slice unit Level Clock Gating override during rstflow 94F0		
Register Space: MMIO: 0/2/0 Size (in bits): 32		
Address: 094F0h		
Unit Level Clock Gating Disable bits		
DWord	**Bit**	**Description**
0	31:0	**ECO Spare Bits** Access: R/W _Custom_GTIReset: BUS

SLM Bank Hash

SLM_BANKHASH - SLM Bank Hash

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	0E660h
Name:	SLM Bank Hash
ShortName:	SLM_BANKHASH

Description
Register for selecting number of SLM banks and setting XOR hashing in bank select bits.
Restriction : To avoid data corruption, this register can only be modified when no threads are running that access SLM. Data corruption can also occur if a mid-thread preemption image with saved SLM data is restored with a different setting.
Restriction : This register is a privileged register (not user-mode non-privileged) to ensure that one context changing it does not corrupt a simultaneously running context.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15	**64 Bank Mode** Format: U1 This bit selects the number of logical banks in SLM (64 or 65). Value 1h: B64 — 64 logical banks in SLM. Value 0h: B65 **[Default]** — 65 logical banks in SLM.
	14:12	**Reserved** Access: RO Format: MBZ
	11:10	**bit7 Hash** Access: R/W Format: U2 Programming this field is only valid when Bit[15] is set to 1 (64-bank mode). This field defines which address bit(s) will be XOR-ed with address bit[7] to produce new address bit[7]. Multiple bits can be set to XOR multiple address bit with bit[7]. Value 2h: A10 — When set, XOR address[7] with address[10] to produce new address[7]. Value 1h: A13 — When set, XOR address[7] with address[13] to produce new

SLM_BANKHASH - SLM Bank Hash

		address[7].
0h	No XOR **[Default]**	Address[7] is not XOR-ed with any other address bit.

9:8 **bit6 Hash**

Access:	R/W
Format:	U2

Programming this field is only valid when Bit[15] is set to 1 (64-bank mode). This field defines which address bit(s) will be XOR-ed with address bit[6] to produce new address bit[6]. Multiple bits can be set to XOR multiple address bit with bit[6].

Value	Name	Description
2h	A9	When set, XOR address[6] with address[9] to produce new address[6].
1h	A12	When set, XOR address[6] with address[12] to produce new address[6].
0h	No XOR **[Default]**	Address[6] is not XOR-ed with any other address bit.

7:6 **bit5 Hash**

Access:	R/W
Format:	U2

Programming this field is only valid when Bit[15] is set to 1 (64-bank mode). This field defines which address bit(s) will be XOR-ed with address bit[5] to produce new address bit[5]. Multiple bits can be set to XOR multiple address bit with bit[5].

Value	Name	Description
2h	A8	When set, XOR address[5] with address[8] to produce new address[5].
1h	A11	When set, XOR address[5] with address[11] to produce new address[5].
0h	No XOR **[Default]**	Address[5] is not XOR-ed with any other address bit.

5:4 **bit4 Hash**

Access:	R/W
Format:	U2

Programming this field is only valid when Bit[15] is set to 1 (64-bank mode). This field defines which address bit(s) will be XOR-ed with address bit[4] to produce new address bit[4]. Multiple bits can be set to XOR multiple address bit with bit[4].

Value	Name	Description
2h	A16	When set, XOR address[4] with address[16] to produce new address[4].
1h	A10	When set, XOR address[4] with address[10] to produce new address[4].

SLM_BANKHASH - SLM Bank Hash

0h	No XOR **[Default]**	Address[4] is not XOR-ed with any other address bit.

3:2 **bit3 Hash**

Access:	R/W
Format:	U2

Programming this field is only valid when Bit[15] is set to 1 (64-bank mode). This field defines which address bit(s) will be XOR-ed with address bit[3] to produce new address bit[3]. Multiple bits can be set to XOR multiple address bit with bit[3].

Value	Name	Description
2h	A15	When set, XOR address[3] with address[15] to produce new address[3].
1h	A9	When set, XOR address[3] with address[9] to produce new address[3].
0h	No XOR **[Default]**	Address[3] is not XOR-ed with any other address bit.

1:0 **bit2 Hash**

Access:	R/W
Format:	U2

Programming this field is only valid when Bit[15] is set to 1 (64-bank mode). This field defines which address bit(s) will be XOR-ed with address bit[2] to produce new address bit[2]. Multiple bits can be set to XOR multiple address bit with bit[2].

Value	Name	Description
2h	A14	When set, XOR address[2] with address[14] to produce new address[2].
1h	A8	When set, XOR address[2] with address[8] to produce new address[2].
0h	No XOR **[Default]**	Address[2] is not XOR-ed with any other address bit.

SNP_FILTER_Q_STATUS

SNP_FILTER_Q_STATUS - SNP_FILTER_Q_STATUS	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	00B00h
Name:	SNP_FILTER_Q_STATUS
ShortName:	SNP_FILTER_Q_STATUS

This is a basic register template

DWord	Bit	Description
0	31	**Reserved**
	30:29	**SF_FUSE_HASH_3** Access: RO _Custom_GTIReset: BUS This bit captures the fuse_idi_hash[3:2] for GuC to use
	28:20	**Reserved** Access: RO Format: MBZ
	19	**Reserved**
	18	**Reserved**
	17	**Reserved**
	16	**Reserved**
	15	**Reserved**
	14	**Reserved**
	13	**Reserved**
	12	**Reserved**
	11	**Reserved**
	10	**Reserved**
	9	**Reserved**
	8	**Reserved**
	7	**Reserved**
	6	**Reserved**
	5	**Reserved**
	4	**Reserved**
	3	**Reserved**
	2	**Reserved**
	1	**Reserved**
	0	**Reserved**

Soft Scratch

SOFT_SCRATCH - Soft Scratch		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	512	
These are 16 dword software scratchpad dword registers defined for use.		
Programming Notes		
These registers are saved in the power context		
DWord	**Bit**	**Description**
0..15	511:0	**Scratch**

Software SCI

SWSCI_0_2_0_PCI - Software SCI

Register Space:	PCI: 0/2/0
Size (in bits):	16
Address:	000E8h

This register serves 2 purposes:
1) Support selection of SMI or SCI event source (SMISCISEL - bit15)
2) SCI Event trigger (GSSCIE - bit 0). To generate a SW SCI event, software should program bit 15 (SMISCISEL) to 1. This is typically programmed once (assuming SMIs are never triggered). On a "0" to "1" subsequent transition in bit 0 of this register (caused by a software write operation), a SCI message will be sent to cause the TCOSCI_STS bit in GPE0 register to be set to 1. The corresponding SCI event handler in BIOS is to be defined as a _Lxx method, indicating level trigger to the operating system. Once written as 1, software must write a "0" to this bit to clear it, and all other write transitions (1-0, 0-0, 1-1) will not cause a SCI message to be sent. To generate a SW SMI event, software should program bit 15 to 0 and trigger SMI via writes to SWSMI register (See SWSMI register for programming details).

DWord	Bit	Description
0	15	**SMI or SCI event select** Default Value: 0b Access: R/W _Custom_GTIReset: BUS 0 = SMI (default)1 = SCI If selected event source is SMI, SMI trigger and associated scratch bits accesses are performed via SWSMI register. If SCI event source is selected, the rest of the bits in this register provide SCI trigger capability and associated SW scratch pad area.
	14:1	**Software scratch bits** Default Value: 00000000000000b Access: R/W _Custom_GTIReset: BUS Read/write bits not used by hardware.
	0	**Software SCI Event** Default Value: 0b Access: R/W _Custom_GTIReset: BUS If SCI event is selected (SMISCISEL = 1), on a 0 to 1 transition of GSSCIE bit, a SCI message will be sent to cause the TCOSCI_STS bit in GPE0 register to be set to 1. Software must write a 0 to clear this bit.

Software SMI

SWSMI_0_2_0_PCI - Software SMI

Register Space: PCI: 0/2/0
Size (in bits): 16

Address: 000E0h

DWord	Bit	Description
0	15:8	**Software Scratch Bits** Default Value: 00000000b Access: R/W _Custom_GTIReset: BUS
	7:1	**Software Flag** Default Value: 0000000b Access: R/W _Custom_GTIReset: BUS Used to indicate caller and SMI function desired, as well as return result.
	0	**GMCH Software SMI Event** Default Value: 0b Access: R/W _Custom_GTIReset: BUS When Set this bit will trigger an SMI. Software must write a "0" to clear this bit.SMI will be triggered only if SWSCI[SMISCISEL] is set to select SMI.

SRD_CTL

SRD_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60800h-60803h
Name:	Transcoder SRD Control
ShortName:	SRD_CTL_A
Reset:	soft
Address:	61800h-61803h
Name:	Transcoder SRD Control
ShortName:	SRD_CTL_B
Reset:	soft
Address:	62800h-62803h
Name:	Transcoder SRD Control
ShortName:	SRD_CTL_C
Reset:	soft
Address:	63800h-63803h
Name:	Transcoder SRD Control
ShortName:	SRD_CTL_D
Reset:	soft

Restriction : PSR needs to be enabled only when at least one plane is enabled.

Programming Notes

To use FBC modification tracking for idleness calculations when FBC is disabled, program FBC_CTL CPU Fence Enable, FBC_CONTROL_SA_REGISTER, FBC_CPU_FENCE_OFFSET_REGISTER, FBC_RT_BASE_ADDR_REGISTER, and BLITTER_TRACKING_REGISTER as they are programmed when FBC is enabled.

Cursor front buffer modifications are not tracked in hardware. If the cursor front buffer is modified, touch (write without changing) any cursor register to trigger the PSR idleness tracking.

Restriction

Only the SRD Enable and Single Frame Update Enable fields can be changed while SRD is enabled. The other fields must not be changed while SRD is enabled.

DWord	Bit	Description
0	31	**SRD Enable** This bit enables the Self Refreshing Display function. Updates will take place at the start of the next vertical blank. The port will send SRD VDMs while enabled. When idleness conditions have been met for the programmed number of idle frames, hardware will enter SRD (sleep) and can disable the link and stop fetching data from memory. When activity occurs, hardware will exit

SRD_CTL

SRD (wake) and re-enable the link and resume fetching data from memory.

Value	Name
0b	Disable
1b	Enable

Restriction
SRD must not be enabled when the PSR Setup time from DPCD 00071h is greater than the time for vertical blank minus one line.
SRD must not be enabled together with Interlacing, Black Frame Insertion (BFI), or audio on the same transcoder.

30 **Single Frame Update Enable**

Access:	Double Buffered

This field enables the single frame update mode where a plane flip will cause a single frame to be sent to the receiver. Updates to this field will take effect at the next vertical blank.

Value	Name
0b	Disable
1b	Enable

Programming Notes
Set register PIPE_MISC field Change Mask for Vblank Vsync Int to 1b (Masked) if vblank or vsync interrupts will be used together with single frame update.

Workaround
When Single Frame Update is enabled, the CRC must be disabled for panel compatibility.

Restriction
This mode should only be enabled with link standby.

29 **Context restore to PSR Active**

This field restores eDP context to PSR Active on a context restore.

Value	Name
0b	Disable
1b	Enable

Restriction
This field is used for hardware communication. Software must not change this field.

28 **Adaptive Sync Frame Update**

This field enables the Adaptive Sync Frame Update mode where a flip will cause a single frame to be sent to the receiver. Updates to this field will take effect at the next vertical blank. This field must be enabled with VRR enable.

SRD_CTL

Restriction : This mode should only be enabled with the SRD Link Disable mode. This mode does not support VRR Max Shift. However, normal and flipline VRR modes are supported.

Value	Name
0b	Disable
1b	Enable

Programming Notes
Set register PIPE_MISC field **Change Mask for Vblank Vsync Int** to 1b (Masked) if vblank or vsync interrupts will be used together with single frame update.

27 **Link Ctrl**

This field controls the behavior of the link when in SRD (sleeping). The timing generator and pixel data fetches are disabled when the link is disabled. Only pixel data fetches are disabled when the link is in standby.

This field is ignored by transcoder A/B/C since they only operate in standby.

Value	Name	Description
0b	Disable	Link is disabled when in SRD (sleeping)

26:25 **Reserved**

Access:	RO
Format:	MBZ

24:20 **Max Sleep Time**

Default Value:	00001b 1/8 second

This field is the maximum time to spend in SRD (sleeping). It is programmed in increments of approximately 1/8 a second. Programming all 1s gives ~3.875 seconds.

Restriction
Programming all 0s is invalid.

19:17 **Reserved**

Access:	RO
Format:	MBZ

16:14 **DSC CRC Last Byte**

This field selects the valid last byte of DSC data stream for SRD CRC calculation. Valid data byte for CRC engine can be from 0 to 5.

DSC CRC Last Byte = (Slice Per Line eDP* Bytes PerChunk * VActive) mod 6

Below is an example with 2 slices per line on each DSC.

SPL = Slices Per Line

Case	DSC A SPL	DSC C SPL	Slices Per Line eDP
CoG enable + VDSC enable (no Joiner)	2	2	2 (DSC A or DSC C)
VDSC enable + Joiner enable (no CoG)	2	2	4 (DSC A + DSC C)
VDSC standalone	0 or 2	2 or 0	2

SRD_CTL

	(no Joiner, no CoG)			

13 **Reserved**

12 **Reserved**

11 **TP2 TP3 Select**

This field controls whether TP1 is followed by TP2 or TP3 for training the link on exiting SRD (waking).

Value	Name	Description
0b	TP2	Use TP1 followed by TP2
1b	TP3	Use TP1 followed by TP3

Programming Notes
This bit impacts PSR2. Clear it before enabling PSR2 and do not set it while PSR2 is enabled.

10 **CRC Enable**

This field controls whether the PSR CRC value will be placed in the VSC packet.

Value	Name	Description
0b	Disable	Disable CRC output in VSC. VSC packet CRC value will be populated by VIDEO_DIP_DATA.
1b	Enable	Enable CRC output in VSC. VSC packet CRC value will be populated by the calculated CRC value.

Programming Notes
When CRC is enabled, the Max Sleep Timer should be disabled to provide additional power savings. Disable the Max Sleep Timer by setting register 0x6F860 bit 28 to 1. Re-enable the Max Sleep Timer by clearing register 0x6F860 bit 28 to 0.

Workaround
When Single Frame Update is enabled, the CRC must be disabled for panel compatibility.

9:8 **TP2 TP3 Time**

This field selects the TP2 or TP3 time when training the link on exiting SRD (waking).

Value	Name
00b	500us
01b	100us
10b	2.5ms
11b	0us Skip TP2/TP3

7:6 **TP4 time**

This field selects the TP4 time when training the link on exiting SRD (waking).
If this field is set to any value other than "11", TP4 pattern will be sent at PSR reentry.

Value	Name	Description
00b	500 us	

SRD_CTL

01b	100 us	
10b	2.5 ms	
11b	0 us	Skip TP4

Programming Notes
Always program TP4 to 11b.

5:4 **TP1 Time**

This field selects the TP1 time when training the link on exiting SRD (waking).

Value	Name	Description
00b	500us	
01b	100us	
10b	2.5ms	
11b	0us	Skip TP1

3:0 **Idle Frames**

Default Value:	0001b 1 idle frame

This field is the number of idle frames required before entering SRD (sleeping).

SRD_PERF_CNT

SRD_PERF_CNT	
Register Space:	MMIO: 0/2/0
Access:	Write/Read Status
Size (in bits):	32
Address:	60844h-60847h
Name:	Transcoder SRD Performance Counter
ShortName:	SRD_PERF_CNT_A
Reset:	soft
Address:	61844h-61847h
Name:	Transcoder SRD Performance Counter
ShortName:	SRD_PERF_CNT_B
Reset:	soft
Address:	62844h-62847h
Name:	Transcoder SRD Performance Counter
ShortName:	SRD_PERF_CNT_C
Reset:	soft
Address:	63844h-63847h
Name:	Transcoder SRD Performance Counter
ShortName:	SRD_PERF_CNT_D
Reset:	soft

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23:0	**SRD Perf Cnt** This field increments every millisecond while in SRD (sleeping) and the display CD clock is running. It will stop incrementing when out of SRD (awake), then resume when back in SRD (sleeping). The value is maintained while SRD is disabled, and counting will resume from the previous value when SRD is re-enabled. Writes to this register will set the count to the written value, then it will increment from that value onwards.

SRD_STATUS

SRD_STATUS

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	60840h-60843h
Name:	Transcoder SRD Status
ShortName:	SRD_STATUS_A
Reset:	soft
Address:	61840h-61843h
Name:	Transcoder SRD Status
ShortName:	SRD_STATUS_B
Reset:	soft
Address:	62840h-62843h
Name:	Transcoder SRD Status
ShortName:	SRD_STATUS_C
Reset:	soft
Address:	63840h-63843h
Name:	Transcoder SRD Status
ShortName:	SRD_STATUS_D
Reset:	soft

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="2">0</td><td>31:29</td><td>SRD State
<table>
<tr><td>Access:</td><td>RO</td></tr>
</table>
This field indicates the live state of SRD
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>000b</td><td>IDLE</td><td>Reset state</td></tr>
<tr><td>001b</td><td>SRDONACK</td><td>Wait for TG/Stream to send on frame of data after SRD conditions are met</td></tr>
<tr><td>010b</td><td>SRDENT</td><td>SRD entry with Link OFF</td></tr>
<tr><td>011b</td><td>BUFOFF</td><td>Wait for buffer turn off</td></tr>
<tr><td>100b</td><td>BUFON</td><td>Wait for buffer turn on</td></tr>
<tr><td>101b</td><td>AUXACK</td><td>Wait for AUX to acknowledge on SRD exit</td></tr>
<tr><td>110b</td><td>SRDOFFACK</td><td>Wait for TG/Stream to acknowledge the SRD VDM exit</td></tr>
<tr><td>111b</td><td>SRDENT_ON</td><td>SRD entry with Link ON</td></tr>
<tr><td>Others</td><td>Reserved</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>28</td><td>Reserved</td></tr>
</table>

SRD_STATUS

Access:	RO
Format:	MBZ

27:26 Link Status

Access:	RO

This field indicates the live status of the link.

Value	Name	Description
00b	Full Off	Link is fully off
01b	Full On	Link is fully on
11b	Reserved	Reserved

25 Reserved

Access:	RO
Format:	MBZ

24:20 Max Sleep Time Counter

Access:	RO

This field provides the live status of the sleep time counter.

19:16 SRD Entry Count

Access:	RO

The value in this register represents the number of times SRD has been entered (gone to sleep). The count will increment with each entry. After reaching the maximum count value the counter will rollover and continue from 0.

15 Aux Error

Access:	RO

This field indicates an error on the last SRD AUX handshake.

Value	Name	Description
0b	No Error	AUX had no error
1b	Error	AUX error (receive error or timeout) occurred

14:13 Reserved

Access:	RO
Format:	MBZ

12 Sending Aux

Access:	RO

This field indicates if the SRD AUX handshake is currently being sent.

Value	Name	Description
0b	Not Sending	Not sending AUX handshake
1b	Sending	Sending AUX handshake

SRD_STATUS

11:10 Reserved

Access:	RO
Format:	MBZ

9 Sending Idle

Access:	RO

This field indicates if idles are currently being sent.

Value	Name	Description
0b	Not Sending	Not sending idle
1b	Sending	Sending idle

8 Sending TP2 TP3

Access:	RO

This field indicates if TP2 or TP3 is currently being sent.

Value	Name	Description
0b	Not Sending	Not sending TP2 or TP3
1b	Sending	Sending TP2 or TP3

7 Sending TP4

Access:	RO

This field indicates if TP4 is currently being sent.

Value	Name
0b	Not Sending
1b	Sending

6:5 Reserved

Access:	RO
Format:	MBZ

4 Sending TP1

Access:	RO

This field indicates if TP1 is currently being sent.

Value	Name	Description
0b	Not Sending	Not sending TP1
1b	Sending	Sending TP1

3:0 Idle Frame Counter

Access:	RO

This field provides the live status of the idle frame counter.

Programming Notes
The value of this field is not preserved across power down states such as DC5 and up.

SSM0 BONUS1 Reg

SSM0SPCBONUS1 - SSM0 BONUS1 Reg

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24414h
Clock Gating Messages Register	

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**BONUS1 BIT 7** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	6	**BONUS1 BIT 6** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	5	**BONUS1 BIT 5** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	4	**BONUS1 BIT 4** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	3	**BONUS1 BIT 3** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe)

SSM0SPCBONUS1 - SSM0 BONUS1 Reg

		'1' : Initiate power up sequence (clk/rst/fwe)
	2	**BONUS1 BIT 2** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	1	**BONUS1 BIT 1** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**BONUS1 BIT 0** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

SSM0 BONUS2 Reg

SSM0SPCBONUS2 - SSM0 BONUS2 Reg

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24418h
Clock Gating Messages Register	

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**BONUS2 BIT 7** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	6	**BONUS2 BIT 6** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	5	**BONUS2 BIT 5** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	4	**BONUS2 BIT 4** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	3	**BONUS2 BIT 3** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe)

SSM0SPCBONUS2 - SSM0 BONUS2 Reg

		'1' : Initiate power up sequence (clk/rst/fwe)
	2	**BONUS2 BIT 2** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	1	**BONUS2 BIT 1** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**BONUS2 BIT 0** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

SSM1 BONUS1 Reg

SSM1SPCBONUS1 - SSM1 BONUS1 Reg

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24494h
Clock Gating Messages Register	

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**BONUS1 BIT 7** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	6	**BONUS1 BIT 6** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	5	**BONUS1 BIT 5** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	4	**BONUS1 BIT 4** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	3	**BONUS1 BIT 3** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe)

SSM1SPCBONUS1 - SSM1 BONUS1 Reg

		'1' : Initiate power up sequence (clk/rst/fwe)
	2	**BONUS1 BIT 2** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	1	**BONUS1 BIT 1** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**BONUS1 BIT 0** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

SSM1 BONUS2 Reg

SSM1SPCBONUS2 - SSM1 BONUS2 Reg

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24498h
Clock Gating Messages Register	

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**BONUS2 BIT 7** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	6	**BONUS2 BIT 6** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	5	**BONUS2 BIT 5** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	4	**BONUS2 BIT 4** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	3	**BONUS2 BIT 3** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe)

SSM1SPCBONUS2 - SSM1 BONUS2 Reg

		'1' : Initiate power up sequence (clk/rst/fwe)
	2	**BONUS2 BIT 2** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	1	**BONUS2 BIT 1** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**BONUS2 BIT 0** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

SSM2 BONUS1 Reg

SSM2SPCBONUS1 - SSM2 BONUS1 Reg

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24514h
Clock Gating Messages Register	

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**BONUS1 BIT 7** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	6	**BONUS1 BIT 6** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	5	**BONUS1 BIT 5** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	4	**BONUS1 BIT 4** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	3	**BONUS1 BIT 3** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe)

SSM2SPCBONUS1 - SSM2 BONUS1 Reg

		'1' : Initiate power up sequence (clk/rst/fwe)
	2	**BONUS1 BIT 2** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	1	**BONUS1 BIT 1** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS1 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**BONUS1 BIT 0** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

SSM2 BONUS2 Reg

SSM2SPCBONUS2 - SSM2 BONUS2 Reg

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 24518h

Clock Gating Messages Register

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**BONUS2 BIT 7** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	6	**BONUS2 BIT 6** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	5	**BONUS2 BIT 5** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	4	**BONUS2 BIT 4** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	3	**BONUS2 BIT 3** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe)

SSM2SPCBONUS2 - SSM2 BONUS2 Reg

		'1' : Initiate power up sequence (clk/rst/fwe)
	2	**BONUS2 BIT 2** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req
	1	**BONUS2 BIT 1** Access: R/W _Custom_GTIReset: BUS SLICE 0 BONUS2 BIT: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**BONUS2 BIT 0** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

ssmcp Sub Slice Power Context Save request

SSMPCTXSAVEREQ - ssmcp Sub Slice Power Context Save request

Register Space: MMIO: 0/2/0
Size (in bits): 32
CustomGTIIs_ContextSaved: false

Address: 08150h

DWord	Bit	Description
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask bots for lower 16 bits
	15:10	**Reserved** Access: RO Format: MBZ
	9	**Power context save request** Access: R/W Set _Custom_GTIReset: BUS Power Context Save Request 1'b0 : Power context save is not being requested 1'b1 : Power context save is being requested CPUnit self-clears this bit upon sampling.
	8:0	**Power Context Save request crdit count** Access: R/W _Custom_GTIReset: BUS QWord Credits for Power Context Save Request Minimum Credits = 1 : Unit may send 1 QWord pair (enough for first LRI at least) Maximum Credits = 511 : Unit may send 511 QWord pairs A QWord pair is defined as a 32-bit register address and the corresponding 32-bits of register data. Note that the LRI header and END commands are 64-bits each (32-bit command followed by 32-bit NOOP) and will consume one QWord credit. Only valid with PWRCTX_SAVE_REQ (Bit9).

Stream Output 0 Num Primitives Written Counter

SO0_NUM_PRIMS_WRITTEN - Stream Output 0 Num Primitives Written Counter

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
Address:	05200h-05207h
Name:	Stream Output 0 Num Primitives Written Counter
ShortName:	SO0_NUM_PRIMS_WRITTEN

There is one 64-bit register for each of the 4 supported streams:

- 5200h-5207h SO_NUM_PRIMS_WRITTEN0 (for Stream Out Stream #0)
- 5208h-520Fh SO_NUM_PRIMS_WRITTEN1 (for Stream Out Stream #1)
- 5210h-5217h SO_NUM_PRIMS_WRITTEN2 (for Stream Out Stream #2)
- 5218h-521Fh SO_NUM_PRIMS_WRITTEN3 (for Stream Out Stream #3)

These registers are used to count the number of primitives (aka objects: points, lines, triangles) which the SO stage has successfully written to a particular "stream's" Streamed Vertex Output buffers, subject to buffer overflow detection. (See the Stream Output section of the 3D pipeline volume).
These registers are part of the context save and restore.

DWord	Bit	Description
0	31:0	**Num Prims Written Count 0** Format: U32 _Custom_GTIReset: DEV This count is incremented (by one) every time a GS thread outputs a DataPort Streamed Vertex Buffer Write message with the Increment Num Prims Written bit set in the message header (see the Geometry Shader and Data Port chapters in the 3D Volume.)
1	31:0	**Num Prims Written Count 1** Format: U32 _Custom_GTIReset: DEV This count is incremented (by one) every time a GS thread outputs a DataPort Streamed Vertex Buffer Write message with the Increment Num Prims Written bit set in the message header (see the Geometry Shader and Data Port chapters in the 3D Volume.)

Stream Output 0 Primitive Storage Needed Counter

SO0_PRIM_STORAGE_NEEDED - Stream Output 0 Primitive Storage Needed Counter

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64

Address:	05240h-05247h
Name:	Stream Output 0 Primitive Storage Needed Counter
ShortName:	SO0_PRIM_STORAGE_NEEDED

There is one 64-bit register for each of the 4 supported streams:

- 5240h-5247h SO_PRIM_STORAGE_NEEDED0 (for Stream Out Stream #0)
- 5248h-524Fh SO_PRIM_STORAGE_NEEDED1 (for Stream Out Stream #1)
- 5250h-5257h SO_PRIM_STORAGE_NEEDED2 (for Stream Out Stream #2)
- 5258h-525Fh SO_PRIM_STORAGE_NEEDED3 (for Stream Out Stream #3)

These registers are used to count the number of primitives (aka objects: points, lines, triangles) which the SO stage has or would have written to a particular "stream's" Streamed Vertex Output buffers if all buffers had been large enough to accommodate the writes. (See the Stream Output section of the 3D pipeline volume).
These registers are part of the context save and restore.

DWord	Bit	Description
0	31:0	**Prim Storage Needed Count 0** Format: U32 _Custom_GTIReset: DEV This count is incremented (by one) by the SOL stage for each object (point, line, triangle) it writes or attempts to write to the corresponding stream's output buffers. The count is not affected by the actual number of buffers bound to the stream.
1	31:0	**Prim Storage Needed Count 1** Format: U32 _Custom_GTIReset: DEV This count is incremented (by one) by the SOL stage for each object (point, line, triangle) it writes or attempts to write to the corresponding stream's output buffers. The count is not affected by the actual number of buffers bound to the stream.

Stream Output 0 Write Offset

SO0_WRITE_OFFSET - Stream Output 0 Write Offset

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	05280h-05283h
Name:	Stream Output 0 Write Offset
ShortName:	SO0_WRITE_OFFSET

There is one R/W 32-bit register for each of the 4 supported stream output buffer slots:
5280h-5283h SO_WRITE_OFFSET0 (for Stream Out Buffer #0)
5284h-5287h SO_WRITE_OFFSET1 (for Stream Out Buffer #1)
5288h-528Bh SO_WRITE_OFFSET2 (for Stream Out Buffer #2)
528Ch-528Fh SO_WRITE_OFFSET3 (for Stream Out Buffer #3)
These registers are used to set and track a DWord-granular Write Offset for each of the 4 Stream Output Buffer slots. Software can directly write them via MI_LOAD_REGxxx commands. The SOL stage will increment them as part of stream output processing. Software can cause them to be written to memory via MI_STORE_REGxxx commands. (See the Stream Output section of the 3D pipeline volume).
These registers are part of the context save and restore.

Programming Notes

- Software must ensure that no HW stream output operations can be in process or otherwise pending at the point that the MI_LOAD/STORE commands are processed. This will likely require a pipeline flush.
- The SOL stage will effectively advance the write offset by the buffer's Surface Pitch after each vertex is written (assuming no overflow is detected in any targeted SO buffer). Under "normal" conditions one would expect software to initialize the WriteOffset to some (possibly zero) multiple of Surface Pitch in order to align vertex writes to the buffer's Base Address, though it is not required to do so.

DWord	Bit	Description
0	31:2	**Write Offset** Format: U30 _Custom_GTIReset: DEV This field contains a DWord offset from the corresponding SO buffer's Base Address value. The SOL stage uses this value as a write offset when performing writes to the buffer. The SOL stage will increment this value as a part of performing stream output to the buffer. Note that the SOL stage uses the buffer's Surface Pitch to advance the Write Offset, without regard to the buffer's Base Address (see Programming Notes above).
	1:0	**Reserved** Access: RO Format: MBZ

Stream Output 1 Num Primitives Written Counter

SO1_NUM_PRIMS_WRITTEN - Stream Output 1 Num Primitives Written Counter

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
Address:	05208h-0520Fh
Name:	Stream Output 1 Num Primitives Written Counter
ShortName:	SO1_NUM_PRIMS_WRITTEN

There is one 64-bit register for each of the 4 supported streams:
5200h-5207h SO_NUM_PRIMS_WRITTEN0 (for Stream Out Stream #0)
5208h-520Fh SO_NUM_PRIMS_WRITTEN1 (for Stream Out Stream #1)
5210h-5217h SO_NUM_PRIMS_WRITTEN2 (for Stream Out Stream #2)
5218h-521Fh SO_NUM_PRIMS_WRITTEN3 (for Stream Out Stream #3)
These registers are used to count the number of primitives (aka objects: points, lines, triangles) which the SO stage has successfully written to a particular "stream's" Streamed Vertex Output buffers, subject to buffer overflow detection. (See the Stream Output section of the 3D pipeline volume).
These registers are part of the context save and restore.

DWord	Bit	Description
0	31:0	**Num Prims Written Count 0** Format: U32 _Custom_GTIReset: DEV This count is incremented (by one) every time a GS thread outputs a DataPort Streamed Vertex Buffer Write message with the Increment Num Prims Written bit set in the message header (see the Geometry Shader and Data Port chapters in the 3D Volume.)
1	31:0	**Num Prims Written Count 1** Format: U32 _Custom_GTIReset: DEV This count is incremented (by one) every time a GS thread outputs a DataPort Streamed Vertex Buffer Write message with the Increment Num Prims Written bit set in the message header (see the Geometry Shader and Data Port chapters in the 3D Volume.)

Stream Output 1 Primitive Storage Needed Counter

SO1_PRIM_STORAGE_NEEDED - Stream Output 1 Primitive Storage Needed Counter

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
Address:	05248h-0524Fh
Name:	Stream Output 1 Primitive Storage Needed Counter
ShortName:	SO1_PRIM_STORAGE_NEEDED

There is one 64-bit register for each of the 4 supported streams:
5240h-5247h SO_PRIM_STORAGE_NEEDED0 (for Stream Out Stream #0)
5248h-524Fh SO_PRIM_STORAGE_NEEDED1 (for Stream Out Stream #1)
5250h-5257h SO_PRIM_STORAGE_NEEDED2 (for Stream Out Stream #2)
5258h-525Fh SO_PRIM_STORAGE_NEEDED3 (for Stream Out Stream #3)
These registers are used to count the number of primitives (aka objects: points, lines, triangles) which the SO stage has or would have written to a particular "stream's" Streamed Vertex Output buffers if all buffers had been large enough to accommodate the writes. (See the Stream Output section of the 3D pipeline volume).
These registers are part of the context save and restore.

DWord	Bit	Description
0	31:0	**Prim Storage Needed Count 0** Format: U32 _Custom_GTIReset: DEV This count is incremented (by one) by the SOL stage for each object (point, line, triangle) it writes or attempts to write to the corresponding stream's output buffers. The count is not affected by the actual number of buffers bound to the stream.
1	31:0	**Prim Storage Needed Count 1** Format: U32 _Custom_GTIReset: DEV This count is incremented (by one) by the SOL stage for each object (point, line, triangle) it writes or attempts to write to the corresponding stream's output buffers. The count is not affected by the actual number of buffers bound to the stream.

Stream Output 1 Write Offset

SO1_WRITE_OFFSET - Stream Output 1 Write Offset

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	05284h-05287h
Name:	Stream Output 1 Write Offset
ShortName:	SO1_WRITE_OFFSET

There is one R/W 32-bit register for each of the 4 supported stream output buffer slots:
5280h-5283h SO_WRITE_OFFSET0 (for Stream Out Buffer #0)
5284h-5287h SO_WRITE_OFFSET1 (for Stream Out Buffer #1)
5288h-528Bh SO_WRITE_OFFSET2 (for Stream Out Buffer #2)
528Ch-528Fh SO_WRITE_OFFSET3 (for Stream Out Buffer #3)
These registers are used to set and track a DWord-granular Write Offset for each of the 4 Stream Output Buffer slots. Software can directly write them via MI_LOAD_REGxxx commands. The SOL stage will increment them as part of stream output processing. Software can cause them to be written to memory via MI_STORE_REGxxx commands. (See the Stream Output section of the 3D pipeline volume).
These registers are part of the context save and restore.

Programming Notes

- Software must ensure that no HW stream output operations can be in process or otherwise pending at the point that the MI_LOAD/STORE commands are processed. This will likely require a pipeline flush.
- The SOL stage will effectively advance the write offset by the buffer's Surface Pitch after each vertex is written (assuming no overflow is detected in any targeted SO buffer). Under "normal" conditions one would expect software to initialize the WriteOffset to some (possibly zero) multiple of Surface Pitch in order to align vertex writes to the buffer's Base Address, though it is not required to do so.

DWord	Bit	Description
0	31:2	**Write Offset** Format: U30 _Custom_GTIReset: DEV This field contains a DWord offset from the corresponding SO buffer's Base Address value. The SOL stage uses this value as a write offset when performing writes to the buffer. The SOL stage will increment this value as a part of performing stream output to the buffer. Note that the SOL stage uses the buffer's Surface Pitch to advance the Write Offset, without regard to the buffer's Base Address (see Programming Notes above).
	1:0	**Reserved** Access: RO Format: MBZ

Stream Output 2 Num Primitives Written Counter

SO2_NUM_PRIMS_WRITTEN - Stream Output 2 Num Primitives Written Counter

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
Address:	05210h-05217h
Name:	Stream Output 2 Num Primitives Written Counter
ShortName:	SO2_NUM_PRIMS_WRITTEN

There is one 64-bit register for each of the 4 supported streams:
5200h-5207h SO_NUM_PRIMS_WRITTEN0 (for Stream Out Stream #0)
5208h-520Fh SO_NUM_PRIMS_WRITTEN1 (for Stream Out Stream #1)
5210h-5217h SO_NUM_PRIMS_WRITTEN2 (for Stream Out Stream #2)
5218h-521Fh SO_NUM_PRIMS_WRITTEN3 (for Stream Out Stream #3)
These registers are used to count the number of primitives (aka objects: points, lines, triangles) which the SO stage has successfully written to a particular "stream's" Streamed Vertex Output buffers, subject to buffer overflow detection. (See the Stream Output section of the 3D pipeline volume).
These registers are part of the context save and restore.

DWord	Bit	Description
0	31:0	**Num Prims Written Count 0** Format: U32 _Custom_GTIReset: DEV This count is incremented (by one) every time a GS thread outputs a DataPort Streamed Vertex Buffer Write message with the Increment Num Prims Written bit set in the message header (see the Geometry Shader and Data Port chapters in the 3D Volume.)
1	31:0	**Num Prims Written Count 1** Format: U32 _Custom_GTIReset: DEV This count is incremented (by one) every time a GS thread outputs a DataPort Streamed Vertex Buffer Write message with the Increment Num Prims Written bit set in the message header (see the Geometry Shader and Data Port chapters in the 3D Volume.)

Stream Output 2 Primitive Storage Needed Counter

SO2_PRIM_STORAGE_NEEDED - Stream Output 2 Primitive Storage Needed Counter

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
Address:	05250h-05257h
Name:	Stream Output 2 Primitive Storage Needed Counter
ShortName:	SO2_PRIM_STORAGE_NEEDED

There is one 64-bit register for each of the 4 supported streams:
5240h-5247h SO_PRIM_STORAGE_NEEDED0 (for Stream Out Stream #0)
5248h-524Fh SO_PRIM_STORAGE_NEEDED1 (for Stream Out Stream #1)
5250h-5257h SO_PRIM_STORAGE_NEEDED2 (for Stream Out Stream #2)
5258h-525Fh SO_PRIM_STORAGE_NEEDED3 (for Stream Out Stream #3)
These registers are used to count the number of primitives (aka objects: points, lines, triangles) which the SO stage has or would have written to a particular "stream's" Streamed Vertex Output buffers if all buffers had been large enough to accommodate the writes. (See the Stream Output section of the 3D pipeline volume).
These registers are part of the context save and restore.

DWord	Bit	Description
0	31:0	**Prim Storage Needed Count 0** Format: U32 _Custom_GTIReset: DEV This count is incremented (by one) by the SOL stage for each object (point, line, triangle) it writes or attempts to write to the corresponding stream's output buffers. The count is not affected by the actual number of buffers bound to the stream.
1	31:0	**Prim Storage Needed Count 1** Format: U32 _Custom_GTIReset: DEV This count is incremented (by one) by the SOL stage for each object (point, line, triangle) it writes or attempts to write to the corresponding stream's output buffers. The count is not affected by the actual number of buffers bound to the stream.

Stream Output 2 Write Offset

SO2_WRITE_OFFSET - Stream Output 2 Write Offset

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	05288h-0528Bh
Name:	Stream Output 2 Write Offset
ShortName:	SO2_WRITE_OFFSET

There is one R/W 32-bit register for each of the 4 supported stream output buffer slots:
5280h-5283h SO_WRITE_OFFSET0 (for Stream Out Buffer #0)
5284h-5287h SO_WRITE_OFFSET1 (for Stream Out Buffer #1)
5288h-528Bh SO_WRITE_OFFSET2 (for Stream Out Buffer #2)
528Ch-528Fh SO_WRITE_OFFSET3 (for Stream Out Buffer #3)
These registers are used to set and track a DWord-granular Write Offset for each of the 4 Stream Output Buffer slots. Software can directly write them via MI_LOAD_REGxxx commands. The SOL stage will increment them as part of stream output processing. Software can cause them to be written to memory via MI_STORE_REGxxx commands. (See the Stream Output section of the 3D pipeline volume).
These registers are part of the context save and restore.

Programming Notes

- Software must ensure that no HW stream output operations can be in process or otherwise pending at the point that the MI_LOAD/STORE commands are processed. This will likely require a pipeline flush.
- The SOL stage will effectively advance the write offset by the buffer's Surface Pitch after each vertex is written (assuming no overflow is detected in any targeted SO buffer). Under "normal" conditions one would expect software to initialize the WriteOffset to some (possibly zero) multiple of Surface Pitch in order to align vertex writes to the buffer's Base Address, though it is not required to do so.

DWord	Bit	Description
0	31:2	**Write Offset** Format: U30 _Custom_GTIReset: DEV This field contains a DWord offset from the corresponding SO buffer's Base Address value. The SOL stage uses this value as a write offset when performing writes to the buffer. The SOL stage will increment this value as a part of performing stream output to the buffer. Note that the SOL stage uses the buffer's Surface Pitch to advance the Write Offset, without regard to the buffer's Base Address (see Programming Notes above).
	1:0	**Reserved** Access: RO Format: MBZ

Stream Output 3 Num Primitives Written Counter

SO3_NUM_PRIMS_WRITTEN - Stream Output 3 Num Primitives Written Counter

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
Address:	05218h-0521Fh
Name:	Stream Output 3 Num Primitives Written Counter
ShortName:	SO3_NUM_PRIMS_WRITTEN

There is one 64-bit register for each of the 4 supported streams:
5200h-5207h SO_NUM_PRIMS_WRITTEN0 (for Stream Out Stream #0)
5208h-520Fh SO_NUM_PRIMS_WRITTEN1 (for Stream Out Stream #1)
5210h-5217h SO_NUM_PRIMS_WRITTEN2 (for Stream Out Stream #2)
5218h-521Fh SO_NUM_PRIMS_WRITTEN3 (for Stream Out Stream #3)
These registers are used to count the number of primitives (aka objects: points, lines, triangles) which the SO stage has successfully written to a particular "stream's" Streamed Vertex Output buffers, subject to buffer overflow detection. (See the Stream Output section of the 3D pipeline volume).
These registers are part of the context save and restore.

DWord	Bit	Description
0	31:0	**Num Prims Written Count 0** Format: U32 _Custom_GTIReset: DEV This count is incremented (by one) every time a GS thread outputs a DataPort Streamed Vertex Buffer Write message with the Increment Num Prims Written bit set in the message header (see the Geometry Shader and Data Port chapters in the 3D Volume.)
1	31:0	**Num Prims Written Count 1** Format: U32 _Custom_GTIReset: DEV This count is incremented (by one) every time a GS thread outputs a DataPort Streamed Vertex Buffer Write message with the Increment Num Prims Written bit set in the message header (see the Geometry Shader and Data Port chapters in the 3D Volume.)

Stream Output 3 Primitive Storage Needed Counter

SO3_PRIM_STORAGE_NEEDED - Stream Output 3 Primitive Storage Needed Counter

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
Address:	05258h-0525Fh
Name:	Stream Output 3 Primitive Storage Needed Counter
ShortName:	SO3_PRIM_STORAGE_NEEDED

There is one 64-bit register for each of the 4 supported streams:
5240h-5247h SO_PRIM_STORAGE_NEEDED0 (for Stream Out Stream #0)
5248h-524Fh SO_PRIM_STORAGE_NEEDED1 (for Stream Out Stream #1)
5250h-5257h SO_PRIM_STORAGE_NEEDED2 (for Stream Out Stream #2)
5258h-525Fh SO_PRIM_STORAGE_NEEDED3 (for Stream Out Stream #3)
These registers are used to count the number of primitives (aka objects: points, lines, triangles) which the SO stage has or would have written to a particular "stream's" Streamed Vertex Output buffers if all buffers had been large enough to accommodate the writes. (See the Stream Output section of the 3D pipeline volume).
These registers are part of the context save and restore.

DWord	Bit	Description
0	31:0	**Prim Storage Needed Count 0** Format: U32 _Custom_GTIReset: DEV This count is incremented (by one) by the SOL stage for each object (point, line, triangle) it writes or attempts to write to the corresponding stream's output buffers. The count is not affected by the actual number of buffers bound to the stream.
1	31:0	**Prim Storage Needed Count 1** Format: U32 _Custom_GTIReset: DEV This count is incremented (by one) by the SOL stage for each object (point, line, triangle) it writes or attempts to write to the corresponding stream's output buffers. The count is not affected by the actual number of buffers bound to the stream.

Stream Output 3 Write Offset

SO3_WRITE_OFFSET - Stream Output 3 Write Offset

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	0528Ch-0528Fh
Name:	Stream Output 3 Write Offset
ShortName:	SO3_WRITE_OFFSET

There is one R/W 32-bit register for each of the 4 supported stream output buffer slots:
5280h-5283h SO_WRITE_OFFSET0 (for Stream Out Buffer #0)
5284h-5287h SO_WRITE_OFFSET1 (for Stream Out Buffer #1)
5288h-528Bh SO_WRITE_OFFSET2 (for Stream Out Buffer #2)
528Ch-528Fh SO_WRITE_OFFSET3 (for Stream Out Buffer #3)
These registers are used to set and track a DWord-granular Write Offset for each of the 4 Stream Output Buffer slots. Software can directly write them via MI_LOAD_REGxxx commands. The SOL stage will increment them as part of stream output processing. Software can cause them to be written to memory via MI_STORE_REGxxx commands. (See the Stream Output section of the 3D pipeline volume).
These registers are part of the context save and restore.

Programming Notes

- Software must ensure that no HW stream output operations can be in process or otherwise pending at the point that the MI_LOAD/STORE commands are processed. This will likely require a pipeline flush.
- The SOL stage will effectively advance the write offset by the buffer's Surface Pitch after each vertex is written (assuming no overflow is detected in any targeted SO buffer). Under "normal" conditions one would expect software to initialize the WriteOffset to some (possibly zero) multiple of Surface Pitch in order to align vertex writes to the buffer's Base Address, though it is not required to do so.

DWord	Bit	Description
0	31:2	**Write Offset** Format: U30 _Custom_GTIReset: DEV This field contains a DWord offset from the corresponding SO buffer's Base Address value. The SOL stage uses this value as a write offset when performing writes to the buffer. The SOL stage will increment this value as a part of performing stream output to the buffer. Note that the SOL stage uses the buffer's Surface Pitch to advance the Write Offset, without regard to the buffer's Base Address (see Programming Notes above).
	1:0	**Reserved** Access: RO Format: MBZ

SubSlice0 Power Down FSM control register with lock

SSM0SPCPOWERDNFSMCTL - SubSlice0 Power Down FSM control register with lock

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24410h

DWord	Bit	Description
0	31	**power down control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of POWERDNFSMCTL register are R/W 1 = All bits of POWERDNFSMCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:13	**Reserved** Access: RO Format: MBZ
	12	**Leave firewall disabled** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not firewall the gated domain for a power down flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e firewall gated domain to ungated domain crossing during power down flows 1 = Leave firewall disabled, i.e don't firewall the gated domain, but complete logical flow
	11	**Leave reset de-asserted** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not assert reset for power off flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e assert resets during power down flows 1 = Leave reset de-asserted mode, i.e don't assert reset, but complete logical flow
	10	**Leave CLKs ON** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not gate clks for power off flow. But it will pretend to complete the flow with PM Encodings:

SSM0SPCPOWERDNFSMCTL - SubSlice0 Power Down FSM control register with lock

		0 = Default mode, i.e gate clocks during power down flows 1 = Leave CLKS ON mode, i.e don't clock gate, but complete logical flow
	9	**Leave FET On** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not turn off the PFET even though it will complete the flow with PM Encodings: 0 = Default mode, i.e power off fets during power down flows 1 = Leave ON mode, i.e don't power off pfet, but complete logical flow
	8:6	**Power Down state 3** Default Value: 010b Access: R/W Lock _Custom_GTIReset: BUS This will be the 3rd state before power is turned OFF in the well Encodings: 000 = Assert Reset 001 = Firewall ON 010 = Gate clocks 1xx = Rsvd for future Default : Gate Clocks
	5:3	**Power Down state 2** Default Value: 001b Access: R/W Lock _Custom_GTIReset: BUS This will be the 2nd state before power is turned OFF in the well Encodings: 000 = Assert Reset 001 = Firewall ON 010 = Gate clocks 1xx = Rsvd for future Default :Firewall ON
	2:0	**Power Down state 1** Default Value: 000b Access: R/W Lock _Custom_GTIReset: BUS This will be the 1st state before power is turned OFF in the well Encodings: 000 = Assert Reset 001 = Firewall ON 010 = Gate clocks 1xx = Rsvd for future

SSM0SPCPOWERDNFSMCTL - SubSlice0 Power Down FSM control register with lock		
		Default : Assert Reset

SubSlice 0 Power Gate Control Request

SSM0PGCTLREQ - SubSlice 0 Power Gate Control Request

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24400h

Clock Gating Messages Register

DWord	Bit	Description
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask In order to write to bits 15:0, the corresponding message mask bits must be written. For example, for bit 14 to be set, bit 30 needs to be 1 : 40004000
	15:2	**Reserved** Access: RO Format: MBZ
	1	**CLK RST FWE Request** Access: R/W _Custom_GTIReset: BUS SLICE 0 CLK RST FWE request: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**Power Gate Request** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

SubSlice 0 Power on FSM control register with lock

SSM0SPCPOWERUPFSMCTL - SubSlice 0 Power on FSM control register with lock

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	2440Ch

DWord	Bit	Description
0	31	**power up control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of POWERUPFSMCTL register are R/W 1 = All bits of POWERUPFSMCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:9	**Reserved** Access: RO Format: MBZ
	8:6	**Power UP state 3** Default Value: 010b Access: R/W Lock _Custom_GTIReset: BUS This will be the 3rd state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - De-assert resets 3'b000: 10ns (or 1 bclk)
	5:3	**Power UP state 2** Default Value: 001b Access: R/W Lock _Custom_GTIReset: BUS This will be the 2nd state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - Firewall OFF

SSM0SPCPOWERUPFSMCTL - SubSlice 0 Power on FSM control register with lock

	2:0	**Power UP state 1** Default Value: 000b Access: R/W Lock _Custom_GTIReset: BUS This will be the 1st state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - Clock Ungate

SubSlice 1 Power Context Save request

SSM1PGCTXREQ - SubSlice 1 Power Context Save request		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	24484h	
DWord	**Bit**	**Description**
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask bots for lower 16 bits
	15:10	**Reserved** Access: RO Format: MBZ
	9	**Power context save request** Access: R/W Set _Custom_GTIReset: BUS Power Context Save Request 1'b0 : Power context save is not being requested 1'b1 : Power context save is being requested CPUnit self-clears this bit upon sampling.
	8:0	**Power Context Save request crdit count** Access: R/W _Custom_GTIReset: BUS QWord Credits for Power Context Save Request Minimum Credits = 1 : Unit may send 1 QWord pair (enough for first LRI at least) Maximum Credits = 511 : Unit may send 511 QWord pairs A QWord pair is defined as a 32-bit register address and the corresponding 32-bits of register data. Note that the LRI header and END commands are 64-bits each (32-bit command followed by 32-bit NOOP) and will consume one QWord credit. Only valid with PWRCTX_SAVE_REQ (Bit9).

SubSlice 1 Power Down FSM control register with lock

SSM1SPCPOWERDNFSMCTL - SubSlice 1 Power Down FSM control register with lock

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24490h

DWord	Bit	Description
0	31	**power down control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of POWERDNFSMCTL register are R/W 1 = All bits of POWERDNFSMCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:13	**Reserved** Access: RO Format: MBZ
	12	**Leave firewall disabled** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not firewall the gated domain for a power down flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e firewall gated domain to ungated domain crossing during power down flows 1 = Leave firewall disabled, i.e don't firewall the gated domain, but complete logical flow
	11	**Leave reset de-asserted** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not assert reset for power off flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e assert resets during power down flows 1 = Leave reset de-asserted mode, i.e don't assert reset, but complete logical flow
	10	**Leave CLKs ON** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not gate clks for power off flow. But it will pretend to complete the flow with PM Encodings:

SSM1SPCPOWERDNFSMCTL - SubSlice 1 Power Down FSM control register with lock

		0 = Default mode, i.e gate clocks during power down flows 1 = Leave CLKS ON mode, i.e don't clock gate, but complete logical flow
	9	**Leave FET On** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not turn off the PFET eventhough it will complete the flow with PM Encodings: 0 = Default mode, i.e power off fets during power down flows 1 = Leave ON mode, i.e don't power off pfet, but complete logical flow
	8:6	**Power Down state 3** Default Value: 010b Access: R/W Lock _Custom_GTIReset: BUS This will be the 3rd state before power is turned OFF in the well Encodings: 000 = Assert Reset 001 = Firewall ON 010 = Gate clocks 1xx = Rsvd for future Default : Gate Clocks
	5:3	**Power Down state 2** Default Value: 001b Access: R/W Lock _Custom_GTIReset: BUS This will be the 2nd state before power is turned OFF in the well Encodings: 000 = Assert Reset 001 = Firewall ON 010 = Gate clocks 1xx = Rsvd for future Default :Firewall ON
	2:0	**Power Down state 1** Default Value: 000b Access: R/W Lock _Custom_GTIReset: BUS This will be the 1st state before power is turned OFF in the well Encodings: 000 = Assert Reset 001 = Firewall ON 010 = Gate clocks

SSM1SPCPOWERDNFSMCTL - SubSlice 1 Power Down FSM control register with lock		
		1xx = Rsvd for future Default : Assert Reset

SubSlice 1 Power Gate Control Request

SSM1PGCTLREQ - SubSlice 1 Power Gate Control Request

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24480h

Clock Gating Messages Register

DWord	Bit	Description
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask In order to write to bits 15:0, the corresponding message mask bits must be written. For example, for bit 14 to be set, bit 30 needs to be 1 : 40004000
	15:2	**Reserved** Access: RO Format: MBZ
	1	**CLK RST FWE Request** Access: R/W _Custom_GTIReset: BUS SLICE 0 CLK RST FWE request: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**Power Gate Request** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

SubSlice 1 Power on FSM control register with lock

SSM1SPCPOWERUPFSMCTL - SubSlice 1 Power on FSM control register with lock

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	2448Ch

DWord	Bit	Description
0	31	**power up control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of POWERUPFSMCTL register are R/W 1 = All bits of POWERUPFSMCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:9	**Reserved** Access: RO Format: MBZ
	8:6	**Power UP state 3** Default Value: 010b Access: R/W Lock _Custom_GTIReset: BUS This will be the 3rd state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - De-assert resets 3'b000: 10ns (or 1 bclk)
	5:3	**Power UP state 2** Default Value: 001b Access: R/W Lock _Custom_GTIReset: BUS This will be the 2nd state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - Firewall OFF

SSM1SPCPOWERUPFSMCTL - SubSlice 1 Power on FSM control register with lock

	2:0	**Power UP state 1** Default Value: 000b Access: R/W Lock _Custom_GTIReset: BUS This will be the 1st state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - Clock Ungate

SubSlice 2 Power Down FSM control register with lock

SSM2SPCPOWERDNFSMCTL - SubSlice 2 Power Down FSM control register with lock

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	24510h

DWord	Bit	Description
0	31	**power down control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of POWERDNFSMCTL register are R/W 1 = All bits of POWERDNFSMCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:13	**Reserved** Access: RO Format: MBZ
	12	**Leave firewall disabled** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not firewall the gated domain for a power down flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e firewall gated domain to ungated domain crossing during power down flows 1 = Leave firewall disabled, i.e dont firewall the gated domain, but complete logical flow
	11	**Leave reset de-asserted** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not assert reset for power off flow. But it will pretend to complete the flow with PM Encodings: 0 = Default mode, i.e assert resets during power down flows 1 = Leave reset de-asserted mode, i.e dont assert reset, but complete logical flow
	10	**Leave CLKs ON** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not gate clks for power off flow. But it will pretend to complete the flow with PM Encodings:

SSM2SPCPOWERDNFSMCTL - SubSlice 2 Power Down FSM control register with lock

		0 = Default mode, i.e gate clocks during power down flows 1 = Leave CLKS ON mode, i.e dont clock gate, but complete logical flow
	9	**Leave FET On** Access: R/W Lock _Custom_GTIReset: BUS When This bit is set SPC will not turn off the PFET eventhough it will complete the flow with PM Encodings: 0 = Default mode, i.e power off fets during power down flows 1 = Leave ON mode, i.e dont power off pfet, but complete logical flow
	8:6	**Power Down state 3** Default Value: 010b Access: R/W Lock _Custom_GTIReset: BUS This will be the 3rd state before power is turned OFF in the well Encodings: 000 = Assert Reset 001 = Firewall ON 010 = Gate clocks 1xx = Rsvd for future Default : Gate Clocks
	5:3	**Power Down state 2** Default Value: 001b Access: R/W Lock _Custom_GTIReset: BUS This will be the 2nd state before power is turned OFF in the well Encodings: 000 = Assert Reset 001 = Firewall ON 010 = Gate clocks 1xx = Rsvd for future Default :Firewall ON
	2:0	**Power Down state 1** Default Value: 000b Access: R/W Lock _Custom_GTIReset: BUS This will be the 1st state before power is turned OFF in the well Encodings: 000 = Assert Reset 001 = Firewall ON 010 = Gate clocks 1xx = Rsvd for future

SSM2SPCPOWERDNFSMCTL - SubSlice 2 Power Down FSM control register with lock		
		Default : Assert Reset

SubSlice 2 Power Gate Control Request

SSM2PGCTLREQ - SubSlice 2 Power Gate Control Request		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	24500h	
Clock Gating Messages Register		
DWord	**Bit**	**Description**
0	31:16	**Message Mask** Access: RO _Custom_GTIReset: BUS Message Mask In order to write to bits 15:0, the corresponding message mask bits must be written. For example, for bit 14 to be set, bit 30 needs to be 1 : 40004000
	15:2	**Reserved** Access: RO Format: MBZ
	1	**CLK RST FWE Request** Access: R/W _Custom_GTIReset: BUS SLICE 0 CLK RST FWE request: '0' : Initiate power down sequence (clk/rst/fwe) '1' : Initiate power up sequence (clk/rst/fwe)
	0	**Power Gate Request** Access: R/W _Custom_GTIReset: BUS SLice 0 power well request: '0' : Initiate Power Down request '1' : Initiate Power UP req

SubSlice 2 Power on FSM control register with lock

SSM2SPCPOWERUPFSMCTL - SubSlice 2 Power on FSM control register with lock

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	2450Ch

DWord	Bit	Description
0	31	**power up control Lock** Access: R/W Lock _Custom_GTIReset: BUS 0 = Bits of POWERUPFSMCTL register are R/W 1 = All bits of POWERUPFSMCTL register are RO (including this lock bit) Once written to 1, the lock is set and cannot be cleared (i.e., writing a 0 will not clear the lock). These bits are not reset on FLR.
	30:9	**Reserved** Access: RO Format: MBZ
	8:6	**Power UP state 3** Default Value: 010b Access: R/W Lock _Custom_GTIReset: BUS This will be the 3rd state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - De-assert resets 3'b000: 10ns (or 1 bclk)
	5:3	**Power UP state 2** Default Value: 001b Access: R/W Lock _Custom_GTIReset: BUS This will be the 2nd state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - Firewall OFF

SSM2SPCPOWERUPFSMCTL - SubSlice 2 Power on FSM control register with lock

	2:0	**Power UP state 1** Default Value: 000b Access: R/W Lock _Custom_GTIReset: BUS This will be the 1st state after power is turned ON in the well Encodings: 000 = Clock Ungate 001 = Firewall OFF 010 = De-assert resets 1xx = Rsvd for future Default - Clock Ungate

SubSlice unit Level Clock Gating Control 9520

SSMCGCTL9520 - SubSlice unit Level Clock Gating Control 9520

Register Space: MMIO: 0/2/0
Size (in bits): 32
CustomGTIIs_ContextSaved: true

Address: 09520h

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31:2	**Reserved** Access: RO Format: MBZ
	1	**GCPunit Clock Gating Disable** Default Value: 1b Access: R/W _Custom_GTIReset: BUS GCPunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	0	**SMCRunit Clock Gating Disable** Default Value: 1b Access: R/W _Custom_GTIReset: BUS SMCRunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

SubSlice unit Level Clock Gating Control 9524

SSMCGCTL9524 - SubSlice unit Level Clock Gating Control 9524

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 09524h

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31	**EU_FPUunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS EU_FPUunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	30	**EU_GAunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS EU_GAunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	29	**EU_TCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS EU_TCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	28	**DSS router Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS DSS router Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	27	**BCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS BCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for

SSMCGCTL9524 - SubSlice unit Level Clock Gating Control 9524

	functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
26	**Reserved** Access: RO Format: MBZ
25	**CREunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS CREunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
24	**DGunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS DGunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
23	**DMunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS DMunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
22	**Reserved** Access: RO Format: MBZ
21	**EUunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS EUunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
20	**FLunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS FLunit Clock Gating Disable Control:

SSMCGCTL9524 - SubSlice unit Level Clock Gating Control 9524

		'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	19	**Reserved** Access: RO Format: MBZ
	18	**Ftunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS Ftunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	17	**Reserved** Access: RO Format: MBZ
	16	**GWunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GWunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	15:12	**HDCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HDCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	11	**ICunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS ICunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	10	**Reserved** Access: RO Format: MBZ

SSMCGCTL9524 - SubSlice unit Level Clock Gating Control 9524

9	**IMEunit's Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS IMEunit's Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
8	**MAunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS MAunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
7	**MTunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS MTunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
6	**OATREPunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS OATREPunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
5	**PLunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS PLunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
4	**PSDunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS PSDunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)

SSMCGCTL9524 - SubSlice unit Level Clock Gating Control 9524

		'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	3	**RAMDFT Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS RAMDFT Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	2	**SCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	1	**SIunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SIunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	0	**SOunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SOunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

SubSlice unit Level Clock Gating Control 9528

SSMCGCTL9528 - SubSlice unit Level Clock Gating Control 9528

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 09528h

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31:21	**Reserved** Access: RO Format: MBZ
	20	**VMESC Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS
	19	**MediaSampler arb unit2 Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS
	18	**MediaSampler arb unit1 Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS
	17	**Reserved** Access: RO Format: MBZ
	16	**VAFE unit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS
	15	**TSL unit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS
	14	**MediaSampler arb unit Clock Gating Disable** Access: R/W Format: Disable

SSMCGCTL9528 - SubSlice unit Level Clock Gating Control 9528

	_Custom_GTIReset:	BUS
13	**DTOunit Clock Gating Disable**	
	Access:	R/W
	Format:	Disable
	_Custom_GTIReset:	BUS
12	**SLMFE unit Clock Gating Disable**	
	Access:	R/W
	Format:	Disable
	_Custom_GTIReset:	BUS
11	**RDEunit Clock Gating Disable**	
	Access:	R/W
	Format:	Disable
	_Custom_GTIReset:	BUS
10	**Noaunit Clock Gating Disable**	
	Access:	R/W
	Format:	Disable
	_Custom_GTIReset:	BUS
9	**CPSSunit Clock Gating Disable**	
	Access:	R/W
	Format:	Disable
	_Custom_GTIReset:	BUS
8	**EUunit Clock Gating Disable**	
	Access:	R/W
	Format:	Disable
	_Custom_GTIReset:	BUS
7	**EU_EMunit Clock Gating Disable**	
	Access:	R/W
	Format:	Disable
	_Custom_GTIReset:	BUS
6	**sbe_ssbuf Clock Gating Disable**	
	Access:	R/W
	Format:	Disable
	_Custom_GTIReset:	BUS
5	**DAPunit Clock Gating Disable**	
	Access:	R/W
	Format:	Disable

SSMCGCTL9528 - SubSlice unit Level Clock Gating Control 9528

		_Custom_GTIReset: BUS
	4	**Reserved** Access: RO Format: MBZ
	3	**STunit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS
	2	**SVRRunit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS
	1	**SVSMunit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS
	0	**TDLunit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS

SubSlice unit Level Clock Gating Control 9530

SSMCGCTL9530 - SubSlice unit Level Clock Gating Control 9530

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	09530h

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31:14	**Reserved** Access: RO Format: MBZ
	13	**SPARE Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SPARE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	12	**CPSSunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS CPSSunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	11	**AVSunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS AVSunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	10	**Reserved** Access: RO Format: MBZ
	9	**BCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS BCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for

SSMCGCTL9530 - SubSlice unit Level Clock Gating Control 9530

		functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	8	**CREunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS CREunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	7	**Reserved** Access: RO Format: MBZ
	6	**GAunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GAunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	5	**ICunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS ICunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	4	**IMEunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS IMEunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	3	**MTunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS MTunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	2	**PSDunit Clock Gating Disable**

SSMCGCTL9530 - SubSlice unit Level Clock Gating Control 9530

		Access: R/W _Custom_GTIReset: BUS PSDunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	1	**SCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	0	**VSunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VSunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

SubSlice unit Level Clock Gating override during rstflow 9540

SSMMISCCP9540 - SubSlice unit Level Clock Gating override during rstflow 9540		
Register Space: MMIO: 0/2/0 Size (in bits): 32		
Address: 09540h		
Unit Level Clock Gating Disable bits		
DWord	**Bit**	**Description**
0	31:0	**ECO Spare Bits** Access: R/W _Custom_GTIReset: BUS

Subsystem Identification

SID2_0_2_0_PCI - Subsystem Identification

Register Space:	PCI: 0/2/0
Size (in bits):	16
Address:	0002Eh

This register is used to uniquely identify the subsystem where the PCI device resides.

DWord	Bit	Description
0	15:0	**Subsystem Identification** Default Value: 0000000000000000b Access: R/W _Custom_GTIReset: BUS This value is used to identify a particular subsystem. This field should be programmed by BIOS during boot-up.

Subsystem Vendor Identification

SVID2_0_2_0_PCI - Subsystem Vendor Identification

Register Space:	PCI: 0/2/0
Size (in bits):	16
Address:	0002Ch

This register is used to uniquely identify the subsystem where the PCI device resides.

DWord	Bit	Description
0	15:0	**Subsystem Vendor ID** Access: R/W _Custom_GTIReset: BUS This value is used to identify the vendor of the subsystem. Value: 0000000000000000b — Name: **[Default]**

Supported Page Sizes

SUPPORTED_PAGE_SIZES_0_2_0_PCI - Supported Page Sizes

Register Space: PCI: 0/2/0
Size (in bits): 32

Address: 0033Ch

DWord	Bit	Description
0	31:0	**SUPPORTED PAGE SIZES VALUE** Access: RO _Custom_GTIReset: BUS This field indicates the page sizes supported by the PF. This PF supports a page size of 2^(n+12) if bit n is Set. For example, if bit 0 is Set, the PF supports 4-KB page sizes. PFs are required to support 4-KB, 8-KB, 64-KB, 256-KB, 1-MB, and 4-MB page sizes. All other page sizes are optional, and not supported in this implementation. Value: 00000000000000000000010101010011b — Name: **[Default]**

SVG Debug Control

SVG_CTL - SVG Debug Control

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 06400h-06403h

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:8	**Debug Output Mux Select** Controls which 32 bits of the debug data are returned

Value	Name
0	Return General State Base Address for the context that has debug enabled
1	Return Surface State Base Address for the context that has debug enabled
2	Return Indirect Object Base Address for the context that has debug enabled
3	Return General State Access Upper Bound for the context that has debug enabled
4	Return Indirect Object Access Upper Bound for the context that has debug enabled
5	Return Instruction base address for the context that has debug enabled
6	Return Instruction address upper bound for the context that has debug enabled
7	Return System Instruction Pointer for the context that has debug enabled
[8,255]	Reserved: MBZ

	7:0	**Reserved** Access: RO Format: MBZ

SVG Debug Return Data

SVG_RDATA - SVG Debug Return Data		
Register Space:	MMIO: 0/2/0	
Access:	RO	
Size (in bits):	32	
Address:	06404h-06407h	
DWord	**Bit**	**Description**
0	31:0	**SVG Debug Data** Returns data based on debug output mux select.

SVL Barrier Done

SVL_BARRIER_DONE - SVL Barrier Done		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	07FF0h	
This register is used to send messages when a render barrier is done. This register may not be written from CPU.		
DWord	**Bit**	**Description**
0	31:16	**Reserved** Access: RO Format: MBZ
	15:8	**Reserved** Access: R/W Format: PBC _Custom_GTIReset: DEV
	7:0	**BarrierID**

SWF

SWF	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	4F000h-4F08Fh
Name:	Software Flags
ShortName:	SWF_*

These registers are used as scratch pad data storage space and have no direct effect on hardware operation. There are 36 instances of this register format.

DWord	Bit	Description
0	31:0	**Software Flags** Default Value: 00000000000000000000000000000000b Access: R/W _Custom_GTIReset: BUS Software flags

System Page Sizes

SYSTEM_PAGE_SIZES_0_2_0_PCI - System Page Sizes		
Register Space:	PCI: 0/2/0	
Size (in bits):	32	
Address:	00340h	
Defines the System Page Size chosen by the VMM.		
DWord	**Bit**	**Description**
0	31:0	**SYSTEM PAGE SIZES VALUE** Default Value: 00000000000000000000000000000001b Access: RO _Custom_GTIReset: BUS This field defines the page size the system will use to map the VFs memory addresses. Software must set the value of the System Page Size to one of the page sizes set in the Supported Page Sizes field (see Section 3.3.12). As with Supported Page Sizes, if bit nis Set in System Page Size, the VFs associated with this PF are required to support a page size of 2^(n+12). For example, if bit 1 is Set, the system is using an 8-KB page size. The results are undefined if System Page Size is zero. The results are undefined if more than one bit is set in System Page Size. The results are undefined if a bit is Set in System Page Size that is not Set in Supported Page Sizes. When System Page Size is set, the VF associated with this PF is required to align all BAR resources 20 on a System Page Size boundary. Each VF BARn or VF BARn pair (see Section 3.3.14) shall be aligned on a System Page Size boundary. Each VF BARn or VF BARn pair defining a non-zero address space shall be sized to consume an integer multiple of System Page Size bytes. All data structures requiring page size alignment within a VF shall be aligned on a System Page Size boundary. VF Enable must be zero when System Page Size is written. The results are undefined if System Page Size is written when VF Enable is Set. Default value is 1h (i.e., 4 KB), and that is the only value allowed for this implementation

Tailpointer delay counter

TP_DELAY_CNTR - Tailpointer delay counter	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	BUS
Address:	1C2DA0h-1C2DA3h
Name:	Tailpointer delay counter
ShortName:	TP_DELAY_CNTR_VDENC_REG0
Address:	1C6DA0h-1C6DA3h
Name:	Tailpointer delay counter
ShortName:	TP_DELAY_CNTR_VDENC_REG1
Address:	1D2DA0h-1D2DA3h
Name:	Tailpointer delay counter
ShortName:	TP_DELAY_CNTR_VDENC_REG2
Address:	1D6DA0h-1D6DA3h
Name:	Tailpointer delay counter
ShortName:	TP_DELAY_CNTR_VDENC_REG3
Address:	1E2DA0h-1E2DA3h
Name:	Tailpointer delay counter
ShortName:	TP_DELAY_CNTR_VDENC_REG4
Address:	1E6DA0h-1E6DA3h
Name:	Tailpointer delay counter
ShortName:	TP_DELAY_CNTR_VDENC_REG5
Address:	1F2DA0h-1F2DA3h
Name:	Tailpointer delay counter
ShortName:	TP_DELAY_CNTR_VDENC_REG6
Address:	1F6DA0h-1F6DA3h
Name:	Tailpointer delay counter
ShortName:	TP_DELAY_CNTR_VDENC_REG7

This register has stall counter for the VDENC. In an ideal case, this value should be zero

DWord	Bit	Description
0	31:0	**Tail pointer count delay** Access: R/W Hardware Clear

Thread Dispatched Count Register

TDL_THR_DISP_COUNT - Thread Dispatched Count Register

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0E4BCh

This register provides the count of threads dispatched/valid in the subslice.

DWord	Bit	Description
0	31:6	**Reserved** Access: RO Format: MBZ
	5:0	**Thread Count** Value: 0-56 — Name: Valid Range

Thread Faulted Count Register

TDL_THR_PF_COUNT - Thread Faulted Count Register

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	0E5BCh

This register provides the count of threads faulted in each subslice.

DWord	Bit	Description
0	31	**Canonical fault indication bit to CS** The bit is set when a canonical fault on data fetch is reported by EU.
	30:6	**Reserved** Access: RO Format: MBZ
	5:0	**Thread Count** Value: 0-56 — Name: Valid Range

Thread Fault Status Register 0

TDL_THR_PF_STATUS0 - Thread Fault Status Register 0			
Register Space:	MMIO: 0/2/0		
Access:	RO		
Size (in bits):	32		
Address:	0E6B8h		
This register provides the status of each thread in the SubSlice. A bit set indicates the thread in the specific thread slot is faulted.			
DWord	**Bit**	**Description**	
0	31:24	**Row0, EU3, [Reserved, T6-T0]**	
	23:16	**Row0, EU2, [Reserved, T6-T0]**	
	15:8	**Row0, EU1, [Reserved, T6-T0]**	
	7:0	**Row0, EU0, [Reserved, T6-T0]**	

Thread Fault Status Register 1

TDL_THR_PF_STATUS1 - Thread Fault Status Register 1	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	0E7B8h

This register provides the status of each thread in the SubSlice. A bit set indicates the thread in the specific thread slot is faulted.

DWord	Bit	Description
0	31:24	**Row1, EU3, [Reserved, T6-T0]**
	23:16	**Row1, EU2, [Reserved, T6-T0]**
	15:8	**Row1, EU1, [Reserved, T6-T0]**
	7:0	**Row1, EU0, [Reserved, T6-T0]**

Thread Load Status Register 0

TDL_THR_STATUS0 - Thread Load Status Register 0	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0E4B8h

This register provides the status of each thread in the SubSlice.

DWord	Bit	Description
0	31:0	**Reserved** Access: RO Format: MBZ

Thread Load Status Register 1

TDL_THR_STATUS1 - Thread Load Status Register 1	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0E5B8h

This register provides the status of each thread in the SubSlice. A bit set indicates a valid thread is loaded in the thread slot.

DWord	Bit	Description
0	31:24	**Row1, EU3, [Reserved, T6-T0]**
	23:16	**Row1, EU2, [Reserved, T6-T0]**
	15:8	**Row1, EU1, [Reserved, T6-T0]**
	7:0	**Row1, EU0, [Reserved, T6-T0]**

Thread Mode Register

FF_MODE - Thread Mode Register

Register Space: MMIO: 0/2/0
Access: R/W
Size (in bits): 32
_Custom_GTIReset: DEV

This register is used to program the FF shader Mode.

DWord	Bit	Description
0	31	**TE Autostrip Disable** Format: U1 Value / Name / Description: 0h / Enable **[Default]** / TE will generate "autostrip" primitives (if/where possible) during tessellation. 1h / Disable / TE will not generate "autostrip" primitives.
	30	**TDS external Cache Disable** Value / Name / Description: 0b / Enable **[Default]** / The external TDS Cache is enabled if there is enough handles to enable the cache. 1b / Disable / The external TDS Cache is disabled even if there is enough handles to enable the cache. Only the internal TDS Cache will be used.
	29:26	**DS Hit Max Value** Format: U4 If the number of hits reaches the DS Hit Max Value and there is a pending miss to be dispatched, the DS will dispatch the pending miss vertex as a single dispatch. Programming a value of 0 will disable the DS Hit Max counter logic and therefore partial dispatches will not be forced due to the number of hits seen during the accumulation of inputs for a thread dispatch. Value / Name: 15 / **[Default]** [0,15] /
	25:20	**VS Hit Max Value** Format: U6 If the number of hits reaches the VS Hit Max Value and there is a pending miss to be dispatched, the VS will dispatch the pending miss vertex as a single dispatch. Since VS Reference Count Full Force miss enable was removed, the value can be [1,63].

FF_MODE - Thread Mode Register

Value	Name
10	**[Default]**
[1,63]	

19 **Tessellation DOP gating Disable**

Format:	Disable

Value	Name	Description
0h	Enable **[Default]**	HS, TE, TETG, DS, GS and SOL units are DOP gated if all units are disabled
1h	Disable	DOP gating is disabled for HS, TE, TETG, DS, GS and SOL units

Programming Notes
Once this bit is set to a 1, it must not be cleared to a 0 until after a reset.

18 **TRI NOINSIDE Autostrip Cache Invalidate Disable**

Format:	Disable

This bit can be used to control the TRI NOINSIDE Autostrip Cache Invalidate feature. By default the invalidation is ENABLED (allowing higher performance).

Value	Name	Description
0h	Enable **[Default]**	TE will suppress the Autostrip cache invalidate for TRI NOINSIDE patches. This setting may improve performance.
1h	Disable	TE will not suppress the Autostrip cache invalidate for TRI NOINSIDE patches.

Programming Notes
The setting of this field impacts the selection of the "provoking vertex" for the center triangle of TRI domains (if a center triangle exists given the tessellation factors), and this will in turn impact the generated image if any PS attributes are enabled as "flat shaded" and that/those attributes are not identical on all 3 vertices of the triangle. Given the fact that the APIs do not impose requirements on the starting/provoking vertices of the tessellation-generated triangles, the selection of provoking vertices can be arbitrary as long as it is deterministic (i.e., repeated rendering the same patch will yield the same results). Note: If FF_mode.TE autostrip Disable is set to 1, then TRI NOINSIDE Autostrip Cache Invalidate will also be disabled.

17 **Thread fusing disable**

Value	Name	Description
1h	Disable	Fusing of threads will be disabled. Threads will not be fused.
0h	**[Default]**	Fusing of threads will be enabled. Threads will be fused if all conditions are met to fuse a thread.

FF_MODE - Thread Mode Register

	16:15	**Reserved** Format: PBC
	14:13	**Reserved** Format: PBC
	12	**Reserved** Default Value: 0h Format: PBC
	11:7	**Reserved** Format: PBC
	6:5	**Reserved** Format: PBC
	4	**Reserved** Default Value: 0h Format: PBC
	3	**Reserved** Format: PBC
	2	**TDS Tracking fifo wrap fix disable** Format: Disable **Value** / **Name** / **Description** 1h / / Disable the tds tracking fifo wrap fix. 0h / **[Default]** / Enable the tds tracking fifo wrap fix.
	1	**Reserved** Format: PBC
	0	**Reserved** Format: PBC

Thread Restart Control Register

TDL_THR_RESTART - Thread Restart Control Register		
Register Space:	MMIO: 0/2/0	
Access:	WO	
Size (in bits):	32	
Address:	0E450h	
This register provides control to restart page faulted and halted threads in each subslice.		
DWord	**Bit**	**Description**
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**Restart All Faulted Threads** A write of 1 to this register restarts all threads that have halted due to page fault.

Tile Cache Control Register

TCCNTLREG - Tile Cache Control Register

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B138h
Name:	Tile Cache Control Register
ShortName:	TCCNTLREG
Address:	0B238h
ShortName:	TCCNTLREG_CCS0

DWord	Bit	Description
0	31:25	**Unified Tile Cache Pool** Access: R/W Number of ways allocated for the unified client pool. This is a combined pool for all streams. 00h **[Default]** When this field is non-zero, Z tile cache pool and C tile cache pool should be 0KB. [0h,68h]
	24:18	**Z Tile Cache Pool** Access: R/W Number of ways allocated for Z streams. 00h **[Default]** Note: This field must be 0KB if Unified Tile cache Pool is non-zero. [0h,68h]
	17:11	**C Tile Cache Pool** Access: R/W Number of ways allocated for Color Streams 00h **[Default]** Note: This field must be 0KB if Unified Tile cache Pool is non-zero. [0h,68h]
	10:4	**Command Streamer Allocation** Access: R/W Number of ways allocated for CS(Command Streamer) 00h **[Default]** [0h,68h]

TCCNTLREG - Tile Cache Control Register

	3	**Reserved** Access: RO Format: MBZ
	2	**L3 Data partial write merging enable** Default Value: 1 Format: Enable
	1	**Color/Z write partial write merging enable** Format: Enable **Value** / **Name** / **Description** 0 / **[Default]** / Partial write merging optimization (in SQDB) will be disabled for Color/Z bound cycles.
	0	**Reserved** Access: RO Format: MBZ

TILECTL

TILECTL - TILECTL		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
SOC_Consumer:	BIOS	
Address:	101000h	
Tile control and TLB control.		

DWord	Bit	Description
0	31:2	**Reserved** Access: RO Format: MBZ
	1	**TLBPF** Default Value: 0b Access: R/W _Custom_GTIReset: BUS Store multiple PTE enable. 0: Only one Page Table Entry is stored in the Translation Lookaside Buffer cache. 1: Multiple Page Table Entries (8) are stored in the Translation Lookaside Buffer cache.
	0	**SWZCTL** Default Value: 0b Access: R/W _Custom_GTIReset: BUS In order to spread DRAM accesses between multiple channels in the most efficient way, address bits can be used as a channel select. The Driver needs to obtain the need for memory address swizzling via DRAM configuration registers and set the following bits. x0b - No Address Swizzling x1b - Address bit [6] needs to be swizzled for tiled surfaces

TiledResources Invalid Tile Detection Register

TRINVTILEDETCT - TiledResources Invalid Tile Detection Register

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	04DECh
Name:	TiledResources Invalid Tile Detection Register
ShortName:	TRINVTILEDETCT

DWord	Bit	Description
0	31:0	**Invalid Tile Detection Value** Access: R/W _Custom_GTIReset: DEV **Value** / **Name** / **Description** 00000000h / **[Default]** / A 32-bit value programmed to enable h/w to perform a match of TR-VA TT entries to detect Invalid Tiles. Hardware will flag each entry and space behind it as Invalid Tile for matched entries.

TiledResources VA Detection Registers

TRVADR - TiledResources VA Detection Registers

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	04DF0h
Name:	TiledResources VA Detection Registers
ShortName:	TRVADR

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7:4	**TR - VA Mask Value** Default Value: 0000b Access: R/W _Custom_GTIReset: DEV 4bit MASK value that is mapped to incoming address bits[47:44] MASK bits are used to identify which address bits need to be considered for compare. If particular mask bit is "1", mapping address bit needs to be compared to DATA value provided. If "0", corresponding address bit is masked which makes it don't care for compare. (This field defaults to "0000" to disable detection.). Note: The only usage model for GFX driver to set this field to "1111". Behavior of h/w for any other setting is not defined. Note: GFX driver shall use same TRVA MASK value for all contexts.
	3:0	**TR- VA Data Value** Access: R/W _Custom_GTIReset: DEV Value / Name / Description: 0000b **[Default]** 4bit Data value that is mapped to incoming address bits[47:44]. Data bits are used to compare address values that are not filtered by the TRVAMV for match Note: GFX driver shall use same TRVA Data value for all contexts

Tiled Resources Wrapper Write Data Port arbitration

TRWRPARB - Tiled Resources Wrapper Write Data Port arbitration

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	04DF8h
Name:	Tiled Resources Wrapper Write Data Port arbitration
ShortName:	TRWRPARB

DWord	Bit	Description
0	31:13	**Reserved** Access: RO Format: MBZ
	12:10	**L3 Max Write Request Limit Count** Default Value: 100b Access: R/W _Custom_GTIReset: DEV This is the MAX number of Allowed writes from L3 before switching the priority to Z Requests Count - Minimum count value must be 1
	9	**Reserved** Access: RO Format: MBZ
	8:6	**Z Max Write Request Limit Count** Default Value: 010b Access: R/W _Custom_GTIReset: DEV This is the MAX number of Allowed writes from Z before switching the priority to C Requests Count - Minimum count value must be = 1
	5	**Reserved** Access: RO Format: MBZ
	4:2	**C Max Write Request Limit Count** Default Value: 010b Access: R/W _Custom_GTIReset: DEV This is the MAX number of Allowed writes from C before switching to L3 Request Count - Minimum count value = 1

TRWRPARB - Tiled Resources Wrapper Write Data Port arbitration

	1	**Reserved** Access: RO Format: MBZ
	0	**Fixed Arbitration enable** Default Value: 1b Access: R/W _Custom_GTIReset: DEV Fixed Arbitration enable when 1'b1 Programmable arbitration when 1'b0

TIMESTAMP_CTR

TIMESTAMP_CTR	
Register Space:	MMIO: 0/2/0
Access:	R/WC
Size (in bits):	32
Address:	44070h-44073h
Name:	Time Stamp Counter
ShortName:	TIMESTAMP_CTR
Reset:	global

The register is not reset by a FLR.

DWord	Bit	Description
0	31:0	**TIMESTAMP Counter** This field increments every microsecond. The value in this field is latched in the Pipe Flip TIMESTAMP registers when flips occur, and in the Pipe Frame TIMESTAMP registers at start of vertical blank. The register value will reset if any value is written to it. The register is not reset by a FLR.

TLBInvalidationRegister_BLT

BLT_TLB_INV_CR - TLBInvalidationRegister_BLT

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0CEE4h
Name:	BLT TLB Invalidation Register
ShortName:	BLT_TLB_INV_CR

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**Invalidate TLBs on the corresponding Engine** Default Value: 0b Access: R/W SW writes 1 to invalidate the TLBs for the associated engine and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's HW pipeline is flushed and cleared from all its memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs. This bit is self clear.

TLBInvalidationRegister_GFX

GFX_TLB_INV_CR - TLBInvalidationRegister_GFX

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0CED8h
Name:	GFX TLB Invalidation Register
ShortName:	GFX_TLB_INV_CR

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**Invalidate TLBs on the corresponding Engine** Default Value: 0b Access: R/W SW writes 1 to invalidate the TLBs for the associated engine and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's HW pipeline is flushed and cleared from all its memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs. This bit is self clear.

TLBInvalidationRegister_GUC

GUC_TLB_INV_CR - TLBInvalidationRegister_GUC

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
_Custom_GTIHardWiredEnable:	clr_reg_ga_cee8

Address:	0CEE8h
Name:	GUC TLB Invalidation Register
ShortName:	GUC_TLB_INV_CR
_Custom_GTIHardWiredEnable:	clr_reg_ga_cee8

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**Invalidate TLBs on the corresponding Engine** Default Value: 0b Access: R/W SW writes 1 to invalidate the TLBs for the associated engine and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's HW pipeline is flushed and cleared from all its memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs. This bit is self clear.

TLBInvalidationRegister_OA

OA_TLB_INV_CR - TLBInvalidationRegister_OA

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0CEECh
Name:	OA TLB Invalidation Register
ShortName:	OA_TLB_INV_CR

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**Invalidate TLBs on the corresponding Engine** Default Value: 0b Access: R/W SW writes 1 to invalidate the TLBs for the associated engine and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's HW pipeline is flushed and cleared from all its memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs. This bit is self clear.

Top of Low Usable DRAM Register

TOLUD_REG - Top of Low Usable DRAM Register

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	108000h

This 32 bit register defines the Top of Low Usable DRAM. GT uses this to ensure no GT memory accesses occur between 4GB and TOLUD.

DWord	Bit	Description
0	31:20	**TOLUD** Default Value: 001h Access: R/W _Custom_GTIReset: BUS This register contains bits 31 to 20 of an address one byte above the maximum DRAM memory below 4G that is usable by the operating system. Address bits 31 down to 20 programmed to 01h implies a minimum memory size of 1MB. BIOS must set this value. Address bits 19:0 are assumed to be 0_0000h for the purposes of address comparison. The Host interface positively decodes an address towards DRAM if the incoming address is less than the value programmed in this register.
	19:0	**Reserved** Access: RO Format: MBZ

TOUUD_LSB_REG

TOUUD_LSB_REG - TOUUD_LSB_REG

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	108080h

This 64 bit register defines the Top of Upper Usable DRAM. Address bits 19:0 are assumed to be 000_0000h for the purposes of address comparison. The Host interface positively decodes an address towards DRAM if the incoming address is less than the value programmed in this register and greater than or equal to 4GB. BIOS Restriction: Minimum value for TOUUD is 4GB.
All the bits in this register have SAI policy group protection.

DWord	Bit	Description
0	31:20	**TOUUD** Default Value: 001h Access: R/W _Custom_GTIReset: BUS This register contains the LSB portion (bits 31 to 20) of an address one byte above the maximum DRAM memory above 4G that is usable by the operating system. Address bits 19:0 are assumed to be 000_0000h for the purposes of address comparison. The Host interface positively decodes an address towards DRAM if the incoming address is less than the value programmed in this register and greater than 4GB.
	19:1	**Reserved** Access: RO Format: MBZ
	0	**SPARE** Default Value: 0b Access: R/W _Custom_GTIReset: BUS

TOUUD_MSB_REG

TOUUD_MSB_REG - TOUUD_MSB_REG		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
SOC_Consumer:	BIOS	
Address:	108084h	
This is the most significant 32 bits of a 64 bit register that defines the Top of Upper Usable DRAM. BIOS Restriction: Minimum value for TOUUD is 4GB.		
DWord	**Bit**	**Description**
0	31:0	**TOUUD** Default Value: 00000000h Access: R/W _Custom_GTIReset: BUS This register contains bits 63 to 32 of an address one byte above the maximum DRAM memory above 4G that is usable by the operating system.

TRANS_CLK_SEL

TRANS_CLK_SEL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	46140h-46143h
Name:	Transcoder A Clock Select
ShortName:	TRANS_CLK_SEL_A
Reset:	soft
Address:	46144h-46147h
Name:	Transcoder B Clock Select
ShortName:	TRANS_CLK_SEL_B
Reset:	soft
Address:	46148h-4614Bh
Name:	Transcoder C Clock Select
ShortName:	TRANS_CLK_SEL_C
Reset:	soft
Address:	4614Ch-4614Fh
Name:	Transcoder D Clock Select
ShortName:	TRANS_CLK_SEL_D
Reset:	soft

This register maps the port clock to the transcoder.

DWord	Bit	Description
0	31:28	**Trans Clock Select** Select which DDI clock to use for this transcoder.

Value	Name
0000b	None - Clock Disabled
0001b	DDI A
0010b	DDI B
0011b	DDI C
0100b	DDI USBC1
0101b	DDI USBC2
0110b	DDI USBC3
0111b	DDI USBC4
1000b	DDI USBC5
1001b	DDI USBC6

TRANS_CLK_SEL		
		Restriction This must not be changed while the transcoder is enabled.
	27:0	**Reserved** Access: RO Format: MBZ

TRANS_CONF

TRANS_CONF	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of vertical blank (WD cap sync) OR transcoder disabled
Address:	7E008h-7E00Bh
Name:	Transcoder WD0 Configuration
ShortName:	TRANS_CONF_WD0
Reset:	soft
Address:	7D008h-7D00Bh
Name:	Transcoder WD1 Configuration
ShortName:	TRANS_CONF_WD1
Reset:	soft
Address:	7B008h-7B00Bh
Name:	Transcoder DSI 0 Configuration
ShortName:	TRANS_CONF_DSI0
Reset:	soft
Address:	7B808h-7B80Bh
Name:	Transcoder DSI 1 Configuration
ShortName:	TRANS_CONF_DSI1
Reset:	soft
Address:	70008h-7000Bh
Name:	Transcoder Configuration
ShortName:	TRANS_CONF_A
Reset:	soft
Address:	71008h-7100Bh
Name:	Transcoder Configuration
ShortName:	TRANS_CONF_B
Reset:	soft
Address:	72008h-7200Bh
Name:	Transcoder Configuration
ShortName:	TRANS_CONF_C
Reset:	soft
Address:	73008h-7300Bh
Name:	Transcoder Configuration

TRANS_CONF

ShortName: TRANS_CONF_D

Reset: soft

DWord	Bit	Description
0	31	**Transcoder Enable** Setting this bit to the value of one, turns on this transcoder. Turning the transcoder off disables the timing generator and synchronization pulses to the display will not be maintained. Enabling the transcoder may be internally delayed for one frame while the display data buffers are re-configured. **Value** / **Name** 0b – Disable 1b – Enable **Restriction** Timing registers must contain valid values before this bit is enabled.
	30	**Transcoder State** Access: RO This read only bit indicates the actual state of the transcoder. **Value** / **Name** 0b – Disabled 1b – Enabled
	29:23	**Reserved** Access: RO Format: MBZ
	22:21	**Interlaced Mode** These bits control the transcoder interlaced mode. This field is ignored by WD. **Value** / **Name** / **Description** 00b – PF-PD – Progressive Fetch with Progressive Display 01b – PF-ID – Progressive Fetch with Interlaced Display 11b – IF-ID – Interlaced Fetch with Interlaced Display Others – Reserved – Reserved **Programming Notes** Progressive Fetch with Interlaced Display requires pipe scaling. **Restriction** VGA display modes do not work while in interlaced fetch mode. Progressive Fetch with Interlaced Display effectively down scales the vertical by 2X, which reduces the maximum supported pixel rate by half. Interlaced fetch mode is not supported with Y Tiling. Interlaced fetch mode is not supported

TRANS_CONF

		with 90/270 rotation. Interlaced fetch mode is not supported with scaling. Interlaced fetch mode is not supported with YUV 420 hybrid planar source pixel formats. In Interlaced mode, the plane height must be a minimum of 2 scanlines.
	20:12	**Reserved** Access: RO Format: MBZ
	11:8	**Reserved**
	7	**Reserved**
	6:0	**DP Audio Symbol Watermark** Default Value: 24h 36 entries This fields set the level to which the DP audio symbol RAM must fill before it starts to drain during horizontal blank. The minimum is 2 entries and the maximum is 64.

TRANS_DDI_FUNC_CTL

TRANS_DDI_FUNC_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B400h-6B403h
Name:	Transcoder DSI 0 DDI Function Control
ShortName:	TRANS_DDI_FUNC_CTL_DSI0
Reset:	soft
Address:	6BC00h-6BC03h
Name:	Transcoder DSI 1 DDI Function Control
ShortName:	TRANS_DDI_FUNC_CTL_DSI1
Reset:	soft
Address:	60400h-60403h
Name:	Transcoder DDI Function Control
ShortName:	TRANS_DDI_FUNC_CTL_A
Reset:	soft
Address:	61400h-61403h
Name:	Transcoder DDI Function Control
ShortName:	TRANS_DDI_FUNC_CTL_B
Reset:	soft
Address:	62400h-62403h
Name:	Transcoder DDI Function Control
ShortName:	TRANS_DDI_FUNC_CTL_C
Reset:	soft
Address:	63400h-63403h
Name:	Transcoder DDI Function Control
ShortName:	TRANS_DDI_FUNC_CTL_D
Reset:	soft

DWord	Bit	Description
0	31	**TRANS DDI Function Enable** This bit enables the transcoder DDI function. Value: 0b = Disable; 1b = Enable
	30:27	**DDI Select**

These bits determine which DDI port this transcoder will connect to. It is not valid to enable and direct more than one transcoder to one DDI, except when using DisplayPort multistreaming.

This field is ignored by the DSI transcoders since they have a fixed DDI mapping.

Value	Name
0000b	None
0001b	DDI A
0010b	DDI B
0011b	DDI C
0100b	DDI USBC1
0101b	DDI USBC2
0110b	DDI USBC3
0111b	DDI USBC4
1000b	DDI USBC5
1001b	DDI USBC6

Restriction
This field must not be changed while the function is enabled.

26:24 **TRANS DDI Mode Select**

This field determines the mode of operation. HDMI mode enables a null packet (32 bytes of a value of 0) to be sent when Vsync=1, and also enables preambles and guardbands prior to the null packets, in accordance with the HDMI specification. DVI mode will function as HDMI if DIP packets or audio are enabled.

This field does not apply to the DSI transcoder.

Value	Name	Description
000b	HDMI	Function in HDMI mode
001b	DVI	Function in DVI mode
010b	DP SST	Function in DisplayPort SST mode
011b	DP MST	Function in DisplayPort MST mode
Others	Reserved	Reserved

Restriction
This field must not be changed while the function is enabled. The DisplayPort mode (SST or MST) selected here must match the mode selected in the DisplayPort Transport Control register for the transport attached to this transcoder.
Trans DDI mode select should be programmed to MST in the same register write as MST transport select (field [11:10] in this register).

TRANS_DDI_FUNC_CTL

23 Reserved

Access:	RO
Format:	MBZ

22:20 Bits Per Color

This field selects the number of bits per color output on the DDI connected to this transcoder. Dithering should be enabled when selecting a pixel color depth higher or lower than the pixel color depth of the frame buffer.

This field does not apply to the DSI transcoder. The Pixel Format for the DSI transcoder is defined within the TRANS_DSI_FUNC_CONF register.

Value	Name	Description
000b	8 bpc	
001b	10 bpc	
010b	6 bpc	
011b	12 bpc	
Others	Reserved	Reserved

Restriction
This field must not be changed while the function is enabled.
6bpc not supported with HDMI.
6bpc not supported with VDSC.

19:18 Reserved

Access:	RO
Format:	MBZ

17:16 Sync Polarity

This field indicates the polarity of Hsync and Vsync.

Field ignored by the DSI transcoder

Value	Name	Description
00b	Low	VS and HS are active low (inverted)
01b	VS Low, HS High	VS is active low (inverted), HS is active high
10b	VS High, HS Low	VS is active high, HS is active low (inverted)
11b	High **[Default]**	VS and HS are active high

15 Reserved

Access:	RO
Format:	MBZ

TRANS_DDI_FUNC_CTL

14:12 DSI Input Select

These bits determine the input to transcoder DSI. These bits are ignored by the other transcoders.

Value	Name
000b	Pipe A
101b	Pipe B
110b	Pipe C
111b	Pipe D
Others	Reserved

Restriction
This field must not be changed while the function is enabled. It is not valid to have the same pipes driving multiple enabled transcoders.

11:10 MST Transport Select

This field selects which DP transport the DP data from this transcoder is sent to for MST stream combining. This field is ignored when MST is disabled.
Restriction : MST transport select should be programmed in the same register write as Trans DDI mode select (field [26:24] in this register).

Value	Name
00b	DPTP A
01b	DPTP B
10b	DPTP C
11b	DPTP D

Restriction
This field must not be changed while the function is enabled.

9 Reserved

8 DP VC Payload Allocate

This bit enables DisplayPort Virtual Channel payload allocation. This bit is ignored by transcoder EDP since it does not support multistreaming.
This bit is ignored by transcoder DSI since it does not support multistreaming

Value	Name
0b	Disable
1b	Enable

TRANS_DDI_FUNC_CTL

7 HDMI Scrambler CTS Enable

This bit enables Compliance Test Specification mode on the HDMI scrambler. This bit must be set before the scrambler is enabled.

This bit is ignored by transcoder DSI

Value	Name
0b	Disable
1b	True

6 HDMI Scrambler Reset frequency

This bit specifies the frequency at which the scrambler is reset when the HDMI Scrambler CTS Enable bit is set.

This bit must be set before or along with the HDMI Scrambler CTS Enable bit.

Value	Name	Description
0b	Every Line	SSCP sent on hsync of every line
1b	Every Other Line	SSCP sent on hsync of every other line

5 Reserved

4 High TMDS Char Rate

This field enables the high TMDS character rate. It must be enabled when the HDMI link symbol rate is greater than 340 MHz. It must be disabled when the HDMI link symbol rate is less than or equal to 340 MHz.

Value	Name	Description
0	Disable	TMDS Character Rate is less than or equal to 340 Mega-characters/second/channel
1	Enable	TMDS Character Rate is greater than 340 Mega-characters/second/channel

3:1 Port Width Selection

Description
This field selects the number of lanes to be enabled on the DDI link for DisplayPort and DSI. for DSI, this field specifies the number of Data lanes to use - the Clock lane is always enabled when the DSI function is enabled. This field is ignored for HDMI and DVI which always use all 4 lanes. The value selected here must match the value selected in the DDI Buffer Control register for the DDI attached to this pipe.

Value	Name	Description
000b	x1	x1 Mode
001b	x2	x2 Mode
010b	x3	[] x3 Mode (DSI only)
011b	x4	x4 Mode
Others	Reserved	Reserved

		Restriction This field must not be changed while the DDI is enabled.
	0	**HDMI Scrambling Enabled** Setting this bit enables scrambling over the HDMI link. Scrambling must be enabled when the HDMI link symbol rate is greater than 340 MHz. Scrambling should be enabled at lower frequencies if the receiver supports it at that speed. This must not changed while the port is enabled. **Value** / **Name** 0b / Disable 1b / Enable

TRANS_DDI_FUNC_CTL2

TRANS_DDI_FUNC_CTL2	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B404h-6B407h
Name:	Transcoder DSI 0 DDI Function Control2
ShortName:	TRANS_DDI_FUNC_CTL2_DSI0
Reset:	soft
Address:	6BC04h-6BC07h
Name:	Transcoder DSI 1 DDI Function Control2
ShortName:	TRANS_DDI_FUNC_CTL2_DSI1
Reset:	soft
Address:	60404h-60407h
Name:	Transcoder DDI Function Control2
ShortName:	TRANS_DDI_FUNC_CTL2_A
Reset:	soft
Address:	61404h-61407h
Name:	Transcoder DDI Function Control2
ShortName:	TRANS_DDI_FUNC_CTL2_B
Reset:	soft
Address:	62404h-62407h
Name:	Transcoder DDI Function Control2
ShortName:	TRANS_DDI_FUNC_CTL2_C
Reset:	soft
Address:	63404h-63407h
Name:	Transcoder DDI Function Control2
ShortName:	TRANS_DDI_FUNC_CTL2_D
Reset:	soft

DWord	Bit	Description
0	31	**Genlock Enable** Value: 1b — Name: Enable Value: 0b — Name: Disable
	30:29	**Genlock Mode**

TRANS_DDI_FUNC_CTL2

Value	Name	Description
10b	Primary	Primary transcoder outputs frame sync for other transcoders to secondary to.
00b	Local Secondary	Local secondary transcoder secondaries to frame sync from a primary transcoder in the same device. The primary transcoder is selected by Port Sync Mode Primary Select.
01b	Remote Secondary	Remote secondary transcoder secondaries to frame sync from a primary transcoder in a different device.
11b	Reserved	

28 **Reserved**

Access:	RO
Format:	MBZ

27:9 **Reserved**

Access:	RO
Format:	MBZ

8 **Double Buffer Vactive**

Value	Name
0b	Normal Vactive
1b	Double Buffer Vactive

7:6 **Audio Mute Override**

This field overrides audio mutesignal in VBID.

Value	Name	Description
00b,01b	Do not override	
10b	Override and reset	Override audio mute bit to '0'.
11b	Override and set	Override audio mute bit to '1'.

5 **Dual Pipe Sync Enable**

This bit informs the DSI transcoder that while it is synchronized with another DSI transcoder, it will also be driven by a separate Pipe

Value	Name	Description
0b	Disabled	Both transcoders are being driven by a single Pipe (Dual Link - Single Pipe)
1b	Enabled	Each transcoder is being driven by a separate Pipe (Dual Link - Dual Pipe)

4 **Port Sync Mode Enable**

This field enables the DisplayPort port sync mode on this transcoder. This mode forces two or more transcoders to be in sync; with one transcoder primary and one or more transcoder secondaries. The primary is unaware that it is operating in this mode. Only the secondary is aware that it is operating in this mode. Port sync mode is only enabled in the secondary transcoder.

For DSI, this bit enables DSI Transcoder 1 to be a secondary to DSI Transcoder 0. DSI

TRANS_DDI_FUNC_CTL2

Transcoder 0 is unaware that it is the primary of DSI Transcoder 1

Value	Name
0b	Disable
1b	Enable

Restriction
Follow the instructions for enabling and disabling Sync Mode in the Display Mode Set Sequence - Sequence for DisplayPort. Port Sync Mode Primary Select must be programmed with a valid value when Port sync Mode is enabled. The secondary and primary transcoders and associated ports must have identical parameters and properties; same color format, link width (number of lanes enabled), resolution, refresh rate, PLL configuration, dot clock, TU size, M and N programming, etc. Spread spectrum clocking cannot be used when the ports use separate PLLs.
Port Sync Mode can be enabled with DisplayPort SST and with DisplayPort MST.

3 **Reserved**

Access:	RO
Format:	MBZ

2:0 **Port Sync Mode Primary Select**

Description
This field indicates which transcoder will be the primary to this transcoder when in port sync mode.
This field is ignored by the DSI transcoders since only DSI 0 can be the primary.
This field is also used for genlock for the local secondary transcoder to select a primary transcoder.

Value	Name
001b	Transcoder A
010b	Transcoder B
011b	Transcoder C
100b	Transcoder D

Restriction
A port cannot be secondaries to itself.
The DSI transcoders cannot be secondary to a non-DSI transcoder - field ignored by the DSI transcoder.

TRANS_DSI_FUNC_CONF

TRANS_DSI_FUNC_CONF

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B030h-6B033h
Name:	Transcoder DSI 0 Function Configuration
ShortName:	TRANS_DSI_FUNC_CONF_0
Reset:	soft
Address:	6B830h-6B833h
Name:	Transcoder DSI 1 Function Configuration
ShortName:	TRANS_DSI_FUNC_CONF_1
Reset:	soft

This register defines the functional transcoder configuration that is specific to the DSI transcoders.

Restriction :
This register must be programmed before the DSI Transcoder function is enabled (i.e. TRANS DDI Function Enable)
The contents of this register must not be changed while the DSI Transcoder function is enabled

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29:28	**Mode of Operation** Access: R/W This defines whether the DSI transcoder is in Video or Command mode. In addition to the main modes, there are two sub-modes per main mode. For the Command sub-modes, when in the "No Gate" mode, the transcoder will begin transmitting the frame pixels as soon as they are received from the Display Engine. When in the "TE Gate" mode, the transcoder will only start transmitting the frame pixels after a TE event is received. For the Video sub-modes, when in the Sync Event mode only Sync Start packets (VSS/HSS) are sent to the Periphery. When in the Sync Pulse mode, both Sync Start (VSS/HSS) and Sync End (VSE/HSE) packets are sent to the Periphery. Note that regardless of the programming of this field, until the Transcoder Enable bit is set within the TRANS_CONF_DSI register, the DSI transcoder will not generate any timing information to the Display Engine or timing packets to the Peripheral Value / Name: 00b — Command Mode (No Gate) 01b — Command Mode (TE Gate)

TRANS_DSI_FUNC_CONF

10b	Video Mode (Sync Event)
11b	Video Mode (Sync Pulse)

27 **TE Source**

Access:	R/W

This bit defines the source of the TE events from the Peripheral when the Transcoder is operating in Command Mode

Value	Name
0	In-band TE event source
1	Out-of-band TE event source (i.e. GPIO)

26 **TE Deglitch Enable**

Access:	R/W

When using the GPIO as the source of TE events in Command Mode, this bit will control whether the signaling from the GPIO pin is debounced or not.

Value	Name
0b	Disabled
1b	Enabled

25 **TE Accumulation**

This bit controls whether the TE events from two Panels are accumulated into a single event (usage would be for a Dual Link mode). The accumulated event will feed into the interrupt registers for each DSI transcoder pair (e.g. DSI0 and DSI1).

Value	Name
0b	Disabled
1b	Enabled

Programming Notes

1. This bit only affects the operation of the transcoder when it is operating in the Command Mode
2. It is the responsibility of Software to set this bit accordingly (i.e. it must ensure TE events are being received from both ports)
3. Even though the accumulated event is being fed to the interrupt registers of both transcoders, it is ultimately up to Software on how it enables the TE Interrupt across both transcoders
4. Hardware will automatically enable this feature (i.e. it will override the bit programming) when Periodic Frame Update and Port Sync Mode are enabled for both transcoders. Hardware will use the accumulated TE events to spawn the Frame Update Requests to each transcoder

24:21 **Reserved**

Access:	RO
Format:	MBZ

TRANS_DSI_FUNC_CONF

20 Link Ready

Access:	RO

This bit advertises whether the Link is ready to receive traffic from the DSI transcoder

Value	Name
0	Link is not ready to accept traffic
1	Link is ready to accept traffic

19 Reserved

Access:	RO
Format:	MBZ

18:16 Pixel Format

Access:	R/W

This field defines the pixel format the DSI Transcoder will be operating in

Value	Name
000b	16-bit RGB, 5-6-5
001b	18-bit RGB, 6-6-6 (Packed)
010b	18-bit RGB, 6-6-6 (Loose)
011b	24-bit RGB, 8-8-8
100b	30-bit RGB, 10-10-10
101b	36-bit RGB, 12-12-12
110b	Compressed
Others	Reserved

Restriction
When in the 18-bit RGB (Packed) pixel format, the H. Active Size must be a multiple of 4 pixels

15 BGR Transmission

Access:	R/W

This field will reverse the order of the RGB channels within the pixels received from the Display Engine

Value	Name
0	Transmit order is R-G-B
1	Transmit order is B-G-R

14 Reserved

Access:	RO
Format:	MBZ

13:12 Pixel Virtual Channel

Access:	R/W

This field defines the Virtual Channel that HW will bind to all DSI packets carrying pixel data

TRANS_DSI_FUNC_CONF

11:10 Pixel Buffer Threshold

Access:	R/W

This field defines the threshold of buffering needed within the Pixel Buffer before the transcoder will start internally processing the pixel stream.

Value	Name
00b	The Pixel Buffer will have to be 1/4 full
01b	The Pixel Buffer will have to be 1/2 full
10b	The Pixel Buffer will have to be 3/4 full
11b	The Pixel Buffer will have to be full

9:8 Continuous Clock

Access:	R/W

This field will control the behavior of the Clock Lane and whether it is allowed to enter the LP state.
Keeping the Clock Lane running while letting the Data Lanes go in and out of the LP state keeps the LP to HS turnaround latency to a minimum, but consumes more power.
Certain panels may also require the Clock Lane to continuously run

Value	Name
00b	Always enter LP after Data Lanes
10b	Opportunistically keep Clock in HS or LP
11b	Continuous HS Clock
Others	Reserved

7 LP Clock during LPM

When the Clock Lane is configured for Continuous HS Clock, this bit will control whether the DSI transcoder places the Clock Lane into the LP state along with the Data Lanes when the per frame LP mode (LPM) is performed.
This bit has no effect ont the Clock Lane for the other Continuous Clock settings.

Value	Name	Description
0b	Disabled	Clock Lane does not follow the Data Lanes
1b	Enable	Clock Lane follows the Data Lanes

6 Reserved

Access:	RO
Format:	MBZ

5:4 Link Calibration

Access:	R/W

This field will control the Link calibration of the DSI Transcoder

Value	Name
00b	Calibration Disabled
10b	Calibration Enabled - Initial only
11b	Calibration Enabled - Initial and Periodic

TRANS_DSI_FUNC_CONF

Others	Reserved

Restriction
Calibration can be enabled for any Link frequency, but it must be enabled when the Link frequency is operating above 1.5 Gbps

3 **Reserved**

Access:	RO
Format:	MBZ

2 **Blanking Packet during BLLP**

This bit will control whether the transcoder allows the link to enter the LP state during BLLP regions (assuming there is enough time), or whether it will keep the link in the HS state with a Blanking Packet

Value	Name	Description
0b	Disabled	LP allowed in BLLP regions
1b	Enabled	Blanking packets transmitted in BLLP regions

Programming Notes

1. This bit is only applicable when the transcoder is operating in the Video Mode
2. When this bit is set, then all BLLP regions will be filled with Blanking Packets regardless of where those regions are located (i.e. Vertical active or blank)
3. Regardless of the setting of this bit, if HW determines it cannot allow the link to enter the LP state between HS bursts, then it will automatically fill the region with a Blanking Packet
4. Regardless of the setting of this bit, HW will still ensure the link enters the LP state once per frame per the DSI spec

1 **S3D Orientation**

Access:	R/W

This bit controls the orientation encoding of the 3DMODE field of the Stereoscopic 3D control function sent to a Panel via the Vertical Source Sync (VSS) packet when in Video Mode.

Value	Name
0b	Portrait Orientation
1b	Landscape Orientation

Programming Notes

This bit will only be sampled by the transcoder when Stereoscopic 3D is enabled for the transcoder
This bit should be programmed before enabling Stereoscopic 3D for the transcoder (TRANS_STEREO3D_CTL)
If Software changes this bit, it must also perform a write to the TRANS_STEREO3D_CTL for the change to be sent within the next VSS

This bit is only applicable when the transcoder is operating in Video Mode. If the transcoder is operating in Command Mode, then Software will have to communicate the Stereoscopic 3D function information to the Panel through a set_3D_control DCS command using the DCS Long Write DSI packet type.

0 **EoTp Disabled**

Access:	R/W

When set, the DSI transcoder will not transmit an End of Transmission packet at the end of High Speed bursts

Value	Name
0	EoTp Enabled
1	EoTp Disabled

TRANS_EXITLINE

TRANS_EXITLINE	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60018h-6001Bh
Name:	Transcoder DC3CO Exit Line
ShortName:	TRANS_EXITLINE_A
Reset:	soft
Address:	61018h-6101Bh
Name:	Transcoder DC3CO Exit Line
ShortName:	TRANS_EXITLINE_B
Reset:	soft
Address:	62018h-6201Bh
Name:	Transcoder DC3CO Exit Line
ShortName:	TRANS_EXITLINE_C
Reset:	soft
Address:	63018h-6301Bh
Name:	Transcoder DC3CO Exit Line
ShortName:	TRANS_EXITLINE_D
Reset:	soft

Restriction
This register should be programmed following the sequence to allow DC3co (i.e., anytime before DC3co is enabled).

DWord	Bit	Description
0	31	**Enable Exit Line** Enable indicates idle frame reset should be applied at exit line Value / Name: 0b – Disable 1b – Enable
	30:13	**Reserved** Access: RO Format: MBZ

TRANS_EXITLINE		
	12:0	**Exit Line** This field represents the scanline at which DC3CO Exit happens. Example: DC3CO Exit time per calculation is10 scanlines SW needs to program this field as (VACTIVE - 10).

TRANS_HBLANK

TRANS_HBLANK	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60004h-60007h
Name:	Transcoder Horizontal Blank
ShortName:	TRANS_HBLANK_A
Reset:	soft
Address:	61004h-61007h
Name:	Transcoder Horizontal Blank
ShortName:	TRANS_HBLANK_B
Reset:	soft
Address:	62004h-62007h
Name:	Transcoder Horizontal Blank
ShortName:	TRANS_HBLANK_C
Reset:	soft
Address:	63004h-63007h
Name:	Transcoder Horizontal Blank
ShortName:	TRANS_HBLANK_D
Reset:	soft

Restriction
This register should not be changed while the transcoder or port are enabled.

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**Horizontal Blank End** This field specifies Horizontal Blank End position relative to the horizontal active display start. **Restriction** The minimum horizontal blank size is 32 pixels. For HDMI Audio transmission the minimum is 138 pixels. This register must always be programmed to the same value as the Horizontal Total.
	15:14	**Reserved** Access: RO Format: MBZ

TRANS_HBLANK		
	13:0	**Horizontal Blank Start** This field specifies the Horizontal Blank Start position relative to the horizontal active display start. **Restriction** This register must always be programmed to the same value as the Horizontal Active.

TRANS_HSYNC

TRANS_HSYNC	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B008h-6B00Bh
Name:	Transcoder DSI 0 Horizontal Sync
ShortName:	TRANS_HSYNC_DSI0
Reset:	soft
Address:	6B808h-6B80Bh
Name:	Transcoder DSI 1 Horizontal Sync
ShortName:	TRANS_HSYNC_DSI1
Reset:	soft
Address:	60008h-6000Bh
Name:	Transcoder Horizontal Sync
ShortName:	TRANS_HSYNC_A
Reset:	soft
Address:	61008h-6100Bh
Name:	Transcoder Horizontal Sync
ShortName:	TRANS_HSYNC_B
Reset:	soft
Address:	62008h-6200Bh
Name:	Transcoder Horizontal Sync
ShortName:	TRANS_HSYNC_C
Reset:	soft
Address:	63008h-6300Bh
Name:	Transcoder Horizontal Sync
ShortName:	TRANS_HSYNC_D
Reset:	soft

Restriction

This register should not be changed while the transcoder or port are enabled.
HSYNC is always programmed to an even number of pixel clock cycles for YUV 4:2:0 pixel format with 10bpc and 12bpc..
HSYNC timings must be even for HDMI operating on 10bpc (regardless of format)

DWord	Bit	Description	
0	31:30	**Reserved**	
		Access:	RO

TRANS_HSYNC

	Format:	MBZ
29:16	**Horizontal Sync End** This field specifies the Horizontal Sync End position relative to the horizontal active display start. It is programmed with HorizontalActive+FrontPorch+Sync-1 **Restriction** This value must be greater than the horizontal sync start and less than Horizontal Total. For the DSI transcoder this field is only relevant if the transcoder is operating in Video Mode (it is ignored in Command Mode). As such, please refer to the "Determining Minimum Horizontal Blanking Regions" section of the Transcoder DSI Function page for minimum programming allowed	
15:14	**Reserved**	
	Access:	RO
	Format:	MBZ
13:0	**Horizontal Sync Start** This field specifies the Horizontal Sync Start position relative to the horizontal active display start. It is programmed with HorizontalActive + FrontPorch - 1 **Restriction** This value must be greater than Horizontal Active for all non-DSI transcoders, or for DSI transcoders operating with non-compressed pixels in Video Mode. For DSI transcoders operating on a compressed pixel stream in Video Mode, this value should be greater than the Horizontal Active size divided by the Compression Ratio (H. Blank Start = H. Active / Compression Ratio). In HDMI modes the minimum gap between horizontal blank start and horizontal sync start is 16 pixels. For DSI transcoders, this field is only relevant if the transcoder is operating in Video Mode (it is ignored in Command Mode). When the DSI transcoder is in Video Mode, then please refer to the "Determining Minimum Horizontal Blanking Regions" section of the Transcoder DSI Function page for minimum programming allowed.	

TRANS_HTOTAL

TRANS_HTOTAL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6E000h-6E003h
Name:	Transcoder WD0 Horizontal Total
ShortName:	TRANS_HTOTAL_WD0
Reset:	soft
Address:	6E800h-6E803h
Name:	Transcoder WD1 Horizontal Total
ShortName:	TRANS_HTOTAL_WD1
Reset:	soft
Address:	6B000h-6B003h
Name:	Transcoder DSI 0 Horizontal Total
ShortName:	TRANS_HTOTAL_DSI0
Reset:	soft
Address:	6B800h-6B803h
Name:	Transcoder DSI 1 Horizontal Total
ShortName:	TRANS_HTOTAL_DSI1
Reset:	soft
Address:	60000h-60003h
Name:	Transcoder Horizontal Total
ShortName:	TRANS_HTOTAL_A
Reset:	soft
Address:	61000h-61003h
Name:	Transcoder Horizontal Total
ShortName:	TRANS_HTOTAL_B
Reset:	soft
Address:	62000h-62003h
Name:	Transcoder Horizontal Total
ShortName:	TRANS_HTOTAL_C
Reset:	soft
Address:	63000h-63003h
Name:	Transcoder Horizontal Total
ShortName:	TRANS_HTOTAL_D

TRANS_HTOTAL

Reset: soft

Programming Notes
For eDP CoG: $Htotal_{COG}$ = (active pixels per segment) + (overlap pixels * number of segments/2) + Original Horizontal Blanking

Restriction
This register should not be changed while the transcoder or port are enabled.
The following restriction applies only to HDMI 4:2:0. All horizontal timings should be a multiple of 4 for 8/12/16 bpc cases and multiple of 8 for 10 bpc case. This applies to full blend and bypass modes.
All horizontal timings must be even for HDMI operating on 10bpc (regardless of format)

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**Horizontal Total** This field specifies Horizontal Total size. This field is programmed to the number of pixels desired minus one. This should be equal to the sum of the horizontal active and the horizontal blank sizes for all non-DSI transcoders, or for DSI transcoders operating with non-compressed pixels in Video Mode. For DSI transcoders operating with compressed pixels in Video Mode, this field should be equal to the sum of the compressed horizontal active size and the horizontal blank size (H. Total = (H. Active + H. Blank size) / Compression Ratio) For DSI transcoders operating in Command Mode, there are no restrictions on the programming of this field. This field is ignored by WD transcoders. **Restriction** This register must always be programmed to the same value as the Horizontal Blank End.
	15:14	**Reserved** Access: RO Format: MBZ

TRANS_HTOTAL		
	13:0	**Horizontal Active** This field specifies Horizontal Active Display size. The first horizontal active display pixel is considered pixel number 0. This field is programmed to the number of pixels desired minus one. **Restriction** The minimum horizontal active display size is 64 pixels. In HDMI modes the minimum is 256 pixels. This register must always be programmed to the same value as the Horizontal Blank Start. DSI requires a minimum Horizontal Active Display of, 256 pixels. Also, when transmitting an 18-bit RGB pixel format, the one-based size must be a multiple of 4 pixels

TRANS_MSA_MISC

TRANS_MSA_MISC	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60410h-60413h
Name:	Transcoder MSA Misc
ShortName:	TRANS_MSA_MISC_A
Reset:	soft
Address:	61410h-61413h
Name:	Transcoder MSA Misc
ShortName:	TRANS_MSA_MISC_B
Reset:	soft
Address:	62410h-62413h
Name:	Transcoder MSA Misc
ShortName:	TRANS_MSA_MISC_C
Reset:	soft
Address:	63410h-63413h
Name:	Transcoder MSA Misc
ShortName:	TRANS_MSA_MISC_D
Reset:	soft

This register selects what value will be sent in the DisplayPort Main Stream Attribute (MSA) Miscellaneous (MISC) fields. The MSA MISC fields are mostly used to indicate the color encoding format and need to be programmed to indicate color space, bits per color, etc.

Programming Notes

See the DisplayPort specification for the details on what to program in these fields.

DWord	Bit	Description
0	31:16	**MSA Unused** This field selects the value that will be sent in the DisplayPort MSA unused fields. **Programming Notes** This should be usually programmed with all 0s.
	15:8	**MSA MISC1** This field selects the value that will be sent in the DisplayPort MSA MISC1 field. When TRANS_STEREO3D_CTL bit FS_MSA_MISC1_Drive_En is enabled, hardware will drive MISC1 bits 2:1 (bits 10:9 of this register) with the field sequential stereo 3D left or right eye indication, and any value written to those bits here will be ignored.

TRANS_MSA_MISC		
	7:0	**MSA MISC0** This field selects the value that will be sent in the DisplayPort MSA MISC0 field. **Restriction** Before enabling DisplayPort, bit 0 should always be set to 1 to indicate link clock and stream clock are synchronous.

TRANS_MULT

TRANS_MULT	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6002Ch-6002Fh
Name:	Transcoder Multiply
ShortName:	TRANS_MULT_A
Reset:	soft
Address:	6102Ch-6102Fh
Name:	Transcoder Multiply
ShortName:	TRANS_MULT_B
Reset:	soft
Address:	6202Ch-6202Fh
Name:	Transcoder Multiply
ShortName:	TRANS_MULT_C
Reset:	soft
Address:	6302Ch-6302Fh
Name:	Transcoder Multiply
ShortName:	TRANS_MULT_D
Reset:	soft

Restriction
This register should not be changed while the transcoder or port are enabled.

DWord	Bit	Description
0	31:3	**Reserved** Access: RO Format: MBZ
	2:0	**Multiplier** This field specifies the data multiplier value used by HDMI and DVI.

Value	Name	Description
000b	X1	Multiply by 1
001b	X2	Multiply by 2
011b	X4	Multiply by 4
Others	Reserved	Reserved

TRANS_PUSH

TRANS_PUSH	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60A70h-60A73h
Name:	Transcoder ASFU VRR Push
ShortName:	TRANS_PUSH_A
Reset:	soft
Address:	61A70h-61A73h
Name:	Transcoder ASFU VRR Push
ShortName:	TRANS_PUSH_B
Reset:	soft
Address:	62A70h-62A73h
Name:	Transcoder ASFU VRR Push
ShortName:	TRANS_PUSH_C
Reset:	soft
Address:	63A70h-63A73h
Name:	Transcoder ASFU VRR Push
ShortName:	TRANS_PUSH_D
Reset:	soft

After programming any pipe registers in ASFU/VRR cases, Software can set push bit. H/W will sync all those updates for that frame.

DWord	Bit	Description
0	31	**Push Enable** This bit enables Push frame functionality. This bit should be set before ASFU/VRR enable. Value: 0b — Disable Value: 1b — Enable
	30	**Send Push** Access: R/W Set Writing a 1b to this field hints that Frame update is desired. Vblank is asserted at the next decision boundary. This is a sticky bit. The bit will be cleared by hardware after double buffer update. Writing a 0b has no effect.
	29:0	**Reserved** Access: RO Format: MBZ

TRANS_SPACE

TRANS_SPACE	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B024h-6B027h
Name:	Transcoder DSI 0 Space
ShortName:	TRANS_SPACE_DSI0
Reset:	soft
Address:	6B824h-6B827h
Name:	Transcoder DSI 1 Space
ShortName:	TRANS_SPACE_DSI1
Reset:	soft
Address:	60024h-60027h
Name:	Transcoder Space
ShortName:	TRANS_SPACE_A
Reset:	soft
Address:	61024h-61027h
Name:	Transcoder Space
ShortName:	TRANS_SPACE_B
Reset:	soft
Address:	62024h-62027h
Name:	Transcoder Space
ShortName:	TRANS_SPACE_C
Reset:	soft
Address:	63024h-63027h
Name:	Transcoder Space
ShortName:	TRANS_SPACE_D
Reset:	soft

Restriction
This register should not be changed while the transcoder or port are enabled.

DWord	Bit	Description
0	31:12	**Reserved** Access: RO Format: MBZ

TRANS_SPACE		
	11:0	**Vertical Active Space** **Description** This field specifies Stereo 3D Vertical Active space. This determines the number of constant pixel value lines inserted between the left and right eye active video regions in the stereo 3D stacked frame mode. This field will only be used when the transcoder is in the stereo 3D stacked frame mode. This field should usually be programmed to be the same as the width of the vertical blank.

TRANS_STEREO3D_CTL

TRANS_STEREO3D_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	7B020h-7B023h
Name:	Transcoder DSI 0 Stereo 3D Control
ShortName:	TRANS_STEREO3D_CTL_DSI0
Reset:	soft
Address:	7B820h-7B823h
Name:	Transcoder DSI 1 Stereo 3D Control
ShortName:	TRANS_STEREO3D_CTL_DSI1
Reset:	soft
Address:	70020h-70023h
Name:	Transcoder Stereo 3D Control
ShortName:	TRANS_STEREO3D_CTL_A
Reset:	soft
Address:	71020h-71023h
Name:	Transcoder Stereo 3D Control
ShortName:	TRANS_STEREO3D_CTL_B
Reset:	soft
Address:	72020h-72023h
Name:	Transcoder Stereo 3D Control
ShortName:	TRANS_STEREO3D_CTL_C
Reset:	soft
Address:	73020h-73023h
Name:	Transcoder Stereo 3D Control
ShortName:	TRANS_STEREO3D_CTL_D
Reset:	soft

This register is sampled one line before vertical blank.

DWord	Bit	Description
0	31	**Transcoder S3D Enable** **Description** This bit enables the stereo 3D modes on this transcoder. Updates will take place at the start of the next vertical blank.

TRANS_STEREO3D_CTL

Value	Name
0b	Disable
1b	Enable

Restriction
These modes are only for use with DisplayPort, HDMI, and DVI.HDMI/DVI: Stereo 3D can only be enabled with a mode set. It must be enabled before transcoder and port are enabled. It must be disabled after transcoder is disabled. DisplayPort: Stereo 3D can be enabled and disabled with a mode set, like HDMI and DVI, or it can be enabled after an enable mode set is complete and disabled prior to a disable mode set. VGA display modes, interlaced modes, SRD/PSR, WD, and frame buffer compression (FBC) do not work with stereo 3D.The left surface base address registers for the planes going to this transcoder must be programmed with valid addresses prior to enabling stereo 3D.

30:29 **Reserved**

Access:	RO
Format:	MBZ

28:27 **S3D Mode**

This field selects between the stereo 3D modes.
The stacked buffer mode combines both stereo 3D fields (left and right eye images) into a single tall frame with the left eye image on top, then a programmable space of black lines, then the right eye image on the bottom.
The field sequential mode sends one stereo 3D field (left or right eye image) out per frame. This mode is only for use with DisplayPort. Field sequential hardware controlled mode automatically toggles between left and right eye at the start of each vertical blank. The starting field is selected using the FS_Field_Ctl register bit. Field sequential software controlled mode will manually select left or right eye using the FS_Field_Ctl register bit.

Value	Name	Description
00b	FS HW Auto	Hardware controlled auto-toggle between left and right eye on each vertical blank.
01b	FS SW Manual	Software controlled selection between left and right eye
10b	Stacked	Stacked frame mode with both left and right eye images combined in a single tall frame
Others	Reserved	Reserved

Programming Notes
In the stacked frame mode, a vertical sync is not generated in the gap between left and right eye images, and the scan line count increments across the entire tall frame.

Restriction
This field should only be changed when stereo 3D is disabled.

TRANS_STEREO3D_CTL

26 **FS Field Ctl**

The operation of this bit depends on the S3D Mode setting. This field is ignored in the S3D stacked mode. In the field sequential software controlled mode this bit selects the field sequential stereo 3D field (left or right eye).In the field sequential hardware controlled mode this bit selects the field sequential stereo 3D starting field, the field used on the frame when field sequential stereo 3D is enabled. Hardware does not wait for a specific eye when disabling.

Value	Name
0b	Right Eye
1b	Left Eye **[Default]**

Restriction
The starting field must be set to the left eye for FS HW Auto usage.

25 **Reserved**

Access:	RO
Format:	MBZ

24 **S3D Current Field**

Access:	RO

This read only bit indicates the current stereo 3D field (left or right eye).This bit should be ignored when stereo 3D is not enabled.

Value	Name
0b	Right Eye
1b	Left Eye

23 **FS MSA MISC1 Drive En**

This bit enables hardware to drive the MSA MISC1 bits 2:1 with the internal field sequential stereo 3D left/right eye field indication. Hardware will drive 00 when field sequential 3D stereo mode is not enabled, 01 when enabled and the upcoming video frame is the right eye, 11 when enabled and the upcoming video frame is the left eye. This is based on the internal left/right indication which could be either generated by hardware in the HW auto mode or by software in the SW manual mode. FS_MSA_Drive_Invert can be programmed to invert the left and right eye selection in the MSA. When this bit is disabled, software may manually program TRANS_MSA_MISC to set MISC1 bits 2:1.

Value	Name	Description
0b	Disable	Disable hardware driving MSA MISC1 bits 2:1. Allow software to manually program MSA MISC1 bits 2:1 through TRANS_MSA_MISC.
1b	Enable	Enable hardware to drive MSA MISC1 bits 2:1 for stereo 3D.

Restriction
This field should only be changed when stereo 3D is disabled and should not be enabled for the stacked frame mode.

22 **Reserved**

TRANS_STEREO3D_CTL			
	21:0	**Reserved**	
		Access:	RO
		Format:	MBZ

TRANS_VBLANK

TRANS_VBLANK	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B010h-6B013h
Name:	Transcoder DSI 0 Vertical Blank
ShortName:	TRANS_VBLANK_DSI0
Reset:	soft
Address:	6B810h-6B813h
Name:	Transcoder DSI 1 Vertical Blank
ShortName:	TRANS_VBLANK_DSI1
Reset:	soft
Address:	60010h-60013h
Name:	Transcoder Vertical Blank
ShortName:	TRANS_VBLANK_A
Reset:	soft
Address:	61010h-61013h
Name:	Transcoder Vertical Blank
ShortName:	TRANS_VBLANK_B
Reset:	soft
Address:	62010h-62013h
Name:	Transcoder Vertical Blank
ShortName:	TRANS_VBLANK_C
Reset:	soft
Address:	63010h-63013h
Name:	Transcoder Vertical Blank
ShortName:	TRANS_VBLANK_D
Reset:	soft

Description
PSR ASU: This register is double buffered when Vactive double buffer bit (bit[8])is asserted in TRANS_DDI_FUNC_CTL2.
The transcoder will generate two Vertical Blanks to the Display Engine. A raw (or unmodified) V. Blank will be generated based off of the vertical timings defined within the TRANS_VTOTAL register (the raw V. Blank will assert after V.Active and de-asserted after V. Total). A delayed V. Blank will be generated from the vertical timings of this register. The delay is in terms of lines, and if this register is programmed with the same value as the TRANS_VTOTAL register (i.e. V. Blank Start = V. Active), then the delay will be zero.

TRANS_VBLANK

Programming Notes

This register is programmed in terms of lines and the values programmed should be zero-based (e.g. for the vertial blank to start on line 10, the Vertical Blank Start will be programmed to 9)

Restriction

This register should not be changed while the transcoder or port are enabled.

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**Vertical Blank End** This field specifies Vertical Blank End position relative to the vertical active display start. **Restriction** 1. The minimum vertical blank size is 5 lines. With SRD/PSR and/or DisplayPort VDIP GMP the minimum is 8 lines 2. The V. Blank End must be programmed to the same value as V. Total in the TRANS_VTOTAL register 3. The V. Blank End must be greater than the V. Blank Start
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**Vertical Blank Start** This field specifies the Vertical Blank Start position relative to the vertical active display start. **Restriction** The V. Blank Start must be greater than or equal to the Vertical Active programming in TRANS_VTOTAL

TRANS_VRR_CTL

TRANS_VRR_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60420h-60423h
Name:	VRR Control Register Transcoder
ShortName:	TRANS_VRR_CTL_A
Reset:	soft
Address:	61420h-61423h
Name:	VRR Control Register Transcoder
ShortName:	TRANS_VRR_CTL_B
Reset:	soft
Address:	62420h-62423h
Name:	VRR Control Register Transcoder
ShortName:	TRANS_VRR_CTL_C
Reset:	soft
Address:	63420h-63423h
Name:	VRR Control Register Transcoder
ShortName:	TRANS_VRR_CTL_D
Reset:	soft

DWord	Bit	Description
0	31	**VRR Enable** This bit enables/disables VRR feature on the fly, no mode set required. Value: 0b = Disable; 1b = Enable
	30	**Ignore Max Shift** This bit when set to '1' will ignore programmed Vblank max shift values (both INC and DEC), allowing Vblank to freely swing between Vmin and Vmax. Value: 1b = IGNORE; 0b = DO NOT IGNORE
	29	**Flip Line Enable** Description: This bit enables/disables Flip Line feature where frame start will be generated at flip line value for VRR. Updates to this field will take effect at the next vertical blank. This field must be

enabled before VRR enable.

Ignore Max Shift must be set to 1 when Flip Line is enabled

Value	Name
0b	Disable
1b	Enable

28:16 **Reserved**

Access:	RO
Format:	MBZ

15:11 **Reserved**

Access:	RO
Format:	MBZ

10:3 **FrameStart to Pipeline Full LineCount**

Default Value:	20h

This 8-bit line count allows a maximum of 255 scanlines (programming 255 gives 255 scanlines duration to fill the pipeline etc.).

Restriction
Always program this field to (Transcoder VRR Vmin - Transcoder Vertical Blank Start - 4) where VRR Vmin and Transcoder Vertical Blank Start are '0' based.

2:1 **Reserved**

Access:	RO
Format:	MBZ

0 **Pipeline Full Override**

This bit tells whether to depend on HW generated signal to start Vactive or to depend on the programmed Pipeline full line count.
Setting this bit assumes pipeline full line count is already programmed.

Value	Name
0b	HW generated Pipeline Full Line Count
1b	Programmed Pipeline Full Line Count

Restriction
SW programmed pipeline full line count must be used when flip line is enabled. This will ensure that Vblank always ends at the same point.

TRANS_VRR_FLIPLINE

TRANS_VRR_FLIPLINE	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60438h-6043Bh
Name:	VRR Flipline Trans
ShortName:	TRANS_VRR_FLIPLINE_A
Reset:	soft
Address:	61438h-6143Bh
Name:	VRR Flipline Trans
ShortName:	TRANS_VRR_FLIPLINE_B
Reset:	soft
Address:	62438h-6243Bh
Name:	VRR Flipline Trans
ShortName:	TRANS_VRR_FLIPLINE_C
Reset:	soft
Address:	63438h-6343Bh
Name:	VRR Flipline Trans
ShortName:	TRANS_VRR_FLIPLINE_D
Reset:	soft

This register defines vertical total size to execute a flip for VRR.

DWord	Bit	Description
0	31:20	**Reserved** Access: RO Format: MBZ
	19:0	**VRR FLIPLINE** This field provides the vertical total size to execute flip for VRR. This is 0-based. So, value should be programmed as N-1.

TRANS_VRR_STATUS

TRANS_VRR_STATUS	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6042Ch-6042Fh
Name:	VRR Status Trans
ShortName:	TRANS_VRR_STATUS_A
Reset:	soft
Address:	6142Ch-6142Fh
Name:	VRR Status Trans
ShortName:	TRANS_VRR_STATUS_B
Reset:	soft
Address:	6242Ch-6242Fh
Name:	VRR Status Trans
ShortName:	TRANS_VRR_STATUS_C
Reset:	soft
Address:	6342Ch-6342Fh
Name:	VRR Status Trans
ShortName:	TRANS_VRR_STATUS_D
Reset:	soft

This register provides VRR status and some additional information

DWord	Bit	Description
0	31	**Vmax Reached** Access: R/WC Sticky bit is set indicating no flip has occurred when Vmax is reached. This bit needs to be polled and cleared by software. Setting of this sticky bit will not result in any interrupt. 0b: Not Reached 1b: Reached
	30	**No flip till decision boundary** Access: R/WC This bit is asserted whenever frame decision boundary is reached without a flip. 0b: 1b:

TRANS_VRR_STATUS

	29	**Flip before decision boundary** Access: R/WC This bit is asserted whenever frame decision boundary is reached with a flip. 0b 1b
	28	**No flip frame** Access: R/WC This bit is asserted whenever current frame terminates without a flip. 0b 1b
	27	**VRR Enable Live** Access: RO This bit is asserted whenever pipe/plane logic is running with VRR enabled. 0b 1b
	26	**flips serviced** Access: R/WC This bit represents acknowledgement from pipe to transcoder indicating that processing for all flips in the current frame is complete and any subsequent flips will be handled in the next frame. 0b 1b
	25:23	**Reserved** Access: RO Format: MBZ
	22:20	**Current Region in Vblank** Access: RO This field indicates current status of VRR FSM.

Value	Name	Description
000b	IDLE	Timing generator disabled.
001b	WAIT_TILL_FDB	Wait till the flip decision boundary
010b	WAIT_TILL_FRAMESTART	After the decision boundary, wait for framestart
011b	WAIT_TILL_FLIP	Transition to this state when past the decision boundary, but no primary flip

TRANS_VRR_STATUS				
		100b	PIPELINE_FILL	State after framestart, waiting for a fixed num of lines
		101b	ACTIVE	
		110b	LEGACY_VBLANK	No VRR
	19:0	**Reserved**		
		Access:		RO
		Format:		MBZ

TRANS_VRR_STATUS2

TRANS_VRR_STATUS2	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6043Ch-6043Fh
Name:	VRR Status2 Trans
ShortName:	TRANS_VRR_STATUS2_A
Reset:	soft
Address:	6143Ch-6143Fh
Name:	VRR Status2 Trans
ShortName:	TRANS_VRR_STATUS2_B
Reset:	soft
Address:	6243Ch-6243Fh
Name:	VRR Status2 Trans
ShortName:	TRANS_VRR_STATUS2_C
Reset:	soft
Address:	6343Ch-6343Fh
Name:	VRR Status2 Trans
ShortName:	TRANS_VRR_STATUS2_D
Reset:	soft

This register provides the live status of vertical line counter. This field should be used to issue flip during VRR Flip Line mode.

DWord	Bit	Description
0	31:20	**Reserved** Access: RO Format: MBZ
	19:0	**Vertical Line Counter Status** Access: RO This field provides the live status of vertical line counter. This field should be used to issue flip during VRR Flip Line mode.

TRANS_VRR_VMAX

TRANS_VRR_VMAX	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60424h-60427h
Name:	VRR Vmax Transcoder
ShortName:	TRANS_VRR_VMAX_A
Reset:	soft
Address:	61424h-61427h
Name:	VRR Vmax Transcoder
ShortName:	TRANS_VRR_VMAX_B
Reset:	soft
Address:	62424h-62427h
Name:	VRR Vmax Transcoder
ShortName:	TRANS_VRR_VMAX_C
Reset:	soft
Address:	63424h-63427h
Name:	VRR Vmax Transcoder
ShortName:	TRANS_VRR_VMAX_D
Reset:	soft

Programming Notes

This register is not double buffered and the changed values are reflected immediately in VRR logic.
This register should not be updated while VRR mode is running.

DWord	Bit	Description
0	31:20	**Reserved** Access: RO Format: MBZ
	19:0	**VRR Vmax** This field provides the maximum vertical total size for VRR. This is a 0-based counter (that is, counted 0 to N-1). This field cannot be zero when VRR is enabled. Vmax= ROUNDDOWN(Dot clock / (Htotal * Min Refresh rate))

TRANS_VRR_VMAXSHIFT

TRANS_VRR_VMAXSHIFT	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60428h-6042Bh
Name:	VRR Vmaxshift Trans
ShortName:	TRANS_VRR_VMAXSHIFT_A
Reset:	soft
Address:	61428h-6142Bh
Name:	VRR Vmaxshift Trans
ShortName:	TRANS_VRR_VMAXSHIFT_B
Reset:	soft
Address:	62428h-6242Bh
Name:	VRR Vmaxshift Trans
ShortName:	TRANS_VRR_VMAXSHIFT_C
Reset:	soft
Address:	63428h-6342Bh
Name:	VRR Vmaxshift Trans
ShortName:	TRANS_VRR_VMAXSHIFT_D
Reset:	soft

This register defines VBLANK maximum shift allowed between successive frames.

Programming Notes

This register is not double buffered and the changed values are reflected immediately in VRR logic.
This register should not be updated while VRR mode is running.

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**Decrement** This register represents the maximum reduction.Smallest Vblank = Previous Frame Vblank End - Vblank Max Shift Decrement.
	15:14	**Reserved** Access: RO Format: MBZ

TRANS_VRR_VMAXSHIFT		
	13:0	**Increment** This register represents the maximum increase.Largest Vblank = Previous Frame Vblank End + Vblank Max Shift Increment.

TRANS_VRR_VMIN

TRANS_VRR_VMIN	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60434h-60437h
Name:	VRR Vmin Transcoder
ShortName:	TRANS_VRR_VMIN_A
Reset:	soft
Address:	61434h-61437h
Name:	VRR Vmin Transcoder
ShortName:	TRANS_VRR_VMIN_B
Reset:	soft
Address:	62434h-62437h
Name:	VRR Vmin Transcoder
ShortName:	TRANS_VRR_VMIN_C
Reset:	soft
Address:	63434h-63437h
Name:	VRR Vmin Transcoder
ShortName:	TRANS_VRR_VMIN_D
Reset:	soft

Programming Notes

This register is not double buffered and the changed values are reflected immediately in VRR logic.
This register should not be updated while VRR mode is running.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**VRR Vmin** Access: R/W This field provides the minimum vertical total size for VRR. This is a 0-based counter (that is, counted0 to N-1). This field cannot be zero when VRR is enabled. Vmin = ROUNDUP(Dot clock / (Htotal * Max Refresh rate))

TRANS_VRR_VTOTAL_PREV

TRANS_VRR_VTOTAL_PREV	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60480h-60483h
Name:	VRR Vtotal Previous Transcoder
ShortName:	TRANS_VRR_VTOTAL_PREV_A
Reset:	soft
Address:	61480h-61483h
Name:	VRR Vtotal Previous Transcoder
ShortName:	TRANS_VRR_VTOTAL_PREV_B
Reset:	soft
Address:	62480h-62483h
Name:	VRR Vtotal Previous Transcoder
ShortName:	TRANS_VRR_VTOTAL_PREV_C
Reset:	soft
Address:	63480h-63483h
Name:	VRR Vtotal Previous Transcoder
ShortName:	TRANS_VRR_VTOTAL_PREV_D
Reset:	soft

This register holds the Vtotal from previous frame and other VRR status bits. If VRR is not enabled, this register will hold software programmed Vtotal.

DWord	Bit	Description
0	31	**Flip Before Boundary** Access: RO This bit indicates that a push has occurred in either Active region or in Vblank before Flip Decision Boundary and will result in a Vblank which is Previous_Frame_Vblank - Vblank_Max_Shift_Dec.
	30	**Flip After Boundary** Access: RO This bit indicates that the first push has occurred after Flip Decision Boundary and will result in a Vblank which is between (Previous_Frame_Vblank - Vblank_Max_Shift_Dec) and (Previous_Frame_Vblank + Vblank_Max_Shift_Inc).
	29	**Flip After Double Buffer** Access: RO This bit indicates that the first push has occurred after the Double buffer update and will result in a Vblank which is (Previous_Frame_Vblank + Vblank_Max_Shift_Inc). This however will cause

TRANS_VRR_VTOTAL_PREV		
		the next frame Vblank to be (Previous_Frame_Vblank - Vblank_Max_Shift_Dec).
	28:20	**Reserved** Access: RO Format: MBZ
	19:0	**Vtotal Previous** Access: RO Vtotal from previous frame.

TRANS_VSYNC

TRANS_VSYNC	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B014h-6B017h
Name:	Transcoder DSI 0 Vertical Sync
ShortName:	TRANS_VSYNC_DSI0
Reset:	soft
Address:	6B814h-6B817h
Name:	Transcoder DSI 1 Vertical Sync
ShortName:	TRANS_VSYNC_DSI1
Reset:	soft
Address:	60014h-60017h
Name:	Transcoder Vertical Sync
ShortName:	TRANS_VSYNC_A
Reset:	soft
Address:	61014h-61017h
Name:	Transcoder Vertical Sync
ShortName:	TRANS_VSYNC_B
Reset:	soft
Address:	62014h-62017h
Name:	Transcoder Vertical Sync
ShortName:	TRANS_VSYNC_C
Reset:	soft
Address:	63014h-63017h
Name:	Transcoder Vertical Sync
ShortName:	TRANS_VSYNC_D
Reset:	soft

Restriction
This register should not be changed while the transcoder or port are enabled.

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ

<table>
<tr><th colspan="3">TRANS_VSYNC</th></tr>
<tr><td></td><td>28:16</td><td>Vertical Sync End
This field specifies the Vertical Sync End position relative to the vertical active display start. It is programmed with VerticalActive+FrontPorch+Sync-1
<table><tr><th>Restriction</th></tr><tr><td>This value must be greater than the vertical sync start and less than Vertical Total.</td></tr></table></td></tr>
<tr><td></td><td>15:13</td><td>Reserved
<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>12:0</td><td>Vertical Sync Start
This field specifies the Vertical Sync Start position relative to the vertical active display start. It is programmed with VerticalActive+FrontPorch-1
<table><tr><th>Restriction</th></tr><tr><td>This value must be greater than Vertical Active.</td></tr></table></td></tr>
</table>

TRANS_VSYNCSHIFT

TRANS_VSYNCSHIFT	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6B028h-6B02Bh
Name:	Transcoder DSI 0 Vertical Sync Shift
ShortName:	TRANS_VSYNCSHIFT_DSI0
Reset:	soft
Address:	6B828h-6B82Bh
Name:	Transcoder DSI 1 Vertical Sync Shift
ShortName:	TRANS_VSYNCSHIFT_DSI1
Reset:	soft
Address:	60028h-6002Bh
Name:	Transcoder Vertical Sync Shift
ShortName:	TRANS_VSYNCSHIFT_A
Reset:	soft
Address:	61028h-6102Bh
Name:	Transcoder Vertical Sync Shift
ShortName:	TRANS_VSYNCSHIFT_B
Reset:	soft
Address:	62028h-6202Bh
Name:	Transcoder Vertical Sync Shift
ShortName:	TRANS_VSYNCSHIFT_C
Reset:	soft
Address:	63028h-6302Bh
Name:	Transcoder Vertical Sync Shift
ShortName:	TRANS_VSYNCSHIFT_D
Reset:	soft

Restriction
This register should not be changed while the transcoder or port are enabled.

DWord	Bit	Description
0	31:13	**Reserved** Access: RO Format: MBZ

TRANS_VSYNCSHIFT		
	12:0	**Second Field VSync Shift** This value specifies the vertical sync alignment for the start of the interlaced second field, expressed in terms of the absolute pixel number relative to the horizontal active display start. This value will only be used if the transcoder is programmed to an interlaced mode. Typically, the interlaced second field vertical sync should start one pixel after the point halfway between successive horizontal syncs, so the value of this register should be programmed = horizontal sync start - floor[horizontal total / 2]Calculate using the actual horizontal sync start and horizontal total values and not the minus one values programmed into the registers. This vertical sync shift only occurs during the interlaced second field. In all other cases the vertical sync start position is aligned with horizontal sync start.

TRANS_VTOTAL

TRANS_VTOTAL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6E00Ch-6E00Fh
Name:	Transcoder WD0 Vertical Total
ShortName:	TRANS_VTOTAL_WD0
Reset:	soft
Address:	6E80Ch-6E80Fh
Name:	Transcoder WD1 Vertical Total
ShortName:	TRANS_VTOTAL_WD1
Reset:	soft
Address:	6B00Ch-6B00Fh
Name:	Transcoder DSI 0 Vertical Total
ShortName:	TRANS_VTOTAL_DSI0
Reset:	soft
Address:	6B80Ch-6B80Fh
Name:	Transcoder DSI 1 Vertical Total
ShortName:	TRANS_VTOTAL_DSI1
Reset:	soft
Address:	6000Ch-6000Fh
Name:	Transcoder Vertical Total
ShortName:	TRANS_VTOTAL_A
Reset:	soft
Address:	6100Ch-6100Fh
Name:	Transcoder Vertical Total
ShortName:	TRANS_VTOTAL_B
Reset:	soft
Address:	6200Ch-6200Fh
Name:	Transcoder Vertical Total
ShortName:	TRANS_VTOTAL_C
Reset:	soft
Address:	6300Ch-6300Fh
Name:	Transcoder Vertical Total
ShortName:	TRANS_VTOTAL_D

TRANS_VTOTAL

Reset: soft

Programming Notes

PSR ASU: This register is double buffered when Vactive double buffer bit (bit[8])is asserted in TRANS_DDI_FUNC_CTL2.

Restriction

This register should not be changed while the transcoder or port are enabled.

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**Vertical Total** **Description** This field specifies Vertical Total size. This should be equal to the sum of the vertical active and the vertical blank sizes. For progressive display modes, this field is programmed to the number of lines desired minus one. For interlaced display modes, this field is programmed with the number of lines desired minus two. The vertical counter is incremented on the leading edge of the horizontal sync. Both even and odd vertical totals are supported. This field is ignored by WD transcoders. Vertical total is double buffered. It updates immediately when transcoder is off. Otherwise, it updates at VBLANK rising edge. **Restriction** This register must always be programmed to the same value as the Vertical Blank End.
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**Vertical Active** This field specifies Vertical Active Display size. The first vertical active display line is considered line number# 0. This field is always programmed to the number of lines desired minus one. Restriction : When using the internal panel fitting logic, the minimum vertical active area must be seven lines. This register must always be programmed to the same value as the Vertical Blank Start.

TRANS_WD_FUNC_CTL

TRANS_WD_FUNC_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6E400h-6E403h
Name:	Transcoder WD0 Function Control
ShortName:	TRANS_WD_FUNC_CTL_0
Reset:	soft
Address:	6EC00h-6EC03h
Name:	Transcoder WD1 Function Control
ShortName:	TRANS_WD_FUNC_CTL_1
Reset:	soft

DWord	Bit	Description
0	31	**WD Function Enable** This bit enables the WD function. Value: 0b = Disable; 1b = Enable
	30	**Triggered Capture Mode Enable** This field enables the triggered capture mode where a frame is only captured after the Start Trigger Frame bit is written with 1, and hardware will ignore the transcoder frame time. This must be set before or when WD Function Enable is set. When triggered capture mode is disabled hardware will periodically capture frames following the transcoder frame time. Value: 0b = Disable; 1b = Enable
	29	**Start Trigger Frame** Access: R/W Set Write a 1 to this field to start a software triggered frame capture when Triggered Capture Mode Enable is 1. Hardware will clear the field when the frame starts.
	28	**Stop Trigger Frame** Access: R/W Set Write a 1 to this field to stop a software triggered frame capture when Triggered Capture Mode Enable is 1. Hardware will clear the field when the frame stops. This is only intended for use in case of an error where capture is never completing and software times out. It must not be set at the same time as Start Trigger Frame. After a stop trigger, VDenc will be out of sync with WD and also need to be stopped. WD and

TRANS_WD_FUNC_CTL

VDenc then need to start from the same frame number.

27 Reserved

26 Chroma Filtering Enable

This field selects how U and V are downsampled from YUV 444 to 422. This field only applies to the YUV 422 formats.

Value	Name	Description
0	Drop	Drop U2 and V2
1	Filter **[Default]**	Use a 15-34-15 three tap filter

25:23 Reserved

Access:	RO
Format:	MBZ

22:20 WD Color Mode

This field selects the capture color format.

Value	Name	Description
000b	YUV 4:4:4	YUV 32-bit 4:4:4 packed (8:8:8:8 MSB-Y:U:X:V)
001b	YUV 4:2:2	YUV 16-bit 4:2:2 packed (8:8:8:8 MSB- Y1:U:Y2:V) Chroma downsampling is programmable according to the Chroma Filtering field.
010b	XYUV 4:4:4	YUV 32-bit 4:4:4 packed (8:8:8:8 MSB-X:Y:U:V)
011b	RGBX	RGBX 32-bit (8:8:8:8 MSB-X:B:G:R)
100b	Y410	YUV 444 10bpc (MSB-X:V:Y:U)
101b	YUY2 8b	8 bit YUV 422 (MSB-V:Y2:U:Y1) Chroma downsampling is programmable according to the Chroma Filtering field.
110b	RGB10	RGB1010102 (MSB-X:B:G:R)

Restriction
This field must not be changed while the function is enabled.

19:18 Control Pointers

This field controls which pointers are sent and followed. If the head pointer is ignored, then the transcoder captures frames without any pointer comparison to stall the capture.

Value	Name	Description
00b	Enable Tail and Head	Send tail pointer to GT. Follow head pointer from GT.
01b	Enable Tail, Disable Head	Send tail pointer to GT. Ignore head pointer from GT. Non-cacheable.
11b	Disable Tail and Head	Do not send tail pointer to GT. Ignore head pointer from GT. Non-cacheable.

TRANS_WD_FUNC_CTL

17:16 VDenc Session Select

This field selects the encode session. Each enabled WD transcoder must select a unique session. It is not valid to have multiple WD transcoders select the same session.

Value	Name
00b	0
01b	1
10b	2
11b	3

15 Reserved

Access:	RO
Format:	MBZ

14:12 WD Input Select

These bits determine the input to WD.

Value	Name
000b	Pipe A
101b	Pipe B
110b	Pipe C
111b	Pipe D
Others	Reserved

Restriction
This field must not be changed while the function is enabled. It is not valid to have the same pipes driving multiple enabled transcoders.

11:4 Reserved

3:0 Frame Number

SW provided frame number. This is sent in the tail pointer message to media, to be used for synchronizing the encode with the display frame.

TRNULLDETCT

REG_TEMPLATE - TRNULLDETCT	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	04DE8h
Name:	TiledResources Null Tile Detection Register
ShortName:	TRNULLDETCT

DWord	Bit	Description
0	31:0	**Null Tile Detection Value** Access: R/W _Custom_GTIReset: DEV **Value** / **Name** / **Description** 00000000h / **Default]** / A 32bit value programmed to enable h/w to perform a match of TR-VA TT entries to detect Null Tiles. Hardware will flag each entry and space behind it as Null Tile for matched entries.

TSEG Base Memory

TSEGMB - TSEG Base Memory

Register Space:	MMIO: 0/2/0
Size (in bits):	64
SOC_Consumer:	BIOS
Address:	108400h

This 64 bit register defines the TSEG Base.

DWord	Bit	Description
0..1	63:20	**TSEG Memory Base** Access: R/W _Custom_GTIReset: BUS This register contains the base address of TSEG DRAM memory. Bios must program the value of TSEGMB to be the same as BGSM when TSEG is disabled. Value: 000000000000b — Name: **[Default]**
	19:0	**Reserved** Access: RO Format: MBZ

Ungated Clock Counter for DFR Testability

SAMPLER_DFR_UNGATED_COUNT - Ungated Clock Counter for DFR Testability

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0E148h

For testability of DFR feature

DWord	Bit	Description
0	31:0	**Counter Bits** Format: U32 Count of full-speed sampler clocks

Unslice unit Level Clock Gating Control 943C

UNSLCGCTL943C - Unslice unit Level Clock Gating Control 943C

Register Space: MMIO: 0/2/0
Size (in bits): 32
CustomGTIIs_ContextSaved: true

Address: 0943Ch

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31	**hprunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS hprunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	30	**Reserved**
	29:26	**Reserved** Access: RO Format: MBZ
	25	**mmcdunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS mmcdunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	24	**bfceunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS bfceunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	23	**ecpunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS ecpunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

UNSLCGCTL943C - Unslice unit Level Clock Gating Control 943C

	22	**vdlunit1 Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS vdlunit1 Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	21	**vhmeunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS vhmeunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	20	**vimeunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS vimeunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	19	**vcreunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS vcreunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	18	**vdxunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS vdxunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	17	**mdcunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS mdcunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)

UNSLCGCTL943C - Unslice unit Level Clock Gating Control 943C

		'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	16	**hpounit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS hpounit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	15	**hrsunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS hrsunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	14	**ftunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS ftunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	13	**fqunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS fqunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	12	**hleunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS hleunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	11	**hlcunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS hlcunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for

UNSLCGCTL943C - Unslice unit Level Clock Gating Control 943C

		functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	10	**hhiunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS hhiunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	9	**mlfunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS mlfunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	8	**mmcunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS mmcunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	7	**mbdunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS mbdunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	6	**mpdunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS mpdunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	5	**mmxunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS

UNSLCGCTL943C - Unslice unit Level Clock Gating Control 943C

		mmxunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	4	**hedunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS hedunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	3	**hlfunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS hlfunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	2	**hmcunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS hmcunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	1	**hmxunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS hmxunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	0	**hppunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS hppunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

Unslice unit Level Clock Gating Control 9430

UNSLCGCTL9430 - Unslice unit Level Clock Gating Control 9430

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	09430h

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29	**oaal2 Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS oaal2 Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	28	**oaal3 Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS oaal3 Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	27	**mertgart Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS mertgart Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	26	**RCPBLAT Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS RCPBLAT Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

UNSLCGCTL9430 - Unslice unit Level Clock Gating Control 9430

25	**RCPBARB Clock Gating Disable**
	Access: R/W _Custom_GTIReset: BUS
	RCPBARB Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
24	**LNIC Clock Gating Disable**
	Access: R/W _Custom_GTIReset: BUS
	LNIC Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
23	**LNIB Clock Gating Disable**
	Access: R/W _Custom_GTIReset: BUS
	LNIB Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
22	**Reserved**
	Access: R/W _Custom_GTIReset: BUS
	Reserved
21	**Reserved**
	Access: RO Format: MBZ
20	**SARB Clock Gating Disable**
	Access: R/W _Custom_GTIReset: BUS
	SARB Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
19	**DAP Clock Gating Disable**
	Access: R/W _Custom_GTIReset: BUS
	DAP Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)

UNSLCGCTL9430 - Unslice unit Level Clock Gating Control 9430

		'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	18	**RCC Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS RCC Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	17	**RAMRCC Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS RAMRCC Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	16	**Reserved** Access: R/W _Custom_GTIReset: BUS Reserved
	15	**SRCPBPIX Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SRCPBPIX Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	14	**SCFE Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SCFE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	13	**LNEP Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS LNEP Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

UNSLCGCTL9430 - Unslice unit Level Clock Gating Control 9430

	12	**MSQC wrapper 2xclk Clock Gating Disable** Default Value: 1b Access: R/W _Custom_GTIReset: BUS MSQC wrapper 2xclk Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	11	**MSQC wrapper Clock Gating Disable** Default Value: 1b Access: R/W _Custom_GTIReset: BUS MSQC wrapper Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	10	**GCPunit Clock Gating Disable** Default Value: 1b Access: R/W _Custom_GTIReset: BUS GCPunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	9	**SPARE6 Clock Gating Disable** Default Value: 1b Access: R/W _Custom_GTIReset: BUS SPARE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	8	**SPARE5 Clock Gating Disable** Default Value: 1b Access: R/W _Custom_GTIReset: BUS SPARE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

UNSLCGCTL9430 - Unslice unit Level Clock Gating Control 9430

7	**SPARE4 Clock Gating Disable**

Default Value:	1b
Access:	R/W
_Custom_GTIReset:	BUS

SPARE Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

6	**GPMunit Clock Gating Disable**

Default Value:	1b
Access:	R/W
_Custom_GTIReset:	BUS

GPMunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

5	**MBGFunit Clock Gating Disable**

Default Value:	1b
Access:	R/W
_Custom_GTIReset:	BUS

MBGFunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

4	**MSQDunit Clock Gating Disable**

Default Value:	1b
Access:	R/W
_Custom_GTIReset:	BUS

MSQDunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

3	**MSQDunit 2xclk Clock Gating Disable**

Default Value:	1b
Access:	R/W
_Custom_GTIReset:	BUS

MSQDunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

UNSLCGCTL9430 - Unslice unit Level Clock Gating Control 9430

	2	**MISDunits 2x Clock Gating Disable** Default Value: 1b Access: R/W _Custom_GTIReset: BUS MISDunits Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	1	**MISDunit Clock Gating Disable** Default Value: 1b Access: R/W _Custom_GTIReset: BUS MISDunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	0	**MCRunit Clock Gating Disable** Default Value: 1b Access: R/W _Custom_GTIReset: BUS MCRunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

Unslice unit Level Clock Gating Control 9434

UNSLCGCTL9434 - Unslice unit Level Clock Gating Control 9434

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	09434h

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31	**GTIunit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS GTIunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always) Value: 1b — Name: **[Default]**
	30	**Reserved** Access: RO Format: MBZ
	29	**GAMunit Clock Gating Disable** Default Value: 1b Access: R/W Format: Disable _Custom_GTIReset: BUS GAMunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	28	**GABunit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS GABunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always) Value: 1b — Name: **[Default]**

UNSLCGCTL9434 - Unslice unit Level Clock Gating Control 9434

27	**DTunit Clock Gating Disable**

Access:	R/W
Format:	Disable
_Custom_GTIReset:	BUS

DTunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

26	**Reserved**

Access:	RO
Format:	MBZ

25	**BLSunit Clock Gating Disable**

Access:	R/W
Format:	Disable
_Custom_GTIReset:	BUS

BLSunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

24	**BLBunit Clock Gating Disable**

Access:	R/W
Format:	Disable
_Custom_GTIReset:	BUS

BLBunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

23	**BFunit Clock Gating Disable**

Access:	R/W
Format:	Disable
_Custom_GTIReset:	BUS

BFunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

22	**BDunit Clock Gating Disable**

Access:	R/W
Format:	Disable
_Custom_GTIReset:	BUS

BDunit Clock Gating Disable Control:

UNSLCGCTL9434 - Unslice unit Level Clock Gating Control 9434

		'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	21	**BCSunit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS BCSunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	20	**gamtlbhitarb Clock Gating Disable** Default Value: 1b Access: R/W Format: Disable _Custom_GTIReset: BUS gamtlbhitarb Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	19	**DTOunit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS DTOunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	18	**gamtlbmisarb Clock Gating Disable** Default Value: 1b Access: R/W Format: Disable _Custom_GTIReset: BUS gamtlbmisarb Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	17:14	**Reserved** Access: RO Format: MBZ

UNSLCGCTL9434 - Unslice unit Level Clock Gating Control 9434

13	**RCPBunit Clock Gating Disable**

Access:	R/W
Format:	Disable
_Custom_GTIReset:	BUS

RCPBunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

12	**QCunit Clock Gating Disable**

Access:	R/W
Format:	Disable
_Custom_GTIReset:	BUS

QCunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

11:8	**Reserved**

Access:	RO
Format:	MBZ

7	**cg3ddisvfgr Clock Gating Disable**

Access:	R/W
Format:	Disable
_Custom_GTIReset:	BUS

cg3ddisvfgr Clock Gating Disable
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

6	**MSPBISTunit Clock Gating Disable**

Access:	R/W
Format:	Disable
_Custom_GTIReset:	BUS

MSPBISTunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

5	**OACSunit Clock Gating Disable**

Access:	R/W
Format:	Disable
_Custom_GTIReset:	BUS

OACSunit Clock Gating Disable Control:

UNSLCGCTL9434 - Unslice unit Level Clock Gating Control 9434

		'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	4	**NOAunit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS NOAunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	3	**OARUNIT Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS OARUNIT Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	2	**VMEunit Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS VMEunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	1	**BCS BE Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS BCS BE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	0	**BCS FE Clock Gating Disable** Access: R/W Format: Disable _Custom_GTIReset: BUS BCS FE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for

UNSLCGCTL9434 - Unslice unit Level Clock Gating Control 9434		
		functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

Unslice unit Level Clock Gating Control 9438

UNSLCGCTL9438 - Unslice unit Level Clock Gating Control 9438

Register Space: MMIO: 0/2/0
Size (in bits): 32
CustomGTIIs_ContextSaved: true

Address: 09438h

Unit Level Clock Gating Disable bits.

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30:29	**MSQCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS MSQCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always) Value: 11b — Name: **[Default]**
	28	**VCOPunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VCOPunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	27	**LNIunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS LNIunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	26	**LNEUNIT Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS LNEUNIT Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for

UNSLCGCTL9438 - Unslice unit Level Clock Gating Control 9438

		functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	25:20	**Reserved** Access: RO Format: MBZ
	19	**RPM Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS RPM Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	18	**OASC Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS OASC Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	17	**VECS Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VECS Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	16	**GAHSV Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GAHSV Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	15	**GAHSD Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GAHSD Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

UNSLCGCTL9438 - Unslice unit Level Clock Gating Control 9438

14	**Reserved** Access: RO Format: MBZ
13	**GAPSL Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GAPSL Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
12	**GAPSU Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GAPSU Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
11	**SPMunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SPMunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
10	**MUCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS MUCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
9	**Reserved** Access: RO Format: MBZ
8	**GUC Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GUC Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

UNSLCGCTL9438 - Unslice unit Level Clock Gating Control 9438		
	7:0	**Reserved** Access: RO Format: MBZ

Unslice unit Level Clock Gating Control 9440

UNSLCGCTL9440 - Unslice unit Level Clock Gating Control 9440

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	09440h

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28	**gamtlboacs Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS gamtlboacs Clock Gating Disable Control: 1b – Clock Gating Disabled **[Default]** – Clocks are toggling, always. 0b – Clock Gating Enabled – Clocks can be gated when they are not required to toggle for functionality.
	27	**gamtlbgfx3 Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS gamtlbgfx3 Clock Gating Disable Control: 1b – Clock Gating Disabled **[Default]** – Clocks are toggling, always. 0b – Clock Gating Enabled – Clocks can be gated when they are not required to toggle for functionality.
	26	**gamtlbgfx2 Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS gamtlbgfx2 Clock Gating Disable Control: 1b – Clock Gating Disabled **[Default]** – Clocks are toggling, always. 0b – Clock Gating Enabled – Clocks can be gated when they are not required to toggle for functionality.

UNSLCGCTL9440 - Unslice unit Level Clock Gating Control 9440

25 **gamdrtnunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

gamdrtnunit Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

24 **gamtlbgfx1 Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

gamtlbgfx1 Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

23 **gamtlbgfx0 Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

gamtlbgfx0 Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

22 **gamtlbvebox0 Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

gamtlbvebox0 Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

21 **gamtlbvdbox2 Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

UNSLCGCTL9440 - Unslice unit Level Clock Gating Control 9440

gamtlbvdbox2 Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

20 **SPARE Clock Gating Disable2**

Access:	R/W
_Custom_GTIReset:	BUS

SPARE Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

19 **SPARE Clock Gating Disable1**

Access:	R/W
_Custom_GTIReset:	BUS

SPARE Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

18 **LSNEPunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

LSNEPUNIT Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

17 **gamtlbvdbox0 Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

gamtlbvdbox0 Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled	Clocks are toggling, always.

UNSLCGCTL9440 - Unslice unit Level Clock Gating Control 9440

	[Default]	
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

16 **gamtlbkcr Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

gamtlbkcr Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

15 **gamtlbguc Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

gamtlbguc Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

14 **gamtlbblt Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

gamtlbblt Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

13 **gamtrtlb Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

gamtrtlb Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

UNSLCGCTL9440 - Unslice unit Level Clock Gating Control 9440

12 **gamstlb Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

gamstlb Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

11 **gamccs Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

gamccs Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

10 **GACFG Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

CACFG Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

9 **GAVARBunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

GAVARBunit Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

8 **gamctrl Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

UNSLCGCTL9440 - Unslice unit Level Clock Gating Control 9440

gamctrl Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

7 **gamcmdi Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

gamcmdi Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

6 **Reserved**

Access:	RO
Format:	MBZ

5 **gamwkrs Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

gamwkrs Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

4 **gamdati Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

gamdati Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

UNSLCGCTL9440 - Unslice unit Level Clock Gating Control 9440

3	**L3_CR Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

L3_CR Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

2	**gamreqstrm Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

gamreqstrm Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

1	**ramdft Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

ramdft Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

0	**hwmunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

hwmunit Clock Gating Disable Control:

Value	Name	Description
1b	Clock Gating Disabled **[Default]**	Clocks are toggling, always.
0b	Clock Gating Enabled	Clocks can be gated when they are not required to toggle for functionality.

Unslice unit Level Clock Gating Control 9444

UNSLCGCTL9444 - Unslice unit Level Clock Gating Control 9444

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	09444h

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31:4	**Reserved** Access: RO Format: MBZ
	3	**KCRunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS KCRunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	2	**WCRunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS WCRunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	1	**lsnunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS lsnunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	0	**warbunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS warbunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

Unslice unit Level Clock Gating Control 9448

UNSLCGCTL9448 - Unslice unit Level Clock Gating Control 9448	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	09448h
Unslice unit Level Clock Gating Control 9448 Unit Level Clock Gating Disable bits	

DWord	Bit	Description
0	31:0	**ECO Spare Bits** Access: R/W _Custom_GTIReset: BUS Reserved.

Unslice unit Level Clock Gating Control 9450

UNSLCGCTL9450 - Unslice unit Level Clock Gating Control 9450

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	09450h

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29	**URBunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS URBunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	28:27	**Reserved** Access: RO Format: MBZ
	26	**RCPBEunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS RCPBEunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	25	**GAMunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GAMunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	24	**HDCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HDCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for

UNSLCGCTL9450 - Unslice unit Level Clock Gating Control 9450

	functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
23	**Reserved** Access: RO Format: MBZ
22	**BLBunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS BLBunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
21	**BFunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS BFunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
20	**MUCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS MUCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
19	**WVISunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS WVISunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
18	**WAVM Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS WAVM Clock Gating Disable Control:

UNSLCGCTL9450 - Unslice unit Level Clock Gating Control 9450

		'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	17	**WHME Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS WHME Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	16	**WIME Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS WIME Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	15	**WMPC Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS WMPC Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	14	**Reserved** Access: RO Format: MBZ
	13	**VSHM Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VSHM Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

UNSLCGCTL9450 - Unslice unit Level Clock Gating Control 9450

12	**DAPRTS Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS DAPRTS Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
11	**Reserved** Access: RO Format: MBZ
10	**GUC Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GUC Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
9:6	**Reserved** Access: RO Format: MBZ
5	**vdlunit1 Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS vdlunit1 Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
4	**vhmeunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS vhmeunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

UNSLCGCTL9450 - Unslice unit Level Clock Gating Control 9450

	3	**vcreunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS vcreunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	2	**hleunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS hleunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	1	**mbdunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS mbdunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	0	**mmxunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS mmxunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

Unslice unit Level Clock Gating Control 9454

UNSLCGCTL9454 - Unslice unit Level Clock Gating Control 9454

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	09454h

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31:22	**Reserved** Access: RO Format: MBZ
	21	**SPARE Clock Gating Disable2** Access: R/W _Custom_GTIReset: BUS SPARE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	20	**ram gamtm2 unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS gamtm2 unit Clock Gating Disable Control for ram: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
	19	**ram gamtm1 unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS gamtm1 unit Clock Gating Disable Control for ram: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
	18	**VDIunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VDIunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)

UNSLCGCTL9454 - Unslice unit Level Clock Gating Control 9454

	'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
17:14	**Reserved** Access: RO Format: MBZ
13	**GAMTGunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GAMTGunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
12	**GAMTOunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GAMTOunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
11:10	**Reserved** Access: RO Format: MBZ
9	**mpdunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS mpdunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
8	**hedunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS hedunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

UNSLCGCTL9454 - Unslice unit Level Clock Gating Control 9454

	7	**hlfunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS hlfunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	6	**hmcunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS hmcunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	5	**hmxunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS hmxunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	4	**hppunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS hppunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	3	**hprunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS hprunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

UNSLCGCTL9454 - Unslice unit Level Clock Gating Control 9454

	2	**hucunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS hucunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	1	**hwmunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS hwmunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	0	**mdcunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS mdcunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

uOS Full Hash High

UOS_FULL_HASH_HI - uOS Full Hash High	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	128

The Full Hash being generated for the uOS. This register is written by Shim hardware. The update happens as the Full Hash is being generated and compared during the GfxMem --> WOPCM DMA operation. IA Writes have no effect.

Programming Notes
This register is saved in the power context

DWord	Bit	Description
0..3	127:0	**Hash**

uOS Full Hash Low

UOS_FULL_HASH_LO - uOS Full Hash Low	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	128

The Full Hash being generated for the uOS. This register is written by Shim hardware. The update happens as the Full Hash is being generated and compared during the GfxMem --> WOPCM DMA operation. IA Writes have no effect.

Programming Notes
This register is saved in the power context

DWord	Bit	Description
0..3	127:0	**Hash**

URB Runtime ECC capture Register

URB_ECC_CAPTURE_REG - URB Runtime ECC capture Register

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B14Ch

This register shows the captured value of the ECC calculated by the internal logic for URB writes.

DWord	Bit	Description
0	31:20	**Reserved** Access: RO Format: MBZ
	19:0	**URB Latched ECC Value** Access: RO This field holds the value of the ECC generated by the URB pipeline for the preceeding write cycle.

URB Runtime ECC Test Control Register

URB_ECC_TEST_CTL - URB Runtime ECC Test Control Register

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B148h

This register is used to control the run time testing of the ECC logic in the URB pipeline.

DWord	Bit	Description
0	31:22	**Reserved** Access: RO Format: MBZ
	21	**URB ECC Latch Enable** Access: R/W This bit enables the latching of the ECC value generated for any write access made to the URB. The latched value of the ECC can be read from the "URB Cache Runtime ECC capture Register". 0 – **[Default]** – Disable latching. 1 – Enable the latching
	20	**URB ECC Override Enable** Access: R/W This bit enables the overriding of the ECC value generated for any write access made to the URB. The value in the field below will be used to override the ECC internally generated. 0 – **[Default]** – Disable override of the ECC value 1 – Enable the override of the ECC value
	19:0	**URB ECC Override Value** Default Value: 00h Access: R/W This is the value to be used as the override for the ECC when enabled in the bit field above.

UTIL_PIN_BUF_CTL

UTIL_PIN_BUF_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	48404h-48407h
Name:	Utility Pin Buffer Control
ShortName:	UTIL_PIN_BUF_CTL
Reset:	soft

This register controls the display utility pin I/O buffer.

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29:28	**Hysteresis**
	27	**Reserved** Access: RO Format: MBZ
	26:24	**Spare**
	23:21	**Reserved** Access: RO Format: MBZ
	20:16	**Pulldown Strength**
	15:12	**Pulldown Slew**
	11:9	**Reserved** Access: RO Format: MBZ
	8:4	**Pullup Strength**
	3:0	**Pullup Slew**

UTIL_PIN_CTL

UTIL_PIN_CTL

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	48400h-48403h
Name:	Utility Pin Control
ShortName:	UTIL_PIN_CTL
Reset:	soft

This register controls the display utility pin. The maximum switching frequency is 100 KHz.

DWord	Bit	Description
0	31	**Util Pin Enable** This bit enables the utility pin. Value: 0b = Disable; 1b = Enable
	30:29	**Pipe Select** This bit selects which pipe will be used when the utility pin is outputting timing related signals. Value: 00b = Pipe A; 01b = Pipe B; 10b = Pipe C; 11b = Pipe D Restriction: The field should only be changed when the utility pin is disabled or not configured to use any timing signals.
	28	**Reserved** Access: RO Format: MBZ
	27:24	**Util Pin Mode** This bit configures the utility pin mode of operation for output. 0000b Data: Output the Util_Pin_Output_Data value. 0001b PWM: Output from the backlight PWM circuit. 0100b Vblank: Output the vertical blank. [] This is the pipe delayed vblank. 0101b Vsync: Output the vertical sync.

UTIL_PIN_CTL

0110b	framestart	Output the framestart
1000b	Right/Left Eye Level	Output the stereo 3D right/left eye level signal. Asserted for the left eye and de-asserted for the right eye.
Others	Reserved	Reserved

Restriction
The field should only be changed when the utility pin is disabled.

23 **Util Pin Output Data**

This bit selects what the value to drive as an output when in the data mode.

Value	Name
0b	0
1b	1

22 **Util Pin Output Polarity**

This bit inverts the polarity of the pin output.

Value	Name
0b	Not inverted
1b	Inverted

21:20 **Reserved**

Access:	RO
Format:	MBZ

19 **Util Pin Direction**

This bit selects whether the pin is used as an output or an input.

Value	Name
0b	Output
1b	Input

Restriction
The field should only be changed when the utility pin is disabled.

18:17 **Reserved**

Access:	RO
Format:	MBZ

16 **Util Pin Input Data**

Access:	RO

This bit gives the value received on the pin. This is only valid when the utility pin is enabled and the direction is input.

15:0 **Reserved**

Access:	RO
Format:	MBZ

UTIL2_PIN_BUF_CTL

UTIL2_PIN_BUF_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	4840Ch-4840Fh
Name:	Utility2 Pin Buffer Control
ShortName:	UTIL2_PIN_BUF_CTL
Reset:	soft

This register controls the display utility2 pin I/O buffer.

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29:28	**Hysteresis**
	27	**Reserved** Access: RO Format: MBZ
	26:24	**Spare**
	23:21	**Reserved** Access: RO Format: MBZ
	20:16	**Pulldown Strength**
	15:12	**Pulldown Slew**
	11:9	**Reserved** Access: RO Format: MBZ
	8:4	**Pullup Strength**
	3:0	**Pullup Slew**

UTIL2_PIN_CTL

UTIL2_PIN_CTL

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	48408h-4840Bh
Name:	Utility2 Pin Control
ShortName:	UTIL2_PIN_CTL
Reset:	soft

This register controls the display utility2 pin.

DWord	Bit	Description
0	31	**Util Pin Enable** This bit enables the utility pin. Value: 0b = Disable; 1b = Enable
	30	**Util Pin Direction** This bit selects whether the pin is used as an output or an input. Value: 0b = Output; 1b = Input Restriction: The field should only be changed when the utility pin is disabled.
	29:0	**Reserved** Access: RO Format: MBZ

Valid Bit Vector 3 for RCC Register

CCTLB_VLD_3 - Valid Bit Vector 3 for RCC Register		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
Address:	04DACh	
This register contains the valid bits for entries 0-31 of RCCTLB.		
DWord	**Bit**	**Description**
0	31:0	**Valid Bit Vector 3 for RCC** Default Value: 00000000h Access: RO _Custom_GTIReset: DEV Valid Bits per Entry.

VCS CSB

VCS_CSB - VCS CSB		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
SW reads this offset to read the CSB entry at the top of the CSB fifo. Reads must occur in pairs to obtain a single 64 bit CSB entry. The second read pops the entry off the CSB fifo.		
DWord	**Bit**	**Description**
0	31:0	**Context Status Buffer DW** Access: RO This DW holds CSB bits[31:0] for the first read and CSB bits[63:32] for the second read.

VCS CSB Fifo Status Register

VCS_CSB_FSR - VCS CSB Fifo Status Register		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
This RO register holds status of the CSB fifo.		
DWord	**Bit**	**Description**
0	31	**Not Empty** Default Value: 000000000000000b Access: RO
	30:16	**Reserved** Access: RO Format: MBZ
	15:8	**FIFO Maximum Occupancy Count** This field is a read-only field. It reflects the depth of the FIFO, which is a static value for each product.
	7:0	**Fifo Occupancy Count** Access: RO

VDBox TLB Invalidation Register

VD_TLB_INV_CR - VDBox TLB Invalidation Register

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0CEDCh

DWord	Bit	Description
0	31:16	**Mask Bits** Default Value: 0000000000000000b Access: R/W
	15:8	**Reserved** Access: RO Format: MBZ
	7	**Invalidate VDBox TLBs bit7** Default Value: 0b Access: R/W SW writes 1 to invalidate the TLBs for the associated VDBox and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's HW pipeline is flushed and cleared from all its memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs.Bit[7:0] invalidates VDBox[7:0] TLBs. No Effect on non-existent VDBoxs. These bits self clear.
	6	**Invalidate VDBox TLBs bit6** Default Value: 0b Access: R/W SW writes 1 to invalidate the TLBs for the associated VDBox and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's HW pipeline is flushed and cleared from all its memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs.Bit[7:0] invalidates VDBox[7:0] TLBs. No Effect on non-existent VDBoxs. These bits self clear.
	5	**Invalidate VDBox TLBs bit5** Default Value: 0b Access: R/W SW writes 1 to invalidate the TLBs for the associated VDBox and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's HW pipeline is flushed and cleared from all its memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs.Bit[7:0] invalidates VDBox[7:0] TLBs. No Effect on non-existent VDBoxs. These bits self clear.

VD_TLB_INV_CR - VDBox TLB Invalidation Register

	4	**Invalidate VDBox TLBs bit4** Default Value: 0b Access: R/W SW writes 1 to invalidate the TLBs for the associated VDBox and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's HW pipeline is flushed and cleared from all its memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs.Bit[7:0] invalidates VDBox[7:0] TLBs. No Effect on non-existent VDBoxs. These bits self clear.
	3	**Invalidate VDBox TLBs bit3** Default Value: 0b Access: R/W SW writes 1 to invalidate the TLBs for the associated VDBox and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's HW pipeline is flushed and cleared from all its memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs.Bit[7:0] invalidates VDBox[7:0] TLBs. No Effect on non-existent VDBoxs. These bits self clear.
	2	**Invalidate VDBox TLBs bit2** Default Value: 0b Access: R/W SW writes 1 to invalidate the TLBs for the associated VDBox and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's HW pipeline is flushed and cleared from all its memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs.Bit[7:0] invalidates VDBox[7:0] TLBs. No Effect on non-existent VDBoxs. These bits self clear.
	1	**Invalidate VDBox TLBs bit1** Default Value: 0b Access: R/W SW writes 1 to invalidate the TLBs for the associated VDBox and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's HW pipeline is flushed and cleared from all its memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs.Bit[7:0] invalidates VDBox[7:0] TLBs. No Effect on non-existent VDBoxs. These bits self clear.
	0	**Invalidate VDBox TLBs bit0** Default Value: 0b Access: R/W SW writes 1 to invalidate the TLBs for the associated VDBox and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's HW pipeline is flushed and cleared from all its memory accesses.

VD_TLB_INV_CR - VDBox TLB Invalidation Register		
		Otherwise HW cannot guarantee the proper invalidation for TLBs.Bit[7:0] invalidates VDBox[7:0] TLBs. No Effect on non-existent VDBoxs. These bits self clear.

Vdbox unit Level Clock Gating Control 3F0C

VDCGCTL3F0C - Vdbox unit Level Clock Gating Control 3F0C

Register Space:	MMIO: 0/2/0
Size (in bits):	32
CustomGTIIs_ContextSaved:	true

Address:	1C3F0Ch-1C3F0Fh
Name:	VDbox registers4
ShortName:	VDCGCTL3F0C_VDBOX0

Address:	1C7F0Ch-1C7F0Fh
Name:	VDbox registers4
ShortName:	VDCGCTL3F0C_VDBOX1

Address:	1D3F0Ch-1D3F0Fh
Name:	VDbox registers4
ShortName:	VDCGCTL3F0C_VDBOX2

Address:	1D7F0Ch-1D7F0Fh
Name:	VDbox registers4
ShortName:	VDCGCTL3F0C_VDBOX3

Address:	1E3F0Ch-1E3F0Fh
Name:	VDbox registers4
ShortName:	VDCGCTL3F0C_VDBOX4

Address:	1E7F0Ch-1E7F0Fh
Name:	VDbox registers4
ShortName:	VDCGCTL3F0C_VDBOX5

Address:	1F3F0Ch-1F3F0Fh
Name:	VDbox registers4
ShortName:	VDCGCTL3F0C_VDBOX6

Address:	1F7F0Ch-1F7F0Fh
Name:	VDbox registers4
ShortName:	VDCGCTL3F0C_VDBOX7

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31	**SPARE Clock Gating Disable12** Access: R/W _Custom_GTIReset: BUS SPARE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for

VDCGCTL3F0C - Vdbox unit Level Clock Gating Control 3F0C

		functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	30	**SPARE Clock Gating Disable11** Access: R/W _Custom_GTIReset: BUS SPARE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	29	**SPARE Clock Gating Disable10** Access: R/W _Custom_GTIReset: BUS SPARE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	28	**SPARE Clock Gating Disable9** Access: R/W _Custom_GTIReset: BUS SPARE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	27	**VNCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VNCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	26	**VMXunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VMXunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	25	**VMTSunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VMTSunit Clock Gating Disable Control:

VDCGCTL3F0C - Vdbox unit Level Clock Gating Control 3F0C

		'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	24	**VMPCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VMPCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	23	**VMDunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VMDunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	22	**VMCRunit Clock Gating Disable** Default Value: 1b Access: R/W _Custom_GTIReset: BUS VMCRunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	21	**VMCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VMCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	20	**VMBunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VMBunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F0C - Vdbox unit Level Clock Gating Control 3F0C

	19	**VLFunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VLFunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	18	**VITunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VITunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	17	**VISunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VISunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	16	**VIPunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VIPunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	15	**VID6 Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VID6 Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F0C - Vdbox unit Level Clock Gating Control 3F0C

	14	**VID5 Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VID5 Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	13	**VID4 Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VID4 Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	12	**VID3 Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VID3 Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	11	**VID2 Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VID2 Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	10	**VID1 Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VID1 Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F0C - Vdbox unit Level Clock Gating Control 3F0C

9	**VIMEunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VIMEunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
8	**VHRunit's Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VHRunit's Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
7	**VHMEunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VHMEunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
6	**VFTunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VFTunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
5	**VDXunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VDXunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F0C - Vdbox unit Level Clock Gating Control 3F0C

	4	**VDSunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VDSunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	3	**vdl1unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS vdl1unit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	2	**Csunit's Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS Csunit's Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	1	**VCREunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VCREunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	0	**VCPunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VCPunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

Vdbox unit Level Clock Gating Control 3F04

VDCGCTL3F04 - Vdbox unit Level Clock Gating Control 3F04

Register Space:	MMIO: 0/2/0
Size (in bits):	32
CustomGTIIs_ContextSaved:	true

Address:	1C3F04h-1C3F07h
Name:	VDbox registers2
ShortName:	VDCGCTL3F04_VDBOX0

Address:	1C7F04h-1C7F07h
Name:	VDbox registers2
ShortName:	VDCGCTL3F04_VDBOX1

Address:	1D3F04h-1D3F07h
Name:	VDbox registers2
ShortName:	VDCGCTL3F04_VDBOX2

Address:	1D7F04h-1D7F07h
Name:	VDbox registers2
ShortName:	VDCGCTL3F04_VDBOX3

Address:	1E3F04h-1E3F07h
Name:	VDbox registers2
ShortName:	VDCGCTL3F04_VDBOX4

Address:	1E7F04h-1E7F07h
Name:	VDbox registers2
ShortName:	VDCGCTL3F04_VDBOX5

Address:	1F3F04h-1F3F07h
Name:	VDbox registers2
ShortName:	VDCGCTL3F04_VDBOX6

Address:	1F7F04h-1F7F07h
Name:	VDbox registers2
ShortName:	VDCGCTL3F04_VDBOX7

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31	**spare Clock Gating Disable4** Access: R/W _Custom_GTIReset: BUS SPARE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for

VDCGCTL3F04 - Vdbox unit Level Clock Gating Control 3F04

		functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	30	**spare Clock Gating Disable3** Access: R/W _Custom_GTIReset: BUS SPARE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	29	**spare Clock Gating Disable2** Access: R/W _Custom_GTIReset: BUS SPARE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	28	**HVSHAREunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HVSHAREunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	27	**HFTunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HFTunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	26	**HFQunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HFQunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F04 - Vdbox unit Level Clock Gating Control 3F04

	25	**HCRESunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HCRESunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	24	**HCREIunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HCREIunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	23	**HCREFunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HCREFunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	22	**MEDunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS MEDunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	21	**GACXunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GACXunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F04 - Vdbox unit Level Clock Gating Control 3F04

20	**GACunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

GACunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

19	**ECPunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

ECPunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

18	**BSPunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

BSPunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

17	**vmmunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

vmmunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

16	**VHLFunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

VHLFunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F04 - Vdbox unit Level Clock Gating Control 3F04

15	**VDKMXunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

VDKMXunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

14	**HWMunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

HWMunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

13	**HUCMXunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

HUCMXunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

12	**HUCunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

HUCunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

11	**HSSEunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

HSSEunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F04 - Vdbox unit Level Clock Gating Control 3F04

	10	**HSFunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HSFunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	9	**HPRunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HPRunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	8	**HPPunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HPPunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	7	**HMXFunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HMXFunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	6	**HMXBunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HMXBunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F04 - Vdbox unit Level Clock Gating Control 3F04

5	**HMCunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

HMCunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

4	**HITunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

HITunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

3	**HHLFunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

HHLFunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

2	**HFCunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

HFCunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

1	**HEDunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

HEDunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F04 - Vdbox unit Level Clock Gating Control 3F04

	0	**HBEunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HBEunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

Vdbox unit Level Clock Gating Control 3F08

VDCGCTL3F08 - Vdbox unit Level Clock Gating Control 3F08

Register Space:	MMIO: 0/2/0
Size (in bits):	32
CustomGTIIs_ContextSaved:	true

Address:	1C3F08h-1C3F0Bh
Name:	VDbox registers3
ShortName:	VDCGCTL3F08_VDBOX0

Address:	1C7F08h-1C7F0Bh
Name:	VDbox registers3
ShortName:	VDCGCTL3F08_VDBOX1

Address:	1D3F08h-1D3F0Bh
Name:	VDbox registers3
ShortName:	VDCGCTL3F08_VDBOX2

Address:	1D7F08h-1D7F0Bh
Name:	VDbox registers3
ShortName:	VDCGCTL3F08_VDBOX3

Address:	1E3F08h-1E3F0Bh
Name:	VDbox registers3
ShortName:	VDCGCTL3F08_VDBOX4

Address:	1E7F08h-1E7F0Bh
Name:	VDbox registers3
ShortName:	VDCGCTL3F08_VDBOX5

Address:	1F3F08h-1F3F0Bh
Name:	VDbox registers3
ShortName:	VDCGCTL3F08_VDBOX6

Address:	1F7F08h-1F7F0Bh
Name:	VDbox registers3
ShortName:	VDCGCTL3F08_VDBOX7

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31	**SPARE Clock Gating Disable8** Access: R/W _Custom_GTIReset: BUS SPARE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for

VDCGCTL3F08 - Vdbox unit Level Clock Gating Control 3F08

		functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	30	**SPARE Clock Gating Disable7** Access: R/W _Custom_GTIReset: BUS SPARE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	29	**SPARE Clock Gating Disable6** Access: R/W _Custom_GTIReset: BUS SPARE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	28	**spare Clock Gating Disable5** Access: R/W _Custom_GTIReset: BUS SPARE Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	27	**VClunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VClunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	26	**VCDunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VCDunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F08 - Vdbox unit Level Clock Gating Control 3F08

	25	**vbspunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS vbspunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	24	**VBPunits Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VBPunits Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	23	**VAMunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VAMunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	22	**VADuit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VADuit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	21	**VACunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VACunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F08 - Vdbox unit Level Clock Gating Control 3F08

	20	**USBunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS USBunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	19	**SECunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SECunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	18	**RDOFunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS RDOFunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	17	**RDOBunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS RDOBunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	16	**QRCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS QRCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F08 - Vdbox unit Level Clock Gating Control 3F08

	15	**MEDunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS MEDunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	14	**MPCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS MPCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	13	**MDCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS MDCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	12	**jusbunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS jusbunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	11	**JPGunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS JPGunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F08 - Vdbox unit Level Clock Gating Control 3F08

	10	**HWOPunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HWOPunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	9	**MARB Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS MARB Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	8	**HVDunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HVDunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	7	**HTQunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HTQunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	6	**HSAOunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HSAOunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F08 - Vdbox unit Level Clock Gating Control 3F08

	5	**HRSunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HRSunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	4	**HPOunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HPOunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	3	**HMDCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HMDCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	2	**HLEunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HLEunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	1	**HLCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HLCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F08 - Vdbox unit Level Clock Gating Control 3F08

	0	**HIMEunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HIMEunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

Vdbox unit Level Clock Gating Control 3F10

VDCGCTL3F10 - Vdbox unit Level Clock Gating Control 3F10

Register Space: MMIO: 0/2/0
Size (in bits): 32
CustomGTIIs_ContextSaved: true

Address: 1C3F10h-1C3F13h
Name: VDbox registers5
ShortName: VDCGCTL3F10_VDBOX0

Address: 1C7F10h-1C7F13h
Name: VDbox registers5
ShortName: VDCGCTL3F10_VDBOX1

Address: 1D3F10h-1D3F13h
Name: VDbox registers5
ShortName: VDCGCTL3F10_VDBOX2

Address: 1D7F10h-1D7F13h
Name: VDbox registers5
ShortName: VDCGCTL3F10_VDBOX3

Address: 1E3F10h-1E3F13h
Name: VDbox registers5
ShortName: VDCGCTL3F10_VDBOX4

Address: 1E7F10h-1E7F13h
Name: VDbox registers5
ShortName: VDCGCTL3F10_VDBOX5

Address: 1F3F10h-1F3F13h
Name: VDbox registers5
ShortName: VDCGCTL3F10_VDBOX6

Address: 1F7F10h-1F7F13h
Name: VDbox registers5
ShortName: VDCGCTL3F10_VDBOX7

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ

VDCGCTL3F10 - Vdbox unit Level Clock Gating Control 3F10

30	**VFWunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VFWunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
29	**VEOunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VEOunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
28	**ECSunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS ECSunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
27	**VDNunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VDNunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
26	**VDMunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VDMunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F10 - Vdbox unit Level Clock Gating Control 3F10

25 **VDIunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

VDIunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

24 **VCWunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

VCWunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

23 **VCUSunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

VCUSunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

22 **IECPunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

IECPunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

21 **GAVARBunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

GAVARBunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F10 - Vdbox unit Level Clock Gating Control 3F10

20	**GAVunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

GAVunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

19	**dec400 unit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

dec400 unit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

18	**decsubunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

decsubunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

17	**SFXunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

SFXunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

16	**SFOunit Clock Gating Disable**

Access:	R/W
_Custom_GTIReset:	BUS

SFOunit Clock Gating Disable Control:
'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)
'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F10 - Vdbox unit Level Clock Gating Control 3F10

	15	**SFMunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SFMunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	14	**SFIunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SFIunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	13	**SFEunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SFEunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	12	**SFDunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SFDunitClock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	11	**SFAunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SFAunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F10 - Vdbox unit Level Clock Gating Control 3F10

10	**hmxbrouterunit Clock Gating Disable**	
	Access:	R/W
	_Custom_GTIReset:	BUS
	hmxbrouterunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)	
9	**RAMDFTunit Clock Gating Disable**	
	Access:	R/W
	_Custom_GTIReset:	BUS
	RAMDFTunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)	
8	**Reserved**	
	Access:	RO
	Format:	MBZ
7	**SWPunit Clock Gating Disable**	
	Access:	R/W
	_Custom_GTIReset:	BUS
	SWPunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)	
6	**VTQunit Clock Gating Disable**	
	Access:	R/W
	_Custom_GTIReset:	BUS
	VTQunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)	
5	**VSLunit Clock Gating Disable**	
	Access:	R/W
	_Custom_GTIReset:	BUS
	VSLunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)	

VDCGCTL3F10 - Vdbox unit Level Clock Gating Control 3F10

		'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	4	**VSECunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VSECunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	3	**VRTunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VRTunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	2	**VPRunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VPRunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	1	**VOPunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VOPunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	0	**VNEunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VNEunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

Vdbox unit Level Clock Gating Control 3F14

VDCGCTL3F14 - Vdbox unit Level Clock Gating Control 3F14

Register Space:	MMIO: 0/2/0
Size (in bits):	32
CustomGTIIs_ContextSaved:	true

Address:	1C3F14h-1C3F17h
Name:	VDbox registers6
ShortName:	VDCGCTL3F14_VDBOX0

Address:	1C7F14h-1C7F17h
Name:	VDbox registers6
ShortName:	VDCGCTL3F14_VDBOX1

Address:	1D3F14h-1D3F17h
Name:	VDbox registers6
ShortName:	VDCGCTL3F14_VDBOX2

Address:	1D7F14h-1D7F17h
Name:	VDbox registers6
ShortName:	VDCGCTL3F14_VDBOX3

Address:	1E3F14h-1E3F17h
Name:	VDbox registers6
ShortName:	VDCGCTL3F14_VDBOX4

Address:	1E7F14h-1E7F17h
Name:	VDbox registers6
ShortName:	VDCGCTL3F14_VDBOX5

Address:	1F3F14h-1F3F17h
Name:	VDbox registers6
ShortName:	VDCGCTL3F14_VDBOX6

Address:	1F7F14h-1F7F17h
Name:	VDbox registers6
ShortName:	VDCGCTL3F14_VDBOX7

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31	**VDIunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VDIunit Clock Gating Disable Control:

VDCGCTL3F14 - Vdbox unit Level Clock Gating Control 3F14

		'0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	30	**SFMunit Clock Gating Disable1** Access: R/W _Custom_GTIReset: BUS SFMunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	29	**SFEunit Clock Gating Disable1** Access: R/W _Custom_GTIReset: BUS SFEunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	28	**SFDunits Clock Gating Disable1** Access: R/W _Custom_GTIReset: BUS SFDunits Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	27	**SFAunit Clock Gating Disable1** Access: R/W _Custom_GTIReset: BUS SFAunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)

VDCGCTL3F14 - Vdbox unit Level Clock Gating Control 3F14

		'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	26	**VEOunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VEOunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	25	**VNCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VNCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	24	**VMXunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VMXunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	23	**vmpcunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS vmpcunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F14 - Vdbox unit Level Clock Gating Control 3F14

	22	**vmmunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS vmmunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	21	**VMCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VMCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	20	**VLFunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VLFunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	19	**VISunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VISunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F14 - Vdbox unit Level Clock Gating Control 3F14

	18	**vhmeunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS vhmeunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	17	**vhlfunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS vhlfunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	16	**VCWunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VCWunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	15	**vcreunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS vcreunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F14 - Vdbox unit Level Clock Gating Control 3F14

	14	**USBunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS USBunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	13	**QRCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS QRCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	12	**MPCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS MPCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	11	**mdcunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS mdcunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F14 - Vdbox unit Level Clock Gating Control 3F14

	10	**HWMunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HWMunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	9	**IECPuit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS IECPuit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	8	**HVDunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HVDunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	7	**HUCMXunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HUCMXunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F14 - Vdbox unit Level Clock Gating Control 3F14

	6	**HUCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HUCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	5	**HTQunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HTQunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	4	**HSAOunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HSAOunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	3	**HPRunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HPRunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VDCGCTL3F14 - Vdbox unit Level Clock Gating Control 3F14

	2	**HPPunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HPPunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	1	**HHLFunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HHLFunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	0	**HFCunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS HFCunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

Vdbox unit Level Clock Gating Control 3F18

VDCGCTL3F18 - Vdbox unit Level Clock Gating Control 3F18

Register Space:	MMIO: 0/2/0
Size (in bits):	32

Address:	1C3F18h-1C3F1Bh
Name:	VDbox registers7
ShortName:	VDCGCTL3F18_VDBOX0

Address:	1C7F18h-1C7F1Bh
Name:	VDbox registers7
ShortName:	VDCGCTL3F18_VDBOX1

Address:	1D3F18h-1D3F1Bh
Name:	VDbox registers7
ShortName:	VDCGCTL3F18_VDBOX2

Address:	1D7F18h-1D7F1Bh
Name:	VDbox registers7
ShortName:	VDCGCTL3F18_VDBOX3

Address:	1E3F18h-1E3F1Bh
Name:	VDbox registers7
ShortName:	VDCGCTL3F18_VDBOX4

Address:	1E7F18h-1E7F1Bh
Name:	VDbox registers7
ShortName:	VDCGCTL3F18_VDBOX5

Address:	1F3F18h-1F3F1Bh
Name:	VDbox registers7
ShortName:	VDCGCTL3F18_VDBOX6

Address:	1F7F18h-1F7F1Bh
Name:	VDbox registers7
ShortName:	VDCGCTL3F18_VDBOX7

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31:27	**Reserved** Access: RO Format: MBZ
	26	**VECS BE unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS

VDCGCTL3F18 - Vdbox unit Level Clock Gating Control 3F18

		VECS BE unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
	25	**VECS FE unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VECS FE unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
	24	**VCS BE unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VCS BE unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
	23	**VCS FE unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VCS FE unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
	22	**amx router unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS amxb router unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for

VDCGCTL3F18 - Vdbox unit Level Clock Gating Control 3F18

		functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
	21	**amx unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS amx unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
	20	**splt unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS splt unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
	19	**tbc unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS tbc unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
	18	**vdl1is unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS vdl1is unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)

VDCGCTL3F18 - Vdbox unit Level Clock Gating Control 3F18

17	**Ibc unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS Ibc unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
16	**amxb unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS amxb unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
15	**Reserved** Access: RO Format: MBZ
14	**awm unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS awm unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
13	**aln unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS aln unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)

VDCGCTL3F18 - Vdbox unit Level Clock Gating Control 3F18

	12	**alf unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS alf unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
	11	**apr unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS apr unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
	10	**amc unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS amc unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
	9	**ait unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS ait unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)

VDCGCTL3F18 - Vdbox unit Level Clock Gating Control 3F18

	8	**app unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS app unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
	7	**aed unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS aed unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
	6	**hfe unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS hfe unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
	5	**scr unit Clock Gating Disable 2** Access: R/W _Custom_GTIReset: BUS scr unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)

VDCGCTL3F18 - Vdbox unit Level Clock Gating Control 3F18

4	**scr unit Clock Gating Disable 1** Access: R/W _Custom_GTIReset: BUS scr unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
3	**scr unit Clock Gating Disable 0** Access: R/W _Custom_GTIReset: BUS scr unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
2	**scr unit Clock Gating Disable 3** Access: R/W _Custom_GTIReset: BUS scr unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)
1	**Reserved** Access: RO Format: MBZ
0	**kin unit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS kin unit Clock Gating Disable Control: '0' : Clock Gating Enabled.; (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled.; (i.e., clocks are toggling, always)

Vdbox unit Level Clock Gating override during rstflow

VDMISCCP3F20 - Vdbox unit Level Clock Gating override during rstflow

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	1C3F20h-1C3F23h
Name:	VDbox registers9
ShortName:	VDMISCCP3F20_VDBOX0
Address:	1C7F20h-1C7F23h
Name:	VDbox registers9
ShortName:	VDMISCCP3F20_VDBOX1
Address:	1D3F20h-1D3F23h
Name:	VDbox registers9
ShortName:	VDMISCCP3F20_VDBOX2
Address:	1D7F20h-1D7F23h
Name:	VDbox registers9
ShortName:	VDMISCCP3F20_VDBOX3
Address:	1E3F20h-1E3F23h
Name:	VDbox registers9
ShortName:	VDMISCCP3F20_VDBOX4
Address:	1E7F20h-1E7F23h
Name:	VDbox registers9
ShortName:	VDMISCCP3F20_VDBOX5
Address:	1F3F20h-1F3F23h
Name:	VDbox registers9
ShortName:	VDMISCCP3F20_VDBOX6
Address:	1F7F20h-1F7F23h
Name:	VDbox registers9
ShortName:	VDMISCCP3F20_VDBOX7

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31:0	**ECO Spare Bits** Access: R/W _Custom_GTIReset: BUS Reserved

VDENC current encoded bitstream quality

CUR_ENC_BITS_QUALITY - VDENC current encoded bitstream quality

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	128
_Custom_GTIReset:	BUS
Address:	1C2D80h-1C2D8Fh
Name:	Current Encoded bitstream quality for LCU rows
ShortName:	CUR_ENC_BITS_QUALITY_VDENC_REG0
Address:	1C6D80h-1C6D8Fh
Name:	Current Encoded bitstream quality for LCU rows
ShortName:	CUR_ENC_BITS_QUALITY_VDENC_REG1
Address:	1D2D80h-1D2D8Fh
Name:	Current Encoded bitstream quality for LCU rows
ShortName:	CUR_ENC_BITS_QUALITY_VDENC_REG2
Address:	1D6D80h-1D6D8Fh
Name:	Current Encoded bitstream quality for LCU rows
ShortName:	CUR_ENC_BITS_QUALITY_VDENC_REG3
Address:	1E2D80h-1E2D8Fh
Name:	Current Encoded bitstream quality for LCU rows
ShortName:	CUR_ENC_BITS_QUALITY_VDENC_REG4
Address:	1E6D80h-1E6D8Fh
Name:	Current Encoded bitstream quality for LCU rows
ShortName:	CUR_ENC_BITS_QUALITY_VDENC_REG5
Address:	1F2D80h-1F2D8Fh
Name:	Current Encoded bitstream quality for LCU rows
ShortName:	CUR_ENC_BITS_QUALITY_VDENC_REG6
Address:	1F6D80h-1F6D8Fh
Name:	Current Encoded bitstream quality for LCU rows
ShortName:	CUR_ENC_BITS_QUALITY_VDENC_REG7

Current frame row quality. Each bit in this register indicates the quality of current encoded bitstream. This is the status information derived based on Maximum QP used for any CU in the LCU row.

DWord	Bit	Description
0	31:0	**Current encoded bitstream quality vector** Access: RO

CUR_ENC_BITS_QUALITY - VDENC current encoded bitstream quality

		This register indicates the quality of current frame for 32 consecutive LCU rows starting from LCU row 0.
1	31:0	**Current encoded bitstream quality vector** Access: RO This register indicates the quality of current frame for 32 consecutive LCU rows starting from LCU row 32.
2	31:0	**Current encoded bitstream quality vector** Access: RO This register indicates the quality of current frame for 32 consecutive LCU rows starting from LCU row 64.
3	31:0	**Current encoded bitstream quality vector** Access: RO This register indicates the quality of current frame for 32 consecutive LCU rows starting from LCU row 96.

VDENC FPAK Tile completion message

VDENC_FPAK_TILE_CMPL_MSG - VDENC FPAK Tile completion message

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	BUS
Address:	1C2D98h-1C2D9Bh
Name:	VDENC FPAK Tile Completion message
ShortName:	VDENC_FPAK_TILE_CMPL_MSG_VDENC_REG0
Address:	1C6D98h-1C6D9Bh
Name:	VDENC FPAK Tile Completion message
ShortName:	VDENC_FPAK_TILE_CMPL_MSG_VDENC_REG1
Address:	1D2D98h-1D2D9Bh
Name:	VDENC FPAK Tile Completion message
ShortName:	VDENC_FPAK_TILE_CMPL_MSG_VDENC_REG2
Address:	1D6D98h-1D6D9Bh
Name:	VDENC FPAK Tile Completion message
ShortName:	VDENC_FPAK_TILE_CMPL_MSG_VDENC_REG3
Address:	1E2D98h-1E2D9Bh
Name:	VDENC FPAK Tile Completion message
ShortName:	VDENC_FPAK_TILE_CMPL_MSG_VDENC_REG4
Address:	1E6D98h-1E6D9Bh
Name:	VDENC FPAK Tile Completion message
ShortName:	VDENC_FPAK_TILE_CMPL_MSG_VDENC_REG5
Address:	1F2D98h-1F2D9Bh
Name:	VDENC FPAK Tile Completion message
ShortName:	VDENC_FPAK_TILE_CMPL_MSG_VDENC_REG6
Address:	1F6D98h-1F6D9Bh
Name:	VDENC FPAK Tile Completion message
ShortName:	VDENC_FPAK_TILE_CMPL_MSG_VDENC_REG7

A write to this register with a value of 1 would result in head pointer update to Display .

DWord	Bit	Description
0	31:1	**Frame num and time num for FPAK** Access: R/W Hardware Clear
	0	**Tile completion Msg to FPAK** Access: R/W Hardware Clear

VDENC_FPAK_TILE_CMPL_MSG - VDENC FPAK Tile completion message

<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>1</td><td></td><td>A write with 1 would result in Tile completion message to FPAK. Reading this register would result in zero.
Tile completion message happens through config write to address 2005_0530 for session id 0 and config write to address 2005_0630 for session id 1.
The data for the tile completion message uses the following format.
<table>
<tr><td>31:15</td><td>Zeros</td></tr>
<tr><td>14:9</td><td>Tile Number</td></tr>
<tr><td>8:5</td><td>Frame number</td></tr>
<tr><td>4</td><td>Reserved</td></tr>
<tr><td>3:0</td><td>Opcode
0001 - Tail pointer update
1000 - Start of frame indicator</td></tr>
</table>
</td></tr>
<tr><td>0</td><td>[Default]</td><td>A write with 0 is ignored.</td></tr>
</table>

VDENC X_Y_POS

VDENC_X_Y_POS - VDENC X_Y_POS

Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	BUS
Address:	1C2DA8h-1C2DABh
Name:	VDENC X_Y_POS
ShortName:	VDENC_X_Y_POS_VDENC_REG0
Address:	1C6DA8h-1C6DABh
Name:	VDENC X_Y_POS
ShortName:	VDENC_X_Y_POS_VDENC_REG1
Address:	1D2DA8h-1D2DABh
Name:	VDENC X_Y_POS
ShortName:	VDENC_X_Y_POS_VDENC_REG2
Address:	1D6DA8h-1D6DABh
Name:	VDENC X_Y_POS
ShortName:	VDENC_X_Y_POS_VDENC_REG3
Address:	1E2DA8h-1E2DABh
Name:	VDENC X_Y_POS
ShortName:	VDENC_X_Y_POS_VDENC_REG4
Address:	1E6DA8h-1E6DABh
Name:	VDENC X_Y_POS
ShortName:	VDENC_X_Y_POS_VDENC_REG5
Address:	1F2DA8h-1F2DABh
Name:	VDENC X_Y_POS
ShortName:	VDENC_X_Y_POS_VDENC_REG6
Address:	1F6DA8h-1F6DABh
Name:	VDENC X_Y_POS
ShortName:	VDENC_X_Y_POS_VDENC_REG7

This register is used to indicate the VDENC processed X and Y position. This change was introduced for Streamin buffer implementation for fulsim to detect till what point VDENC has processed the data.

DWord	Bit	Description
0	31:8	**vdenc_x_pos_map** Access: RO
	7:0	**vdenc_y_pos** Access: RO

VDMBDFBARKVM

VDMBDFBARKVM - VDMBDFBARKVM

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	101070h

Allows indirection of KVM traffic for manageability.

DWord	Bit	Description
0	31:19	**Reserved** Access: RO Format: MBZ
	18:16	**BARNUM** Default Value: 111b Access: R/W _Custom_GTIReset: BUS Indicates to which base address register VDM packets should be addressed.
	15:8	**BUSNUM** Default Value: 00000000b Access: R/W _Custom_GTIReset: BUS Indicates to which bus number VDM packets should be addressed.
	7:3	**DEVNUM** Default Value: 10110b Access: R/W _Custom_GTIReset: BUS Indicates to which Device number VDM packets should be addressed.
	2:0	**FUNNUM** Default Value: 000b Access: R/W _Custom_GTIReset: BUS Indicates to which Function number VDM packets should be addressed.

VEBOX TLB Invalidation Register

VE_TLB_INV_CR - VEBOX TLB Invalidation Register

Register Space: MMIO: 0/2/0
Size (in bits): 32

Address: 0CEE0h

DWord	Bit	Description
0	31:16	**Mask Bits** Default Value: 0000000000000000b Access: R/W _Custom_GTIReset: DEV
	15:4	**Reserved** Access: RO Format: MBZ
	3	**Invalidate VEBox TLBs bit3** Default Value: 0b Access: R/W _Custom_GTIReset: DEV SW writes 1 to invalidate the TLBs for the associated VEBox and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's HW pipeline is flushed and cleared from all its memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs.Bit[3:0] invalidates VEBox[3:0] TLBs. No Effect on non-existent VEBoxs. These bits self clear.
	2	**Invalidate VEBox TLBs bit2** Default Value: 0b Access: R/W _Custom_GTIReset: DEV SW writes 1 to invalidate the TLBs for the associated VEBox and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's HW pipeline is flushed and cleared from all its memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs.Bit[3:0] invalidates VEBox[3:0] TLBs. No Effect on non-existent VEBoxs. These bits self clear.
	1	**Invalidate VEBox TLBs bit1** Default Value: 0b Access: R/W _Custom_GTIReset: DEV SW writes 1 to invalidate the TLBs for the associated VEBox and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the

VE_TLB_INV_CR - VEBOX TLB Invalidation Register

		corresponding engine's HW pipeline is flushed and cleared from all its memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs.Bit[3:0] invalidates VEBox[3:0] TLBs. No Effect on non-existent VEBoxs. These bits self clear.
	0	**Invalidate VEBox TLBs bit0** Default Value: 0b Access: R/W _Custom_GTIReset: DEV SW writes 1 to invalidate the TLBs for the associated VEBox and HW clears the bit when invalidation is complete. To ensure proper invalidation of the TLBs, SW has to ensure the corresponding engine's HW pipeline is flushed and cleared from all its memory accesses. Otherwise HW cannot guarantee the proper invalidation for TLBs.Bit[3:0] invalidates VEBox[3:0] TLBs. No Effect on non-existent VEBoxs. These bits self clear.

Vebox unit Level Clock Gating Control 3F04

VECGCTL3F04 - Vebox unit Level Clock Gating Control 3F04	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	1CBF04h-1CBF07h
Name:	VEbox registers
ShortName:	VECGCTL3F04_VEBOX0
Address:	1DBF04h-1DBF07h
Name:	VEbox registers
ShortName:	VECGCTL3F04_VEBOX1
Address:	1EBF04h-1EBF07h
Name:	VEbox registers
ShortName:	VECGCTL3F04_VEBOX2
Address:	1FBF04h-1FBF07h
Name:	VEbox registers
ShortName:	VECGCTL3F04_VEBOX3

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23	**ramdftunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS ramdftunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	22	**cg3ddis_spare2 Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS cg3ddis_spare2 Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	21	**cg3ddis_spare1 Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS

VECGCTL3F04 - Vebox unit Level Clock Gating Control 3F04

		cg3ddis_spare1 Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	20	**MCRunit Clock Gating Disable** Default Value: 1b Access: R/W _Custom_GTIReset: BUS MCRunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	19	**VFWunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VFWunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	18	**VEOunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VEOunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	17	**ECSunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS ECSunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	16	**VDNunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VDNunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

VECGCTL3F04 - Vebox unit Level Clock Gating Control 3F04

	15	**VDMunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VDMunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	14	**VDIunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VDIunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	13	**VCWunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VCWunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	12	**VCUSunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VCUSunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	11	**SFXunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SFXunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	10	**SFOunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SFOunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality)

VECGCTL3F04 - Vebox unit Level Clock Gating Control 3F04

		'1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	9	**SFMunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SFMunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	8	**SFIunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SFIunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	7	**SFEunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SFEunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	6	**SFDunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SFDunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	5	**SFAunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SFAunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	4	**NOAunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS NOAunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for

VECGCTL3F04 - Vebox unit Level Clock Gating Control 3F04

		functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	3	**IECPunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS IECPunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	2	**GCPunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GCPunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	1	**GAVARBunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GAVARBunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	0	**GAVunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS GAVunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

Vebox unit Level Clock Gating Control 3F08

VECGCTL3F08 - Vebox unit Level Clock Gating Control 3F08

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	1CBF08h-1CBF0Bh
Name:	VEbox registers
ShortName:	VECGCTL3F08_VEBOX0
Address:	1DBF08h-1DBF0Bh
Name:	VEbox registers
ShortName:	VECGCTL3F08_VEBOX1
Address:	1EBF08h-1EBF0Bh
Name:	VEbox registers
ShortName:	VECGCTL3F08_VEBOX2
Address:	1FBF08h-1FBF0Bh
Name:	VEbox registers
ShortName:	VECGCTL3F08_VEBOX3

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**VEOunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VEOunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	6	**VDIunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS VDIunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	5	**VCWunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS

VECGCTL3F08 - Vebox unit Level Clock Gating Control 3F08

		VCWunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	4	**SFMunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SFMunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	3	**SFEunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SFEunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	2	**SFDunits Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SFDunits Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	1	**SFAunit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS SFAunit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)
	0	**IECPuit Clock Gating Disable** Access: R/W _Custom_GTIReset: BUS IECPuit Clock Gating Disable Control: '0' : Clock Gating Enabled. (i.e., clocks can be gated when they are not required to toggle for functionality) '1' : Clock Gating Disabled. (i.e., clocks are toggling, always)

Vebox unit Level Clock Gating override during rstflow

VEMISCCP3F10 - Vebox unit Level Clock Gating override during rstflow

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	03F10h

Unit Level Clock Gating Disable bits

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**misccp Clock Gating Disable during rstflow** Default Value: 1b Access: R/W _Custom_GTIReset: BUS misccp Clock Gating Disable Control: '0' : Clock Gating Enabled during reset flows (i.e., clocks can be gated when they are not required to toggle for functionality, NOT Recommended) '1' : Clock Gating Disabled during reset flows. (i.e., clocks are toggling, always) Randomizing this bit will result in X flush not completing during the simulation

VECS CSB

VECS_CSB - VECS CSB		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
SW reads this offset to read the CSB entry at the top of the CSB fifo. Reads must occur in pairs to obtain a single 64 bit entry. The second read pops the entry off the CSB fifo.		
DWord	**Bit**	**Description**
0	31:0	**Context Status Buffer DW** Access: RO This DW holds CSB bits[31:0] for the first read and CSB bits[63:32] for the second read.

VECS CSB Fifo Status Register

VECS_CSB_FSR - VECS CSB Fifo Status Register

Register Space: MMIO: 0/2/0

Size (in bits): 32

This RO register holds status of the CSB fifo

DWord	Bit	Description
0	31	**Not Empty** Access: RO
	30:16	**Reserved** Access: RO Format: MBZ
	15:8	**Fifo Maximum Occupancy Count** This field is a read-only field. It reflects the depth of the FIFO, which is a static value for each product.
	7:0	**FIFO Occupancy Count** Access: RO

Vendor Identification

VID2_0_2_0_PCI - Vendor Identification		
Register Space:	PCI: 0/2/0	
Size (in bits):	16	
Address:	00000h	
This register combined with the Device Identification register uniquely identifies any PCI device.		
DWord	**Bit**	**Description**
0	15:0	**Vendor Identification Number** Default Value: 1000000010000110b Access: RO _Custom_GTIReset: BUS PCI standard identification for Intel.

VEO Current Pipe 0 XY Register

VEO_CURRENT0_XY - VEO Current Pipe 0 XY Register	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1CA054h
ShortName:	VEO_CURRENT0_XY_01
Address:	1DA054h
ShortName:	VEO_CURRENT0_XY_02
Address:	1EA054h
ShortName:	VEO_CURRENT0_XY_03
Address:	1FA054h
ShortName:	VEO_CURRENT0_XY_04

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**Current Input Pipe 0 X** Default Value: 0h
	15	**Reserved** Access: RO Format: MBZ
	14:0	**Current Input Pipe 0 Y** Default Value: 0h

VEO DN Pipe 0 XY Register

VEO_DN0_XY - VEO DN Pipe 0 XY Register

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1CA04Ch
ShortName:	VEO_DN0_XY_01
Address:	1DA04Ch
ShortName:	VEO_DN0_XY_02
Address:	1EA04Ch
ShortName:	VEO_DN0_XY_03
Address:	1FA04Ch
ShortName:	VEO_DN0_XY_04

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**DN Pipe 0 X** Default Value: 0h dn_input_x[13:0]
	15	**Reserved** Access: RO Format: MBZ
	14:0	**DN Pipe 0 Y** Default Value: 0h dn_input_y[14:0]

VEO DN Pipe 1 XY Register

VEO_DN1_XY - VEO DN Pipe 1 XY Register

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1CA050h
ShortName:	VEO_DN1_XY_01
Address:	1DA050h
ShortName:	VEO_DN1_XY_02
Address:	1EA050h
ShortName:	VEO_DN1_XY_03
Address:	1FA050h
ShortName:	VEO_DN1_XY_04

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**DN Pipe 1 X** Default Value: 0h
	15	**Reserved** Access: RO Format: MBZ
	14:0	**DN Pipe 1 Y** Default Value: 0h

VEO DV Count Register

VEO_DV_COUNT - VEO DV Count Register		
Register Space:	MMIO: 0/2/0	
Access:	RO	
Size (in bits):	32	
Address:	1CA044h	
ShortName:	VEO_DV_COUNT_01	
Address:	1DA044h	
ShortName:	VEO_DV_COUNT_02	
Address:	1EA044h	
ShortName:	VEO_DV_COUNT_03	
Address:	1FA044h	
ShortName:	VEO_DV_COUNT_04	

DWord	Bit	Description	
0	31:24	**Pipe1 Motion History DV/Hold Maxcount**	
		Default Value:	0h
	23:16	**Pipe1 Pixel History DV/Hold Maxcount**	
		Default Value:	0h
	15:8	**Pipe0 Motion History DV/Hold Maxcount**	
		Default Value:	0h
	7:0	**Pipe0 Pixel History DV/Hold Maxcount**	
		Default Value:	0h

VEO DV Hold Register

VEO_DVHOLD - VEO DV Hold Register	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1CA05Ch
ShortName:	VEO_DVHOLD_01
Address:	1DA05Ch
ShortName:	VEO_DVHOLD_02
Address:	1EA05Ch
ShortName:	VEO_DVHOLD_03
Address:	1FA05Ch
ShortName:	VEO_DVHOLD_04
Datavalid/Hold signals for VEO interface	

DWord	Bit	Description
0	31	**vdn_p0_veo_pixel_dv** Default Value: 0h
	30	**veo_vdn_p0_pixel_hold** Default Value: 0h
	29	**vdn_p0_veo_mh_dv** Default Value: 0h
	28	**veo_vdn_p0_mh_hold** Default Value: 0h
	27	**vdn_p0_veo_bne_luma_dv** Default Value: 0h
	26	**veo_vdn_p0_bne_luma_hold** Default Value: 0h
	25	**vdn_p0_veo_bne_chroma_dv** Default Value: 0h
	24	**veo_vdn_p0_bne_chroma_hold** Default Value: 0h
	23	**vdi_p0_veo_pixel_dv** Default Value: 0h
	22	**veo_vdi_p0_pixel_hold** Default Value: 0h

VEO_DVHOLD - VEO DV Hold Register

	21	**vdi_p0_veo_stmm_dv** Default Value: 0h
	20	**veo_vdi_p0_stmm_hold** Default Value: 0h
	19	**vdi_p0_veo_fmd_dv** Default Value: 0h
	18	**veo_vdi_p0_fmd_hold** Default Value: 0h
	17	**iecp_p0_veo_dv** Default Value: 0h
	16	**veo_iecp_p0_hold** Default Value: 0h
	15	**vdn_p1_veo_pixel_dv** Default Value: 0h
	14	**veo_vdn_p1_pixel_hold** Default Value: 0h
	13	**vdn_p1_veo_mh_dv** Default Value: 0h
	12	**veo_vdn_p1_mh_hold** Default Value: 0h
	11	**vdn_p1_veo_bne_luma_dv** Default Value: 0h
	10	**veo_vdn_p1_bne_luma_hold** Default Value: 0h
	9	**vdn_p1_veo_bne_chroma_dv** Default Value: 0h
	8	**veo_vdn_p1_bne_chroma_hold** Default Value: 0h
	7	**vdi_p1_veo_pixel_dv** Default Value: 0h
	6	**veo_vdi_p1_pixel_hold** Default Value: 0h
	5	**vdi_p1_veo_stmm_dv** Default Value: 0h
	4	**veo_vdi_p1_stmm_hold** Default Value: 0h

VEO_DVHOLD - VEO DV Hold Register		
	3	**vdi_p1_veo_fmd_dv** Default Value: 0h
	2	**veo_vdi_p1_fmd_hold** Default Value: 0h
	1	**iecp_p1_veo_dv** Default Value: 0h
	0	**veo_iecp_p1_hold** Default Value: 0h

VEO IECP DV Count Register

VEO_IECP_DV_COUNT - VEO IECP DV Count Register	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1CA048h
ShortName:	VEO_IECP_DV_COUNT_01
Address:	1DA048h
ShortName:	VEO_IECP_DV_COUNT_02
Address:	1EA048h
ShortName:	VEO_IECP_DV_COUNT_03
Address:	1FA048h
ShortName:	VEO_IECP_DV_COUNT_04

DWord	Bit	Description
0	31:24	**IECP DV/Hold Maxcount** Default Value: 0h
	23:16	**DI/FMD DV/Hold Maxcount** Default Value: 0h
	15:8	**DI/STMM DV/Hold Maxcount** Default Value: 0h
	7:0	**DI Pixel DV/Hold Maxcount** Default Value: 0h

VEO Previous Pipe 0 XY Register

VEO_PREVIOUS0_XY - VEO Previous Pipe 0 XY Register

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1CA058h
ShortName:	VEO_PREVIOUS0_XY_01
Address:	1DA058h
ShortName:	VEO_PREVIOUS0_XY_02
Address:	1EA058h
ShortName:	VEO_PREVIOUS0_XY_03
Address:	1FA058h
ShortName:	VEO_PREVIOUS0_XY_04

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**Previous Input Pipe 0 X** Default Value: 0h
	15	**Reserved** Access: RO Format: MBZ
	14:0	**Previous Input Pipe 0 Y** Default Value: 0h

VEO State Register

VEO_STATE - VEO State Register

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1CA040h
ShortName:	VEO_STATE_01
Address:	1DA040h
ShortName:	VEO_STATE_02
Address:	1EA040h
ShortName:	VEO_STATE_03
Address:	1FA040h
ShortName:	VEO_STATE_04

Data valids and holds for the statistics interface

DWord	Bit	Description
0	31	**iecp_p0_veo_his_dv** Default Value: 0h
	30	**iecp_p0_veo_skin_dv** Default Value: 0h
	29	**iecp_p0_veo_rgb_his_dv** Default Value: 0h
	28	**iecp_p0_veo_out_dist_dv** Default Value: 0h
	27	**iecp_p1_veo_his_dv** Default Value: 0h
	26	**iecp_p1_veo_skin_dv** Default Value: 0h
	25	**iecp_p1_veo_out_dist_dv** Default Value: 0h
	24	**veo_iecp_p0_rgb_his_hold** Default Value: 0h
	23	**Reserved** Access: RO Format: MBZ

VEO_STATE - VEO State Register

	22:19	**VSC_FSM_State** Default Value: 0h State of the VEO_VSC_CNTRL state machine
	18:16	**GAV Command Credit Count** Default Value: 4h
	15:12	**GAV Data Credit Count** Default Value: 8h
	11:8	**Reserved** Access: RO Format: MBZ
	7:0	**GAV Stall Clk Cnt Max** Default Value: 0h The longest stall from GAV since the beginning of the frame.

VE SFC Forced Lock Acknowledgement Register

VE_SFC_FORCED_LOCK_ACK - VE SFC Forced Lock Acknowledgement Register

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1CA018h
ShortName:	VE_SFC_FORCED_LOCK_ACK_VECS0
Address:	1DA018h
ShortName:	VE_SFC_FORCED_LOCK_ACK_VECS1
Address:	1EA018h
ShortName:	VE_SFC_FORCED_LOCK_ACK_VECS2
Address:	1FA018h
ShortName:	VE_SFC_FORCED_LOCK_ACK_VECS3

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**VE_SFC_FORCED_LOCK_ACK** Format: U1 This bit can only be set by hardware and it has to be clear by hardware as well. This bit is going to be polled by driver. This bit indicates that VE has received MFX_SFC_Forced_Lock from driver and it has sent that signal to SFC. Once this bit is set, it indicates SFC status (lock or unlock) will not be changed anymore. Driver will be safe to start the reset process after this bit is set. Hardware has to de-assert this bit after driver de-assert VE_SFC_Forced_Lock as well.

VE SFC Forced Lock Register

VE_SFC_FORCED_LOCK - VE SFC Forced Lock Register	
Register Space:	MMIO: 0/2/0
Access:	WO
Size (in bits):	32
Address:	1CA01Ch
ShortName:	VE_SFC_FORCED_LOCK_01
Address:	1DA01Ch
ShortName:	VE_SFC_FORCED_LOCK_02
Address:	1EA01Ch
ShortName:	VE_SFC_FORCED_LOCK_03
Address:	1FA01Ch
ShortName:	VE_SFC_FORCED_LOCK_04

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**VE_SFC_FORCED_LOCK** Format: U1 This bit can only be set by driver and it has to be clear by driver as well. Driver should set this bit before issuing the software (watchdog timer) reset. It tells VEBox that a software reset is going to happen. VE then issues a forced lock to SFC. If SFC is currently locked to VE, SFC should not unlock itself from VE. If SFC is NOT currently locked to VE, SFC should not accept the lock request from VE. Driver needs to clear this bit after the software reset sequence is complete.

VE VFW SFC Usage Register

VE_SFC_USAGE - VE VFW SFC Usage Register

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	1CA014h
ShortName:	VE_SFC_USAGE_VECS0
Address:	1DA014h
ShortName:	VE_SFC_USAGE_VECS1
Address:	1EA014h
ShortName:	VE_SFC_USAGE_VECS2
Address:	1FA014h
ShortName:	VE_SFC_USAGE_VECS3

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**VE_SFC_USAGE** Format: U1 This bit can only be set by hardware and it has to be clear by hardware as well. This bit indicates SFC is currently locked to VE. This bit should be set after SFC accepts the lock request from VE. This bit should be clear once SFC finishes the workload and unlocked from VEBox. In case a reset happens on MFX, this bit must be reset once a new workload is received

VF_CAPABILITY_REGISTER

VF_CAP_REG - VF_CAPABILITY_REGISTER

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	1901F8h

This register is used to communicate information about the VF to the VM Drivers. The same offset (0x1901F8) is used for all VF and the PF.

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**Virtual Function** Access: RO _Custom_GTIReset: BUS Value / Name: 0b – Physical Function **[Default]** 1b – Virtual Function

VF_SW_FLAG

VF_SW_FLAG	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	190240h
Name:	VF_SW_FLAG_0
ShortName:	VF_SW_FLAG_0
Address:	190244h
Name:	VF_SW_FLAG_1
ShortName:	VF_SW_FLAG_1
Address:	190248h
Name:	VF_SW_FLAG_2
ShortName:	VF_SW_FLAG_2
Address:	19024Ch
Name:	VF_SW_FLAG_3
ShortName:	VF_SW_FLAG_3

Each Virtual Function has 4x32bit Software Flag registers, which can be used as scratch registers.

DWord	Bit	Description
0	31:0	**Data** Default Value: 00000000h Access: R/W _Custom_GTIReset: BUS The format of this register is defined by Software.

VF BAR0 LDW

VF_BAR0_LDW_0_2_0_PCI - VF BAR0 LDW

Register Space:	PCI: 0/2/0
Size (in bits):	32
Address:	00344h

Lower DW of the BAR that defines the base Host Physical Address (HPA) of GTTMMADR for all VFs.
The HPA of the GTTMMADR for Virtual Function n = VF GTTMMADDR (Upper and Lower DW) + (n - 1) * (16MB * num Tiles)

DWord	Bit	Description
0	31:24	**VF GTTMMADDR Lower DW** Default Value: 00000000b Access: R/W Mask: VF GTTMMADDR Lower DW Mask Format: PhysicalAddress[31:24] _Custom_GTIReset: BUS VF GTTMMADDR Lower DW
	23:4	**VF GTTMMADDR Lower DW Mask** Default Value: 00000000000000000000000000b Access: RO _Custom_GTIReset: BUS VF GTTMMADDR Lower DW Mask
	3	**Prefetchable** Default Value: 0b Access: RO _Custom_GTIReset: BUS Prefetchable
	2:1	**Type** Default Value: 10b Access: RO _Custom_GTIReset: BUS Type. Value 10 indicates 64 bit BAR

VF_BAR0_LDW_0_2_0_PCI - VF BAR0 LDW

	0	**Memory Space Indicator** Default Value: 0b Access: RO _Custom_GTIReset: BUS Memory space Indicator. Value 0 indicates memory space.

VF BAR0 UDW

VF_BAR0_UDW_0_2_0_PCI - VF BAR0 UDW		
Register Space:	PCI: 0/2/0	
Size (in bits):	32	
Address:	00348h	
Upper DW of the BAR that defines the base Host Physical Address of the GTTMMADR for all VFs		
DWord	**Bit**	**Description**
0	31:0	**VF GTTMMADDR Upper DWord** Default Value: 00000000000000000000000000000000b Access: R/W Format: GraphicsAddress[63:32] _Custom_GTIReset: BUS VF GTTMMADDR Upper DW

VF BAR1 LDW

VF_BAR1_LDW_0_2_0_PCI - VF BAR1 LDW

Register Space:	PCI: 0/2/0
Size (in bits):	32
Address:	0034Ch

Lower DW of the BAR that defines the base Host Physical Address of GMADR for all VFs.

DWord	Bit	Description
0	31:29	**VF GMADDR Lower DW** Default Value: 000b Access: R/W Mask: VF GMADDR Lower DW Mask Format: PhysicalAddress[31:29] _Custom_GTIReset: BUS VF GMADDR Lower DW
	28:4	**VF GMADDR Lower DW Mask** Default Value: 00000000000000000000b Access: RO _Custom_GTIReset: BUS VF GMADDR Lower DW Mask
	3	**Prefetchable** Default Value: 1b Access: RO _Custom_GTIReset: BUS Prefetchable
	2:1	**Type** Default Value: 10b Access: RO _Custom_GTIReset: BUS Type. Value 10 indicates 64 bit BAR
	0	**Memory Space Indicator** Default Value: 0b Access: RO _Custom_GTIReset: BUS Memory space Indicator. Value 0 indicates memory space.

VF BAR1 UDW

VF_BAR1_UDW_0_2_0_PCI - VF BAR1 UDW		
Register Space: PCI: 0/2/0 Size (in bits): 32		
Address: 00350h		
Upper DW of the BAR that defines the base Host Physical Address of GMADR for all VFs		
DWord	**Bit**	**Description**
0	31:0	**VF GMADDR Upper DWord** Default Value: 00000000000000000000000000000000b Access: R/W Format: PhysicalAddress[63:32] _Custom_GTIReset: BUS VF GMADDR Upper DW

VF BAR2 LDW

VF_BAR2_LDW_0_2_0_PCI - VF BAR2 LDW		
Register Space:	PCI: 0/2/0	
Size (in bits):	32	
Address:	00354h	
Lower DW of Unused BAR		
DWord	**Bit**	**Description**
0	31:0	**Reserved Bar** Default Value: 00000000000000000000000000000000b Access: RO _Custom_GTIReset: BUS Reserved

VF BAR2 UDW

VF_BAR2_UDW_0_2_0_PCI - VF BAR2 UDW		
Register Space:	PCI: 0/2/0	
Size (in bits):	32	
Address:	00358h	
Upper DW of Unused BAR		
DWord	**Bit**	**Description**
0	31:0	**Reserved Bar** Default Value: 00000000000000000000000000000000b Access: RO _Custom_GTIReset: BUS Reserved

VF Device ID

VF_DEVICEID_0_2_0_PCI - VF Device ID

Register Space: PCI: 0/2/0
Size (in bits): 16

Address: 0033Ah

Defines the Device ID to be used by all Virtual Functions

DWord	Bit	Description
0	15:0	**VF DEVICE ID VALUE** Access: RO Variant _Custom_GTIReset: BUS Mirror the same device ID as the PF Value: 1001101001000000b — Name: **[Default]**

VF Migration State Array Offset

VF_MIGST_OFFSET_0_2_0_PCI - VF Migration State Array Offset

Register Space:	PCI: 0/2/0
Size (in bits):	32
Address:	0035Ch

Defines offset from a PF BAR to the VF Migration State Array. VF Migration not supported in this implementation

DWord	Bit	Description
0	31:0	**Reserved bits** Default Value: 00000000000000000000000000000000b Access: RO _Custom_GTIReset: BUS Reserved

VF Scratch Pad

VFSKPD - VF Scratch Pad

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	083A8h-083ABh
Name:	VF Scratch Pad
ShortName:	VFSKPD_VFUNIT
Address:	16EA8h-16EABh
Name:	VF Scratch Pad
ShortName:	VFSKPD_VFRUNIT

DWord	Bit	Description
0	31:16	**Mask** Access: WO Format: Mask _Custom_GTIReset: DEV Must be set to modify corresponding bit in Bits 15:0. (All bits implemented)
	15	**Reserved** Access: RO Format: MBZ
	14	**Reserved** Access: R/W Format: PBC _Custom_GTIReset: DEV
	13	**Mid Object Preemption Topology Fix Disable** Access: R/W Format: Disable _Custom_GTIReset: DEV Value / Name / Description: 0h / Enable **[Default]** / VF replay topology fix is enabled - VF will output correct topology after Object Level Pre-emption. 1h / / VF replay topology fix is disabled
	12	**Index Buffer Fetch Enhancement Disable** Access: R/W Format: Disable

VFSKPD - VF Scratch Pad

_Custom_GTIReset:	DEV

Value	Name	Description
0h	Enable **[Default]**	The index buffer fetch enhancement is enabled. If the number of vertex per instance is less than the index buffer depth, VF will only fetch the indices from the index buffer on the first instance. All other instances will re-use the indices stored in the index buffer.
1h		The index fetch enhancement is disabled. All indices are always fetched.

11 **POLYGON PrimitiveID Fix Disable**

Access:	R/W
Format:	Disable
_Custom_GTIReset:	DEV

Value	Name	Description
0h	Enable **[Default]**	PrimitiveID is constant across all triangles of a POLYGON.
1h	Disable	PrimitiveID is incremented for each triangle of a POLYGON.

10 **VF POSH Starvation Disable**

Access:	R/W
Format:	Disable
_Custom_GTIReset:	DEV

Value	Name	Description
0h	Enable **[Default]**	The VF will inform OVR when it is starved for POSH token data.
1h	Disable	The VF will not inform OVR when it is starved for POSH token data.

9 **Partial Autostrip Disable**

Access:	R/W
Format:	Disable
_Custom_GTIReset:	DEV

Value	Name	Description
0h	Enable **[Default]**	The VF can generate "partial autostrip" primitives from TRILIST inputs (if/when possible).
1h	Disable	VF will not generate "partial autostrip" primitives

8 **Reserved**

Access:	R/W
Format:	PBC
_Custom_GTIReset:	DEV

VFSKPD - VF Scratch Pad

7 **Reserved**

Access:	R/W
Format:	PBC
_Custom_GTIReset:	DEV

6 **Autostrip Disable**

Access:	R/W
Format:	U1
_Custom_GTIReset:	DEV

Value	Name	Description
0h	Enable **[Default]**	The VF can generate "autostrip" primitives from TRILIST inputs (if/when possible).
1h	Disable	VF will not generate "autostrip" primitives.

5 **TLB Prefectch Enable**

Access:	R/W
Format:	U1
_Custom_GTIReset:	DEV

Value	Name	Description
0h	Disable **[Default]**	The VF will generate prefetch of TLB when it is fetching sequential vertex data and four or fewer vertex buffers are valid.
1h	Enable	VF will disable prefetch of TLB entries.

4 **4th Vertex Data Pipe Disable**

Access:	R/W
Format:	Disable
_Custom_GTIReset:	DEV

Value	Name	Description
0h	**[Default]**	The 4th Vertex Data Pipe is enabled.
1h		The 4th Vertex Data Pipe is disabled.

Programming Notes
This is only valid when there are 3 or more vertex data pipes.

3 **Nullprim early credit release disable**

Access:	R/W
_Custom_GTIReset:	DEV

VFSKPD - VF Scratch Pad

Value	Name	Description
0h	**[Default]**	The nullprim credit release will be returned to csunit when the upper pipe of vfunit is emtpy.
1h		The nullprim credit release will be returned to csunit when the upper pipe and the lower pipe of vfunit are emtpy.

2 **Vertex Cache Implicit Disable Inhibit**

Access:	R/W
Format:	U1
_Custom_GTIReset:	DEV

Value	Name	Description
0h	**[Default]**	Allow VF to disable VS0 when Sequential index or Prim ID is a valid Element.
1h		VF never implicitly disables the vertex cache. Software must disable the VS0 Cache when required.

1 **Disable Over Fetch Cache**

Access:	R/W
_Custom_GTIReset:	DEV

Value	Name	Description
0h	**[Default]**	Cache will check for data in cache before making a request to memory
1h		Always re-fetch new data from memory.

Programming Notes
Note that the Disable Multiple Miss Read squash bit must be cleared for Disable Over Fetch Cache to be set.

0 **Disable Multiple Miss Read squash**

Access:	R/W
Format:	Disable
_Custom_GTIReset:	DEV

Value	Name	Description
0h	**[Default]**	Allow VF to squash reads that are to the same cacheline for vertex buffer requests.
1h		Disallow VF from squashing reads that are to the same cacheline for vertex buffer requests.

VF Stride

VF_STRIDE_0_2_0_PCI - VF Stride		
Register Space:	PCI: 0/2/0	
Size (in bits):	16	
Address:	00336h	
Defines the stride of the function number from one VF to the next.		
DWord	**Bit**	**Description**
0	15:0	**VF STRIDE VALUE** Default Value: 0000000000000001b Access: RO _Custom_GTIReset: BUS Defines the Routing ID offset from one VF to the next one for all VFs associated with the PF that contains this Capability structure. The next VFs 16-bit Routing ID is calculated by adding the contents of this field to the Routing ID of the current VF, ignoring any carry, using unsigned 16-bit arithmetic. The value of this field is hardwired to 0001h.

VFW Credit Count Register

VFW_CREDIT_CNT - VFW Credit Count Register		
Register Space:	MMIO: 0/2/0	
Access:	RO	
Size (in bits):	32	
Address:	1CA010h	
ShortName:	VFW_CREDIT_CNT_01	
Address:	1DA010h	
ShortName:	VFW_CREDIT_CNT_02	
Address:	1EA010h	
ShortName:	VFW_CREDIT_CNT_03	
Address:	1FA010h	
ShortName:	VFW_CREDIT_CNT_04	

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7:0	**Credit Count** Default Value: 4h The number of outstanding credits between VFW and GAV. If zero VEBOX cannot proceed due to GAV not releasing credits.

VGA_CONTROL

VGA_CONTROL

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	41000h-41003h
Name:	VGA Control
ShortName:	VGA_CONTROL
Reset:	global

Restriction

VGA requires panel fitting to be enabled. VGA is always connected to pipe A.VGA cannot be enabled while the display power well is powered down. VGA display should only be enabled if all display planes other than VGA are disabled.

DWord	Bit	Description
0	31	**VGA Display Disable** This bit will disable the VGA compatible display mode. It has no effect on VGA register or A0000-BFFFF memory aperture accesses which are controlled by the PCI configuration and VGA I/O register settings. **Value** / **Name** 0b – Enable 1b – Disable **[Default]** **Restriction** The VGA SR01 screen off bit must be programmed when enabling and disabling VGA. See the VGA Registers document.
	30:27	**Reserved** Access: RO Format: MBZ
	26	**VGA Border Enable** This bit determines if the VGA border areas are included in the active display area. The border will be scaled along with the pixel data. **Value** / **Name** 0b – Disable 1b – Enable
	25	**DBuf Clock Gate** Access: R/W The bit controls the Display buffer clocking when VGA is used. Software must set this bit to 0b before enabling VGA and set it to 1b after VGA gets disabled.

VGA_CONTROL

24	**Pipe CSC Enable** This bit enables pipe color space conversion and the pipe pre-CSC gamma for the VGA pixel data. **Value** / **Name** 0b / Disable 1b / Enable
23:21	**Reserved** Access: RO Format: MBZ
20	**Legacy 8Bit Palette En** This bit affects reads and writes to the palette through VGA I/O addresses. In the 6-bit mode, the 8-bits of data are shifted up two bits on the write (upper two bits are lost) and shifted two bits down on the read. This provides backward compatibility for original VGA programs as well as VESA VBE support for 8-bit palette. It does not affect palette accesses through the palette register MMIO path. **Value** / **Name** 0b / 6 bit DAC 1b / 8 bit DAC
19	**Reserved**
18	**Reserved**
17:16	**Reserved** Access: RO Format: MBZ
15:12	**Reserved**
11:8	**Reserved**
7:6	**Blink Duty Cycle** Controls the VGA text mode blink duty cycle relative to the VGA cursor blink duty cycle. **Value** / **Name** / **Description** 00b / 100% / 100% Duty Cycle, Full Cursor Rate 01b / 25% / 25% Duty Cycle, 1/2 Cursor Rate 10b / 50% / 50% Duty Cycle, 1/2 Cursor Rate 11b / 75% / 75% Duty Cycle, 1/2 Cursor Rate
5:0	**VSYNC Blink Rate** Controls the VGA blink rate in terms of the number of VSYNCs per on/off cycle. **Programming Notes** Program with (VSYNCs/cycle)/2-1

VIDEO_DIP_CTL

VIDEO_DIP_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60200h-60203h
Name:	Transcoder Video Data Island Packet Control
ShortName:	VIDEO_DIP_CTL_A
Reset:	soft
Address:	61200h-61203h
Name:	Transcoder Video Data Island Packet Control
ShortName:	VIDEO_DIP_CTL_B
Reset:	soft
Address:	62200h-62203h
Name:	Transcoder Video Data Island Packet Control
ShortName:	VIDEO_DIP_CTL_C
Reset:	soft
Address:	63200h-63203h
Name:	Transcoder Video Data Island Packet Control
ShortName:	VIDEO_DIP_CTL_D
Reset:	soft

Each type of Video DIP will be sent once each frame while it is enabled.

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28	**DRM DIP enable** Access: R/W This bit enables the output of the DRM DIP. Value: 1b — Name: DRM DIP enable Value: 0b — Name: DRM DIP disable **Programming Notes** This needs to be enabled with HDMI only.

VIDEO_DIP_CTL

27 PSR PSR2 VSC bit 7

This bit manually sets PSR VSC bit for future specification changes.

Value	Name
0b	Do not set
1b	Set

26:25 VSC select

Select between hardware and software control of the VSC DIP data and header. When hardware controls it will send PSR1 header and data values when PSR1 is enabled, and PSR2 + Y-coordinate header and data values when PSR2 is enabled.

If PSR2/SU SDP without Y-coordinate is enabled SW must set the header bits of the VSC packet to the PSR2 without Y-coordinate value from the eDP standard.

Value	Name	Description
00b	Header and Data	Hardware controls all headers bytes. Hardware controls data bytes 0-13. Software controls data bytes 14+. Recommended for use when PSR is enabled and software is not using VSC for other types of data.
10b	Data Only	Software controls all headers bytes. Hardware controls data bytes 0-13. Software controls data bytes 14+. Recommended for use when software is sending data with VSC. If PSR is enabled, software must configure the header to include the PSR revision and byte count.
11b	None	Software controls all headers bytes. Software controls all data bytes. Not recommended for use.

24 VDIP Enable PPS

Access:	R/W

This field controls enable/disable of PPS DIP. It should be enabled prior to enabling VDSC.
SW should enable DSC hardware feature only after programming PPS.
Hardware still sends PPS once at the beginning of enable cycle in disable PPS case.
PPS DIP can only be enabled with DisplayPort.

Value	Name
0b	Disable
1b	Enable

23:22 Reserved

Access:	RO
Format:	MBZ

21 Reserved

Access:	RO
Format:	MBZ

VIDEO_DIP_CTL

20	**VDIP Enable VSC** Access: R/W This bit enables the output of the Video Stream Configuration DIP. Value: 0b — Disable VSC DIP Value: 1b — Enable VSC DIP **Restriction** VSC can only be enabled with DisplayPort. VSC should be enabled prior to enabling PSR or stereo 3D if VSC will be used to transmit stereo 3D related information.
19:17	**Reserved** Access: RO Format: MBZ
16	**VDIP Enable GCP** Access: R/W This bit enables the output of the General Control Packet (GCP) DIP.GCP is different from other DIPs in that much of the payload is automatically reflected in the packet, and therefore there is a VIDEO_DIP_GCP register instead of DIP data buffers for GCP. Value: 0b — Disable GCP DIP Value: 1b — Enable GCP DIP **Restriction** GCP is only supported with HDMI when the bits per color is not equal to 8. GCP must be enabled prior to enabling TRANS_DDI_FUNC_CTL for HDMI with bits per color not equal to 8 and disabled after disabling TRANS_DDI_FUNC_CTL
15:13	**Reserved** Access: RO Format: MBZ
12	**VDIP Enable AVI** Access: R/W This bit enables the output of the Auxiliary Video Information DIP. Value: 0b — Disable AVI DIP Value: 1b — Enable AVI DIP **Restriction** Only enable with HDMI.

VIDEO_DIP_CTL

11:9	**Reserved** Access: RO Format: MBZ
8	**VDIP Enable VS** Access: R/W This bit enables the output of the Vendor Specific (VS) DIP. **Value** / **Name** 0b – Disable VS DIP 1b – Enable VS DIP **Restriction** Only enable with HDMI.
7:5	**Reserved** Access: RO Format: MBZ
4	**VDIP Enable GMP** Access: R/W This bit enables the output of the Gamut Metadata Packet (GMP) DIP.GMP can be enabled with either DisplayPort or HDMI. **Value** / **Name** 0b – Disable GMP DIP 1b – Enable GMP DIP
3:1	**Reserved** Access: RO Format: MBZ
0	**VDIP Enable SPD** Access: R/W This bit enables the output of the Source Product Description (SPD) DIP. **Value** / **Name** 0b – Disable SPD DIP 1b – Enable SPD DIP **Restriction** Only enable with HDMI.

VIDEO_DIP_DATA

VIDEO_DIP_DATA	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60220h-60223h
Name:	Transcoder Video Data Island Packet AVI Data 0
ShortName:	VIDEO_DIP_AVI_DATA_0_A
Reset:	soft
Address:	60224h-60227h
Name:	Transcoder Video Data Island Packet AVI Data 1
ShortName:	VIDEO_DIP_AVI_DATA_1_A
Reset:	soft
Address:	60228h-6022Bh
Name:	Transcoder Video Data Island Packet AVI Data 2
ShortName:	VIDEO_DIP_AVI_DATA_2_A
Reset:	soft
Address:	6022Ch-6022Fh
Name:	Transcoder Video Data Island Packet AVI Data 3
ShortName:	VIDEO_DIP_AVI_DATA_3_A
Reset:	soft
Address:	60230h-60233h
Name:	Transcoder Video Data Island Packet AVI Data 4
ShortName:	VIDEO_DIP_AVI_DATA_4_A
Reset:	soft
Address:	60234h-60237h
Name:	Transcoder Video Data Island Packet AVI Data 5
ShortName:	VIDEO_DIP_AVI_DATA_5_A
Reset:	soft
Address:	60238h-6023Bh
Name:	Transcoder Video Data Island Packet AVI Data 6
ShortName:	VIDEO_DIP_AVI_DATA_6_A
Reset:	soft
Address:	6023Ch-6023Fh
Name:	Transcoder Video Data Island Packet AVI Data 7
ShortName:	VIDEO_DIP_AVI_DATA_7_A

VIDEO_DIP_DATA	
Reset:	soft
Address:	60260h-60263h
Name:	Transcoder Video Data Island Packet VS Data 0
ShortName:	VIDEO_DIP_VS_DATA_0_A
Reset:	soft
Address:	60264h-60267h
Name:	Transcoder Video Data Island Packet VS Data 1
ShortName:	VIDEO_DIP_VS_DATA_1_A
Reset:	soft
Address:	60268h-6026Bh
Name:	Transcoder Video Data Island Packet VS Data 2
ShortName:	VIDEO_DIP_VS_DATA_2_A
Reset:	soft
Address:	6026Ch-6026Fh
Name:	Transcoder Video Data Island Packet VS Data 3
ShortName:	VIDEO_DIP_VS_DATA_3_A
Reset:	soft
Address:	60270h-60273h
Name:	Transcoder Video Data Island Packet VS Data 4
ShortName:	VIDEO_DIP_VS_DATA_4_A
Reset:	soft
Address:	60274h-60277h
Name:	Transcoder Video Data Island Packet VS Data 5
ShortName:	VIDEO_DIP_VS_DATA_5_A
Reset:	soft
Address:	60278h-6027Bh
Name:	Transcoder Video Data Island Packet VS Data 6
ShortName:	VIDEO_DIP_VS_DATA_6_A
Reset:	soft
Address:	6027Ch-6027Fh
Name:	Transcoder Video Data Island Packet VS Data 7
ShortName:	VIDEO_DIP_VS_DATA_7_A
Reset:	soft
Address:	602A0h-602A3h
Name:	Transcoder Video Data Island Packet SPD Data 0
ShortName:	VIDEO_DIP_SPD_DATA_0_A

VIDEO_DIP_DATA	
Reset:	soft
Address:	602A4h-602A7h
Name:	Transcoder Video Data Island Packet SPD Data 1
ShortName:	VIDEO_DIP_SPD_DATA_1_A
Reset:	soft
Address:	602A8h-602ABh
Name:	Transcoder Video Data Island Packet SPD Data 2
ShortName:	VIDEO_DIP_SPD_DATA_2_A
Reset:	soft
Address:	602ACh-602AFh
Name:	Transcoder Video Data Island Packet SPD Data 3
ShortName:	VIDEO_DIP_SPD_DATA_3_A
Reset:	soft
Address:	602B0h-602B3h
Name:	Transcoder Video Data Island Packet SPD Data 4
ShortName:	VIDEO_DIP_SPD_DATA_4_A
Reset:	soft
Address:	602B4h-602B7h
Name:	Transcoder Video Data Island Packet SPD Data 5
ShortName:	VIDEO_DIP_SPD_DATA_5_A
Reset:	soft
Address:	602B8h-602BBh
Name:	Transcoder Video Data Island Packet SPD Data 6
ShortName:	VIDEO_DIP_SPD_DATA_6_A
Reset:	soft
Address:	602BCh-602BFh
Name:	Transcoder Video Data Island Packet SPD Data 7
ShortName:	VIDEO_DIP_SPD_DATA_7_A
Reset:	soft
Address:	602E0h-602E3h
Name:	Transcoder Video Data Island Packet GMP Data 0
ShortName:	VIDEO_DIP_GMP_DATA_0_A
Reset:	soft
Address:	602E4h-602E7h
Name:	Transcoder Video Data Island Packet GMP Data 1
ShortName:	VIDEO_DIP_GMP_DATA_1_A

VIDEO_DIP_DATA	
Reset:	soft
Address:	602E8h-602EBh
Name:	Transcoder Video Data Island Packet GMP Data 2
ShortName:	VIDEO_DIP_GMP_DATA_2_A
Reset:	soft
Address:	602ECh-602EFh
Name:	Transcoder Video Data Island Packet GMP Data 3
ShortName:	VIDEO_DIP_GMP_DATA_3_A
Reset:	soft
Address:	602F0h-602F3h
Name:	Transcoder Video Data Island Packet GMP Data 4
ShortName:	VIDEO_DIP_GMP_DATA_4_A
Reset:	soft
Address:	602F4h-602F7h
Name:	Transcoder Video Data Island Packet GMP Data 5
ShortName:	VIDEO_DIP_GMP_DATA_5_A
Reset:	soft
Address:	602F8h-602FBh
Name:	Transcoder Video Data Island Packet GMP Data 6
ShortName:	VIDEO_DIP_GMP_DATA_6_A
Reset:	soft
Address:	602FCh-602FFh
Name:	Transcoder Video Data Island Packet GMP Data 7
ShortName:	VIDEO_DIP_GMP_DATA_7_A
Reset:	soft
Address:	60300h-60303h
Name:	Transcoder Video Data Island Packet GMP Data 8
ShortName:	VIDEO_DIP_GMP_DATA_8_A
Reset:	soft
Address:	60320h-60323h
Name:	Transcoder Video Data Island Packet VSC Data 0
ShortName:	VIDEO_DIP_VSC_DATA_0_A
Reset:	soft
Address:	60324h-60327h
Name:	Transcoder Video Data Island Packet VSC Data 1
ShortName:	VIDEO_DIP_VSC_DATA_1_A

VIDEO_DIP_DATA	
Reset:	soft
Address:	60328h-6032Bh
Name:	Transcoder Video Data Island Packet VSC Data 2
ShortName:	VIDEO_DIP_VSC_DATA_2_A
Reset:	soft
Address:	6032Ch-6032Fh
Name:	Transcoder Video Data Island Packet VSC Data 3
ShortName:	VIDEO_DIP_VSC_DATA_3_A
Reset:	soft
Address:	60330h-60333h
Name:	Transcoder Video Data Island Packet VSC Data 4
ShortName:	VIDEO_DIP_VSC_DATA_4_A
Reset:	soft
Address:	60334h-60337h
Name:	Transcoder Video Data Island Packet VSC Data 5
ShortName:	VIDEO_DIP_VSC_DATA_5_A
Reset:	soft
Address:	60338h-6033Bh
Name:	Transcoder Video Data Island Packet VSC Data 6
ShortName:	VIDEO_DIP_VSC_DATA_6_A
Reset:	soft
Address:	6033Ch-6033Fh
Name:	Transcoder Video Data Island Packet VSC Data 7
ShortName:	VIDEO_DIP_VSC_DATA_7_A
Reset:	soft
Address:	60340h-60343h
Name:	Transcoder Video Data Island Packet VSC Data 8
ShortName:	VIDEO_DIP_VSC_DATA_8_A
Reset:	soft
Address:	61220h-61223h
Name:	Transcoder Video Data Island Packet AVI Data 0
ShortName:	VIDEO_DIP_AVI_DATA_0_B
Reset:	soft
Address:	61224h-61227h
Name:	Transcoder Video Data Island Packet AVI Data 1
ShortName:	VIDEO_DIP_AVI_DATA_1_B

VIDEO_DIP_DATA	
Reset:	soft
Address:	61228h-6122Bh
Name:	Transcoder Video Data Island Packet AVI Data 2
ShortName:	VIDEO_DIP_AVI_DATA_2_B
Reset:	soft
Address:	6122Ch-6122Fh
Name:	Transcoder Video Data Island Packet AVI Data 3
ShortName:	VIDEO_DIP_AVI_DATA_3_B
Reset:	soft
Address:	61230h-61233h
Name:	Transcoder Video Data Island Packet AVI Data 4
ShortName:	VIDEO_DIP_AVI_DATA_4_B
Reset:	soft
Address:	61234h-61237h
Name:	Transcoder Video Data Island Packet AVI Data 5
ShortName:	VIDEO_DIP_AVI_DATA_5_B
Reset:	soft
Address:	61238h-6123Bh
Name:	Transcoder Video Data Island Packet AVI Data 6
ShortName:	VIDEO_DIP_AVI_DATA_6_B
Reset:	soft
Address:	6123Ch-6123Fh
Name:	Transcoder Video Data Island Packet AVI Data 7
ShortName:	VIDEO_DIP_AVI_DATA_7_B
Reset:	soft
Address:	61260h-61263h
Name:	Transcoder Video Data Island Packet VS Data 0
ShortName:	VIDEO_DIP_VS_DATA_0_B
Reset:	soft
Address:	61264h-61267h
Name:	Transcoder Video Data Island Packet VS Data 1
ShortName:	VIDEO_DIP_VS_DATA_1_B
Reset:	soft
Address:	61268h-6126Bh
Name:	Transcoder Video Data Island Packet VS Data 2
ShortName:	VIDEO_DIP_VS_DATA_2_B

VIDEO_DIP_DATA	
Reset:	soft
Address:	6126Ch-6126Fh
Name:	Transcoder Video Data Island Packet VS Data 3
ShortName:	VIDEO_DIP_VS_DATA_3_B
Reset:	soft
Address:	61270h-61273h
Name:	Transcoder Video Data Island Packet VS Data 4
ShortName:	VIDEO_DIP_VS_DATA_4_B
Reset:	soft
Address:	61274h-61277h
Name:	Transcoder Video Data Island Packet VS Data 5
ShortName:	VIDEO_DIP_VS_DATA_5_B
Reset:	soft
Address:	61278h-6127Bh
Name:	Transcoder Video Data Island Packet VS Data 6
ShortName:	VIDEO_DIP_VS_DATA_6_B
Reset:	soft
Address:	6127Ch-6127Fh
Name:	Transcoder Video Data Island Packet VS Data 7
ShortName:	VIDEO_DIP_VS_DATA_7_B
Reset:	soft
Address:	612A0h-612A3h
Name:	Transcoder Video Data Island Packet SPD Data 0
ShortName:	VIDEO_DIP_SPD_DATA_0_B
Reset:	soft
Address:	612A4h-612A7h
Name:	Transcoder Video Data Island Packet SPD Data 1
ShortName:	VIDEO_DIP_SPD_DATA_1_B
Reset:	soft
Address:	612A8h-612ABh
Name:	Transcoder Video Data Island Packet SPD Data 2
ShortName:	VIDEO_DIP_SPD_DATA_2_B
Reset:	soft
Address:	612ACh-612AFh
Name:	Transcoder Video Data Island Packet SPD Data 3
ShortName:	VIDEO_DIP_SPD_DATA_3_B

VIDEO_DIP_DATA	
Reset:	soft
Address:	612B0h-612B3h
Name:	Transcoder Video Data Island Packet SPD Data 4
ShortName:	VIDEO_DIP_SPD_DATA_4_B
Reset:	soft
Address:	612B4h-612B7h
Name:	Transcoder Video Data Island Packet SPD Data 5
ShortName:	VIDEO_DIP_SPD_DATA_5_B
Reset:	soft
Address:	612B8h-612BBh
Name:	Transcoder Video Data Island Packet SPD Data 6
ShortName:	VIDEO_DIP_SPD_DATA_6_B
Reset:	soft
Address:	612BCh-612BFh
Name:	Transcoder Video Data Island Packet SPD Data 7
ShortName:	VIDEO_DIP_SPD_DATA_7_B
Reset:	soft
Address:	612E0h-612E3h
Name:	Transcoder Video Data Island Packet GMP Data 0
ShortName:	VIDEO_DIP_GMP_DATA_0_B
Reset:	soft
Address:	612E4h-612E7h
Name:	Transcoder Video Data Island Packet GMP Data 1
ShortName:	VIDEO_DIP_GMP_DATA_1_B
Reset:	soft
Address:	612E8h-612EBh
Name:	Transcoder Video Data Island Packet GMP Data 2
ShortName:	VIDEO_DIP_GMP_DATA_2_B
Reset:	soft
Address:	612ECh-612EFh
Name:	Transcoder Video Data Island Packet GMP Data 3
ShortName:	VIDEO_DIP_GMP_DATA_3_B
Reset:	soft
Address:	612F0h-612F3h
Name:	Transcoder Video Data Island Packet GMP Data 4
ShortName:	VIDEO_DIP_GMP_DATA_4_B

VIDEO_DIP_DATA	
Reset:	soft
Address:	612F4h-612F7h
Name:	Transcoder Video Data Island Packet GMP Data 5
ShortName:	VIDEO_DIP_GMP_DATA_5_B
Reset:	soft
Address:	612F8h-612FBh
Name:	Transcoder Video Data Island Packet GMP Data 6
ShortName:	VIDEO_DIP_GMP_DATA_6_B
Reset:	soft
Address:	612FCh-612FFh
Name:	Transcoder Video Data Island Packet GMP Data 7
ShortName:	VIDEO_DIP_GMP_DATA_7_B
Reset:	soft
Address:	61300h-61303h
Name:	Transcoder Video Data Island Packet GMP Data 8
ShortName:	VIDEO_DIP_GMP_DATA_8_B
Reset:	soft
Address:	61320h-61323h
Name:	Transcoder Video Data Island Packet VSC Data 0
ShortName:	VIDEO_DIP_VSC_DATA_0_B
Reset:	soft
Address:	61324h-61327h
Name:	Transcoder Video Data Island Packet VSC Data 1
ShortName:	VIDEO_DIP_VSC_DATA_1_B
Reset:	soft
Address:	61328h-6132Bh
Name:	Transcoder Video Data Island Packet VSC Data 2
ShortName:	VIDEO_DIP_VSC_DATA_2_B
Reset:	soft
Address:	6132Ch-6132Fh
Name:	Transcoder Video Data Island Packet VSC Data 3
ShortName:	VIDEO_DIP_VSC_DATA_3_B
Reset:	soft
Address:	61330h-61333h
Name:	Transcoder Video Data Island Packet VSC Data 4
ShortName:	VIDEO_DIP_VSC_DATA_4_B

VIDEO_DIP_DATA	
Reset:	soft
Address:	61334h-61337h
Name:	Transcoder Video Data Island Packet VSC Data 5
ShortName:	VIDEO_DIP_VSC_DATA_5_B
Reset:	soft
Address:	61338h-6133Bh
Name:	Transcoder Video Data Island Packet VSC Data 6
ShortName:	VIDEO_DIP_VSC_DATA_6_B
Reset:	soft
Address:	6133Ch-6133Fh
Name:	Transcoder Video Data Island Packet VSC Data 7
ShortName:	VIDEO_DIP_VSC_DATA_7_B
Reset:	soft
Address:	61340h-61343h
Name:	Transcoder Video Data Island Packet VSC Data 8
ShortName:	VIDEO_DIP_VSC_DATA_8_B
Reset:	soft
Address:	62220h-62223h
Name:	Transcoder Video Data Island Packet AVI Data 0
ShortName:	VIDEO_DIP_AVI_DATA_0_C
Reset:	soft
Address:	62224h-62227h
Name:	Transcoder Video Data Island Packet AVI Data 1
ShortName:	VIDEO_DIP_AVI_DATA_1_C
Reset:	soft
Address:	62228h-6222Bh
Name:	Transcoder Video Data Island Packet AVI Data 2
ShortName:	VIDEO_DIP_AVI_DATA_2_C
Reset:	soft
Address:	6222Ch-6222Fh
Name:	Transcoder Video Data Island Packet AVI Data 3
ShortName:	VIDEO_DIP_AVI_DATA_3_C
Reset:	soft
Address:	62230h-62233h
Name:	Transcoder Video Data Island Packet AVI Data 4
ShortName:	VIDEO_DIP_AVI_DATA_4_C

VIDEO_DIP_DATA	
Reset:	soft
Address:	62234h-62237h
Name:	Transcoder Video Data Island Packet AVI Data 5
ShortName:	VIDEO_DIP_AVI_DATA_5_C
Reset:	soft
Address:	62238h-6223Bh
Name:	Transcoder Video Data Island Packet AVI Data 6
ShortName:	VIDEO_DIP_AVI_DATA_6_C
Reset:	soft
Address:	6223Ch-6223Fh
Name:	Transcoder Video Data Island Packet AVI Data 7
ShortName:	VIDEO_DIP_AVI_DATA_7_C
Reset:	soft
Address:	62260h-62263h
Name:	Transcoder Video Data Island Packet VS Data 0
ShortName:	VIDEO_DIP_VS_DATA_0_C
Reset:	soft
Address:	62264h-62267h
Name:	Transcoder Video Data Island Packet VS Data 1
ShortName:	VIDEO_DIP_VS_DATA_1_C
Reset:	soft
Address:	62268h-6226Bh
Name:	Transcoder Video Data Island Packet VS Data 2
ShortName:	VIDEO_DIP_VS_DATA_2_C
Reset:	soft
Address:	6226Ch-6226Fh
Name:	Transcoder Video Data Island Packet VS Data 3
ShortName:	VIDEO_DIP_VS_DATA_3_C
Reset:	soft
Address:	62270h-62273h
Name:	Transcoder Video Data Island Packet VS Data 4
ShortName:	VIDEO_DIP_VS_DATA_4_C
Reset:	soft
Address:	62274h-62277h
Name:	Transcoder Video Data Island Packet VS Data 5
ShortName:	VIDEO_DIP_VS_DATA_5_C

VIDEO_DIP_DATA	
Reset:	soft
Address:	62278h-6227Bh
Name:	Transcoder Video Data Island Packet VS Data 6
ShortName:	VIDEO_DIP_VS_DATA_6_C
Reset:	soft
Address:	6227Ch-6227Fh
Name:	Transcoder Video Data Island Packet VS Data 7
ShortName:	VIDEO_DIP_VS_DATA_7_C
Reset:	soft
Address:	622A0h-622A3h
Name:	Transcoder Video Data Island Packet SPD Data 0
ShortName:	VIDEO_DIP_SPD_DATA_0_C
Reset:	soft
Address:	622A4h-622A7h
Name:	Transcoder Video Data Island Packet SPD Data 1
ShortName:	VIDEO_DIP_SPD_DATA_1_C
Reset:	soft
Address:	622A8h-622ABh
Name:	Transcoder Video Data Island Packet SPD Data 2
ShortName:	VIDEO_DIP_SPD_DATA_2_C
Reset:	soft
Address:	622ACh-622AFh
Name:	Transcoder Video Data Island Packet SPD Data 3
ShortName:	VIDEO_DIP_SPD_DATA_3_C
Reset:	soft
Address:	622B0h-622B3h
Name:	Transcoder Video Data Island Packet SPD Data 4
ShortName:	VIDEO_DIP_SPD_DATA_4_C
Reset:	soft
Address:	622B4h-622B7h
Name:	Transcoder Video Data Island Packet SPD Data 5
ShortName:	VIDEO_DIP_SPD_DATA_5_C
Reset:	soft
Address:	622B8h-622BBh
Name:	Transcoder Video Data Island Packet SPD Data 6
ShortName:	VIDEO_DIP_SPD_DATA_6_C

VIDEO_DIP_DATA	
Reset:	soft
Address:	622BCh-622BFh
Name:	Transcoder Video Data Island Packet SPD Data 7
ShortName:	VIDEO_DIP_SPD_DATA_7_C
Reset:	soft
Address:	622E0h-622E3h
Name:	Transcoder Video Data Island Packet GMP Data 0
ShortName:	VIDEO_DIP_GMP_DATA_0_C
Reset:	soft
Address:	622E4h-622E7h
Name:	Transcoder Video Data Island Packet GMP Data 1
ShortName:	VIDEO_DIP_GMP_DATA_1_C
Reset:	soft
Address:	622E8h-622EBh
Name:	Transcoder Video Data Island Packet GMP Data 2
ShortName:	VIDEO_DIP_GMP_DATA_2_C
Reset:	soft
Address:	622ECh-622EFh
Name:	Transcoder Video Data Island Packet GMP Data 3
ShortName:	VIDEO_DIP_GMP_DATA_3_C
Reset:	soft
Address:	622F0h-622F3h
Name:	Transcoder Video Data Island Packet GMP Data 4
ShortName:	VIDEO_DIP_GMP_DATA_4_C
Reset:	soft
Address:	622F4h-622F7h
Name:	Transcoder Video Data Island Packet GMP Data 5
ShortName:	VIDEO_DIP_GMP_DATA_5_C
Reset:	soft
Address:	622F8h-622FBh
Name:	Transcoder Video Data Island Packet GMP Data 6
ShortName:	VIDEO_DIP_GMP_DATA_6_C
Reset:	soft
Address:	622FCh-622FFh
Name:	Transcoder Video Data Island Packet GMP Data 7
ShortName:	VIDEO_DIP_GMP_DATA_7_C

VIDEO_DIP_DATA	
Reset:	soft
Address:	62300h-62303h
Name:	Transcoder Video Data Island Packet GMP Data 8
ShortName:	VIDEO_DIP_GMP_DATA_8_C
Reset:	soft
Address:	62320h-62323h
Name:	Transcoder Video Data Island Packet VSC Data 0
ShortName:	VIDEO_DIP_VSC_DATA_0_C
Reset:	soft
Address:	62324h-62327h
Name:	Transcoder Video Data Island Packet VSC Data 1
ShortName:	VIDEO_DIP_VSC_DATA_1_C
Reset:	soft
Address:	62328h-6232Bh
Name:	Transcoder Video Data Island Packet VSC Data 2
ShortName:	VIDEO_DIP_VSC_DATA_2_C
Reset:	soft
Address:	6232Ch-6232Fh
Name:	Transcoder Video Data Island Packet VSC Data 3
ShortName:	VIDEO_DIP_VSC_DATA_3_C
Reset:	soft
Address:	62330h-62333h
Name:	Transcoder Video Data Island Packet VSC Data 4
ShortName:	VIDEO_DIP_VSC_DATA_4_C
Reset:	soft
Address:	62334h-62337h
Name:	Transcoder Video Data Island Packet VSC Data 5
ShortName:	VIDEO_DIP_VSC_DATA_5_C
Reset:	soft
Address:	62338h-6233Bh
Name:	Transcoder Video Data Island Packet VSC Data 6
ShortName:	VIDEO_DIP_VSC_DATA_6_C
Reset:	soft
Address:	6233Ch-6233Fh
Name:	Transcoder Video Data Island Packet VSC Data 7
ShortName:	VIDEO_DIP_VSC_DATA_7_C

VIDEO_DIP_DATA	
Reset:	soft
Address:	62340h-62343h
Name:	Transcoder Video Data Island Packet VSC Data 8
ShortName:	VIDEO_DIP_VSC_DATA_8_C
Reset:	soft
Address:	63220h-63223h
Name:	Transcoder Video Data Island Packet AVI Data 0
ShortName:	VIDEO_DIP_AVI_DATA_0_D
Reset:	soft
Address:	63224h-63227h
Name:	Transcoder Video Data Island Packet AVI Data 1
ShortName:	VIDEO_DIP_AVI_DATA_1_D
Reset:	soft
Address:	63228h-6322Bh
Name:	Transcoder Video Data Island Packet AVI Data 2
ShortName:	VIDEO_DIP_AVI_DATA_2_D
Reset:	soft
Address:	6322Ch-6322Fh
Name:	Transcoder Video Data Island Packet AVI Data 3
ShortName:	VIDEO_DIP_AVI_DATA_3_D
Reset:	soft
Address:	63230h-63233h
Name:	Transcoder Video Data Island Packet AVI Data 4
ShortName:	VIDEO_DIP_AVI_DATA_4_D
Reset:	soft
Address:	63234h-63237h
Name:	Transcoder Video Data Island Packet AVI Data 5
ShortName:	VIDEO_DIP_AVI_DATA_5_D
Reset:	soft
Address:	63238h-6323Bh
Name:	Transcoder Video Data Island Packet AVI Data 6
ShortName:	VIDEO_DIP_AVI_DATA_6_D
Reset:	soft
Address:	6323Ch-6323Fh
Name:	Transcoder Video Data Island Packet AVI Data 7
ShortName:	VIDEO_DIP_AVI_DATA_7_D

VIDEO_DIP_DATA	
Reset:	soft
Address:	63260h-63263h
Name:	Transcoder Video Data Island Packet VS Data 0
ShortName:	VIDEO_DIP_VS_DATA_0_D
Reset:	soft
Address:	63264h-63267h
Name:	Transcoder Video Data Island Packet VS Data 1
ShortName:	VIDEO_DIP_VS_DATA_1_D
Reset:	soft
Address:	63268h-6326Bh
Name:	Transcoder Video Data Island Packet VS Data 2
ShortName:	VIDEO_DIP_VS_DATA_2_D
Reset:	soft
Address:	6326Ch-6326Fh
Name:	Transcoder Video Data Island Packet VS Data 3
ShortName:	VIDEO_DIP_VS_DATA_3_D
Reset:	soft
Address:	63270h-63273h
Name:	Transcoder Video Data Island Packet VS Data 4
ShortName:	VIDEO_DIP_VS_DATA_4_D
Reset:	soft
Address:	63274h-63277h
Name:	Transcoder Video Data Island Packet VS Data 5
ShortName:	VIDEO_DIP_VS_DATA_5_D
Reset:	soft
Address:	63278h-6327Bh
Name:	Transcoder Video Data Island Packet VS Data 6
ShortName:	VIDEO_DIP_VS_DATA_6_D
Reset:	soft
Address:	6327Ch-6327Fh
Name:	Transcoder Video Data Island Packet VS Data 7
ShortName:	VIDEO_DIP_VS_DATA_7_D
Reset:	soft
Address:	632A0h-632A3h
Name:	Transcoder Video Data Island Packet SPD Data 0
ShortName:	VIDEO_DIP_SPD_DATA_0_D

VIDEO_DIP_DATA	
Reset:	soft
Address:	632A4h-632A7h
Name:	Transcoder Video Data Island Packet SPD Data 1
ShortName:	VIDEO_DIP_SPD_DATA_1_D
Reset:	soft
Address:	632A8h-632ABh
Name:	Transcoder Video Data Island Packet SPD Data 2
ShortName:	VIDEO_DIP_SPD_DATA_2_D
Reset:	soft
Address:	632ACh-632AFh
Name:	Transcoder Video Data Island Packet SPD Data 3
ShortName:	VIDEO_DIP_SPD_DATA_3_D
Reset:	soft
Address:	632B0h-632B3h
Name:	Transcoder Video Data Island Packet SPD Data 4
ShortName:	VIDEO_DIP_SPD_DATA_4_D
Reset:	soft
Address:	632B4h-632B7h
Name:	Transcoder Video Data Island Packet SPD Data 5
ShortName:	VIDEO_DIP_SPD_DATA_5_D
Reset:	soft
Address:	632B8h-632BBh
Name:	Transcoder Video Data Island Packet SPD Data 6
ShortName:	VIDEO_DIP_SPD_DATA_6_D
Reset:	soft
Address:	632BCh-632BFh
Name:	Transcoder Video Data Island Packet SPD Data 7
ShortName:	VIDEO_DIP_SPD_DATA_7_D
Reset:	soft
Address:	632E0h-632E3h
Name:	Transcoder Video Data Island Packet GMP Data 0
ShortName:	VIDEO_DIP_GMP_DATA_0_D
Reset:	soft
Address:	632E4h-632E7h
Name:	Transcoder Video Data Island Packet GMP Data 1
ShortName:	VIDEO_DIP_GMP_DATA_1_D

VIDEO_DIP_DATA	
Reset:	soft
Address:	632E8h-632EBh
Name:	Transcoder Video Data Island Packet GMP Data 2
ShortName:	VIDEO_DIP_GMP_DATA_2_D
Reset:	soft
Address:	632ECh-632EFh
Name:	Transcoder Video Data Island Packet GMP Data 3
ShortName:	VIDEO_DIP_GMP_DATA_3_D
Reset:	soft
Address:	632F0h-632F3h
Name:	Transcoder Video Data Island Packet GMP Data 4
ShortName:	VIDEO_DIP_GMP_DATA_4_D
Reset:	soft
Address:	632F4h-632F7h
Name:	Transcoder Video Data Island Packet GMP Data 5
ShortName:	VIDEO_DIP_GMP_DATA_5_D
Reset:	soft
Address:	632F8h-632FBh
Name:	Transcoder Video Data Island Packet GMP Data 6
ShortName:	VIDEO_DIP_GMP_DATA_6_D
Reset:	soft
Address:	632FCh-632FFh
Name:	Transcoder Video Data Island Packet GMP Data 7
ShortName:	VIDEO_DIP_GMP_DATA_7_D
Reset:	soft
Address:	63300h-63303h
Name:	Transcoder Video Data Island Packet GMP Data 8
ShortName:	VIDEO_DIP_GMP_DATA_8_D
Reset:	soft
Address:	63320h-63323h
Name:	Transcoder Video Data Island Packet VSC Data 0
ShortName:	VIDEO_DIP_VSC_DATA_0_D
Reset:	soft
Address:	63324h-63327h
Name:	Transcoder Video Data Island Packet VSC Data 1
ShortName:	VIDEO_DIP_VSC_DATA_1_D

VIDEO_DIP_DATA

Reset:	soft
Address:	63328h-6332Bh
Name:	Transcoder Video Data Island Packet VSC Data 2
ShortName:	VIDEO_DIP_VSC_DATA_2_D
Reset:	soft
Address:	6332Ch-6332Fh
Name:	Transcoder Video Data Island Packet VSC Data 3
ShortName:	VIDEO_DIP_VSC_DATA_3_D
Reset:	soft
Address:	63330h-63333h
Name:	Transcoder Video Data Island Packet VSC Data 4
ShortName:	VIDEO_DIP_VSC_DATA_4_D
Reset:	soft
Address:	63334h-63337h
Name:	Transcoder Video Data Island Packet VSC Data 5
ShortName:	VIDEO_DIP_VSC_DATA_5_D
Reset:	soft
Address:	63338h-6333Bh
Name:	Transcoder Video Data Island Packet VSC Data 6
ShortName:	VIDEO_DIP_VSC_DATA_6_D
Reset:	soft
Address:	6333Ch-6333Fh
Name:	Transcoder Video Data Island Packet VSC Data 7
ShortName:	VIDEO_DIP_VSC_DATA_7_D
Reset:	soft
Address:	63340h-63343h
Name:	Transcoder Video Data Island Packet VSC Data 8
ShortName:	VIDEO_DIP_VSC_DATA_8_D
Reset:	soft

There are multiple instances of this register format per DIP type and per transcoder.

DWord	Bit	Description
0	31:0	**Video DIP DATA** Access: R/W This field contains the video DIP data to be transmitted. **Restriction** Data should be loaded before enabling the transmission through the DIP type enable bit.

VIDEO_DIP_DRM_DATA

VIDEO_DIP_DRM_DATA - VIDEO_DIP_DRM_DATA	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60440h-60443h
Name:	Transcoder Video Data Island Packet for DRM 0
ShortName:	VIDEO_DIP_DRM_DATA_0_A
Reset:	soft
Address:	60444h-60447h
Name:	Transcoder Video Data Island Packet for DRM 1
ShortName:	VIDEO_DIP_DRM_DATA_1_A
Reset:	soft
Address:	60448h-6044Bh
Name:	Transcoder Video Data Island Packet for DRM 2
ShortName:	VIDEO_DIP_DRM_DATA_2_A
Reset:	soft
Address:	6044Ch-6044Fh
Name:	Transcoder Video Data Island Packet for DRM 3
ShortName:	VIDEO_DIP_DRM_DATA_3_A
Reset:	soft
Address:	60450h-60453h
Name:	Transcoder Video Data Island Packet for DRM 4
ShortName:	VIDEO_DIP_DRM_DATA_4_A
Reset:	soft
Address:	60454h-60457h
Name:	Transcoder Video Data Island Packet for DRM 5
ShortName:	VIDEO_DIP_DRM_DATA_5_A
Reset:	soft
Address:	60458h-6045Bh
Name:	Transcoder Video Data Island Packet for DRM 6
ShortName:	VIDEO_DIP_DRM_DATA_6_A
Reset:	soft
Address:	6045Ch-6045Fh
Name:	Transcoder Video Data Island Packet for DRM 7
ShortName:	VIDEO_DIP_DRM_DATA_7_A

VIDEO_DIP_DRM_DATA - VIDEO_DIP_DRM_DATA	
Reset:	soft
Address:	61440h-61443h
Name:	Transcoder Video Data Island Packet for DRM 0
ShortName:	VIDEO_DIP_DRM_DATA_0_B
Reset:	soft
Address:	61444h-61447h
Name:	Transcoder Video Data Island Packet for DRM 1
ShortName:	VIDEO_DIP_DRM_DATA_1_B
Reset:	soft
Address:	61448h-6144Bh
Name:	Transcoder Video Data Island Packet for DRM 2
ShortName:	VIDEO_DIP_DRM_DATA_2_B
Reset:	soft
Address:	6144Ch-6144Fh
Name:	Transcoder Video Data Island Packet for DRM 3
ShortName:	VIDEO_DIP_DRM_DATA_3_B
Reset:	soft
Address:	61450h-61453h
Name:	Transcoder Video Data Island Packet for DRM 4
ShortName:	VIDEO_DIP_DRM_DATA_4_B
Reset:	soft
Address:	61454h-61457h
Name:	Transcoder Video Data Island Packet for DRM 5
ShortName:	VIDEO_DIP_DRM_DATA_5_B
Reset:	soft
Address:	61458h-6145Bh
Name:	Transcoder Video Data Island Packet for DRM 6
ShortName:	VIDEO_DIP_DRM_DATA_6_B
Reset:	soft
Address:	6145Ch-6145Fh
Name:	Transcoder Video Data Island Packet for DRM 7
ShortName:	VIDEO_DIP_DRM_DATA_7_B
Reset:	soft
Address:	62440h-62443h
Name:	Transcoder Video Data Island Packet for DRM 0
ShortName:	VIDEO_DIP_DRM_DATA_0_C

VIDEO_DIP_DRM_DATA - VIDEO_DIP_DRM_DATA	
Reset:	soft
Address:	62444h-62447h
Name:	Transcoder Video Data Island Packet for DRM 1
ShortName:	VIDEO_DIP_DRM_DATA_1_C
Reset:	soft
Address:	62448h-6244Bh
Name:	Transcoder Video Data Island Packet for DRM 2
ShortName:	VIDEO_DIP_DRM_DATA_2_C
Reset:	soft
Address:	6244Ch-6244Fh
Name:	Transcoder Video Data Island Packet for DRM 3
ShortName:	VIDEO_DIP_DRM_DATA_3_C
Reset:	soft
Address:	62450h-62453h
Name:	Transcoder Video Data Island Packet for DRM 4
ShortName:	VIDEO_DIP_DRM_DATA_4_C
Reset:	soft
Address:	62454h-62457h
Name:	Transcoder Video Data Island Packet for DRM 5
ShortName:	VIDEO_DIP_DRM_DATA_5_C
Reset:	soft
Address:	62458h-6245Bh
Name:	Transcoder Video Data Island Packet for DRM 6
ShortName:	VIDEO_DIP_DRM_DATA_6_C
Reset:	soft
Address:	6245Ch-6245Fh
Name:	Transcoder Video Data Island Packet for DRM 7
ShortName:	VIDEO_DIP_DRM_DATA_7_C
Reset:	soft
Address:	63440h-63443h
Name:	Transcoder Video Data Island Packet for DRM 0
ShortName:	VIDEO_DIP_DRM_DATA_0_D
Reset:	soft
Address:	63444h-63447h
Name:	Transcoder Video Data Island Packet for DRM 1
ShortName:	VIDEO_DIP_DRM_DATA_1_D

VIDEO_DIP_DRM_DATA - VIDEO_DIP_DRM_DATA

Reset:	soft
Address:	63448h-6344Bh
Name:	Transcoder Video Data Island Packet for DRM 2
ShortName:	VIDEO_DIP_DRM_DATA_2_D
Reset:	soft
Address:	6344Ch-6344Fh
Name:	Transcoder Video Data Island Packet for DRM 3
ShortName:	VIDEO_DIP_DRM_DATA_3_D
Reset:	soft
Address:	63450h-63453h
Name:	Transcoder Video Data Island Packet for DRM 4
ShortName:	VIDEO_DIP_DRM_DATA_4_D
Reset:	soft
Address:	63454h-63457h
Name:	Transcoder Video Data Island Packet for DRM 5
ShortName:	VIDEO_DIP_DRM_DATA_5_D
Reset:	soft
Address:	63458h-6345Bh
Name:	Transcoder Video Data Island Packet for DRM 6
ShortName:	VIDEO_DIP_DRM_DATA_6_D
Reset:	soft
Address:	6345Ch-6345Fh
Name:	Transcoder Video Data Island Packet for DRM 7
ShortName:	VIDEO_DIP_DRM_DATA_7_D
Reset:	soft

HDMI 2.0 DRM Infoframe DIP data.

DWord	Bit	Description
0	31:0	**DRM DIP data** Access: R/W

VIDEO_DIP_DRM_ECC

VIDEO_DIP_DRM_ECC - VIDEO_DIP_DRM_ECC	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	60460h-60463h
Name:	Transcoder Video Data Island Packet for DRM ECC 0
ShortName:	VIDEO_DIP_DRM_ECC_0_A
Reset:	soft
Address:	60464h-60467h
Name:	Transcoder Video Data Island Packet for DRM ECC 1
ShortName:	VIDEO_DIP_DRM_ECC_1_A
Reset:	soft
Address:	61460h-61463h
Name:	Transcoder Video Data Island Packet for DRM ECC 0
ShortName:	VIDEO_DIP_DRM_ECC_0_B
Reset:	soft
Address:	61464h-61467h
Name:	Transcoder Video Data Island Packet for DRM ECC 1
ShortName:	VIDEO_DIP_DRM_ECC_1_B
Reset:	soft
Address:	62460h-62463h
Name:	Transcoder Video Data Island Packet for DRM ECC 0
ShortName:	VIDEO_DIP_DRM_ECC_0_C
Reset:	soft
Address:	62464h-62467h
Name:	Transcoder Video Data Island Packet for DRM ECC 1
ShortName:	VIDEO_DIP_DRM_ECC_1_C
Reset:	soft
Address:	63460h-63463h
Name:	Transcoder Video Data Island Packet for DRM ECC 0
ShortName:	VIDEO_DIP_DRM_ECC_0_D
Reset:	soft
Address:	63464h-63467h
Name:	Transcoder Video Data Island Packet for DRM ECC 1
ShortName:	VIDEO_DIP_DRM_ECC_1_D

VIDEO_DIP_DRM_ECC - VIDEO_DIP_DRM_ECC		
Reset: soft		
HDMI 2.0 DRM Infoframe ECC data.		
DWord	**Bit**	**Description**
0	31:0	**DRM ECC data**

VIDEO_DIP_ECC

VIDEO_DIP_ECC	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	60240h-60243h
Name:	Transcoder Video Data Island Packet AVI ECC 0
ShortName:	VIDEO_DIP_AVI_ECC_0_A
Reset:	soft
Address:	60244h-60247h
Name:	Transcoder Video Data Island Packet AVI ECC 1
ShortName:	VIDEO_DIP_AVI_ECC_1_A
Reset:	soft
Address:	60280h-60283h
Name:	Transcoder Video Data Island Packet VS ECC 0
ShortName:	VIDEO_DIP_VS_ECC_0_A
Reset:	soft
Address:	60284h-60287h
Name:	Transcoder Video Data Island Packet VS ECC 1
ShortName:	VIDEO_DIP_VS_ECC_1_A
Reset:	soft
Address:	602C0h-602C3h
Name:	Transcoder Video Data Island Packet SPD ECC 0
ShortName:	VIDEO_DIP_SPD_ECC_0_A
Reset:	soft
Address:	602C4h-602C7h
Name:	Transcoder Video Data Island Packet SPD ECC 1
ShortName:	VIDEO_DIP_SPD_ECC_1_A
Reset:	soft
Address:	602D0h-602D3h
Name:	Transcoder Video Data Island Packet VSC ECC COG2 0
ShortName:	VIDEO_DIP_VSC_ECC_COG2_0_A
Reset:	soft
Address:	602D4h-602D7h
Name:	Transcoder Video Data Island Packet VSC ECC COG2 1
ShortName:	VIDEO_DIP_VSC_ECC_COG2_1_A

VIDEO_DIP_ECC	
Reset:	soft
Address:	602D8h-602DBh
Name:	Transcoder Video Data Island Packet VSC ECC COG2 2
ShortName:	VIDEO_DIP_VSC_ECC_COG2_2_A
Reset:	soft
Address:	60304h-60307h
Name:	Transcoder Video Data Island Packet GMP ECC 0
ShortName:	VIDEO_DIP_GMP_ECC_0_A
Reset:	soft
Address:	60308h-6030Bh
Name:	Transcoder Video Data Island Packet GMP ECC 1
ShortName:	VIDEO_DIP_GMP_ECC_1_A
Reset:	soft
Address:	6030Ch-6030Fh
Name:	Transcoder Video Data Island Packet GMP ECC 2
ShortName:	VIDEO_DIP_GMP_ECC_2_A
Reset:	soft
Address:	60310h-60313h
Name:	Transcoder Video Data Island Packet GMP ECC 3
ShortName:	VIDEO_DIP_GMP_ECC_3_A
Reset:	soft
Address:	60314h-60317h
Name:	Transcoder Video Data Island Packet GMP ECC 4
ShortName:	VIDEO_DIP_GMP_ECC_4_A
Reset:	soft
Address:	60344h-60347h
Name:	Transcoder Video Data Island Packet VSC ECC 0
ShortName:	VIDEO_DIP_VSC_ECC_0_A
Reset:	soft
Address:	60348h-6034Bh
Name:	Transcoder Video Data Island Packet VSC ECC 1
ShortName:	VIDEO_DIP_VSC_ECC_1_A
Reset:	soft
Address:	6034Ch-6034Fh
Name:	Transcoder Video Data Island Packet VSC ECC 2
ShortName:	VIDEO_DIP_VSC_ECC_2_A

VIDEO_DIP_ECC	
Reset:	soft
Address:	61240h-61243h
Name:	Transcoder Video Data Island Packet AVI ECC 0
ShortName:	VIDEO_DIP_AVI_ECC_0_B
Reset:	soft
Address:	61244h-61247h
Name:	Transcoder Video Data Island Packet AVI ECC 1
ShortName:	VIDEO_DIP_AVI_ECC_1_B
Reset:	soft
Address:	61280h-61283h
Name:	Transcoder Video Data Island Packet VS ECC 0
ShortName:	VIDEO_DIP_VS_ECC_0_B
Reset:	soft
Address:	61284h-61287h
Name:	Transcoder Video Data Island Packet VS ECC 1
ShortName:	VIDEO_DIP_VS_ECC_1_B
Reset:	soft
Address:	612C0h-612C3h
Name:	Transcoder Video Data Island Packet SPD ECC 0
ShortName:	VIDEO_DIP_SPD_ECC_0_B
Reset:	soft
Address:	612C4h-612C7h
Name:	Transcoder Video Data Island Packet SPD ECC 1
ShortName:	VIDEO_DIP_SPD_ECC_1_B
Reset:	soft
Address:	612D0h-612D3h
Name:	Transcoder Video Data Island Packet VSC ECC COG2 0
ShortName:	VIDEO_DIP_VSC_ECC_COG2_0_B
Reset:	soft
Address:	612D4h-612D7h
Name:	Transcoder Video Data Island Packet VSC ECC COG2 1
ShortName:	VIDEO_DIP_VSC_ECC_COG2_1_B
Reset:	soft
Address:	612D8h-612DBh
Name:	Transcoder Video Data Island Packet VSC ECC COG2 2
ShortName:	VIDEO_DIP_VSC_ECC_COG2_2_B

VIDEO_DIP_ECC	
Reset:	soft
Address:	61304h-61307h
Name:	Transcoder Video Data Island Packet GMP ECC 0
ShortName:	VIDEO_DIP_GMP_ECC_0_B
Reset:	soft
Address:	61308h-6130Bh
Name:	Transcoder Video Data Island Packet GMP ECC 1
ShortName:	VIDEO_DIP_GMP_ECC_1_B
Reset:	soft
Address:	6130Ch-6130Fh
Name:	Transcoder Video Data Island Packet GMP ECC 2
ShortName:	VIDEO_DIP_GMP_ECC_2_B
Reset:	soft
Address:	61310h-61313h
Name:	Transcoder Video Data Island Packet GMP ECC 3
ShortName:	VIDEO_DIP_GMP_ECC_3_B
Reset:	soft
Address:	61314h-61317h
Name:	Transcoder Video Data Island Packet GMP ECC 4
ShortName:	VIDEO_DIP_GMP_ECC_4_B
Reset:	soft
Address:	61344h-61347h
Name:	Transcoder Video Data Island Packet VSC ECC 0
ShortName:	VIDEO_DIP_VSC_ECC_0_B
Reset:	soft
Address:	61348h-6134Bh
Name:	Transcoder Video Data Island Packet VSC ECC 1
ShortName:	VIDEO_DIP_VSC_ECC_1_B
Reset:	soft
Address:	6134Ch-6134Fh
Name:	Transcoder Video Data Island Packet VSC ECC 2
ShortName:	VIDEO_DIP_VSC_ECC_2_B
Reset:	soft
Address:	62240h-62243h
Name:	Transcoder Video Data Island Packet AVI ECC 0
ShortName:	VIDEO_DIP_AVI_ECC_0_C

VIDEO_DIP_ECC	
Reset:	soft
Address:	62244h-62247h
Name:	Transcoder Video Data Island Packet AVI ECC 1
ShortName:	VIDEO_DIP_AVI_ECC_1_C
Reset:	soft
Address:	62280h-62283h
Name:	Transcoder Video Data Island Packet VS ECC 0
ShortName:	VIDEO_DIP_VS_ECC_0_C
Reset:	soft
Address:	62284h-62287h
Name:	Transcoder Video Data Island Packet VS ECC 1
ShortName:	VIDEO_DIP_VS_ECC_1_C
Reset:	soft
Address:	622C0h-622C3h
Name:	Transcoder Video Data Island Packet SPD ECC 0
ShortName:	VIDEO_DIP_SPD_ECC_0_C
Reset:	soft
Address:	622C4h-622C7h
Name:	Transcoder Video Data Island Packet SPD ECC 1
ShortName:	VIDEO_DIP_SPD_ECC_1_C
Reset:	soft
Address:	622D0h-622D3h
Name:	Transcoder Video Data Island Packet VSC ECC COG2 0
ShortName:	VIDEO_DIP_VSC_ECC_COG2_0_C
Reset:	soft
Address:	622D4h-622D7h
Name:	Transcoder Video Data Island Packet VSC ECC COG2 1
ShortName:	VIDEO_DIP_VSC_ECC_COG2_1_C
Reset:	soft
Address:	622D8h-622DBh
Name:	Transcoder Video Data Island Packet VSC ECC COG2 2
ShortName:	VIDEO_DIP_VSC_ECC_COG2_2_C
Reset:	soft
Address:	62304h-62307h
Name:	Transcoder Video Data Island Packet GMP ECC 0
ShortName:	VIDEO_DIP_GMP_ECC_0_C

VIDEO_DIP_ECC	
Reset:	soft
Address:	62308h-6230Bh
Name:	Transcoder Video Data Island Packet GMP ECC 1
ShortName:	VIDEO_DIP_GMP_ECC_1_C
Reset:	soft
Address:	6230Ch-6230Fh
Name:	Transcoder Video Data Island Packet GMP ECC 2
ShortName:	VIDEO_DIP_GMP_ECC_2_C
Reset:	soft
Address:	62310h-62313h
Name:	Transcoder Video Data Island Packet GMP ECC 3
ShortName:	VIDEO_DIP_GMP_ECC_3_C
Reset:	soft
Address:	62314h-62317h
Name:	Transcoder Video Data Island Packet GMP ECC 4
ShortName:	VIDEO_DIP_GMP_ECC_4_C
Reset:	soft
Address:	62344h-62347h
Name:	Transcoder Video Data Island Packet VSC ECC 0
ShortName:	VIDEO_DIP_VSC_ECC_0_C
Reset:	soft
Address:	62348h-6234Bh
Name:	Transcoder Video Data Island Packet VSC ECC 1
ShortName:	VIDEO_DIP_VSC_ECC_1_C
Reset:	soft
Address:	6234Ch-6234Fh
Name:	Transcoder Video Data Island Packet VSC ECC 2
ShortName:	VIDEO_DIP_VSC_ECC_2_C
Reset:	soft
Address:	63240h-63243h
Name:	Transcoder Video Data Island Packet AVI ECC 0
ShortName:	VIDEO_DIP_AVI_ECC_0_D
Reset:	soft
Address:	63244h-63247h
Name:	Transcoder Video Data Island Packet AVI ECC 1
ShortName:	VIDEO_DIP_AVI_ECC_1_D

VIDEO_DIP_ECC	
Reset:	soft
Address:	63280h-63283h
Name:	Transcoder Video Data Island Packet VS ECC 0
ShortName:	VIDEO_DIP_VS_ECC_0_D
Reset:	soft
Address:	63284h-63287h
Name:	Transcoder Video Data Island Packet VS ECC 1
ShortName:	VIDEO_DIP_VS_ECC_1_D
Reset:	soft
Address:	632C0h-632C3h
Name:	Transcoder Video Data Island Packet SPD ECC 0
ShortName:	VIDEO_DIP_SPD_ECC_0_D
Reset:	soft
Address:	632C4h-632C7h
Name:	Transcoder Video Data Island Packet SPD ECC 1
ShortName:	VIDEO_DIP_SPD_ECC_1_D
Reset:	soft
Address:	632D0h-632D3h
Name:	Transcoder Video Data Island Packet VSC ECC COG2 0
ShortName:	VIDEO_DIP_VSC_ECC_COG2_0_D
Reset:	soft
Address:	632D4h-632D7h
Name:	Transcoder Video Data Island Packet VSC ECC COG2 1
ShortName:	VIDEO_DIP_VSC_ECC_COG2_1_D
Reset:	soft
Address:	632D8h-632DBh
Name:	Transcoder Video Data Island Packet VSC ECC COG2 2
ShortName:	VIDEO_DIP_VSC_ECC_COG2_2_D
Reset:	soft
Address:	63304h-63307h
Name:	Transcoder Video Data Island Packet GMP ECC 0
ShortName:	VIDEO_DIP_GMP_ECC_0_D
Reset:	soft
Address:	63308h-6330Bh
Name:	Transcoder Video Data Island Packet GMP ECC 1
ShortName:	VIDEO_DIP_GMP_ECC_1_D

VIDEO_DIP_ECC	
Reset:	soft
Address:	6330Ch-6330Fh
Name:	Transcoder Video Data Island Packet GMP ECC 2
ShortName:	VIDEO_DIP_GMP_ECC_2_D
Reset:	soft
Address:	63310h-63313h
Name:	Transcoder Video Data Island Packet GMP ECC 3
ShortName:	VIDEO_DIP_GMP_ECC_3_D
Reset:	soft
Address:	63314h-63317h
Name:	Transcoder Video Data Island Packet GMP ECC 4
ShortName:	VIDEO_DIP_GMP_ECC_4_D
Reset:	soft
Address:	63344h-63347h
Name:	Transcoder Video Data Island Packet VSC ECC 0
ShortName:	VIDEO_DIP_VSC_ECC_0_D
Reset:	soft
Address:	63348h-6334Bh
Name:	Transcoder Video Data Island Packet VSC ECC 1
ShortName:	VIDEO_DIP_VSC_ECC_1_D
Reset:	soft
Address:	6334Ch-6334Fh
Name:	Transcoder Video Data Island Packet VSC ECC 2
ShortName:	VIDEO_DIP_VSC_ECC_2_D
Reset:	soft

There are multiple instances of this register format per DIP type and per transcoder.

DWord	Bit	Description
0	31:0	**Video DIP ECC** This field contains the video DIP ECC value for read back.

VIDEO_DIP_GCP

VIDEO_DIP_GCP	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60210h-60213h
Name:	Transcoder Video Data Island Packet GCP
ShortName:	VIDEO_DIP_GCP_A
Reset:	soft
Address:	61210h-61213h
Name:	Transcoder Video Data Island Packet GCP
ShortName:	VIDEO_DIP_GCP_B
Reset:	soft
Address:	62210h-62213h
Name:	Transcoder Video Data Island Packet GCP
ShortName:	VIDEO_DIP_GCP_C
Reset:	soft
Address:	63210h-63213h
Name:	Transcoder Video Data Island Packet GCP
ShortName:	VIDEO_DIP_GCP_D
Reset:	soft

DWord	Bit	Description
0	31:3	**Reserved** Access: RO Format: MBZ
	2	**GCP color indication** Access: R/W Value / Name / Description: 0b / Don't Indicate / Don't indicate color depth. CD and PP bits in GCP set to zero. 1b / Indicate / Indicate color depth using CD bits in GCP. The color depth value comes from the TRANS_DDI_FUNC_CTL register. Restriction: This bit must be set when in HDMI deep color (>8 BPC) mode.

VIDEO_DIP_GCP

1 **GCP default phase enable**

Access:	R/W

GCP default phase indicates that video timings meet alignment requirements such that the following conditions are met:

1. Htotal is a multiple of the value given in the table below
2. Hactive is an even number
3. Front and back porches for Hsync are even numbers
4. Vsync always starts on an even-numbered pixel within a line in interlaced modes (starting counting with 0)

BPC	Color Format	Htotal Multiple Requirement
8	RGB	2
8	YUV420	4
10	RGB	4
10	YUV420	8
12	RGB	2
12	YUV420	4

Value	Name	Description
0b	Clear	Default phase bit in GCP is cleared.
1b	Set	Default phase bit in GCP is set.

Restriction
Do not set this bit if these requirements are not met.

0 **Reserved**

VIDEO_DIP_PPS_DATA

VIDEO_DIP_PPS_DATA	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60350h-60353h
Name:	Transcoder Video Data Island Packet for PPS 0
ShortName:	VIDEO_DIP_PPS_DATA_0_A
Reset:	soft
Address:	60354h-60357h
Name:	Transcoder Video Data Island Packet for PPS 1
ShortName:	VIDEO_DIP_PPS_DATA_1_A
Reset:	soft
Address:	60358h-6035Bh
Name:	Transcoder Video Data Island Packet for PPS 2
ShortName:	VIDEO_DIP_PPS_DATA_2_A
Reset:	soft
Address:	6035Ch-6035Fh
Name:	Transcoder Video Data Island Packet for PPS 3
ShortName:	VIDEO_DIP_PPS_DATA_3_A
Reset:	soft
Address:	60360h-60363h
Name:	Transcoder Video Data Island Packet for PPS 4
ShortName:	VIDEO_DIP_PPS_DATA_4_A
Reset:	soft
Address:	60364h-60367h
Name:	Transcoder Video Data Island Packet for PPS 5
ShortName:	VIDEO_DIP_PPS_DATA_5_A
Reset:	soft
Address:	60368h-6036Bh
Name:	Transcoder Video Data Island Packet for PPS 6
ShortName:	VIDEO_DIP_PPS_DATA_6_A
Reset:	soft
Address:	6036Ch-6036Fh
Name:	Transcoder Video Data Island Packet for PPS 7
ShortName:	VIDEO_DIP_PPS_DATA_7_A

VIDEO_DIP_PPS_DATA	
Reset:	soft
Address:	60370h-60373h
Name:	Transcoder Video Data Island Packet for PPS 8
ShortName:	VIDEO_DIP_PPS_DATA_8_A
Reset:	soft
Address:	60374h-60377h
Name:	Transcoder Video Data Island Packet for PPS 9
ShortName:	VIDEO_DIP_PPS_DATA_9_A
Reset:	soft
Address:	60378h-6037Bh
Name:	Transcoder Video Data Island Packet for PPS 10
ShortName:	VIDEO_DIP_PPS_DATA_10_A
Reset:	soft
Address:	6037Ch-6037Fh
Name:	Transcoder Video Data Island Packet for PPS 11
ShortName:	VIDEO_DIP_PPS_DATA_11_A
Reset:	soft
Address:	60380h-60383h
Name:	Transcoder Video Data Island Packet for PPS 12
ShortName:	VIDEO_DIP_PPS_DATA_12_A
Reset:	soft
Address:	60384h-60387h
Name:	Transcoder Video Data Island Packet for PPS 13
ShortName:	VIDEO_DIP_PPS_DATA_13_A
Reset:	soft
Address:	60388h-6038Bh
Name:	Transcoder Video Data Island Packet for PPS 14
ShortName:	VIDEO_DIP_PPS_DATA_14_A
Reset:	soft
Address:	6038Ch-6038Fh
Name:	Transcoder Video Data Island Packet for PPS 15
ShortName:	VIDEO_DIP_PPS_DATA_15_A
Reset:	soft
Address:	60390h-60393h
Name:	Transcoder Video Data Island Packet for PPS 16
ShortName:	VIDEO_DIP_PPS_DATA_16_A

VIDEO_DIP_PPS_DATA	
Reset:	soft
Address:	60394h-60397h
Name:	Transcoder Video Data Island Packet for PPS 17
ShortName:	VIDEO_DIP_PPS_DATA_17_A
Reset:	soft
Address:	60398h-6039Bh
Name:	Transcoder Video Data Island Packet for PPS 18
ShortName:	VIDEO_DIP_PPS_DATA_18_A
Reset:	soft
Address:	6039Ch-6039Fh
Name:	Transcoder Video Data Island Packet for PPS 19
ShortName:	VIDEO_DIP_PPS_DATA_19_A
Reset:	soft
Address:	603A0h-603A3h
Name:	Transcoder Video Data Island Packet for PPS 20
ShortName:	VIDEO_DIP_PPS_DATA_20_A
Reset:	soft
Address:	603A4h-603A7h
Name:	Transcoder Video Data Island Packet for PPS 21
ShortName:	VIDEO_DIP_PPS_DATA_21_A
Reset:	soft
Address:	603A8h-603ABh
Name:	Transcoder Video Data Island Packet for PPS 22
ShortName:	VIDEO_DIP_PPS_DATA_22_A
Reset:	soft
Address:	603ACh-603AFh
Name:	Transcoder Video Data Island Packet for PPS 23
ShortName:	VIDEO_DIP_PPS_DATA_23_A
Reset:	soft
Address:	603B0h-603B3h
Name:	Transcoder Video Data Island Packet for PPS 24
ShortName:	VIDEO_DIP_PPS_DATA_24_A
Reset:	soft
Address:	603B4h-603B7h
Name:	Transcoder Video Data Island Packet for PPS 25
ShortName:	VIDEO_DIP_PPS_DATA_25_A

VIDEO_DIP_PPS_DATA	
Reset:	soft
Address:	603B8h-603BBh
Name:	Transcoder Video Data Island Packet for PPS 26
ShortName:	VIDEO_DIP_PPS_DATA_26_A
Reset:	soft
Address:	603BCh-603BFh
Name:	Transcoder Video Data Island Packet for PPS 27
ShortName:	VIDEO_DIP_PPS_DATA_27_A
Reset:	soft
Address:	603C0h-603C3h
Name:	Transcoder Video Data Island Packet for PPS 28
ShortName:	VIDEO_DIP_PPS_DATA_28_A
Reset:	soft
Address:	603C4h-603C7h
Name:	Transcoder Video Data Island Packet for PPS 29
ShortName:	VIDEO_DIP_PPS_DATA_29_A
Reset:	soft
Address:	603C8h-603CBh
Name:	Transcoder Video Data Island Packet for PPS 30
ShortName:	VIDEO_DIP_PPS_DATA_30_A
Reset:	soft
Address:	603CCh-603CFh
Name:	Transcoder Video Data Island Packet for PPS 31
ShortName:	VIDEO_DIP_PPS_DATA_31_A
Reset:	soft
Address:	603D0h-603D3h
Name:	Transcoder Video Data Island Packet for PPS 32
ShortName:	VIDEO_DIP_PPS_DATA_32_A
Reset:	soft
Address:	61350h-61353h
Name:	Transcoder Video Data Island Packet for PPS 0
ShortName:	VIDEO_DIP_PPS_DATA_0_B
Reset:	soft
Address:	61354h-61357h
Name:	Transcoder Video Data Island Packet for PPS 1
ShortName:	VIDEO_DIP_PPS_DATA_1_B

VIDEO_DIP_PPS_DATA	
Reset:	soft
Address:	61358h-6135Bh
Name:	Transcoder Video Data Island Packet for PPS 2
ShortName:	VIDEO_DIP_PPS_DATA_2_B
Reset:	soft
Address:	6135Ch-6135Fh
Name:	Transcoder Video Data Island Packet for PPS 3
ShortName:	VIDEO_DIP_PPS_DATA_3_B
Reset:	soft
Address:	61360h-61363h
Name:	Transcoder Video Data Island Packet for PPS 4
ShortName:	VIDEO_DIP_PPS_DATA_4_B
Reset:	soft
Address:	61364h-61367h
Name:	Transcoder Video Data Island Packet for PPS 5
ShortName:	VIDEO_DIP_PPS_DATA_5_B
Reset:	soft
Address:	61368h-6136Bh
Name:	Transcoder Video Data Island Packet for PPS 6
ShortName:	VIDEO_DIP_PPS_DATA_6_B
Reset:	soft
Address:	6136Ch-6136Fh
Name:	Transcoder Video Data Island Packet for PPS 7
ShortName:	VIDEO_DIP_PPS_DATA_7_B
Reset:	soft
Address:	61370h-61373h
Name:	Transcoder Video Data Island Packet for PPS 8
ShortName:	VIDEO_DIP_PPS_DATA_8_B
Reset:	soft
Address:	61374h-61377h
Name:	Transcoder Video Data Island Packet for PPS 9
ShortName:	VIDEO_DIP_PPS_DATA_9_B
Reset:	soft
Address:	61378h-6137Bh
Name:	Transcoder Video Data Island Packet for PPS 10
ShortName:	VIDEO_DIP_PPS_DATA_10_B

VIDEO_DIP_PPS_DATA	
Reset:	soft
Address:	6137Ch-6137Fh
Name:	Transcoder Video Data Island Packet for PPS 11
ShortName:	VIDEO_DIP_PPS_DATA_11_B
Reset:	soft
Address:	61380h-61383h
Name:	Transcoder Video Data Island Packet for PPS 12
ShortName:	VIDEO_DIP_PPS_DATA_12_B
Reset:	soft
Address:	61384h-61387h
Name:	Transcoder Video Data Island Packet for PPS 13
ShortName:	VIDEO_DIP_PPS_DATA_13_B
Reset:	soft
Address:	61388h-6138Bh
Name:	Transcoder Video Data Island Packet for PPS 14
ShortName:	VIDEO_DIP_PPS_DATA_14_B
Reset:	soft
Address:	6138Ch-6138Fh
Name:	Transcoder Video Data Island Packet for PPS 15
ShortName:	VIDEO_DIP_PPS_DATA_15_B
Reset:	soft
Address:	61390h-61393h
Name:	Transcoder Video Data Island Packet for PPS 16
ShortName:	VIDEO_DIP_PPS_DATA_16_B
Reset:	soft
Address:	61394h-61397h
Name:	Transcoder Video Data Island Packet for PPS 17
ShortName:	VIDEO_DIP_PPS_DATA_17_B
Reset:	soft
Address:	61398h-6139Bh
Name:	Transcoder Video Data Island Packet for PPS 18
ShortName:	VIDEO_DIP_PPS_DATA_18_B
Reset:	soft
Address:	6139Ch-6139Fh
Name:	Transcoder Video Data Island Packet for PPS 19
ShortName:	VIDEO_DIP_PPS_DATA_19_B

VIDEO_DIP_PPS_DATA	
Reset:	soft
Address:	613A0h-613A3h
Name:	Transcoder Video Data Island Packet for PPS 20
ShortName:	VIDEO_DIP_PPS_DATA_20_B
Reset:	soft
Address:	613A4h-613A7h
Name:	Transcoder Video Data Island Packet for PPS 21
ShortName:	VIDEO_DIP_PPS_DATA_21_B
Reset:	soft
Address:	613A8h-613ABh
Name:	Transcoder Video Data Island Packet for PPS 22
ShortName:	VIDEO_DIP_PPS_DATA_22_B
Reset:	soft
Address:	613ACh-613AFh
Name:	Transcoder Video Data Island Packet for PPS 23
ShortName:	VIDEO_DIP_PPS_DATA_23_B
Reset:	soft
Address:	613B0h-613B3h
Name:	Transcoder Video Data Island Packet for PPS 24
ShortName:	VIDEO_DIP_PPS_DATA_24_B
Reset:	soft
Address:	613B4h-613B7h
Name:	Transcoder Video Data Island Packet for PPS 25
ShortName:	VIDEO_DIP_PPS_DATA_25_B
Reset:	soft
Address:	613B8h-613BBh
Name:	Transcoder Video Data Island Packet for PPS 26
ShortName:	VIDEO_DIP_PPS_DATA_26_B
Reset:	soft
Address:	613BCh-613BFh
Name:	Transcoder Video Data Island Packet for PPS 27
ShortName:	VIDEO_DIP_PPS_DATA_27_B
Reset:	soft
Address:	613C0h-613C3h
Name:	Transcoder Video Data Island Packet for PPS 28
ShortName:	VIDEO_DIP_PPS_DATA_28_B

VIDEO_DIP_PPS_DATA	
Reset:	soft
Address:	613C4h-613C7h
Name:	Transcoder Video Data Island Packet for PPS 29
ShortName:	VIDEO_DIP_PPS_DATA_29_B
Reset:	soft
Address:	613C8h-613CBh
Name:	Transcoder Video Data Island Packet for PPS 30
ShortName:	VIDEO_DIP_PPS_DATA_30_B
Reset:	soft
Address:	613CCh-613CFh
Name:	Transcoder Video Data Island Packet for PPS 31
ShortName:	VIDEO_DIP_PPS_DATA_31_B
Reset:	soft
Address:	613D0h-613D3h
Name:	Transcoder Video Data Island Packet for PPS 32
ShortName:	VIDEO_DIP_PPS_DATA_32_B
Reset:	soft
Address:	62350h-62353h
Name:	Transcoder Video Data Island Packet for PPS 0
ShortName:	VIDEO_DIP_PPS_DATA_0_C
Reset:	soft
Address:	62354h-62357h
Name:	Transcoder Video Data Island Packet for PPS 1
ShortName:	VIDEO_DIP_PPS_DATA_1_C
Reset:	soft
Address:	62358h-6235Bh
Name:	Transcoder Video Data Island Packet for PPS 2
ShortName:	VIDEO_DIP_PPS_DATA_2_C
Reset:	soft
Address:	6235Ch-6235Fh
Name:	Transcoder Video Data Island Packet for PPS 3
ShortName:	VIDEO_DIP_PPS_DATA_3_C
Reset:	soft
Address:	62360h-62363h
Name:	Transcoder Video Data Island Packet for PPS 4
ShortName:	VIDEO_DIP_PPS_DATA_4_C

VIDEO_DIP_PPS_DATA	
Reset:	soft
Address:	62364h-62367h
Name:	Transcoder Video Data Island Packet for PPS 5
ShortName:	VIDEO_DIP_PPS_DATA_5_C
Reset:	soft
Address:	62368h-6236Bh
Name:	Transcoder Video Data Island Packet for PPS 6
ShortName:	VIDEO_DIP_PPS_DATA_6_C
Reset:	soft
Address:	6236Ch-6236Fh
Name:	Transcoder Video Data Island Packet for PPS 7
ShortName:	VIDEO_DIP_PPS_DATA_7_C
Reset:	soft
Address:	62370h-62373h
Name:	Transcoder Video Data Island Packet for PPS 8
ShortName:	VIDEO_DIP_PPS_DATA_8_C
Reset:	soft
Address:	62374h-62377h
Name:	Transcoder Video Data Island Packet for PPS 9
ShortName:	VIDEO_DIP_PPS_DATA_9_C
Reset:	soft
Address:	62378h-6237Bh
Name:	Transcoder Video Data Island Packet for PPS 10
ShortName:	VIDEO_DIP_PPS_DATA_10_C
Reset:	soft
Address:	6237Ch-6237Fh
Name:	Transcoder Video Data Island Packet for PPS 11
ShortName:	VIDEO_DIP_PPS_DATA_11_C
Reset:	soft
Address:	62380h-62383h
Name:	Transcoder Video Data Island Packet for PPS 12
ShortName:	VIDEO_DIP_PPS_DATA_12_C
Reset:	soft
Address:	62384h-62387h
Name:	Transcoder Video Data Island Packet for PPS 13
ShortName:	VIDEO_DIP_PPS_DATA_13_C

VIDEO_DIP_PPS_DATA	
Reset:	soft
Address:	62388h-6238Bh
Name:	Transcoder Video Data Island Packet for PPS 14
ShortName:	VIDEO_DIP_PPS_DATA_14_C
Reset:	soft
Address:	6238Ch-6238Fh
Name:	Transcoder Video Data Island Packet for PPS 15
ShortName:	VIDEO_DIP_PPS_DATA_15_C
Reset:	soft
Address:	62390h-62393h
Name:	Transcoder Video Data Island Packet for PPS 16
ShortName:	VIDEO_DIP_PPS_DATA_16_C
Reset:	soft
Address:	62394h-62397h
Name:	Transcoder Video Data Island Packet for PPS 17
ShortName:	VIDEO_DIP_PPS_DATA_17_C
Reset:	soft
Address:	62398h-6239Bh
Name:	Transcoder Video Data Island Packet for PPS 18
ShortName:	VIDEO_DIP_PPS_DATA_18_C
Reset:	soft
Address:	6239Ch-6239Fh
Name:	Transcoder Video Data Island Packet for PPS 19
ShortName:	VIDEO_DIP_PPS_DATA_19_C
Reset:	soft
Address:	623A0h-623A3h
Name:	Transcoder Video Data Island Packet for PPS 20
ShortName:	VIDEO_DIP_PPS_DATA_20_C
Reset:	soft
Address:	623A4h-623A7h
Name:	Transcoder Video Data Island Packet for PPS 21
ShortName:	VIDEO_DIP_PPS_DATA_21_C
Reset:	soft
Address:	623A8h-623ABh
Name:	Transcoder Video Data Island Packet for PPS 22
ShortName:	VIDEO_DIP_PPS_DATA_22_C

VIDEO_DIP_PPS_DATA	
Reset:	soft
Address:	623ACh-623AFh
Name:	Transcoder Video Data Island Packet for PPS 23
ShortName:	VIDEO_DIP_PPS_DATA_23_C
Reset:	soft
Address:	623B0h-623B3h
Name:	Transcoder Video Data Island Packet for PPS 24
ShortName:	VIDEO_DIP_PPS_DATA_24_C
Reset:	soft
Address:	623B4h-623B7h
Name:	Transcoder Video Data Island Packet for PPS 25
ShortName:	VIDEO_DIP_PPS_DATA_25_C
Reset:	soft
Address:	623B8h-623BBh
Name:	Transcoder Video Data Island Packet for PPS 26
ShortName:	VIDEO_DIP_PPS_DATA_26_C
Reset:	soft
Address:	623BCh-623BFh
Name:	Transcoder Video Data Island Packet for PPS 27
ShortName:	VIDEO_DIP_PPS_DATA_27_C
Reset:	soft
Address:	623C0h-623C3h
Name:	Transcoder Video Data Island Packet for PPS 28
ShortName:	VIDEO_DIP_PPS_DATA_28_C
Reset:	soft
Address:	623C4h-623C7h
Name:	Transcoder Video Data Island Packet for PPS 29
ShortName:	VIDEO_DIP_PPS_DATA_29_C
Reset:	soft
Address:	623C8h-623CBh
Name:	Transcoder Video Data Island Packet for PPS 30
ShortName:	VIDEO_DIP_PPS_DATA_30_C
Reset:	soft
Address:	623CCh-623CFh
Name:	Transcoder Video Data Island Packet for PPS 31
ShortName:	VIDEO_DIP_PPS_DATA_31_C

VIDEO_DIP_PPS_DATA	
Reset:	soft
Address:	623D0h-623D3h
Name:	Transcoder Video Data Island Packet for PPS 32
ShortName:	VIDEO_DIP_PPS_DATA_32_C
Reset:	soft
Address:	63350h-63353h
Name:	Transcoder Video Data Island Packet for PPS 0
ShortName:	VIDEO_DIP_PPS_DATA_0_D
Reset:	soft
Address:	63354h-63357h
Name:	Transcoder Video Data Island Packet for PPS 1
ShortName:	VIDEO_DIP_PPS_DATA_1_D
Reset:	soft
Address:	63358h-6335Bh
Name:	Transcoder Video Data Island Packet for PPS 2
ShortName:	VIDEO_DIP_PPS_DATA_2_D
Reset:	soft
Address:	6335Ch-6335Fh
Name:	Transcoder Video Data Island Packet for PPS 3
ShortName:	VIDEO_DIP_PPS_DATA_3_D
Reset:	soft
Address:	63360h-63363h
Name:	Transcoder Video Data Island Packet for PPS 4
ShortName:	VIDEO_DIP_PPS_DATA_4_D
Reset:	soft
Address:	63364h-63367h
Name:	Transcoder Video Data Island Packet for PPS 5
ShortName:	VIDEO_DIP_PPS_DATA_5_D
Reset:	soft
Address:	63368h-6336Bh
Name:	Transcoder Video Data Island Packet for PPS 6
ShortName:	VIDEO_DIP_PPS_DATA_6_D
Reset:	soft
Address:	6336Ch-6336Fh
Name:	Transcoder Video Data Island Packet for PPS 7
ShortName:	VIDEO_DIP_PPS_DATA_7_D

VIDEO_DIP_PPS_DATA	
Reset:	soft
Address:	63370h-63373h
Name:	Transcoder Video Data Island Packet for PPS 8
ShortName:	VIDEO_DIP_PPS_DATA_8_D
Reset:	soft
Address:	63374h-63377h
Name:	Transcoder Video Data Island Packet for PPS 9
ShortName:	VIDEO_DIP_PPS_DATA_9_D
Reset:	soft
Address:	63378h-6337Bh
Name:	Transcoder Video Data Island Packet for PPS 10
ShortName:	VIDEO_DIP_PPS_DATA_10_D
Reset:	soft
Address:	6337Ch-6337Fh
Name:	Transcoder Video Data Island Packet for PPS 11
ShortName:	VIDEO_DIP_PPS_DATA_11_D
Reset:	soft
Address:	63380h-63383h
Name:	Transcoder Video Data Island Packet for PPS 12
ShortName:	VIDEO_DIP_PPS_DATA_12_D
Reset:	soft
Address:	63384h-63387h
Name:	Transcoder Video Data Island Packet for PPS 13
ShortName:	VIDEO_DIP_PPS_DATA_13_D
Reset:	soft
Address:	63388h-6338Bh
Name:	Transcoder Video Data Island Packet for PPS 14
ShortName:	VIDEO_DIP_PPS_DATA_14_D
Reset:	soft
Address:	6338Ch-6338Fh
Name:	Transcoder Video Data Island Packet for PPS 15
ShortName:	VIDEO_DIP_PPS_DATA_15_D
Reset:	soft
Address:	63390h-63393h
Name:	Transcoder Video Data Island Packet for PPS 16
ShortName:	VIDEO_DIP_PPS_DATA_16_D

VIDEO_DIP_PPS_DATA	
Reset:	soft
Address:	63394h-63397h
Name:	Transcoder Video Data Island Packet for PPS 17
ShortName:	VIDEO_DIP_PPS_DATA_17_D
Reset:	soft
Address:	63398h-6339Bh
Name:	Transcoder Video Data Island Packet for PPS 18
ShortName:	VIDEO_DIP_PPS_DATA_18_D
Reset:	soft
Address:	6339Ch-6339Fh
Name:	Transcoder Video Data Island Packet for PPS 19
ShortName:	VIDEO_DIP_PPS_DATA_19_D
Reset:	soft
Address:	633A0h-633A3h
Name:	Transcoder Video Data Island Packet for PPS 20
ShortName:	VIDEO_DIP_PPS_DATA_20_D
Reset:	soft
Address:	633A4h-633A7h
Name:	Transcoder Video Data Island Packet for PPS 21
ShortName:	VIDEO_DIP_PPS_DATA_21_D
Reset:	soft
Address:	633A8h-633ABh
Name:	Transcoder Video Data Island Packet for PPS 22
ShortName:	VIDEO_DIP_PPS_DATA_22_D
Reset:	soft
Address:	633ACh-633AFh
Name:	Transcoder Video Data Island Packet for PPS 23
ShortName:	VIDEO_DIP_PPS_DATA_23_D
Reset:	soft
Address:	633B0h-633B3h
Name:	Transcoder Video Data Island Packet for PPS 24
ShortName:	VIDEO_DIP_PPS_DATA_24_D
Reset:	soft
Address:	633B4h-633B7h
Name:	Transcoder Video Data Island Packet for PPS 25
ShortName:	VIDEO_DIP_PPS_DATA_25_D

VIDEO_DIP_PPS_DATA

Reset:	soft
Address:	633B8h-633BBh
Name:	Transcoder Video Data Island Packet for PPS 26
ShortName:	VIDEO_DIP_PPS_DATA_26_D
Reset:	soft
Address:	633BCh-633BFh
Name:	Transcoder Video Data Island Packet for PPS 27
ShortName:	VIDEO_DIP_PPS_DATA_27_D
Reset:	soft
Address:	633C0h-633C3h
Name:	Transcoder Video Data Island Packet for PPS 28
ShortName:	VIDEO_DIP_PPS_DATA_28_D
Reset:	soft
Address:	633C4h-633C7h
Name:	Transcoder Video Data Island Packet for PPS 29
ShortName:	VIDEO_DIP_PPS_DATA_29_D
Reset:	soft
Address:	633C8h-633CBh
Name:	Transcoder Video Data Island Packet for PPS 30
ShortName:	VIDEO_DIP_PPS_DATA_30_D
Reset:	soft
Address:	633CCh-633CFh
Name:	Transcoder Video Data Island Packet for PPS 31
ShortName:	VIDEO_DIP_PPS_DATA_31_D
Reset:	soft
Address:	633D0h-633D3h
Name:	Transcoder Video Data Island Packet for PPS 32
ShortName:	VIDEO_DIP_PPS_DATA_32_D
Reset:	soft

DWord	Bit	Description
0	31:0	**Video DIP PPS data** Access: R/W Video data island packet for PPS data.

VIDEO_DIP_PPS_ECC

VIDEO_DIP_PPS_ECC	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	603D4h-603D7h
Name:	Transcoder Video DIP for PPS ECC 0
ShortName:	VIDEO_DIP_PPS_ECC_0_A
Reset:	soft
Address:	603D8h-603DBh
Name:	Transcoder Video DIP for PPS ECC 1
ShortName:	VIDEO_DIP_PPS_ECC_1_A
Reset:	soft
Address:	603DCh-603DFh
Name:	Transcoder Video DIP for PPS ECC 2
ShortName:	VIDEO_DIP_PPS_ECC_2_A
Reset:	soft
Address:	603E0h-603E3h
Name:	Transcoder Video DIP for PPS ECC 3
ShortName:	VIDEO_DIP_PPS_ECC_3_A
Reset:	soft
Address:	603E4h-603E7h
Name:	Transcoder Video DIP for PPS ECC 4
ShortName:	VIDEO_DIP_PPS_ECC_4_A
Reset:	soft
Address:	603E8h-603EBh
Name:	Transcoder Video DIP for PPS ECC 5
ShortName:	VIDEO_DIP_PPS_ECC_5_A
Reset:	soft
Address:	603ECh-603EFh
Name:	Transcoder Video DIP for PPS ECC 6
ShortName:	VIDEO_DIP_PPS_ECC_6_A
Reset:	soft
Address:	603F0h-603F3h
Name:	Transcoder Video DIP for PPS ECC 7
ShortName:	VIDEO_DIP_PPS_ECC_7_A

VIDEO_DIP_PPS_ECC	
Reset:	soft
Address:	603F4h-603F7h
Name:	Transcoder Video DIP for PPS ECC 8
ShortName:	VIDEO_DIP_PPS_ECC_8_A
Reset:	soft
Address:	613D4h-613D7h
Name:	Transcoder Video DIP for PPS ECC 0
ShortName:	VIDEO_DIP_PPS_ECC_0_B
Reset:	soft
Address:	613D8h-613DBh
Name:	Transcoder Video DIP for PPS ECC 1
ShortName:	VIDEO_DIP_PPS_ECC_1_B
Reset:	soft
Address:	613DCh-613DFh
Name:	Transcoder Video DIP for PPS ECC 2
ShortName:	VIDEO_DIP_PPS_ECC_2_B
Reset:	soft
Address:	613E0h-613E3h
Name:	Transcoder Video DIP for PPS ECC 3
ShortName:	VIDEO_DIP_PPS_ECC_3_B
Reset:	soft
Address:	613E4h-613E7h
Name:	Transcoder Video DIP for PPS ECC 4
ShortName:	VIDEO_DIP_PPS_ECC_4_B
Reset:	soft
Address:	613E8h-613EBh
Name:	Transcoder Video DIP for PPS ECC 5
ShortName:	VIDEO_DIP_PPS_ECC_5_B
Reset:	soft
Address:	613ECh-613EFh
Name:	Transcoder Video DIP for PPS ECC 6
ShortName:	VIDEO_DIP_PPS_ECC_6_B
Reset:	soft
Address:	613F0h-613F3h
Name:	Transcoder Video DIP for PPS ECC 7
ShortName:	VIDEO_DIP_PPS_ECC_7_B

VIDEO_DIP_PPS_ECC	
Reset:	soft
Address:	613F4h-613F7h
Name:	Transcoder Video DIP for PPS ECC 8
ShortName:	VIDEO_DIP_PPS_ECC_8_B
Reset:	soft
Address:	623D4h-623D7h
Name:	Transcoder Video DIP for PPS ECC 0
ShortName:	VIDEO_DIP_PPS_ECC_0_C
Reset:	soft
Address:	623D8h-623DBh
Name:	Transcoder Video DIP for PPS ECC 1
ShortName:	VIDEO_DIP_PPS_ECC_1_C
Reset:	soft
Address:	623DCh-623DFh
Name:	Transcoder Video DIP for PPS ECC 2
ShortName:	VIDEO_DIP_PPS_ECC_2_C
Reset:	soft
Address:	623E0h-623E3h
Name:	Transcoder Video DIP for PPS ECC 3
ShortName:	VIDEO_DIP_PPS_ECC_3_C
Reset:	soft
Address:	623E4h-623E7h
Name:	Transcoder Video DIP for PPS ECC 4
ShortName:	VIDEO_DIP_PPS_ECC_4_C
Reset:	soft
Address:	623E8h-623EBh
Name:	Transcoder Video DIP for PPS ECC 5
ShortName:	VIDEO_DIP_PPS_ECC_5_C
Reset:	soft
Address:	623ECh-623EFh
Name:	Transcoder Video DIP for PPS ECC 6
ShortName:	VIDEO_DIP_PPS_ECC_6_C
Reset:	soft
Address:	623F0h-623F3h
Name:	Transcoder Video DIP for PPS ECC 7
ShortName:	VIDEO_DIP_PPS_ECC_7_C

VIDEO_DIP_PPS_ECC	
Reset:	soft
Address:	623F4h-623F7h
Name:	Transcoder Video DIP for PPS ECC 8
ShortName:	VIDEO_DIP_PPS_ECC_8_C
Reset:	soft
Address:	633D4h-633D7h
Name:	Transcoder Video DIP for PPS ECC 0
ShortName:	VIDEO_DIP_PPS_ECC_0_D
Reset:	soft
Address:	633D8h-633DBh
Name:	Transcoder Video DIP for PPS ECC 1
ShortName:	VIDEO_DIP_PPS_ECC_1_D
Reset:	soft
Address:	633DCh-633DFh
Name:	Transcoder Video DIP for PPS ECC 2
ShortName:	VIDEO_DIP_PPS_ECC_2_D
Reset:	soft
Address:	633E0h-633E3h
Name:	Transcoder Video DIP for PPS ECC 3
ShortName:	VIDEO_DIP_PPS_ECC_3_D
Reset:	soft
Address:	633E4h-633E7h
Name:	Transcoder Video DIP for PPS ECC 4
ShortName:	VIDEO_DIP_PPS_ECC_4_D
Reset:	soft
Address:	633E8h-633EBh
Name:	Transcoder Video DIP for PPS ECC 5
ShortName:	VIDEO_DIP_PPS_ECC_5_D
Reset:	soft
Address:	633ECh-633EFh
Name:	Transcoder Video DIP for PPS ECC 6
ShortName:	VIDEO_DIP_PPS_ECC_6_D
Reset:	soft
Address:	633F0h-633F3h
Name:	Transcoder Video DIP for PPS ECC 7
ShortName:	VIDEO_DIP_PPS_ECC_7_D

VIDEO_DIP_PPS_ECC

Reset:	soft
Address:	633F4h-633F7h
Name:	Transcoder Video DIP for PPS ECC 8
ShortName:	VIDEO_DIP_PPS_ECC_8_D
Reset:	soft

This represents 36 bytes of ECC over PPS DIP.

DWord	Bit	Description
0	31:0	**Video DIP PPS ECC** Access: RO ECC for PPS DIP.

Video BIOS ROM Base Address

ROMADR_0_2_0_PCI - Video BIOS ROM Base Address

Register Space:	PCI: 0/2/0
Size (in bits):	32
Address:	00030h

The IGD does not use a separate BIOS ROM, therefore this register is hardwired to 0s.

DWord	Bit	Description
0	31:18	**ROM Base Address** Default Value: 00000000000000b Access: RO _Custom_GTIReset: BUS Hardwired to 0's.
	17:11	**Address Mask** Default Value: 0000000b Access: RO _Custom_GTIReset: BUS Hardwired to 0s to indicate 256 KB address range.
	10:1	**Reserved** Access: RO Format: MBZ
	0	**ROM BIOS Enable** Default Value: 0b Access: RO _Custom_GTIReset: BUS Hardwired to 0 to indicate ROM not accessible.

VIRTUALIZATION CONTROL REGISTER

VIRTUAL_CTRL_REG - VIRTUALIZATION CONTROL REGISTER

Register Space:	MMIO: 0/2/0
Size (in bits):	32
SOC_Consumer:	BIOS
Address:	10108Ch

This register is used to identify which Virtual Function, if any, is allowed to access or configure the Global GTT Entry.

DWord	Bit	Description
0	31:9	**Reserved** Access: RO Format: MBZ
	8	**Guest Direct GGTT Update Enable** Access: R/W _Custom_GTIReset: BUS 1'b0 : Guest VM can't directly update GGTT Entries via GSM range within VF GTTMMADR 1'b1 : Guest VM can modify the physical address field of a GGTT Entry if the Function Number field of that entry matches the Function Number associated with the GTTMMADR range that was targeted by the access. No other field of the GGTT Entry can be modified by the Guest VM. A Guest VM Read to a GGTT Entry is allowed if the Function Number field matches, but the Function Number field itself will read as 0 (a Guest VM should not know which VF it has been assigned to). Value: 0b — **[Default]** Value: 1b
	7:0	**Reserved** Access: RO Format: MBZ

VSC_EXT_SDP_CONF

VSC_EXT_SDP_CONF	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60288h-6028Bh
Name:	Transcoder VSD Extended SDP Configuration
ShortName:	VSC_EXT_SDP_CONF_A
Reset:	soft
Address:	61288h-6128Bh
Name:	Transcoder VSD Extended SDP Configuration
ShortName:	VSC_EXT_SDP_CONF_B
Reset:	soft
Address:	62288h-6228Bh
Name:	Transcoder VSD Extended SDP Configuration
ShortName:	VSC_EXT_SDP_CONF_C
Reset:	soft
Address:	63288h-6328Bh
Name:	Transcoder VSD Extended SDP Configuration
ShortName:	VSC_EXT_SDP_CONF_D
Reset:	soft

DWord	Bit	Description
0	31	**Chain done** Access: RO Read by software/DSB to know if it can start sending new chain. Chained metadata straddling 2 buffers need to know this status. Value 0b: HW sending previous chain Value 1b: HW completed sending previous chain
	30:25	**Reserved** Access: RO Format: MBZ
	24	**Reserved** Access: RO Format: MBZ

VSC_EXT_SDP_CONF

23 — SDP Active Error

Access:	R/WC

This bit represents an error condition that not all SDPs were read when video active region is reached.
HW will stop transmitting metadata once active region is reached.
This bit is set by HW and cleared by SW by writing '1'.
To avoid false indication of an error, SW can choose either of the following methods:

1. Avoid programming VSC_EXT registers and setting the buffer ready bit in VSC_EXT_SDP_CTL before vblank rises.
2. Program VSC_EXT registers and set the buffer ready bit in VSC_EXT_SDP_CTL anytime, but clear the SDP Active Error at the start of vblank, so that it can be rechecked at the end of vblank to indicate real issues.

Value	Name
0b	All SDPs transmitted in VBLANK
1b	All SDPs not transmitted in VBLANK

22 — Block mode

When this bit is set to 1, it is software responsibility to set Middle Of Chaining bit [23] and Packet sequence bits [28:24] in the header packet that it programs in metadata register.
HW does not support changing this field during transmission.

Value	Name
0b	Header values from Header register
1b	Header values from SW/DSB written buffer

21:10 — HBLANK early indication

In HW, this field is expressed in terms of pixels and not clocks.
It provides a guardband to stop the VSC_EXT_SDP transmission from going into Hblank region (for Audio).
For eg: If this field is programed to 20 pixels, then when horizontal counter reaches (HActive -20), VSC_EXT_SDP transmission stops.
If a packet is in the middle of transmission, then it completes before stopping. Transmission resumes on the next HActive.
In MST/DP2 this transmission can happen only when an MTP slot is available.
For all SST/MST/DP2 use cases, recommendation is to use 10% of HActive.

9:0 — Packets per chain

This field specifies the total number of packets (each packet has 32 bytes) in the chain.
= Number of bytes/32

VSC_EXT_SDP_CTL

VSC_EXT_SDP_CTL	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60290h-60293h
Name:	Transcoder VSD Extended SDP Control 0
ShortName:	VSC_EXT_SDP_CTL_0_A
Reset:	soft
Address:	60294h-60297h
Name:	Transcoder VSD Extended SDP Control 1
ShortName:	VSC_EXT_SDP_CTL_1_A
Reset:	soft
Address:	61290h-61293h
Name:	Transcoder VSD Extended SDP Control 0
ShortName:	VSC_EXT_SDP_CTL_0_B
Reset:	soft
Address:	61294h-61297h
Name:	Transcoder VSD Extended SDP Control 1
ShortName:	VSC_EXT_SDP_CTL_1_B
Reset:	soft
Address:	62290h-62293h
Name:	Transcoder VSD Extended SDP Control 0
ShortName:	VSC_EXT_SDP_CTL_0_C
Reset:	soft
Address:	62294h-62297h
Name:	Transcoder VSD Extended SDP Control 1
ShortName:	VSC_EXT_SDP_CTL_1_C
Reset:	soft
Address:	63290h-63293h
Name:	Transcoder VSD Extended SDP Control 0
ShortName:	VSC_EXT_SDP_CTL_0_D
Reset:	soft
Address:	63294h-63297h
Name:	Transcoder VSD Extended SDP Control 1
ShortName:	VSC_EXT_SDP_CTL_1_D

VSC_EXT_SDP_CTL

Reset: soft

The VSC_EXT_VESA SDP allows a DP Source device to send video stream-related metadata aligned with the stream. A single VSC_EXT SDP is composed of four header bytes and 32 payload bytes. The VSC_EXT_VESA SDP maybe chained. Chaining framework enables a flexible metadata transportation framework to allow the transport of variable bytes of payload using a single VSC_EXT_VESA.

For transmitting frame synchronous metadata beyond 1KB, software must alternate between Buffer0 and Buffer1.

Metadata must be programmed in multiple of 32 bytes (8 dwords).

DWord	Bit	Description
0	31	**VSC extension SDP metadata enable** **NOTE:** This field is ignored on the register instance VSC_EXT_SDP_CTL_1. Value: 0b — Name: Disable Value: 1b — Name: Enable
	30:25	**Reserved** Access: RO Format: MBZ
	24	**Buffer Empty** Access: RO This field is set by hardware when buffer is empty and reset when buffer has metadata to be sent over the link. This field is read by software (or DSB) to know when it can start writing to the buffer (buffer reuse cases included). Value: 1b — Name: buffer empty **[Default]** Value: 0b — Name: buffer not empty
	23:17	**Reserved** Access: RO Format: MBZ
	16	**Buffer Ready** Access: Write/Read Status This field is set by software (or DSB) to indicate that the buffer is ready for hardware use. Hardware clears this field after the buffer processing is complete. SW can also clear this bit. Both SW clear and SW set will have higher priority over HW clear. Value: 1b — Name: Ready For HW Use Value: 0b — Name: Not ready for HW Use

VSC_EXT_SDP_CTL		
	15	**Reserved** Access: RO Format: MBZ
	14	**Index Auto Increment** This field controls the index auto increment.
	13:8	**Reserved** Access: RO Format: MBZ
	7:0	**Index Value** This field indicates the data location to be accessed through the data register. This value can be automatically incremented by a write to the data register if the index auto increment bit (14) is set. When automatically incrementing, the current automatically calculated index value can be read here, and the index will roll over to 0 after reaching the end of the allowed range. 8-bit index supports up to 1KB. Examples: 00h =Dword at location 0 FFh = Dword at location 255

VSC_EXT_SDP_DATA

VSC_EXT_SDP_DATA	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	60298h-6029Bh
Name:	Transcoder VSD Extended SDP DATA 0
ShortName:	VSC_EXT_SDP_DATA_0_A
Reset:	soft
Address:	6029Ch-6029Fh
Name:	Transcoder VSD Extended SDP DATA 1
ShortName:	VSC_EXT_SDP_DATA_1_A
Reset:	soft
Address:	61298h-6129Bh
Name:	Transcoder VSD Extended SDP DATA 0
ShortName:	VSC_EXT_SDP_DATA_0_B
Reset:	soft
Address:	6129Ch-6129Fh
Name:	Transcoder VSD Extended SDP DATA 1
ShortName:	VSC_EXT_SDP_DATA_1_B
Reset:	soft
Address:	62298h-6229Bh
Name:	Transcoder VSD Extended SDP DATA 0
ShortName:	VSC_EXT_SDP_DATA_0_C
Reset:	soft
Address:	6229Ch-6229Fh
Name:	Transcoder VSD Extended SDP DATA 1
ShortName:	VSC_EXT_SDP_DATA_1_C
Reset:	soft
Address:	63298h-6329Bh
Name:	Transcoder VSD Extended SDP DATA 0
ShortName:	VSC_EXT_SDP_DATA_0_D
Reset:	soft
Address:	6329Ch-6329Fh
Name:	Transcoder VSD Extended SDP DATA 1
ShortName:	VSC_EXT_SDP_DATA_1_D

VSC_EXT_SDP_DATA		
Reset:	soft	
DWord	**Bit**	**Description**
0	31:0	**Data** This field specifies the data for the index specified in VSC EXT SDP CTL register. Index [0-31] allows up to 1KB data. This register must be written only as a full 32 bit dword. Byte or word writes are not supported. This field can be read back only when "VSC extension SDP metadata enable" field in VSC_EXT_SDP_CTL is deasserted.

VSC_EXT_SDP_HEADER

VSC_EXT_SDP_HEADER

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6028Ch-6028Fh
Name:	Transcoder VSD Extended SDP Header
ShortName:	VSC_EXT_SDP_HEADER_A
Reset:	soft
Address:	6128Ch-6128Fh
Name:	Transcoder VSD Extended SDP Header
ShortName:	VSC_EXT_SDP_HEADER_B
Reset:	soft
Address:	6228Ch-6228Fh
Name:	Transcoder VSD Extended SDP Header
ShortName:	VSC_EXT_SDP_HEADER_C
Reset:	soft
Address:	6328Ch-6328Fh
Name:	Transcoder VSD Extended SDP Header
ShortName:	VSC_EXT_SDP_HEADER_D
Reset:	soft

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:24	**Packet sequence** Range is from 0 to 31. Shall be cleared to 0 on the first SDP of a new VSC_EXT packet sequence. When bit 6 of HB2 is set to 1, this field shall increment by 1 on each subsequent chained packet. This counter rolls over to 0 when it exceeds 31.
	23	**Middle Of Chaining** This bit specifies if there are any chained packets to follow. This bit has to be set if block mode(VSC_EXT_SDP_CONF[block mode]) is set. Value: 0b — Name: No chained packets to follow Value: 1b — Name: Chained packet(s) to follow.
	22	**Variable Packet Sequence Number** This bit specifies how the packed sequence field is used.

VSC_EXT_SDP_HEADER

		Value / **Name** 0b — Packet Sequence Field is fixed 1b — Packet Sequence field is incremented with chained packets
	21:18	**VSC EXT VESA SDP Framework Ver 1** Specifies the VSC EXT VESA SDP Framework Version, to be left at default '0'.
	17:16	**VSC EXT VESA SDP Framework Ver 0** Specifies the VSC EXT VESA SDP Framework Version, to be left at default '0'.
	15:8	**SDP Type** Specifies the Secondary Data Packet type. **Value** / **Name** 20h — VESA **[Default]** 21h — CEA
	7:0	**SDP ID** Specifies the Secondary Data Packet ID. Specific to stream (usually 00h).

VS Invocation Counter

VS_INVOCATION_COUNT - VS Invocation Counter	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	64
_Custom_GTIReset:	DEV
Address:	02320h-02327h
Name:	VS Invocation Counter
ShortName:	VS_INVOCATION_COUNT_RCSUNIT_BE_GEOMETRY
Address:	18320h-18327h
Name:	VS Invocation Counter
ShortName:	VS_INVOCATION_COUNT_POCSUNIT_BE_GEOMETRY

This register stores the value of the vertex count shaded by VS. This register is part of the context save and restore.

DWord	Bit	Description
0..1	63:32	**VS Invocation Count Report UDW** Number of vertices that are dispatched as threads by the VS stage. Updated only when **Statistics Enable** is set in VS_STATE (see the Vertex Shader Chapter in the 3D Volume.)
	31:0	**VS Invocation Count Report LDW** Number of vertices that are dispatched as threads by the VS stage. Updated only when **Statistics Enable** is set in VS_STATE (see the Vertex Shader Chapter in the 3D Volume.)

VSR_PUSH_CONSTANT_BASE

VSR_PUSH_CONSTANT_BASE - VSR_PUSH_CONSTANT_BASE

Register Space:	MMIO: 0/2/0
Size (in bits):	32
Address:	0E518h
Name:	VSR_PUSH_CONSTANT_BASE
ShortName:	VSR_PUSH_CONSTANT_BASE

DWord	Bit	Description
0	31:17	**Reserved** Access: RO Format: MBZ
	16:6	**VSR PUSH CONSTANT BASE** Default Value: 200h Access: R/W **Description** This is an 64 Byte aligned offset in to the URB indicating the base of the push constant allocation space for the push constant allocation space for VSR unit in POSH pipeline. The offset and size of the VSR allocation is relative to this base address.
	5:0	**Reserved** Access: RO Format: MBZ

VTd Status

VTD_STATUS_0_2_0_PCI - VTd Status

Register Space:	PCI: 0/2/0
Size (in bits):	8
Address:	0006Ch

This register contains indicator bits for Graphics VTd mode.

DWord	Bit	Description
0	7:1	**Reserved** Access: RO Format: MBZ
	0	**GFX VTd Active** Default Value: 0b Access: R/W _Custom_GTIReset: BUS Reflects GFX VTd Mode is active. 1 - if active, 0 if inactive. Acts as R/W register only during Punit restore - when iommu freeze bit is set. RO otherwise

Wait For Event and Display Flip Flags Register

SYNC_FLIP_STATUS - Wait For Event and Display Flip Flags Register

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV

Address:	022D0h-022D3h
Name:	Wait For Event and Display Flip Flags Register
ShortName:	SYNC_FLIP_STATUS_RCSUNIT_DISP

Address:	222D0h-222D3h
Name:	Wait For Event and Display Flip Flags Register
ShortName:	SYNC_FLIP_STATUS_BCSUNIT_DISP

This register is the saved value of what wait for events are still valid. This register is part of context save and restore for RC6 feature.

Programming Notes

Programming Restriction: This register should NEVER be programmed by SW, this is for HW internal use only.

VideoCS/VideoCS2/VideoEnhancementCS: This register functionality is not supported and must not be exercised for reads or writes.

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Display Plane 1 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 1 "Flip Pending" condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	29	**Display Plane 1 Syncronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 1 "Flip Pending" condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	28	**Display Plane 4 Syncronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 4 "Flip Pending" condition. If a flip

request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.

27 **Reserved**

Access:	RO
Format:	MBZ

26 **Display Plane 2 Asyncronous Display Flip Pending**

Format:	Enable

This field enables a wait for the duration of a Display Plane 2 "Flip Pending" condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.

25 **Display Plane 2 Syncronous Flip Display Pending**

Format:	Enable

This field enables a wait for the duration of a Display Plane 2 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.

24 **Display Plane 5 Syncronous Flip Display Pending**

Format:	Enable

This field enables a wait for the duration of a Display Plane 5 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.

23 **Display Plane 1 Asyncronous Performance Flip Pending Wait Enable**

Source:	RenderCS
Format:	Enable

This field enables a wait for the duration of a Display Plane 1 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.

22 **Display Plane 1 Asyncronous Flip Pending Wait Enable**

Format:	Enable

This field enables a wait for the duration of a Display Plane 1 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending

SYNC_FLIP_STATUS - Wait For Event and Display Flip Flags Register

		Condition (in the Device Programming Interface chapter of MI Functions.
	21	**Display Plane 1 Syncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 1 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	20	**Display Plane 4 Syncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 4 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	19	**Reserved** Access: RO Format: MBZ
	18	**Display Pipe A Scan Line Wait Enable** Format: Enable This field enables a wait while a Display Pipe A Scan Line condition exists. This condition is defined as the the start of the scan line specified in the Pipe A Display Scan Line Count Range Compare Register. See Scan Line Event in the Device Programming Interface chapter of MI Functions.
	17	**Display Pipe A Vertical Blank Wait Enable** Format: Enable This field enables a wait until the next Display Pipe A Vertical Blank event occurs. This event is defined as the start of the next Display Pipe A vertical blank period. Note that this can cause a wait for up to an entire refresh period. See Vertical Blank Event (See Programming Interface).
	16	**Reserved** Access: RO Format: MBZ
	15	**Display Plane 2 Asyncronous Performance Flip Pending Wait Enable** Source: RenderCS Format: Enable This field enables a wait for the duration of a Display Plane 2 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending

SYNC_FLIP_STATUS - Wait For Event and Display Flip Flags Register

		Condition (in the Device Programming Interface chapter of MI Functions.
	14	**Display Plane 2 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 2 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	13	**Display Plane 2 Syncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 2 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	12	**Display Plane 5 Syncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 5 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	11	**Reserved** Access: RO Format: MBZ
	10	**Display Pipe B Scan Line Wait Enable** Format: Enable This field enables a wait while a Display Pipe B Scan Line condition exists. This condition is defined as the the start of the scan line specified in the Pipe B Display Scan Line Count Range Compare Register. See Scan Line Event in the Device Programming Interface chapter of MI Functions.
	9	**Display Pipe B Vertical Blank Wait Enable** Format: Enable This field enables a wait until the next Display Pipe B Vertical Blank event occurs. This event is defined as the start of the next Display Pipe B vertical blank period. Note that this can cause a wait for up to an entire refresh period. See Vertical Blank Event (See Programming Interface).
	8:0	**Reserved** Access: RO Format: MBZ

Wait For Event and Display Flip Flags Register 1

SYNC_FLIP_STATUS_1 - Wait For Event and Display Flip Flags Register 1

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV

Address:	022D4h-022D7h
Name:	Wait For Event and Display Flip Flags Register 1
ShortName:	SYNC_FLIP_STATUS_1_RCSUNIT_DISP

Address:	222D4h-222D7h
Name:	Wait For Event and Display Flip Flags Register 1
ShortName:	SYNC_FLIP_STATUS_1_BCSUNIT_DISP

This register is the saved value of what wait for events are still valid. This register is part of context save and restore for RC6 feature.

Programming Notes

Programming Restriction: This register should NEVER be programmed by SW, this is for HW internal use only.

VideoCS/VideoCS2/VideoEnhancementCS: This register functionality is not supported and must not be exercised for reads or writes.

DWord	Bit	Description
0	31:27	**Reserved** Access: RO Format: MBZ
	26	**Display Plane 12 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 12 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	25	**Display Plane 12 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 12 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_1 - Wait For Event and Display Flip Flags Register 1

	24	**Display Plane 11 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 11 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	23	**Display Plane 11 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 11 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	22	**Display Plane 10 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 10 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	21	**Display Plane 10 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 10 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	20	**Display Plane 9 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 9 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	19	**Display Plane 9 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 9 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_1 - Wait For Event and Display Flip Flags Register 1

	18	**Display Plane 8 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 8 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	17	**Display Plane 8 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 8 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	16	**Display Plane 7 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 7 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	15	**Display Plane 7 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 7 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	14	**Display Pipe C Scan Line Event Pending** Format: Enable This field indicates scan line event operation is pending from Display Pipe C. This field gets set when MI_SCANLINE_INCL/EXCL command is parsed for Display Plane-C and gets reset on scan line event completion for Display Plane-C.
	13	**Display Pipe B Scan Line Event Pending** Format: Enable This field indicates scan line event operation is pending from Display Pipe B. This field gets set when MI_SCANLINE_INCL/EXCL command is parsed for Display Plane-B and gets reset on scan line event completion for Display Plane 3.

SYNC_FLIP_STATUS_1 - Wait For Event and Display Flip Flags Register 1

	12	**Display Pipe A Scan Line Event Pending** Format: Enable This field indicates scan line event operation is pending from Display Pipe A. This field gets set when MI_SCANLINE_INCL/EXCL command is parsed for Display Plane-A and gets reset on scan line event completion for Display Plane 1.
	11	**Reserved** Access: RO Format: MBZ
	10	**Display Plane 3 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 3 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	9	**Display Plane 3 Syncronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 3 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	8	**Display Plane 6 Syncronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 6 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	7	**Display Plane 3 Asyncronous Performance Flip Pending Wait Enable** Source: RenderCS Format: Enable This field enables a wait for the duration of a Display Plane 3 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_1 - Wait For Event and Display Flip Flags Register 1

	6	**Display Plane 3 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 3 "Flip Pending" condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	5	**Display Plane 3 Syncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 3 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	4	**Display Plane 6 Syncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 6 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	3	**Reserved** Access: RO Format: MBZ
	2	**Display Pipe C Scan Line Wait Enable** Format: Enable This field enables a wait while a Display Pipe C Scan Line condition exists. This condition is defined as the the start of the scan line specified in the Pipe C Display Scan Line Count Range Compare Register. See Scan Line Event in the Device Programming Interface chapter of MI Functions.
	1	**Display Pipe C Vertical Blank Wait Enable** Format: Enable This field enables a wait until the next Display Pipe C Vertical Blank event occurs. This event is defined as the start of the next Display Pipe C vertical blank period. Note that this can cause a wait for up to an entire refresh period. See Vertical Blank Event (See Programming Interface).
	0	**Reserved** Access: RO Format: MBZ

Wait For Event and Display Flip Flags Register 2

SYNC_FLIP_STATUS_2 - Wait For Event and Display Flip Flags Register 2

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV

Address:	022ECh-022EFh
Name:	Wait For Event and Display Flip Flags Register 2
ShortName:	SYNC_FLIP_STATUS_2_RCSUNIT_DISP

Address:	222ECh-222EFh
Name:	Wait For Event and Display Flip Flags Register 2
ShortName:	SYNC_FLIP_STATUS_2_BCSUNIT_DISP

This register is the saved value of what wait for events are still valid. This register is part of context save and restore for RC6 feature.

Programming Notes

Programming Restriction: This register should NEVER be programmed by SW, this is for HW internal use only.

VideoCS/VideoCS2/VideoEnhancementCS: This register functionality is not supported and must not be exercised for reads or writes.

DWord	Bit	Description
0	31:27	**Reserved** Access: RO Format: MBZ
	26	**Display Plane 12 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 12 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	25	**Display Plane 12 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 12 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_2 - Wait For Event and Display Flip Flags Register 2

	24	**Display Plane 12 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 12 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	23	**Display Plane 11 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 11 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	22	**Display Plane 11 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 11 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	21	**Display Plane 11 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 11 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	20	**Display Plane 10 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 10 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	19	**Display Plane 10 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 10 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_2 - Wait For Event and Display Flip Flags Register 2

	18	**Display Plane 10 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 10 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	17	**Display Plane 9 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 9 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	16	**Display Plane 9 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 9 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	15	**Display Plane 9 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 9 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	14	**Display Plane 8 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 8 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	13	**Display Plane 8 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 8 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_2 - Wait For Event and Display Flip Flags Register 2

	12	**Display Plane 8 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 8 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	11	**Display Plane 7 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 7 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	10	**Display Plane 7 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 7 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	9	**Display Plane 7 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 7 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	8	**Display Plane 6 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 6 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	7	**Display Plane 6 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 6 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_2 - Wait For Event and Display Flip Flags Register 2

	6	**Display Plane 6 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 6 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	5	**Display Plane 5 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 5 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	4	**Display Plane 5 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 5 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	3	**Display Plane 5 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 5 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	2	**Display Plane 4 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 4 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	1	**Display Plane 4 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 4 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_2 - Wait For Event and Display Flip Flags Register 2

	0	**Display Plane 4 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 4 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.

Wait For Event and Display Flip Flags Register 3

SYNC_FLIP_STATUS_3 - Wait For Event and Display Flip Flags Register 3

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	022B8h-022BBh
Name:	SYNC_FLIP_STATUS_3
ShortName:	SYNC_FLIP_STATUS_3_RCSUNIT_DISP
Address:	222B8h-222BBh
Name:	SYNC_FLIP_STATUS_3
ShortName:	SYNC_FLIP_STATUS_3_BCSUNIT_DISP

This register is the saved value of what wait for events are still valid. This register is part of context save and restore for RC6 feature.

Programming Notes

Programming Restriction: This register should NEVER be programmed by SW, this is for HW internal use only.

VideoCS/VideoCS2/VideoEnhancementCS: This register functionality is not supported and must not be exercised for reads or writes.

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29	**Display Plane 18 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 18 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	28	**Display Plane 18 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 18 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_3 - Wait For Event and Display Flip Flags Register 3

	27	**Display Plane 18 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 18 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	26	**Display Plane 18 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 18 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	25	**Display Plane 18 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 18 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	24	**Display Plane 17 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 17 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	23	**Display Plane 17 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 17 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	22	**Display Plane 17 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 17 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_3 - Wait For Event and Display Flip Flags Register 3

	21	**Display Plane 17 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 17 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	20	**Display Plane 17 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 17 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	19	**Display Plane 16 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 1616 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	18	**Display Plane 16 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	17	**Display Plane 16 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 16 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	16	**Display Plane 16 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 16 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_3 - Wait For Event and Display Flip Flags Register 3

	15	**Display Plane 16 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 16 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	14	**Display Plane 15 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 15 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	13	**Display Plane 15 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 15 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	12	**Display Plane 15 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 15 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	11	**Display Plane 15 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 15 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	10	**Display Plane 15 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 15 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_3 - Wait For Event and Display Flip Flags Register 3

	9	**Display Plane 14 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 4 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	8	**Display Plane 14 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 14 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	7	**Display Plane 14 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 14 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	6	**Display Plane 14 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 14 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	5	**Display Plane 14 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 14 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	4	**Display Plane 13 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 13 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_3 - Wait For Event and Display Flip Flags Register 3

	3	**Display Plane 13 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 13 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	2	**Display Plane 13 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 13 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	1	**Display Plane 13 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 13 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	0	**Display Plane 13 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 13 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.

Wait For Event and Display Flip Flags Register 4

SYNC_FLIP_STATUS_4 - Wait For Event and Display Flip Flags Register 4

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV

Address:	022C0h-022C3h
Name:	SYNC_FLIP_STATUS_4
ShortName:	SYNC_FLIP_STATUS_4_RCSUNIT_DISP

Address:	222C0h-222C3h
Name:	SYNC_FLIP_STATUS_4
ShortName:	SYNC_FLIP_STATUS_4_BCSUNIT_DISP

This register is the saved value of what wait for events are still valid. This register is part of context save and restore for RC6 feature.

Programming Notes

Programming Restriction: This register should NEVER be programmed by SW, this is for HW internal use only.

VideoCS/VideoCS2/VideoEnhancementCS: This register functionality is not supported and must not be exercised for reads or writes.

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29	**Display Plane 24 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 24 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	28	**Display Plane 24 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 24 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_4 - Wait For Event and Display Flip Flags Register 4

	27	**Display Plane 24 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 24 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	26	**Display Plane 24 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 24 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	25	**Display Plane 24 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 24 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	24	**Display Plane 23 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 23 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	23	**Display Plane 23 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 23 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	22	**Display Plane 23 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 23 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_4 - Wait For Event and Display Flip Flags Register 4

	21	**Display Plane 23 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 23 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	20	**Display Plane 23 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 23 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	19	**Display Plane 22 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 22 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	18	**Display Plane 22 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 22 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	17	**Display Plane 22 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 22 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	16	**Display Plane 22 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 22 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_4 - Wait For Event and Display Flip Flags Register 4

	15	**Display Plane 22 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 22 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	14	**Display Plane 21 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 21 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	13	**Display Plane 21 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 21 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	12	**Display Plane 21 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 21 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	11	**Display Plane 21 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 21 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	10	**Display Plane 21 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 21 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_4 - Wait For Event and Display Flip Flags Register 4

	9	**Display Plane 20 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 20 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	8	**Display Plane 20 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 20 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	7	**Display Plane 20 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 20 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	6	**Display Plane 20 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 20 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	5	**Display Plane 20 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 20 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	4	**Display Plane 19 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 19 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_4 - Wait For Event and Display Flip Flags Register 4

	3	**Display Plane 19 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 19 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	2	**Display Plane 19 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 19 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	1	**Display Plane 19 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 19 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	0	**Display Plane 19 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 19 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.

Wait For Event and Display Flip Flags Register 5

SYNC_FLIP_STATUS_5 - Wait For Event and Display Flip Flags Register 5

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	022C4h-022C7h
Name:	SYNC_FLIP_STATUS_5
ShortName:	SYNC_FLIP_STATUS_5_RCSUNIT_DISP
Address:	222C4h-222C7h
Name:	SYNC_FLIP_STATUS_5
ShortName:	SYNC_FLIP_STATUS_5_BCSUNIT_DISP

This register is the saved value of what wait for events are still valid. This register is part of context save and restore for RC6 feature.

Programming Notes

Programming Restriction: This register should NEVER be programmed by SW, this is for HW internal use only.

VideoCS/VideoCS2/VideoEnhancementCS: This register functionality is not supported and must not be exercised for reads or writes.

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29	**Display Plane 30 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 30 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	28	**Display Plane 30 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 30 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_5 - Wait For Event and Display Flip Flags Register 5

	27	**Display Plane 30 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 30 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	26	**Display Plane 30 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 30 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	25	**Display Plane 30 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 30 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	24	**Display Plane 29 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 29 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	23	**Display Plane 29 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 29 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	22	**Display Plane 29 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 29 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_5 - Wait For Event and Display Flip Flags Register 5

	21	**Display Plane 29 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 29 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	20	**Display Plane 29 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 29 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	19	**Display Plane 28 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 28 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	18	**Display Plane 28 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 28 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	17	**Display Plane 28 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 28 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	16	**Display Plane 28 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 28 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_5 - Wait For Event and Display Flip Flags Register 5

	15	**Display Plane 28 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 28 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	14	**Display Plane 27 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 27 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	13	**Display Plane 27 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 27 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	12	**Display Plane 27 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 27 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	11	**Display Plane 27 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 27 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	10	**Display Plane 27 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 27 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_5 - Wait For Event and Display Flip Flags Register 5

	9	**Display Plane 26 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 26 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	8	**Display Plane 26 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 26 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	7	**Display Plane 26 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 26 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	6	**Display Plane 26 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 26 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	5	**Display Plane 26 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 26 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	4	**Display Plane 25 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 25 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_5 - Wait For Event and Display Flip Flags Register 5

	3	**Display Plane 25 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 25 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	2	**Display Plane 25 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 25 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	1	**Display Plane 25 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 25 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	0	**Display Plane 25 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 25 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.

Wait For Event and Display Flip Flags Register 6

SYNC_FLIP_STATUS_6 - Wait For Event and Display Flip Flags Register 6

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV

Address:	021F8h-021FBh
Name:	SYNC_FLIP_STATUS_6
ShortName:	SYNC_FLIP_STATUS_6_RCSUNIT_DISP

Address:	221F8h-221FBh
Name:	SYNC_FLIP_STATUS_6
ShortName:	SYNC_FLIP_STATUS_6_BCSUNIT_DISP

This register is the saved value of what wait for events are still valid. This register is part of context save and restore for RC6 feature.

Programming Notes

Programming Restriction: This register should NEVER be programmed by SW, this is for HW internal use only.

VideoCS/VideoCS2/VideoEnhancementCS: This register functionality is not supported and must not be exercised for reads or writes.

DWord	Bit	Description
0	31:19	**Reserved** Access: RO Format: MBZ
	18	**Display Pipe D Scan Line Event Pending** Format: Enable This field indicates scan line event operation is pending from Display Pipe D. This field gets set when MI_SCANLINE_INCL/EXCL command is parsed for Display Pipe D and gets reset on scan line event completion for Display Plane-C.
	17	**Display Pipe D Scan Line Wait Enable** Format: Enable This field enables a wait while a Display Pipe D Scan Line condition exists. This condition is defined as the start of the scan line specified in the Pipe D Display Scan Line Count Range Compare Register. See Scan Line Event in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_6 - Wait For Event and Display Flip Flags Register 6

	16	**Display Pipe D Vertical Blank Wait Enable** Format: Enable This field enables a wait until the next Display Pipe D Vertical Blank event occurs. This event is defined as the start of the next Display Pipe D vertical blank period. Note that this can cause a wait for up to an entire refresh period. See Vertical Blank Event (See Programming Interface).
	15:10	**Reserved** Access: RO Format: MBZ
	9	**Display Plane 32 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 32 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	8	**Display Plane 32 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 32 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	7	**Display Plane 32 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 32 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	6	**Display Plane 32 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 32 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.

SYNC_FLIP_STATUS_6 - Wait For Event and Display Flip Flags Register 6

	5	**Display Plane 32 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 32 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	4	**Display Plane 31 Asyncronous Performance Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 31 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	3	**Display Plane 31 Asyncronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 31 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	2	**Display Plane 31 Asyncronous Display Flip Pending** Format: Enable This field enables a wait for the duration of a Display Plane 31 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition (in the Device Programming Interface chapter of MI Functions.
	1	**Display Plane 31 Synchronous Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 31 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.
	0	**Display Plane 31 Synchronous Flip Display Pending** Format: Enable This field enables a wait for the duration of a Display Plane 31 Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers). See Display Flip Pending Condition in the Device Programming Interface chapter of MI Functions.

Watchdog Counter

PR_CTR - Watchdog Counter	
Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02190h-02193h
Name:	Watchdog Counter
ShortName:	PR_CTR_RCSUNIT
Address:	18190h-18193h
Name:	Watchdog Counter
ShortName:	PR_CTR_POCSUNIT
Address:	22190h-22193h
Name:	Watchdog Counter
ShortName:	PR_CTR_BCSUNIT
Address:	1C0190h-1C0193h
Name:	Watchdog Counter
ShortName:	PR_CTR_VCSUNIT0
Address:	1C4190h-1C4193h
Name:	Watchdog Counter
ShortName:	PR_CTR_VCSUNIT1
Address:	1C8190h-1C8193h
Name:	Watchdog Counter
ShortName:	PR_CTR_VECSUNIT0
Address:	1D0190h-1D0193h
Name:	Watchdog Counter
ShortName:	PR_CTR_VCSUNIT2
Address:	1D4190h-1D4193h
Name:	Watchdog Counter
ShortName:	PR_CTR_VCSUNIT3
Address:	1D8190h-1D8193h
Name:	Watchdog Counter
ShortName:	PR_CTR_VECSUNIT1
Address:	1E0190h-1E0193h
Name:	Watchdog Counter
ShortName:	PR_CTR_VCSUNIT4

PR_CTR - Watchdog Counter

Address:	1E4190h-1E4193h
Name:	Watchdog Counter
ShortName:	PR_CTR_VCSUNIT5

Address:	1E8190h-1E8193h
Name:	Watchdog Counter
ShortName:	PR_CTR_VECSUNIT2

Address:	1F0190h-1F0193h
Name:	Watchdog Counter
ShortName:	PR_CTR_VCSUNIT6

Address:	1F4190h-1F4193h
Name:	Watchdog Counter
ShortName:	PR_CTR_VCSUNIT7

Address:	1F8190h-1F8193h
Name:	Watchdog Counter
ShortName:	PR_CTR_VECSUNIT3

Address:	1A190h-1A193h
Name:	Watchdog Counter
ShortName:	PR_CTR_CCSUNIT0

DWord	Bit	Description
0	31:0	**Counter Value** Format: U32 This register reflects the render watchdog counter value itself. It cannot be written to.

Watchdog Counter Control

PR_CTR_CTL - Watchdog Counter Control	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	02178h-0217Bh
Name:	Watchdog Counter Control
ShortName:	PR_CTR_CTL_RCSUNIT
Address:	18178h-1817Bh
Name:	Watchdog Counter Control
ShortName:	PR_CTR_CTL_POCSUNIT
Address:	22178h-2217Bh
Name:	Watchdog Counter Control
ShortName:	PR_CTR_CTL_BCSUNIT
Address:	1C0178h-1C017Bh
Name:	Watchdog Counter Control
ShortName:	PR_CTR_CTL_VCSUNIT0
Address:	1C4178h-1C417Bh
Name:	Watchdog Counter Control
ShortName:	PR_CTR_CTL_VCSUNIT1
Address:	1C8178h-1C817Bh
Name:	Watchdog Counter Control
ShortName:	PR_CTR_CTL_VECSUNIT0
Address:	1D0178h-1D017Bh
Name:	Watchdog Counter Control
ShortName:	PR_CTR_CTL_VCSUNIT2
Address:	1D4178h-1D417Bh
Name:	Watchdog Counter Control
ShortName:	PR_CTR_CTL_VCSUNIT3
Address:	1D8178h-1D817Bh
Name:	Watchdog Counter Control
ShortName:	PR_CTR_CTL_VECSUNIT1
Address:	1E0178h-1E017Bh
Name:	Watchdog Counter Control
ShortName:	PR_CTR_CTL_VCSUNIT4

PR_CTR_CTL - Watchdog Counter Control

Address:	1E4178h-1E417Bh
Name:	Watchdog Counter Control
ShortName:	PR_CTR_CTL_VCSUNIT5

Address:	1E8178h-1E817Bh
Name:	Watchdog Counter Control
ShortName:	PR_CTR_CTL_VECSUNIT2

Address:	1F0178h-1F017Bh
Name:	Watchdog Counter Control
ShortName:	PR_CTR_CTL_VCSUNIT6

Address:	1F4178h-1F417Bh
Name:	Watchdog Counter Control
ShortName:	PR_CTR_CTL_VCSUNIT7

Address:	1F8178h-1F817Bh
Name:	Watchdog Counter Control
ShortName:	PR_CTR_CTL_VECSUNIT3

Address:	1A178h-1A17Bh
Name:	Watchdog Counter Control
ShortName:	PR_CTR_CTL_CCSUNIT0

Programming Notes

Ring Buffer Mode of scheduling SW must enable and disable watch dog counter inline to a command sequence of any given workload within the same command buffer dispatch. Watch Dog counter once enabled doesn't stop unless it is explicitly disabled. SW must explicitly reset the watch dog counter by disabling it before enabling the watch dog counter for a new command sequence. Preemption could happen in a command sequence prior to watch dog counter getting disabled resulting in watch dog counter enabled following preemption. SW must explicitly take care of disabling the watch dog counter as part of the preemption sequence. Execution List Mode of Scheduling: SW must enable and disable watch dog counter inline to a command sequence of any given workload within the same command buffer dispatch. On a context switch Watch Dog Counter Control and Watch dog Threshold are context save restored, whereas watch dog counter gets reset to 0x0 and remains disabled until it gets enabled by another context during context restore or due to explicit programming. Watch dog counter value doesn't get accumulated across multiple submissions of a given context.

This register functionality is not supported and must not be programmed for Position command streamer.

DWord	Bit	Description
0	31	**Count Select** Format: U1 **Value / Name / Description** 0h **[Default]** Use eight times the time stamp base unit to increment the watch dog count. The granularity of the time stamp base unit is defined in the Time Stamp

PR_CTR_CTL - Watchdog Counter Control

				Bases subsection in Power Management chapter.
		1h		Use the fixed function clock (csclk) to increment the watchdog count
	30:0	**Counter Logic Op**		
		Default Value:		1h
		This field specifies the action to be taken by the clock counter to generate interrupts. Writing a Zero value to this register starts the counting. Writing a Value of 0000_0001 to this counter stops the counter.		

Watchdog Counter Threshold

PR_CTR_THRSH - Watchdog Counter Threshold	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0217Ch-0217Fh
Name:	Watchdog Counter Threshold
ShortName:	PR_CTR_THRSH_RCSUNIT
Address:	1817Ch-1817Fh
Name:	Watchdog Counter Threshold
ShortName:	PR_CTR_THRSH_POCSUNIT
Address:	2217Ch-2217Fh
Name:	Watchdog Counter Threshold
ShortName:	PR_CTR_THRSH_BCSUNIT
Address:	1C017Ch-1C017Fh
Name:	Watchdog Counter Threshold
ShortName:	PR_CTR_THRSH_VCSUNIT0
Address:	1C417Ch-1C417Fh
Name:	Watchdog Counter Threshold
ShortName:	PR_CTR_THRSH_VCSUNIT1
Address:	1C817Ch-1C817Fh
Name:	Watchdog Counter Threshold
ShortName:	PR_CTR_THRSH_VECSUNIT0
Address:	1D017Ch-1D017Fh
Name:	Watchdog Counter Threshold
ShortName:	PR_CTR_THRSH_VCSUNIT2
Address:	1D417Ch-1D417Fh
Name:	Watchdog Counter Threshold
ShortName:	PR_CTR_THRSH_VCSUNIT3
Address:	1D817Ch-1D817Fh
Name:	Watchdog Counter Threshold
ShortName:	PR_CTR_THRSH_VECSUNIT1
Address:	1E017Ch-1E017Fh
Name:	Watchdog Counter Threshold
ShortName:	PR_CTR_THRSH_VCSUNIT4

PR_CTR_THRSH - Watchdog Counter Threshold

Address:	1E417Ch-1E417Fh
Name:	Watchdog Counter Threshold
ShortName:	PR_CTR_THRSH_VCSUNIT5

Address:	1E817Ch-1E817Fh
Name:	Watchdog Counter Threshold
ShortName:	PR_CTR_THRSH_VECSUNIT2

Address:	1F017Ch-1F017Fh
Name:	Watchdog Counter Threshold
ShortName:	PR_CTR_THRSH_VCSUNIT6

Address:	1F417Ch-1F417Fh
Name:	Watchdog Counter Threshold
ShortName:	PR_CTR_THRSH_VCSUNIT7

Address:	1F817Ch-1F817Fh
Name:	Watchdog Counter Threshold
ShortName:	PR_CTR_THRSH_VECSUNIT3

Address:	1A17Ch-1A17Fh
Name:	Watchdog Counter Threshold
ShortName:	PR_CTR_THRSH_CCSUNIT0

Programming Notes
This register functionality is not supported and must not be programmed for Position command streamer.
This register must never be programmed with zero. This will cause the watchdog counter to exceed and not allow the engine to go into IDLE state.

DWord	Bit	Description
0	31:0	**Counter Logic Threshold** Default Value: 00145855h Format: U32 This field specifies the threshold that the hardware checks against for the value of the render clock counter before generating an interrupt. The counter in hardware generates an interrupt when the threshold is reached, rolls over and starts counting again. The interrupt generated is the "Media Hang Notify" interrupt since this watchdog timer is intended primarily to remedy VLD hangs on the main pipeline.

WD_27_M

WD_27_M

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6E524h-6E527h
Name:	WD0 27 MHz M
ShortName:	WD_27_M_0
Reset:	soft
Address:	6ED24h-6ED27h
Name:	WD1 27 MHz M
ShortName:	WD_27_M_1
Reset:	soft

DWord	Bit	Description
0	31	**Counter Force** This field forces the 27 MHz counter to enabled even if the WD function is disabled. This may be necessary when WD audio is used while WD video is disabled. Value: 1b — Name: Force Enabled Value: 0b — Name: Do Not Force
	30:24	**Reserved** Access: RO Format: MBZ
	23:0	**WD Link M** This field specifies the M value for the 27 MHz clock. Link M / Link N = 27 MHz / CD clock frequency See the Sequences for Changing CD Clock Frequency for the values to use. Value: 000002h — Name: 2 **[Default]** — Description: M=2

WD_27_N

WD_27_N

Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6E528h-6E52Bh
Name:	WD0 27 MHz N
ShortName:	WD_27_N_0
Reset:	soft
Address:	6EC28h-6EC2Bh
Name:	WD1 27 MHz N
ShortName:	WD_27_N_1
Reset:	soft

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23:0	**WD Link N** This field specifies the N value for the 27 MHz clock. Link M / Link N = 27 MHz / CD clock frequency See the Sequences for Changing CD Clock Frequency for the values to use. Value: 000019h; Name: 25 **[Default]**; Description: N=25

WD_FRAME_STATUS

WD_FRAME_STATUS

Register Space:	MMIO: 0/2/0
Access:	RO
Size (in bits):	32
Address:	6E568h-6E56Bh
Name:	WD0 Frame Status
ShortName:	WD_FRAME_STATUS_0
Reset:	soft
Address:	6ED68h-6ED6Bh
Name:	WD1 Frame Status
ShortName:	WD_FRAME_STATUS_1
Reset:	soft

DWord	Bit	Description
0	31	**Frame Complete** Access: R/WC This field is a sticky bit set when WD fully completes a frame. Clear by writing a 1 to it.
	30:27	**Reserved** Access: RO Format: MBZ
	26:24	**WD State** Access: RO This field indicates the live state of WD capture. Value / Name / Description: 000b / IDLE / Reset state 001b / CAPSTART / Start timing generator for normal capture 010b / FRAME_START / Send framestart to display pipe 011b / CAPACTIVE / Capturing data 100b / TG_DONE / Completed writing pixels. Waiting for frame completion. 101b / WDX_DONE / Fully completed frame. Waiting to start next frame. 110b / QUICK_CAP / Quick capture entry
	23	**Reserved**
	22:21	**Reserved** Access: RO Format: MBZ
	20	**Reserved**

WD_FRAME_STATUS		
	19:0	**Reserved** Access: RO Format: MBZ

WD_IIR

WD_IIR	
Register Space:	MMIO: 0/2/0
Access:	R/WC
Size (in bits):	32
Address:	6E564h-6E567h
Name:	WD0 Interrupt Identity
ShortName:	WD_IIR_0
Reset:	soft
Address:	6ED64h-6ED67h
Name:	WD1 Interrupt Identity
ShortName:	WD_IIR_1
Reset:	soft

See the WD interrupt bit definition to find the source event for each interrupt bit.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Interrupt Identity Bits** This field holds the persistent values of the WD interrupt bits which are unmasked by the WD_IMR. Bits set in this register will propagate to the WD interrupt in the Display Engine Miscellaneous Interrupts. Bits set in this register will remain set (persist) until the interrupt condition is cleared by writing a '1' to the appropriate bits. 0b: Condition Not Detected 1b: Condition Detected

WD_IMR

WD_IMR	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	6E560h-6E563h
Name:	WD0 Interrupt Mask
ShortName:	WD_IMR_0
Reset:	soft
Address:	6ED60h-6ED63h
Name:	WD1 Interrupt Mask
ShortName:	WD_IMR_1
Reset:	soft

See the WD interrupt bit definition to find the source event for each interrupt bit.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Interrupt Mask Bits** This field contains a bit mask which selects which WD events are reported in the WD_IIR. Value: 0b — Name: Not Masked Value: 1b — Name: Masked Value: 0000FFFFh — Name: All interrupts masked **[Default]**

WD_PERF_CNT

WD_PERF_CNT		
Register Space:	MMIO: 0/2/0	
Access:	Write/Read Status	
Size (in bits):	32	
Address:	6E55Ch-6E55Fh	
Name:	WD0 Performance Counter	
ShortName:	WD_PERF_CNT_0	
Reset:	soft	
Address:	6ED5Ch-6ED5Fh	
Name:	WD1 Performance Counter	
ShortName:	WD_PERF_CNT_1	
Reset:	soft	

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23:0	**WD Perf Cnt** This field increments every millisecond while capturing. It does not count the time after capture is completed and waiting for the next capsync. Writes to this register will set the count to the written value, then it will increment from that value onwards.

WD_STRIDE

WD_STRIDE	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of capture sync or transcoder not enabled

Address:	6E510h-6E513h
Name:	WD0 Stride
ShortName:	WD_STRIDE_0
Reset:	soft

Address:	6ED10h-6ED13h
Name:	WD1 Stride
ShortName:	WD_STRIDE_1
Reset:	soft

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:6	**WD Stride** This field specifies the stride bits 15:6.This value is used to determine the line to line increment for the capture data writes. This field is programmed in units of 64 bytes. **Programming Notes** The stride has to be at least large enough to encompass all the pixels in a line, and rounded up to 64 byte alignment. Stride bytes >= CEILING[(Horizontal Active * WD Color Mode bytes per pixel) / 64] * 64 **Restriction** The stride is limited to a maximum of 32K bytes.
	5:0	**Reserved** Access: RO Format: MBZ

WD_SURF

WD_SURF	
Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferUpdatePoint:	Start of capture sync or transcoder not enabled

Address:	6E514h-6E517h
Name:	WD0 Surface Base Address
ShortName:	WD_SURF_0
Reset:	soft

Address:	6ED14h-6ED17h
Name:	WD1 Surface Base Address
ShortName:	WD_SURF_1
Reset:	soft

Writes to this register arm WD registers.

DWord	Bit	Description
0	31:12	**WD Surface Base Address** Format: GraphicsAddress[31:12] This address specifies the surface base address bits 31:12. It is mapped to physical pages through the global GTT. **Programming Notes** The mapped pages must be located outside graphics data stolen memory. **Restriction** It must be at least 4KB aligned.It must use linear memory.
	11:0	**Reserved** Access: RO Format: MBZ

WD_TAIL_CFG

WD_TAIL_CFG

Register Space:	MMIO: 0/2/0
Access:	Double Buffered
Size (in bits):	32
_Custom_Display_DoubleBufferArmedBy:	Write to WD_SURF or WD not enabled
_Custom_Display_DoubleBufferUpdatePoint:	Start of capture sync or transcoder not enabled; after armed

Address:	6E520h-6E523h
Name:	WD0 Tail Pointer Config
ShortName:	WD_TAIL_CFG_0
Reset:	soft

Address:	6ED20h-6ED23h
Name:	WD1 Tail Pointer Config
ShortName:	WD_TAIL_CFG_1
Reset:	soft

DWord	Bit	Description
0 If Delay < Period; first tail pointer is sent at the next multiple of Period, after the Delay. If Delay Period; first tail pointer is set at the Period. The tail pointer is also sent at the end of the frame. If Delay or Period are greater than the vertical size, only the tail pointer at the end of the frame is sent.	31:28	**Reserved** Access: RO Format: MBZ
	27:16	**Tail Initial Update Delay** This field specifies the minimum number of scan lines that WD capture must wait for at the beginning of each frame before any tail pointer updates will be sent to media. This must be programmed smaller than the vertical active size.
	15:12	**Reserved** Access: RO Format: MBZ
	11:4	**Tail Update Period** Default Value: 01h 16 lines This field specifies the number of scan lines that the WD capture will write back to memory before sending each tail pointer message to media. This field is programmed in multiples of 16 scan lines. **Restriction** A value of 0 is not valid.

WD_TAIL_CFG		
	3:0	**Reserved** Access: RO Format: MBZ

WD_TAIL_CFG2

WD_TAIL_CFG2		
Register Space:	MMIO: 0/2/0	
Access:	Double Buffered	
Size (in bits):	32	
_Custom_Display_DoubleBufferArmedBy:	Write to WD_SURF or WD not enabled	
_Custom_Display_DoubleBufferUpdatePoint:	Start of capture sync or transcoder not enabled; after armed	
Address:	6E52Ch-6E52Fh	
Name:	WD0 Tail Pointer Config2	
ShortName:	WD_TAIL_CFG2_0	
Reset:	soft	
Address:	6ED2Ch-6ED2Fh	
Name:	WD1 Tail Pointer Config2	
ShortName:	WD_TAIL_CFG2_1	
Reset:	soft	

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:4	**Tail Pointer Offset** If during selective read/write we are compositing to smaller than a full resolution surface, this tail pointer offset is used to synchronize with the VDEnc relative to the full resolution surface. This field is programmed in multiples of 16 scan lines.
	3:0	**Reserved** Access: RO Format: MBZ

WIDI Session 0

WIDI_SESSION_0 - WIDI Session 0		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00ED0h	
DWord	**Bit**	**Description**
0	31:0	**Static Data** Default Value: 0b Access: R/W Scratch registers provided for Display/VDEnc/GUC WIDI Session access

WIDI Session 1

WIDI_SESSION_1 - WIDI Session 1		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00ED4h	
DWord	**Bit**	**Description**
0	31:0	**Static Data** Default Value: 0b Access: R/W Scratch registers provided for Display/VDEnc/GUC WIDI Session access

WIDI Session 2

WIDI_SESSION_2 - WIDI Session 2		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00ED8h	
DWord	**Bit**	**Description**
0	31:0	**Static Data** Default Value: 0b Access: R/W Scratch registers provided for Display/VDEnc/GUC WIDI Session access

WIDI Session 3

WIDI_SESSION_3 - WIDI Session 3		
Register Space:	MMIO: 0/2/0	
Size (in bits):	32	
_Custom_GTIReset:	BUS	
Address:	00EDCh	
DWord	**Bit**	**Description**
0	31:0	**Static Data** Default Value: 0b Access: R/W Scratch registers provided for Display/VDEnc/GUC WIDI Session access

WiDi VDEnc Stall counter

WIDI_VDENC_STALL_CNTR - WiDi VDEnc Stall counter	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	BUS
Address:	1C2D94h-1C2D97h
Name:	WiDi mode VDENC stall counter value
ShortName:	WIDI_VDENC_STALL_CNTR_VDENC_REG0
Address:	1C6D94h-1C6D97h
Name:	WiDi mode VDENC stall counter value
ShortName:	WIDI_VDENC_STALL_CNTR_VDENC_REG1
Address:	1D2D94h-1D2D97h
Name:	WiDi mode VDENC stall counter value
ShortName:	WIDI_VDENC_STALL_CNTR_VDENC_REG2
Address:	1D6D94h-1D6D97h
Name:	WiDi mode VDENC stall counter value
ShortName:	WIDI_VDENC_STALL_CNTR_VDENC_REG3
Address:	1E2D94h-1E2D97h
Name:	WiDi mode VDENC stall counter value
ShortName:	WIDI_VDENC_STALL_CNTR_VDENC_REG4
Address:	1E6D94h-1E6D97h
Name:	WiDi mode VDENC stall counter value
ShortName:	WIDI_VDENC_STALL_CNTR_VDENC_REG5
Address:	1F2D94h-1F2D97h
Name:	WiDi mode VDENC stall counter value
ShortName:	WIDI_VDENC_STALL_CNTR_VDENC_REG6
Address:	1F6D94h-1F6D97h
Name:	WiDi mode VDENC stall counter value
ShortName:	WIDI_VDENC_STALL_CNTR_VDENC_REG7

This register has stall counter for the VDENC. In an ideal case, this value should be zero.

DWord	Bit	Description
0	31:0	**WiDi Stall clock count** Access: RO

Window Hardware Generated Clear Value

WMHWCLRVAL - Window Hardware Generated Clear Value		
Register Space:	MMIO: 0/2/0	
Access:	RO	
Size (in bits):	32	
Address:	05524h	
DWord	**Bit**	**Description**
0	31:0	**WM HW Generated Clear Value** Format: MBZ This register stores HW generated Z clear value.

WM_LINETIME

WM_LINETIME		
Register Space:	MMIO: 0/2/0	
Access:	R/W	
Size (in bits):	32	
Address:	45270h-45273h	
Name:	Pipe Watermark Line Time	
ShortName:	WM_LINETIME_A	
Reset:	soft	
Address:	45274h-45277h	
Name:	Pipe Watermark Line Time	
ShortName:	WM_LINETIME_B	
Reset:	soft	
Address:	45278h-4527Bh	
Name:	Pipe Watermark Line Time	
ShortName:	WM_LINETIME_C	
Reset:	soft	
Address:	4527Ch-4527Fh	
Name:	Pipe Watermark Line Time	
ShortName:	WM_LINETIME_D	
Reset:	soft	

DWord	Bit	Description
0	31:9	**Reserved** Access: RO Format: MBZ
	8:0	**Line Time** This field specifies the line time for the current screen resolution in units of 0.125us. **Programming Notes** Line time in microseconds = Pipe horizontal total number of pixels / pixel rate in MHz. Multiply by 8 to get units of 0.125us and round to nearest integer. Program the smallest line time when using multiple refresh rates. **Restriction** The line time value must be programmed before enabling any display low power watermark. Maximum supported line time is 63.875us (111111111b).

WM_MISC

WM_MISC	
Register Space:	MMIO: 0/2/0
Access:	R/W
Size (in bits):	32
Address:	45260h-45263h
Name:	Watermark Miscellaneous
ShortName:	WM_MISC
Reset:	soft

DWord	Bit	Description
0	31	**Reserved** Format: PBC
	30:28	**Reserved**
	27	**MIPI DBI Method** This field controls the behavior for the TTNF calculation for MIPI DBI. Value / Name / Description 0b / Calculated / TTNF calculated in active area and all 1s outside of active. TTVBI all 1s. 1b / Simple / TTNF all 0s in active area and all 1s outside of active. TTVBI all 1s.
	26:20	**Reserved** Format: PBC
	19:0	**Reserved** Access: RO Format: MBZ

Intel® Iris® Xe and UHD Graphics Open Source

Programmer's Reference Manual

For the 2020-2021 11th Generation Intel Xeon®, Core™, Celeron®, Pentium® Gold Processors based on the "Tiger Lake" Platform

Volume 2d: Command Reference: Structures

December 2021, Revision 1.0

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Table of Contents

3DSTATE_AMFS_BODY 1
3DSTATE_BINDING_TABLE_POINTERS_BODY 2
3DSTATE_BLEND_STATE_POINTERS_BODY 3
3DSTATE_CC_STATE_POINTERS_BODY 4
3DSTATE_CLEAR_PARAMS_BODY 5
3DSTATE_CLIP_BODY 6
3DSTATE_CONSTANT_ALL_BODY 12
3DSTATE_CONSTANT_ALL_DATA 13
3DSTATE_CONSTANT_TS_POINTER_BODY 14
3DSTATE_CONSTANT(Body) 15
3DSTATE_CPS_POINTERS_BODY 18
3DSTATE_DEPTH_BOUNDS_BODY 19
3DSTATE_DS_BODY 20
3DSTATE_GS_BODY 28
3DSTATE_HIER_DEPTH_BUFFER_BODY 38
3DSTATE_HS_BODY 41
3DSTATE_INDEX_BUFFER_BODY 48
3DSTATE_MULTISAMPLE_BODY 50
3DSTATE_PRIMITIVE_REPLICATION_BODY 52
3DSTATE_PS_BLEND_BODY 56
3DSTATE_PS_BODY 58
3DSTATE_PS_EXTRA_BODY 68
3DSTATE_PTBR_MARKER_BODY 74
3DSTATE_PTBR_TILE_SELECT_BODY 75
3DSTATE_RASTER_BODY 77
3DSTATE_SAMPLE_MASK_BODY 82
3DSTATE_SAMPLER_STATE_POINTERS_BODY 83
3DSTATE_SBE_BODY 84
3DSTATE_SBE_SWIZ_BODY 91
3DSTATE_SCISSOR_STATE_POINTERS_BODY 93
3DSTATE_SF_BODY 94
3DSTATE_SO_BUFFER_INDEX_BODY 99

3DSTATE_STENCIL_BUFFER_BODY 101
3DSTATE_STREAMOUT_BODY 109
3DSTATE_TE_BODY 113
3DSTATE_URB_ALLOC_DS_BODY 117
3DSTATE_URB_ALLOC_GS_BODY 119
3DSTATE_URB_ALLOC_HS_BODY 121
3DSTATE_URB_ALLOC_VS_BODY 123
3DSTATE_URB_DS_BODY 125
3DSTATE_URB_GS_BODY 126
3DSTATE_URB_HS_BODY 127
3DSTATE_URB_VS_BODY 128
3DSTATE_VF_BODY 130
3DSTATE_VF_COMPONENT_PACKING_BODY 131
3DSTATE_VF_INSTANCING_BODY 132
3DSTATE_VF_SGVS_2_BODY 134
3DSTATE_VF_SGVS_BODY 137
3DSTATE_VF_TOPOLOGY_BODY 139
3DSTATE_VIEWPORT_STATE_POINTERS_CC_BODY 140
3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP_BODY 141
3DSTATE_VS_BODY 142
3DSTATE_WM_BODY 150
3DSTATE_WM_CHROMAKEY_BODY 154
3DSTATE_WM_DEPTH_STENCIL_BODY 155
3DSTATE_WM_HZ_OP_BODY 158
A32 Buffer Base Address Message Header Control 164
A32 Scaled Header Present Message Descriptor Control Field 165
A32 Sideband Scale and Offset Enable Message Descriptor Control Field 166
A64 Data Size Message Descriptor Control Field 167
A64 Hword Block Message Header 168
A64 Hword Data Blocks Message Descriptor Control Field 169
A64 Oword Block Message Header 170
A64 Oword Data Blocks Message Descriptor Control Field 171
A64 Scaled Header Present Message Descriptor Control Field 172
AddrSubRegNum 173

Any Binding Table Index Message Descriptor Control Field 174
Atomic Float Binary Operation Message Descriptor Control Field 175
Atomic Float Operation Message Descriptor Control Field 176
Atomic Float Ternary Operation Message Descriptor Control Field 177
Atomic Half Float Binary Operation Message Descriptor Control Field 178
Atomic Integer Binary Operation Message Descriptor Control Field 179
Atomic Integer Operation Message Descriptor Control Field 180
Atomic Integer Ternary Operation Message Descriptor Control Field 181
Atomic Integer Unary Operation Message Descriptor Control Field 182
Audio Power State Format 183
AVC CABAC 184
AVC CAVLC 186
AVP_PAK_INSERT_OBJECT_INDIRECT_PAYLOAD 188
AVP_REF_LIST_ENTRY_OLD 189
Barrier Data Payload 190
BaseAddress4KByteAligned 192
Batch Buffer Stack Structure 193
Batch Stack Entry 194
BCS Hardware-Detected Error Bit Definitions 195
BINDING_TABLE_EDIT_ENTRY 196
Bit Definition for Interrupt Control Registers - Media 197
BLEND_STATE_ENTRY 199
BLEND_STATE 203
Blitter Interrupt Vector 205
Block Dimensions Message Header Control 206
Block Message Header 207
BR00 - BLT Opcode and Control 208
BR01 - Setup BLT Raster OP, Control, and Destination Offset 212
BR05 - Setup Expansion Background Color 215
BR06 - Setup Expansion Foreground Color 216
BR07 - Setup Blit Color Pattern Address Lower Order Address bits 217
BR09 - Destination Address Lower Order Address Bits 218
BR11 - BLT Source Pitch (Offset) 219
BR12 - Source Address Lower order Address bits 220

BR13 - BLT Raster OP, Control, and Destination Pitch 221
BR14 - Destination Width and Height 223
BR15 - Color Pattern Address Lower order Address bits 224
BR16 - Pattern Expansion Background and Solid Pattern Color 225
BR17 - Pattern Expansion Foreground Color 226
BR18 - Source Expansion Background and Destination Color 227
BR19 - Source Expansion Foreground Color 228
BR27 - Destination Higher Order Address 229
BR28 - Source Higher Order Address 230
BR29 - Color Pattern Higher Order Address 231
BR30 - Setup Blit Color Pattern Higher Order Address 232
Byte Masked Media Block Message Header 233
Byte Masked Media Block Message Header Control 234
CC_VIEWPORT 236
Channel Mask Message Descriptor Control Field 237
Channel Mode Message Descriptor Control Field 238
Clear Color 239
Clock Gating Disable Format 241
COLOR_CALC_STATE 242
COLOR_PROCESSING_STATE - ACE State 244
COLOR_PROCESSING_STATE - PROCAMP State 250
COLOR_PROCESSING_STATE - STD/STE State 251
COLOR_PROCESSING_STATE - TCC State 264
Color Calculator State Pointer Message Header Control 269
Color Code Message Header Control 270
COMPRESSION_PAIR_BIT 271
ComputeCS Hardware-Detected Error Bit Definitions 272
Compute Engine Interrupt Vector 273
Context Descriptor Format 275
Context Status 279
CPS_STATE 282
CSC COEFFICIENT FORMAT 286
Data Port 0 Message Types 287
Data Port 1 Message Types 288

Data Port Bindless Surface Extended Message Descriptor 290
Data Size Message Descriptor Control Field 291
Depth Clear Value Format 292
Deptrh Clear Value Format 293
DirectOperand 294
Display Engine Render Response Message Definition 295
DUALSUBSLICE_HASH_TABLE_8x8 301
DUALSUBSLICE_HASH_TABLE_16x8 302
Dword Data Payload Register 303
Dword SIMD8 Atomic Operation CMPWR Message Data Payload 304
Dword SIMD8 Data Payload 305
Dword SIMD16 Atomic Operation CMPWR Message Data Payload 306
Dword SIMD16 Data Payload 307
Encoder Statistics Format 308
EU_INSTRUCTION_BASIC_ONE_SRC 312
EU_INSTRUCTION_BASIC_THREE_SRC 316
EU_INSTRUCTION_BASIC_TWO_SRC 321
EU_INSTRUCTION_BRANCH_ONE_SRC 326
EU_INSTRUCTION_BRANCH_TWO_SRC 329
EU_INSTRUCTION_ILLEGAL 332
EU_INSTRUCTION_MATH 333
EU_INSTRUCTION_NOP 338
EU_INSTRUCTION_SEND 339
EU_INSTRUCTION_SYNC 344
Event Data Payload 347
ExMsgDesc 348
Extended Message Descriptor Render Target 349
Extended Message Descriptor - Sampling Engine 351
FFTID Message Header Control 352
Filter_Coefficient 353
Filter_Coefficients 354
FrameDeltaQp 355
FrameDeltaQpRange 356
Gamut_Expansion_Gamma_Correction 357

GGTT Entry **532**
GraphicsAddress63-0 ..... **533**
GraphicsAddress63-1 ..... **534**
GraphicsAddress63-2 ..... **535**
GraphicsAddress63-3 ..... **536**
GraphicsAddress63-4 ..... **537**
GraphicsAddress63-5 ..... **538**
GraphicsAddress63-6 ..... **539**
GraphicsAddress63-12 ..... **540**
GTC Interrupt Bit Definition **541**
GTPM Interrupt Vector **543**
GUC Interrupt Vector **544**
G-Unit Interrupt Vector **545**
Half Precision Dual Source SIMD8 Message Data Payload Register **546**
Half Precision OM Replicated SIMD16 Render Target Data Payload **548**
Half Precision OM S0A SIMD8 Render Target Data Payload **549**
Half Precision OM S0A SIMD16 Render Target Data Payload **550**
Half Precision OM SIMD8 Dual Source Render Target Data Payload **551**
Half Precision OM SIMD8 Render Target Data Payload **552**
Half Precision OM SIMD16 Render Target Data Payload **553**
Half Precision OS OM S0A SIMD8 Render Target Data Payload **554**
Half Precision OS OM SIMD8 Dual Source Render Target Data Payload **556**
Half Precision OS OM SIMD8 Render Target Data Payload **557**
Half Precision OS S0A SIMD8 Render Target Data Payload **558**
Half Precision OS SIMD8 Dual Source Render Target Data Payload **559**
Half Precision OS SIMD8 Render Target Data Payload **560**
Half Precision OS SZ OM S0A SIMD8 Render Target Data Payload **561**
Half Precision OS SZ OM SIMD8 Dual Source Render Target Data Payload **563**
Half Precision OS SZ OM SIMD8 Render Target Data Payload **565**
Half Precision OS SZ S0A SIMD8 Render Target Data Payload **567**
Half Precision OS SZ SIMD8 Dual Source Render Target Data Payload **569**
Half Precision OS SZ SIMD8 Render Target Data Payload **570**
Half Precision Replicated Pixel Render Target Data Payload Register **571**
Half Precision Replicated SIMD16 Render Target Data Payload **572**

Half Precision S0A SIMD8 Render Target Data Payload **573**
Half Precision S0A SIMD16 Render Target Data Payload **574**
Half Precision SIMD8 Dual Source Render Target Data Payload **575**
Half Precision SIMD8 Message Data Payload Register **576**
Half Precision SIMD8 Render Target Data Payload **578**
Half Precision SIMD16 Message Data Payload Register **579**
Half Precision SIMD16 Render Target Data Payload **581**
Half Precision SZ OM S0A SIMD8 Render Target Data Payload **582**
Half Precision SZ OM S0A SIMD16 Render Target Data Payload **584**
Half Precision SZ OM SIMD8 Dual Source Render Target Data Payload **586**
Half Precision SZ OM SIMD8 Render Target Data Payload **587**
Half Precision SZ OM SIMD16 Render Target Data Payload **588**
Half Precision SZ S0A SIMD8 Render Target Data Payload **590**
Half Precision SZ S0A SIMD16 Render Target Data Payload **591**
Half Precision SZ SIMD8 Dual Source Render Target Data Payload **593**
Half Precision SZ SIMD8 Render Target Data Payload **594**
Half Precision SZ SIMD16 Render Target Data Payload **595**
Hardware-Detected Error Bit Definitions **596**
Hardware Status Page Layout **597**
HCP_PAK_INSERT_OBJECT_INDIRECT_PAYLOAD **602**
HCP_REF_LIST_ENTRY **603**
HCP_TILE_POSITION_IN_CTB **605**
HCP_TILE_POSITION_IN_CTB_MSB **606**
HCP_WEIGHTOFFSET_CHROMA_ENTRY **608**
HCP_WEIGHTOFFSET_CHROMA_EXT_ENTRY **610**
HCP_WEIGHTOFFSET_LUMA_ENTRY **611**
Header **612**
Header Forbidden Message Descriptor Control Field **613**
Header Present Message Descriptor Control Field **614**
Header Required Message Descriptor Control Field **615**
HEVC_ARBITRATION_PRIORITY **616**
HEVC_VP9_RDOQ_LAMBDA_FIELDS **617**
HUC_VIRTUAL_ADDR_REGION **618**
HW Generated BINDING_TABLE_STATE **619**

Hword 1 Block Data Payload 620
Hword 2 Block Data Payload 621
Hword 4 Block Data Payload 622
Hword 8 Block Data Payload 623
Hword Channel Mode Message Header Control 625
Hword Register Blocks Message Descriptor Control Field 626
Ignored Message Header 627
IndirectOperand 628
Inline Data Description for MFD_AVC_BSD_Object 629
Inline Data Description in MPEG2-IT Mode 637
Inline Data Description - VP8 PAK OBJECT 640
INTERFACE_DESCRIPTOR_DATA 647
INTERRUPT 653
Invalidate After Read Message Descriptor Control Field 655
JPEG 656
LNCF_POWER_CONTEXT 657
LOD Message Address Payload Control 659
Lower Oword Block Data Payload 660
LPFC_POWER_CONTEXT 661
LRI Data Entry 662
MacroOperand 663
Manageability Engine Interrupt Vector 664
MEDIA_SURFACE_STATE 665
MEMORY_OBJECT_CONTROL_STATE 674
MemoryAddressAttributes 675
Merged Media Block Message Header Control 677
Message Descriptor - Render Target Write 679
Message Descriptor - Sampling Engine 681
MFD_MPEG2_BSD_OBJECT Inline Data Description 684
MFX_REFERENCE_PICTURE_BASE_ADDR 687
MPEG2 688
MSAA Sample Number Message Address Control 689
MsgDesc 690
Named Barrier State 692

No Event Data Payload....693
Normal Media Block Message Header....694
Normal Media Block Message Header Control....695
oMask Message Data Payload Register....697
OM Replicated SIMD16 Render Target Data Payload....699
OM S0A SIMD8 Render Target Data Payload....700
OM S0A SIMD16 Render Target Data Payload....701
OM SIMD8 Dual Source Render Target Data Payload....703
OM SIMD8 Render Target Data Payload....705
OM SIMD16 Render Target Data Payload....706
OS OM S0A SIMD8 Render Target Data Payload....708
OS OM SIMD8 Dual Source Render Target Data Payload....710
OS OM SIMD8 Render Target Data Payload....712
OS S0A SIMD8 Render Target Data Payload....713
OS SIMD8 Dual Source Render Target Data Payload....714
OS SIMD8 Render Target Data Payload....716
OS SZ OM S0A SIMD8 Render Target Data Payload....717
OS SZ OM SIMD8 Dual Source Render Target Data Payload....719
OS SZ OM SIMD8 Render Target Data Payload....721
OS SZ S0A SIMD8 Render Target Data Payload....722
OS SZ SIMD8 Dual Source Render Target Data Payload....723
OS SZ SIMD8 Render Target Data Payload....725
Oword 2 Block Data Payload....726
Oword 4 Block Data Payload....727
Oword 8 Block Data Payload....728
Oword A64 SIMD8 Atomic Operation CMPWR16B Message Data Payload....729
Oword Data Blocks Message Descriptor Control Field....731
Oword Data Payload Register....732
Oword Dual Data Blocks Message Descriptor Control Field....733
Performance Counter Report Format 101b....734
Per Thread Scratch Space Message Header Control....737
PIXEL_HASH_TABLE_1BIT_32ENTRY....738
PIXEL_HASH_TABLE_1BIT_64ENTRY....739
PIXEL_HASH_TABLE_1BIT_128ENTRY....740

PIXEL_HASH_TABLE_2BIT_64ENTRY 741
PIXEL_HASH_TABLE_2BIT_128ENTRY 748
Pixel Sample Mask Render Target Message Header Control 762
PPHWSP_LAYOUT 763
Predicate Barrier Message Data Payload 769
Qword Data Payload Register 770
Qword SIMD8 Atomic Operation CMPWR8B Message Data Payload 771
Qword SIMD8 Atomic Operation CMPWR Message Data Payload 772
Qword SIMD8 Atomic Operation Return Data Message Data Payload 773
Qword SIMD8 Data Payload 774
Qword SIMD16 Atomic Operation CMPWR8B Message Data Payload 775
Qword SIMD16 Atomic Operation Return Data Message Data Payload 777
Qword SIMD16 Data Payload 778
Read-Only Data Port Message Types 779
Read Surface Info 32-Bit Address Payload 780
Read Surface Info Address Payload 781
Read Surface Info Data Payload 782
RENDER_SURFACE_STATE 784
Render Data Port Message Types 817
Render Engine Interrupt Vector 818
Render Target Index Message Header Control 820
Render Target Message Header 821
Render Target Message Header Control 823
Render Target Message Header Poly 825
Replicated Pixel Render Target Data Payload Register 827
Replicated SIMD16 Render Target Data Payload 828
Reversed SIMD Mode 2 Message Descriptor Control Field 829
S0A SIMD8 Render Target Data Payload 830
S0A SIMD16 Render Target Data Payload 831
SAMPLER_BORDER_COLOR_STATE 833
SAMPLER_INDIRECT_STATE_BORDER_COLOR 836
SAMPLER_INDIRECT_STATE 838
SAMPLER_STATE_8x8_AVS 841
SAMPLER_STATE_8x8_AVS_COEFFICIENTS 846

SAMPLER_STATE 850
Sampler Message Header 864
SCALER_COEFFICIENT_FORMAT 868
SCISSOR_RECT 869
Scratch Hword Block Message Header 870
SF_CLIP_VIEWPORT 871
SF_OUTPUT_ATTRIBUTE_DETAIL 873
SFC_8x8_AVS_COEFFICIENTS 875
SFC_AVS_CHROMA_COEFF_TABLE_BODY 878
SFC_AVS_LUMA_COEFF_TABLE_BODY 880
SFC_AVS_STATE_BODY 884
SFC_FRAME_START_BODY 887
SFC_HDR_STATE 888
SFC_HISTOGRAM_SURFACE 890
SFC_IEF_STATE_BODY 892
SFC_LOCK_BODY 904
SFC_STATE_BODY 906
SIMD1 Untyped BUFFER Surface 64-Bit Address Payload 945
SIMD8 Dual Source Render Target Data Payload 946
SIMD8 LOD Message Address Payload Control 948
SIMD8 Render Target Data Payload 949
SIMD8 Typed Surface 32-Bit Address Payload 950
SIMD8 Untyped BUFFER Surface 32-Bit Address Payload 951
SIMD8 Untyped BUFFER Surface 64-Bit Address Payload 952
SIMD8 Untyped SCRATCH Surface 32-Bit Address Payload 953
SIMD8 URB Channel Mask Message Address Payload 954
SIMD8 URB Offset Message Address Payload 955
SIMD16 Render Target Data Payload 956
SIMD16 Typed Surface 16-Bit Address Payload 958
SIMD16 Untyped BUFFER Surface 16-Bit Address Payload 959
SIMD16 Untyped BUFFER Surface 32-Bit Address Payload 960
SIMD16 Untyped BUFFER Surface 64-Bit Address Payload 961
SIMD16 Untyped SCRATCH Surface 16-Bit Address Payload 962
SIMD16 Untyped SCRATCH Surface 32-Bit Address Payload 963

SIMD 32-Bit Address Payload Control 964
SIMD 64-Bit Address Payload Control 965
SIMD8 32-Bit Address Payload 966
SIMD8 64-Bit Address Payload 967
SIMD16 16-Bit Address Payload 968
SIMD16 32-Bit Address Payload 969
SIMD16 64-Bit Address Payload 970
SIMD Mode 2 Message Descriptor Control Field 971
SIMD Mode 3 Message Descriptor Control Field 972
SLICE_HASH_TABLE 973
SLICE_HASHING_TABLE_ENTRY 974
SLM Block Message Header 975
Slot Group 2 Message Descriptor Control Field 976
Slot Group 3 Message Descriptor Control Field 977
Slot Group Select Render Cache Message Descriptor Control Field 978
SO_DECL 979
SO_DECL_ENTRY 981
Split_coding_unit_flags 982
SplitBaseAddress4KByteAligned 983
SplitBaseAddress64ByteAligned 984
Stateless Binding Table Index Message Descriptor Control Field 985
Stateless Block Message Header 986
Stateless Surface Message Header 987
Stateless Surface Pixel Mask Message Header 988
Stencil Message Data Payload Register 989
Subset Atomic Integer Trinary Operation Message Descriptor Control Field 990
Subset Reversed SIMD Mode 2 Message Descriptor Control Field 991
Subset SIMD Mode 2 Message Descriptor Control Field 992
Subset SIMD Mode 3 Message Descriptor Control Field 993
Subspan Render Target Message Header Control 994
Surface Binding Table Index Message Descriptor Control Field 995
Surface or Stateless Binding Table Index Message Descriptor Control Field 996
Surface Pixel Mask Message Header 997
SW Generated BINDING_TABLE_STATE 998

SWSB 999
SZ OM S0A SIMD8 Render Target Data Payload 1001
SZ OM S0A SIMD16 Render Target Data Payload 1002
SZ OM SIMD8 Dual Source Render Target Data Payload 1004
SZ OM SIMD8 Render Target Data Payload 1006
SZ OM SIMD16 Render Target Data Payload 1007
SZ S0A SIMD8 Render Target Data Payload 1009
SZ S0A SIMD16 Render Target Data Payload 1010
SZ SIMD8 Dual Source Render Target Data Payload 1012
SZ SIMD8 Render Target Data Payload 1014
SZ SIMD16 Render Target Data Payload 1015
Thread EOT Message Descriptor 1017
TILE_RECT 1018
Timeout Data Payload 1019
U24_X8 1020
Untyped Write Channel Mask Message Descriptor Control Field 1021
Upper Oword Block Data Payload 1022
URB Channel Mask Payload Control 1023
URB Handle Message Header 1024
URB Handle Message Header Control 1025
VC1 1026
VCS Hardware-Detected Error Bit Definitions 1027
VD_CONTROL_STATE_BODY 1028
VDENC_64B_Aligned_Lower_Address 1029
VDENC_64B_Aligned_Upper_Address 1030
VDENC_Block_8x8_4 1031
VDENC_Colocated_MV_Picture 1032
VDENC_Delta_MV_XY 1033
VDENC_Down_Scaled_Reference_Picture 1035
VDENC_Original_Uncompressed_Picture 1036
VDENC_Reference_Picture 1037
VDENC_Row_Store_Scratch_Buffer_Picture 1038
VDENC_Sub_Mb_Pred_Mode 1039
VDENC_Surface_Control_Bits 1040

VDENC_Surface_State_Fields 1042
VEBOX_ACE_LACE_STATE 1047
VEBOX_ALPHA_AOI_STATE 1060
VEBOX_CAPTURE_PIPE_STATE 1062
VEBOX_CCM_STATE 1066
VEBOX_Ch_Dir_Filter_Coefficient 1070
VEBOX_CSC_STATE 1071
VEBOX_DNDI_STATE 1074
VEBOX_Filter_Coefficient 1096
VEBOX_FORWARD_GAMMA_CORRECTION_STATE 1097
VEBOX_FRONT_END_CSC_STATE 1105
VEBOX_GAMUT_CONTROL_STATE 1108
VEBOX_PROCAMP_STATE 1115
VEBOX_RGB_TO_GAMMA_CORRECTION 1116
VEBOX_STD_STE_STATE 1117
VEBOX_TCC_STATE 1133
VEBOX_VERTEX_TABLE_ENTRY 1139
VECS Hardware-Detected Error Bit Definitions 1140
VERTEX_BUFFER_STATE 1141
VERTEX_ELEMENT_STATE 1144
Vertical Line Stride Override Message Descriptor Control Field 1148
VideoDecoder Interrupt Vector 1149
VideoEnhancement Interrupt Vector 1151
VP8 Encoder StreamOut Format 1152
WDBoxOAInterrupt Vector 1154
WD Interrupt Bit Definition 1155
Word Data Payload Register 1156
Word SIMD8 Atomic Operation CMPWR Message Data Payload 1158
Word SIMD8 Data Payload 1159
Word SIMD16 Atomic Operation CMPWR Message Data Payload 1160
Word SIMD16 Data Payload 1161

3DSTATE_AMFS_BODY

3DSTATE_AMFS_BODY

Source: RenderCS

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:6	**TS State Array Pointer** Format: GraphicsAddress[31:6] Specifies the 64-byte aligned offset of the TS_STATE_ARRAY, with entries per texel shader. This offset is relative to the **Dynamic State Base Address**.
	5	**Reserved**
	4:3	**AMFS mode** Format: U2 **Value** / **Name** / **Description** 0h / normal mode **[Default]** / AMFS shades unshaded texel blocks only, marks them as shaded, no special handling of out of memory condition other than setting indicator bit in CACHE_MODE1 bit[6] 1h / touch mode / forces out of memory operation, AMFS marks unshaded texel blocks as touched, texel shader dispatch is disabled 2h / backfill mode / AMFS shades only touched texel blocks, touched blocks get marked as shaded, originally unshaded and shaded are left unchanged 3h / fallback mode / AMFS shades unshaded texel blocks only, marks them as shaded, special handling of out of memory condition. When AMFS runs out of scratch space, texels are not shaded. Instead, they are marked as "touched". AMFS also sets indicator bit in CACHE_MODE1 bit[6] **Programming Notes** For backwards compatibility mode 0h acts the same as A0 AMFS. Mode 1h can be also used for functional validation of out of memory fallback condition
	2:0	**Num Valid Texel Shaders** Format: U3-1 Specifies number of valid texel shader entries in the TS_STATE_ARRAY using n-1 encoding (1..8 valid entries).

3DSTATE_BINDING_TABLE_POINTERS_BODY

3DSTATE_BINDING_TABLE_POINTERS_BODY

Source: RenderCS
Size (in bits): 32
Default Value: 0x00000000

DWord	Bit	Description
0	31:21	**Reserved** Access: RO Format: MBZ
	20:16	**Reserved** Access: RO Format: MBZ
	15:5	**Pointer to Binding Table** Format: SurfaceStateOffset[15:5]SW Generated BINDING_TABLE_STATE*256 When Binding Table Pool is disabled and HW Binding Table Alignment is not set to 256B alignment. Format: SurfaceStateOffset[16:6]SW Generated BINDING_TABLE_STATE*256 When Binding Table Pool is enabled and HW Binding Table Alignment is not set to 256B alignment. Format: SurfaceStateOffset[18:8]SW Generated BINDING_TABLE_STATE*256 When HW Binding Table Alignment is set to 256B alignment. Specifies an aligned address offset of the function's BINDING_TABLE_STATE. The offset's base and alignment differ depending on whether HW Binding Table is enabled and the setting of HW Binding Table Alignment field: If HW Binding Table Pool is **disabled** and the HW Binding Table Alignment is not set to 256B, the offset is relative to **Surface State Base Address** and the alignment is **32B**. If HW Binding Table Pool is **enabled** and the HW Binding Table Alignment is not set to 256B, the offset is relative to the **Binding Table Pool Base Address** and the alignment is **64B**. If HW Binding Table Pool is **disabled** and the HW Binding Table Alignment is set to 256B, the offset is relative to the **Surface State Base Address** and the alignment is **256B**. If HW Binding Table Pool is **enabled** and the HW Binding Table Alignment is set to 256B, the offset is relative to the **Binding Table Pool Base Address** and the alignment is **256B**.
	4:0	**Reserved** Access: RO Format: MBZ

3DSTATE_BLEND_STATE_POINTERS_BODY

3DSTATE_BLEND_STATE_POINTERS_BODY		
Source:	RenderCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:6	**Blend State Pointer** Format: DynamicStateOffset[31:6]BLEND_STATE*8 Specifies the 64-byte aligned offset of the BLEND_STATE. This offset is relative to the **Dynamic State Base Address**.
	5:1	**Reserved** Access: RO Format: MBZ
	0	**Blend State Pointer Valid** Format: Enable This bit, if set, indicates that the BLEND_STATE pointer has changed and new state needs to be fetched.

3DSTATE_CC_STATE_POINTERS_BODY

3DSTATE_CC_STATE_POINTERS_BODY		
Source:	RenderCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:6	**Color Calc State Pointer** Format: DynamicStateOffset[31:6]COLOR_CALC_STATE Specifies the 64-byte aligned offset of the COLOR_CALC_STATE. This offset is relative to the **Dynamic State Base Address**.
	5:1	**Reserved** Access: RO Format: MBZ
	0	**Color Calc State Pointer Valid** Format: Enable If set, the hardware will fetch the CC state. This bit is context saved and restored so the CC state is considered undefined once this bit is cleared due to the possibility of the CC state changing between context switches.

3DSTATE_CLEAR_PARAMS_BODY

RenderCS - 3DSTATE_CLEAR_PARAMS_BODY

Size (in bits): 64

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0	31:0	**Depth Clear Value** Format: IEEE_FLOAT32 This field defines the clear value that will be applied to the depth buffer if the Depth Buffer Clear field is enabled. It is valid only if Depth Buffer Clear Value Valid is set. **Programming Notes** The clear value must be between the min and max depth values (inclusive) defined in the CC_VIEWPORT. If the depth buffer format is D32_FLOAT, then values must be limited to the range of +0.0f and 1.0f inclusive; values outside this range are reserved.
1	31:1	**Reserved** Access: RO Format: MBZ
	0	**Depth Clear Value Valid** Format: Boolean This field enables the **Depth Clear Value**. If clear, the depth clear value is obtained from interpolated depth of an arbitrary pixel of the primitive rendered with **Depth Buffer Clear** set in WM_STATE or 3DSTATE_WM. If set, the depth clear value is obtained from the **Depth Clear Value** field of this command.

3DSTATE_CLIP_BODY

3DSTATE_CLIP_BODY

Source: RenderCS

Size (in bits): 96

Default Value: 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:21	**Reserved** Access: RO Format: MBZ
	20	**Force User Clip Distance Cull Test Enable Bitmask** Format: Enable This field provides a work around override for the computation of SOL_INT::Render_Enable Value / Name / Description 0h / Normal / Clip_INT::User Clip Distance Cull Test Enable Bitmask normally 1h / Force / Forces Clip_INT::User Clip Distance Cull Test Enable Bitmask to use the value in 3DSTATE_CLIP:: User Clip Distance Cull Test Enable Bitmask
	19	**Vertex Sub Pixel Precision Select** Format: U1 Selects the number of fractional bits maintained in the vertex data Value / Name / Description 0h / 8 Bit / 8 sub pixel precision bits maintained 1h / 4 Bit / 4 sub pixel precision bits maintained
	18	**Early Cull Enable** Format: Enable This field is used to enable/disable the EarlyCull function. When this bit is set triangles are checked if they are backface culled before proceeding through must clip function. **Programming Notes** Setting this bit must not impact functionality, this state only controls the performance of the must clip function. Vertex Sub Pixel Precision Select precision must be set to "8 bit" in order avoid precision issues.
	17	**Force User Clip Distance Clip Test Enable Bitmask** Format: Enable This field provides a work around override for the computation of SOL_INT::Render_Enable. Value / Name / Description 0b / Normal / Clip_INT:: User Clip Distance Clip Test Enable Bitmask normally 1b / Force / Forces Clip_INT:: User Clip Distance Clip Test Enable Bitmask to use the value in 3DSTATE_CLIP::User Clip Distance Clip Test Enable Bitmask

3DSTATE_CLIP_BODY

	16	**Force Clip Mode** Format: Enable This field provides a work around override for the computation of SOL_INT::Render_Enable. **Value** / **Name** / **Description** 0b / Normal / Clip_INT::Clip Mode is computed normally. 1b / Force / Forces Clip_INT::Clip Mode to use the value in 3DSTATE_CLIP::User Clip Mode.
	15:12	**Reserved** Access: RO Format: MBZ
	11:10	**Clipper Statistics Enable** This bit controls whether Clip-unit-specific statistics register(s) can be incremented. **Value** / **Name** / **Description** 00h / Disable / CL_INVOCATIONS_COUNT cannot increment 01h / Increment by one / CL_INVOCATIONS_COUNT can increment 03h / Reserved /
	9	**Reserved** Access: RO Format: MBZ
	8	**Reserved** Access: RO Format: MBZ
	7:0	**User Clip Distance Cull Test Enable Bitmask** Format: Enable[8] This 8 bit mask field selects which of the 8 user clip distances against which trivial reject / trivial accept determination needs to be made (does not cause a must clip).DX10 allows simultaneous use of ClipDistance and Cull Distance test of up to 8 distances.
1	31	**Clip Enable** Format: Enable Specifies whether the Clip function is enabled or disabled (pass-through).
	30	**API Mode** Controls the definition of the NEAR clipping plane **Value** / **Name** / **Description** 0h / OGL / NEAR VP boundary == 0.0 (NDC)
	29	**Reserved** Access: RO Format: MBZ

3DSTATE_CLIP_BODY

28 Viewport XY Clip Test Enable

Format:	Enable

This field is used to control whether the Viewport X, Y extents [-1,1] are considered in VertexClipTest.
If both the Guardband and Viewport XY ClipTest are DISABLED, all vertices are considered "visible" with respect to the XY directions.

27 Reserved

Access:	RO
Format:	MBZ

26 Guardband Clip Test Enable

Format:	Enable

This field is used to control whether the Guardband X, Y extents are considered in VertexClipTest for non-point objects. If the Guardband ClipTest is DISABLED but the Viewport XY ClipTest is ENABLED, ClipDetermination operates as if the Guardband were coincident with the Viewport. If both the Guardband and Viewport XY ClipTest are DISABLED, all vertices are considered "visible" with respect to the XY directions.

25:24 Reserved

Access:	RO
Format:	MBZ

23:16 User Clip Distance Clip Test Enable Bitmask

Format:	Enable[8]

This 8 bit mask field selects which of the 8 user clip distances against which trivial reject / trivial accept / must clip determination needs to be made.DX10 allows simultaneous use of ClipDistance and Cull Distance test of up to 8 distances.

15:13 Clip Mode

This field specifies a general mode of the CLIP unit, when the CLIP unit is ENABLED.

Value	Name	Description
0h	NORMAL	TrivialAccept objects are passed down the pipeline, MustClip objects Clipped in the Fixed Function Clipper HW, TrivialReject and BAD objects are discarded
1h	Reserved	
2h	Reserved	
3h	REJECT_ALL	All objects are discarded
4h	ACCEPT_ALL	All objects (except BAD objects) are trivially accepted. This effectively disables the clip-test/clip-determination function. Note that the CLIP unit will still filter out adjacency information, which may be required since the SF unit does not accept primitives with adjacency.
5h-7h	Reserved	

3DSTATE_CLIP_BODY

	12:10	**Reserved** Access: RO Format: MBZ
	9	**Perspective Divide Disable** Format: Disable This field disables the Perspective Divide function performed on homogeneous position read from the URB. This feature can be used by software to submit pre-transformed "screen-space" geometry for rasterization. This likely requires the W component of positions to contain "rhw" (aka 1/w) in order to support perspective-correct interpolation of vertex attributes. Likewise, the X, Y, Z components will likely be required to be X/W, Y/W, Z/W. Note that the device does not support clipping when perspective divide is disabled. Software must specify CLIPMODE_ACCEPT_ALL whenever it disables perspective divide. This implies that software must ensure that object positions are completely contained within the "guardband" screen-space limits imposed by the SF unit (e.g., by clipping in CPU SW before submitting the objects).
	8	**Non-Perspective Barycentric Enable** Format: Enable This field enables computation of non-perspective barycentric parameters in the clipper, which are sent to SF unit in the must clip case. This field must be enabled if any non-perspective interpolation modes are used in pixel shader. **Programming Notes** This field must be set whenever Enable bits 3 or 4 or 5 of 3DSTATE_WM:Barycentric Interpolation Mode is set. This indicates that one of the Non-perspective barycentric interpolation modes are used. This field must be set if the 3DSTATE_PS_EXTRA:Pixel Shader Requires Non-Perspective Bary Plane Coefficients is set.
	7:6	**Reserved** Access: RO Format: MBZ
	5:4	**Triangle Strip/List Provoking Vertex Select** Format: U2 enumerated type This field selects which vertex of a triangle (in a triangle strip or list primitive) is considered the "provoking vertex". **Value** / **Name** 0h / 0 1h / 1 2h / 2

3DSTATE_CLIP_BODY

	3h	Reserved

3:2 Line Strip/List Provoking Vertex Select

Format:	U2

enumerated type

This field selects which vertex of a line (in a line strip or list primitive) is considered the "provoking vertex".

Value	Name
0h	0
1h	1
2h	Reserved
3h	Reserved

1:0 Triangle Fan Provoking Vertex Select

Format:	U2

enumerated type

This field selects which vertex of a triangle (in a triangle fan primitive) is considered the "provoking vertex".

Value	Name
0h	0
1h	1
2h	2
3h	Reserved

2 31:28 Reserved

Access:	RO
Format:	MBZ

27:17 Minimum Point Width

Format:	U8.3

This value is used to clamp read-back PointWidth values.

16:6 Maximum Point Width

Format:	U8.3

This value is used to clamp read-back PointWidth values.

5 Force Zero RTA Index Enable

Format:	Enable

If set, the Clip unit will ignore the read-back RTAIndex and operate as if the value 0 was read-

		back. If clear, the read-back value is used.
	4	**Reserved** Access: RO Format: MBZ
	3:0	**Maximum VP Index** Format: U4-1 This field specifies the maximum valid VPIndex value, corresponding to the number of active viewports. If the source of the VPIndex exceeds this maximum value, a VPIndex value of 0 is passed down the pipeline. Note that this clamping does not affect a VPIndex value stored in the URB.

3DSTATE_CONSTANT_ALL_BODY

3DSTATE_CONSTANT_ALL_BODY		
Source:	RenderCS	
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0	255:0	**Constant Body** Format: 3DSTATE_CONSTANT_ALL_DATA[4]

3DSTATE_CONSTANT_ALL_DATA

3DSTATE_CONSTANT_ALL_DATA

Source: RenderCS

Size (in bits): 64

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0..1	63:5	**Pointer To Constant Buffer** Format: VIRTUAL_ADDR[63:5] The value of this field is the virtual address of the location of the push constant buffer. **Programming Notes** Constant buffers must be allocated in linear (not tiled) graphics memory.
	4:0	**Constant Buffer Read Length** Format: U5 This field specifies the length of the constant data to be loaded from memory in 256-bit units. **Programming Notes** • The sum of the read length fields for all pointers must be less than or equal to the size of 64 • Zero means there no data to fetch for this buffer pointer.

3DSTATE_CONSTANT_TS_POINTER_BODY

3DSTATE_CONSTANT_TS_POINTER_BODY		
Source:	RenderCS	
Size (in bits):	64	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0..1	63:0	**Constant Body** Format: 3DSTATE_CONSTANT_ALL_DATA Specifies the 64-byte aligned graphics address and length of constant data to be pushed as Texel Shader payload. The push constant payload and its length is common to all Texel Shader slots.

3DSTATE_CONSTANT(Body)

3DSTATE_CONSTANT(Body)

Source: RenderCS

Size (in bits): 320

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:16	**Constant Buffer 1 Read Length** Format: U16 This field specifies the length of the constant data to be loaded from memory in 256-bit units. **Programming Notes** • The sum of all four read length fields must be less than or equal to the size of 64 • Setting the value of the register to zero will disable buffer 1. • If disabled, the **Pointer to Constant Buffer 1** must be programmed to zero.
	15:0	**Constant Buffer 0 Read Length** Format: U16 This field specifies the length of the constant data to be loaded from memory in 256-bit units. **Programming Notes** • The sum of all four read length fields must be less than or equal to the size of 64 • Setting the value of the register to zero will disable buffer 0. • If disabled, the **Pointer to Constant Buffer 0** must be programmed to zero.
1	31:16	**Constant Buffer 3 Read Length** Format: U16 This field specifies the length of the constant data to be loaded from memory in 256-bit units. **Programming Notes** • The sum of all four read length fields must be less than or equal to the size of 64 • Setting the value of the register to zero will disable buffer 3. • If disabled, the **Pointer to Constant Buffer 3** must be programmed to zero.
	15:0	**Constant Buffer 2 Read Length** Format: U16 This field specifies the length of the constant data to be loaded from memory in 256-bit units. **Programming Notes** • The sum of all four read length fields must be less than or equal to the size of 64 • Setting the value of the register to zero will disable buffer 2.

3DSTATE_CONSTANT(Body)

		• If disabled, the **Pointer to Constant Buffer 2** must be programmed to zero.
2..3	63:5	**Pointer To Constant Buffer 0** Format: VIRTUAL_ADDR[63:5] **Description** The value of this field is the virtual address of the location of the push constant buffer 0. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47]. **Programming Notes** Constant buffers must be allocated in linear (not tiled) graphics memory.
	4:0	**Reserved** Access: RO Format: MBZ
4..5	63:5	**Pointer To Constant Buffer 1** Format: VIRTUAL_ADDR[63:5] This field points to the location of Constant Buffer 1. If gather constants are enabled This field is an offset of constant Buffer1 from the Gather Pool BASE ADDRESS. If gather constants is disabled, the value of this field is the virtual address of the location of the push constant buffer. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47]. **Programming Notes** Constant buffers must be allocated in linear (not tiled) graphics memory.
	4:0	**Reserved** Access: RO Format: MBZ
6..7	63:5	**Pointer To Constant Buffer 2** Format: VIRTUAL_ADDR[63:5] The value of this field is the virtual address of the location of the push constant buffer 2. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47]. **Programming Notes** Constant buffers must be allocated in linear (not tiled) graphics memory.
	4:0	**Reserved** Access: RO

3DSTATE_CONSTANT(Body)			
		Format:	MBZ
8..9	63:5	**Pointer To Constant Buffer 3** Format: VIRTUAL_ADDR[63:5] The value of this field is the virtual address of the location of the push constant buffer 3. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47]. **Programming Notes** Constant buffers must be allocated in linear (not tiled) graphics memory.	
	4:0	**Reserved** Access: RO Format: MBZ	

3DSTATE_CPS_POINTERS_BODY

3DSTATE_CPS_POINTERS_BODY		
Source:	RenderCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:5	**Coarse Pixel Shading State Array Pointer** Format: DynamicStateOffset[31:5] Specifies the 32-byte aligned address offset of the array of CPS_STATE states. Each CPS_STATE in the array corresponds to a Viewport index in the range [0..15]. SW must program all the CPS_STATES in the array corresponding to valid Viewport indices. This offset is relative to the Dynamic State Base Address. When 3DSTATE_PS:Pixel Shader Is Per Coarse Pixel is not set, HW does not fetch or depend on any CPS pointers to be valid. The hardware might not order pixels across viewports if multiple CPS_STATE entries map any(x,y) to different CPsizes.
	4:0	**Reserved** Access: RO Format: MBZ

3DSTATE_DEPTH_BOUNDS_BODY

3DSTATE_DEPTH_BOUNDS_BODY

Source: RenderCS

Size (in bits): 96

Default Value: 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:1	**Reserved** Access: RO Format: MBZ
	0	**Depth Bounds Test Enable** Enables the depth bounds test Value / Name / Description: 0 / Disabled / Depth Bounds test is disabled. 1 / Enabled / Depth Bounds test is enabled. If (Z Min Value) <= (Destination Z Value) <= (Z Max Value) the depth bounds test passes. Otherwise the depth bounds test fails and the sample is discarded. Value / Name: 0 / Disabled 1 / Enabled
1	31:0	**Depth Bounds Test Min Value** Format: IEEE_FLOAT This field specifies the minimum Z value to be used in the depth bounds test. This value should be in 32-bit Float. HW will clamp to min value of +0 if set to below +0.
2	31:0	**Depth Bounds Test Max Value** Format: IEEE_FLOAT This field specifies the maximum Z value to be used in the depth bounds test. This value should be in 32-bit Float. HW will clamp to max value of +1 if set to greater than +1.

3DSTATE_DS_BODY

3DSTATE_DS_BODY

Source: RenderCS

Size (in bits): 320

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0..1	63:6	**Kernel Start Pointer** Format: InstructionBaseOffset[63:6] This field specifies the starting location of the kernel program run by threads spawned by this FF unit. It is specified as a 64-byte-granular offset from the Instruction Base Address. This field is ignored if DS Function Enable is DISABLED.
	5:0	**Reserved** Access: RO Format: MBZ
2	31	**Reserved** Access: RO Format: MBZ
	30	**Vector Mask Enable** Format: Enable Upon subsequent DS thread dispatches, this bit is loaded into the EU's Vector Mask Enable (VME, cr0.0[3]) thread state. Refer to EU documentation for the definition and use of VME state. **Value / Name / Description** 0h / Dmask / The EU will use the Dispatch Mask (supplied by the DS stage) for instruction execution. 1h / Vmask / The EU will use the Vector Mask (derived from the Dispatch Mask) for instruction execution. **Programming Notes** Under normal conditions SW shall specify DMask, as the DS stage will provide a Dispatch Mask appropriate to SIMD4x2 or SIMD8 thread execution (as a function of dispatch mode). E.g., for SIMD4x2 thread execution, the DS stage will generate a Dispatch Mask that is equal to what the EU would use as the Vector Mask. For SIMD8 execution there is no known usage model for use of Vector Mask (as there is for PS shaders).
	29:27	**Sampler Count** Format: U3 Specifies how many samplers (in multiples of 4) the kernel uses. Used only for prefetching the associated sampler state entries. This field is ignored if DS Function Enable is DISABLED.

3DSTATE_DS_BODY

Value	Name	Description
0h	No Samplers	No samplers used
1h	1-4 Samplers	between 1 and 4 samplers used
2h	5-8 Samplers	between 5 and 8 samplers used
3h	9-12 Samplers	between 9 and 12 samplers used
4h	13-16 Samplers	between 13 and 16 samplers used

26 **Reserved**

Access:	RO
Format:	MBZ

25:18 **Binding Table Entry Count**

Format:	U8

When HW Generated Binding Table is disabled: Specifies how many binding table entries the kernel uses. Used only for prefetching of the binding table entries and associated surface state. **Note:** For kernels using a large number of binding table entries, it may be wise to set this field to zero to avoid prefetching too many entries and thrashing the state cache. This field is ignored if DS Function Enable is DISABLED.

When HW Generated Binding Table bit is enabled: This field indicates which cache lines (512bit units - 32 Binding Table Entry section) should be fetched. Each bit in this field corresponds to a cache line. Only the 1st 4 non-zero Binding Table entries of each 32 Binding Table entry section prefetched will have its surface state prefetched.

Value	Name
[0,255]	

Programming Notes
When HW binding table bit is set, it is assumed that the Binding Table Entry Count field will be generated at JIT time.

17 **Thread Dispatch Priority**

Format:	U1

Specifies the priority of the thread for dispatch: This field is ignored if DS Function Enable is DISABLED.

Value	Name	Description
0h	Normal	Normal Priority
1h	High	High Priority

16 **Floating Point Mode**

Format:	U1

Specifies the initial floating point mode used by the dispatched thread. This field is ignored if DS Function Enable is DISABLED.

3DSTATE_DS_BODY

		Value / **Name** / **Description** 0h / IEEE-754 / Use IEEE-754 Rules 1h / Alternate / Use alternate rules
	15	**Reserved** Access: RO Format: MBZ
	14	**Accesses UAV** Format: Enable This bit gets loaded into EU CR0.1[12] (note the bit # difference). See Exceptions and ISA Execution Environment. **Programming Notes** This field must not be set when DS Function Enable is disabled.
	13	**Illegal Opcode Exception Enable** Format: Enable This bit gets loaded into EU CR0.1[12] (note the bit # difference). See Exceptions and ISA Execution Environment. This field is ignored if DS Function Enable is DISABLED.
	12:11	**Reserved** Access: RO Format: MBZ
	10:8	**Reserved** Format: MBZ
	7	**Software Exception Enable** Format: Enable This bit gets loaded into EU CR0.1[13] (note the bit # difference). See Exceptions and ISA Execution Environment. This field is ignored if DS Function Enable is DISABLED.
	6:0	**Reserved** Access: RO Format: MBZ
3..4	63:32	**Reserved** Access: RO Format: MBZ
	31:10	**Scratch Space Base Pointer** Format: GeneralStateOffset[31:10] Specifies the starting location of the scratch space area allocated to this FF unit as a 1K-byte aligned offset from the General State Base Address. If required, each thread spawned by this FF unit will be allocated some portion of this space, as specified by Per-Thread Scratch Space. The

computed offset of the thread-specific portion will be passed in the thread payload as Scratch Space Offset. The thread is expected to utilize "stateless" DataPort read/write requests to access scratch space, where the DataPort will cause the General State Base Address to be added to the offset passed in the request header. This field is ignored if DS Function Enable is DISABLED.

9:4 **Reserved**

Access:	RO
Format:	MBZ

3:0 **Per-Thread Scratch Space**

Format:	U4

Specifies the amount of scratch space to be allocated to each thread spawned by this FF unit. The driver must allocate enough contiguous scratch space, starting at the Scratch Space Base Pointer, to ensure that the Maximum Number of Threads can each get Per-Thread Scratch Space size without exceeding the driver-allocated scratch space. This field is ignored if DS Function Enable is DISABLED.

Value	Name
[0,11]	indicating [1K Bytes, 2M Bytes]

Programming Notes
This amount is available to the kernel for information only. It will be passed verbatim (if not altered by the kernel) to the Data Port in any scratch space access messages, but the Data Port will ignore it.

5 31:25 **Reserved**

Access:	RO
Format:	MBZ

24:20 **Dispatch GRF Start Register For URB Data**

Format:	U5

Description
Specifies the starting GRF register number for the URB portion (Constant + Vertices) of the thread payload. This field is ignored if DS Function Enable is DISABLED. When SIMD8_SINGLE_OR_DUAL_PATCH dispatch mode is selected, HW shall increment the GRF start register by 1 when a dual patch simd8 thread is dispatched AND 3DSTATE_DS::PrimitiveIDNotRequired is not set.

Value	Name	Description
[0,31]		indicating GRF [R0, R31]

19:18 **Reserved**

Access:	RO
Format:	MBZ

3DSTATE_DS_BODY

17:11 Patch URB Entry Read Length

Format:	U7

Specifies how much data (in 256-bit units) is to be read from the Patch URB entry and passed in the DS thread payload. This field is ignored if DS Function Enable is DISABLED.

Value	Name
[0,64]	

10 Reserved

Access:	RO
Format:	MBZ

9:4 Patch URB Entry Read Offset

Format:	U6

Specifies the offset (in 256-bit units) at which Patch URB data is to be read from the URB before being included in the thread payload. This field is ignored if DS Function Enable is DISABLED.

Value	Name
[0,63]	

3:0 Reserved

Access:	RO
Format:	MBZ

6 — 31 Reserved

Access:	RO
Format:	MBZ

30:21 Maximum Number of Threads

Format:	U10-1

Specifies the maximum number of simultaneous DS threads allowed to be active. Used to avoid using up the scratch space. Programming the value of the max threads over the number of threads based off number of threads supported in the execution units may improve performance since the architecture allows threads to be buffered between the check for max threads and the actual dispatch into the EU. Programming the max values to a number less than the number of threads supported in the execution units may reduce performance. This field is ignored if DS Function Enable is DISABLED.

Value	Name	Description
[0,545]		indicating thread count of [1,546]

20:11 Reserved

Access:	RO
Format:	MBZ

10 Statistics Enable

Format:	Enable

If ENABLED, this FF unit will engage in statistics gathering. Refer to the Statistics Gathering section.

If DISABLED, statistics information associated with this FF stage will be left unchanged.
This field is ignored if DS Function Enable is DISABLED.

9 **PrimitiveID Not Required**

Format:	Boolean

Description
Software shall set this bit whenever the active DS kernel(s) do not require PrimitiveID as input. When this bit is set, (a) the R1 PrimitiveID phase will not be included in the thread payload (DUAL_PATCH) and (b) the PrimitiveID field in the R0 payload (SINGLE_PATCH) will become UNDEFINED.

8:5 **Reserved**

Access:	RO
Format:	MBZ

4:3 **Dispatch Mode**

Format:	U2

This field specifies how the DS stage generates DS thread requests, and correspondingly impacts the DS thread payload. The setting of this field must agree with how the DS kernel was compiled.This field is ignored if DS Function Enable is DISABLED.

Value	Name	Description	Programming Notes
1h	SIMD8_SINGLE_PATCH	DS threads are passed one patch, up to 8 domain point inputs, and up to 8 output vertex handles. All the domain points are associated with the single input patch. The DS kernel (at KSP) is expected to run in SIMD8 execution mode. The DUAL_PATCH KSP is ignored.	
2h	SIMD8_SINGLE_OR_DUAL_PATCH	This mode enables use of both the KSP and the DUAL_PATCH KSP. The KSP kernel operates just like in SIMD8_SINGLE_PATCH mode. See DUAL_PATCH Thread Execution for a discussion of how the DUAL_PATCH KSP is used.	At least 2 HS URB handles must be allocated in order to enable this mode.
3h	Reserved		

Programming Notes
SIMD4X2 mode is no longer allowed.

3DSTATE_DS_BODY

	2	**Compute W Coordinate Enable** Format: Enable If ENABLED, the DS unit will (for each domain point) compute W = 1 - (U + V) and pass the result as a floating point value in the DS thread payload. If DISABLED, 0.0 will be passed. This field must only be ENABLED for the tessellation of TRI domains, where UVW coordinates are required. This field must be DISABLED for other domains (as they only require UV coordinates) otherwise the computed W coordinate is UNDEFINED. This field is ignored if DS Function Enable is DISABLED.
	1	**Cache Disable** Format: Disable This bit controls the operation of the DS Cache. This field is ignored if DS Function Enable is DISABLED. If the DS Cache is DISABLED and the DS Function is ENABLED, the DS Cache is not used and all incoming domain points will be passed to DS threads. If the DS Cache is ENABLED and the DS Function is ENABLED, incoming domain points that do not hit in the DS Cache will be passed to DS threads. The DS Cache is invalidated whenever the DS Cache becomes DISABLED , whenever the DS Function Enable toggles, and between patches.
	0	**Function Enable** Format: Enable If ENABLED, DS threads will be spawned to process incoming domain points which miss in the DS cache. If DISABLED, the DS stage goes into pass-through mode and performs no specific processing. This field is always used. **Programming Notes** The tessellation stages (HS, TE and DS) must be enabled/disabled as a group. I.e., draw commands can only be issued if all three stages are enabled or all three stages are disabled, otherwise the behavior is UNDEFINED.
7	31:27	**Reserved** Access: RO Format: MBZ
	26:21	**Vertex URB Entry Output Read Offset** Format: U6 Specifies the offset (in 256-bit units) at which Vertex URB data is to be read from the URB by SBE. Value: [0,63] — Name:
	20:16	**Vertex URB Entry Output Length** Format: U5 Specifies the amount of URB data written for each Vertex URB entry, in 256-bit register increments.

3DSTATE_DS_BODY

		Value / **Name**: [1,16] /
		Programming Notes: This length does not include the vertex header.
	15:8	**User Clip Distance Clip Test Enable Bitmask** Format: U8 This 8 bit mask field selects which of the 8 user clip distances against which trivial reject / trivial accept / must clip determination needs to be made.DX10 allows simultaneous use of ClipDistance and Cull Distance test of up to 8 distances.
	7:0	**User Clip Distance Cull Test Enable Bitmask** Format: U8 This 8 bit mask field selects which of the 8 user clip distances against which trivial reject / trivial accept determination needs to be made (does not cause a must clip).DX10 allows simultaneous use of ClipDistance and Cull Distance test of up to 8 distances.
8..9	63:6	**DUAL_PATCH Kernel Start Pointer** Format: InstructionBaseOffset[63:6] This field specifies the starting location of the DUAL_PATCH kernel program run by threads spawned by this FF unit.It is specified as a 64-byte-granular offset from the Instruction Base Address.This field is ignored if DS Function Enable is DISABLED. See DUAL_PATCH Thread Execution for a discussion of how the DUAL_PATCH KSP is used.
	5:0	**Reserved** Access: RO Format: MBZ

3DSTATE_GS_BODY

3DSTATE_GS_BODY

Source: RenderCS

Size (in bits): 288

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0..1	63:6	**Kernel Start Pointer** Format: InstructionBaseOffset[63:6] This field specifies the starting location of the kernel program run by threads spawned by this FF unit. It is specified as a 64-byte-granular offset from the Instruction Base Address.
	5:0	**Reserved** Access: RO Format: MBZ
2	31	**Single Program Flow** Format: Enable Specifies the initial condition of the kernel program as either a single program flow (SIMDnxm with m = 1) or as multiple program flows (SIMDnxm with m > 1). See CR0 description in ISA Execution Environment. Value / Name / Description: 0h / Disable / Single Program Flow disabled 1h / Enable / Single Program Flow enabled
	30	**Vector Mask Enable** Format: Enable Upon subsequent GS thread dispatches, this bit is loaded into the EU's Vector Mask Enable (VME, cr0.0[3]) thread state. Refer to EU documentation for the definition and use of VME state. Value / Name / Description: 0h / Dmask / The EU will use the Dispatch Mask (supplied by the GS stage) for instruction execution. 1h / Vmask / The EU will use the Vector Mask (derived from Dispatch Mask) for instruction execution. **Programming Notes** Under normal conditions SW shall specify DMask, as the GS stage will provide a Dispatch Mask appropriate to SIMD4x2 or SIMD8 thread execution (as a function of dispatch mode). E.g., for SIMD4x2 execution, the GS stage will generate a Dispatch Mask that is equal to what the EU would use as the Vector Mask. For SIMD8 execution there is no known usage model for use of Vector Mask (as there is for PS shaders).

3DSTATE_GS_BODY

29:27 **Sampler Count**

Format:	U3

Specifies how many samplers (in multiples of 4) the geometry shader kernel uses. Used only for prefetching the associated sampler state entries.

Value	Name	Description
0h	No Samplers	No Samplers used
1h	1-4 Samplers	Between 1 and 4 samplers used
2h	5-8 Samplers	Between 5 and 8 samplers used
3h	9-12 Samplers	Between 9 and 12 samplers used
4h	13-16 Samplers	Between 13 and 16 samplers used
5h-7h	Reserved	

26 **Reserved**

Access:	RO
Format:	MBZ

25:18 **Binding Table Entry Count**

Format:	U8

When **HW Generated Binding Table** is disabled: Specifies how many binding table entries the kernel uses. Used only for prefetching of the binding table entries and associated surface state. Note: For kernels using a large number of binding table entries, it may be wise to set this field to zero to avoid prefetching too many entries and thrashing the state cache. When **HW Generated Binding Table** bit is enabled: This field indicates which cache lines (512bit units - 32 Binding Table Entry section) should be fetched. Each bit in this field corresponds to a cache line. Only the 1st 4 non-zero Binding Table entries of each 32 Binding Table entry section prefetched will have its surface state prefetched.

Programming Notes
When HW binding table bit is set, it is assumed that the Binding Table Entry Count field will be generated at JIT time.

17 **Thread Dispatch Priority**

Specifies the priority of the thread for dispatch.

Value	Name	Description
0h	Normal	Normal thread dispatch priority
1h	High	High thread dispatch priority

16 **Floating Point Mode**

Specifies the initial floating point mode used by the dispatched thread.

Value	Name	Description
0h	IEEE-754	Use IEEE-754 Rules
1h	Alternate	Use alternate rules

3DSTATE_GS_BODY

	15:14	**Reserved** Access: RO Format: MBZ
	13	**Illegal Opcode Exception Enable** Format: Enable This bit gets loaded into EU CR0.1[12] (note the bit # difference). See *Exceptions and ISA Execution Environment*.
	12	**Accesses UAV** Format: Enable This field must be set when GS has a UAV access. **Programming Notes** This field must not be set when GS Function Enable is disabled.
	11	**Mask Stack Exception Enable** Format: Enable This bit gets loaded into EU CR0.1[11]. See *Exceptions and ISA Execution Environment.*
	10:8	**Reserved** Format: MBZ
	7	**Software Exception Enable** Format: Enable This bit gets loaded into EU CR0.1[13] (note the bit # difference). See *Exceptions and ISA Execution Environment.*
	6	**Reserved** Access: RO Format: MBZ
	5:0	**Expected Vertex Count** Format: U6 Specifies the number of vertices per input object expected by the GS thread. Input topologies not matching this expect value are discarded. Note that **DiscardAdjacency** is also considered (e.g., if the value programmed is 3 and DiscardAdjacency is set, TRILIST_ADJ and TRISTRIP_ADJ topologies are not discarded as they will pass 3 vertices/object to the GS threads). **Value** | **Name** [1,32] |
3..4	63:32	**Reserved** Access: RO Format: MBZ

3DSTATE_GS_BODY

	31:10	**Scratch Space Base Pointer** Format: GeneralStateOffset[31:10] Specifies the starting location of the scratch space area allocated to this FF unit as a 1K-byte aligned offset from the General State Base Address. If required, each thread spawned by this FF unit will be allocated some portion of this space, as specified by Per-Thread Scratch Space. The computed offset of the thread-specific portion will be passed in the thread payload as Scratch Space Offset. The thread is expected to utilize "stateless" DataPort read/write requests to access scratch space, where the DataPort will cause the General State Base Address to be added to the offset passed in the request header. This field is ignored if VS Function Enable is DISABLED.
	9:4	**Reserved** Access: RO Format: MBZ
	3:0	**Per-Thread Scratch Space** Format: U4 Specifies the amount of scratch space to be allocated to each thread spawned by this FF unit.The driver must allocate enough contiguous scratch space, starting at the Scratch Space Base Pointer, to ensure that the Maximum Number of Threads can each get Per-Thread Scratch Space size without exceeding the driver-allocated scratch space. **Value** / **Name** / **Description** [0,11] / / indicating [1K Bytes, 2M Bytes]
5	31	**Reserved** Access: RO Format: MBZ
	30:29	**Dispatch GRF Start Register For URB Data [5:4]** Format: U2 Specifies bit [5:4] of the starting GRF register number for the URB portion (Constant + Vertices) of the thread payload. The **Dispatch GRF Start Register For URB Data [3:0]** field is used to specify bits [3:0] of the starting GRF register number.
	28:23	**Output Vertex Size** Format: U6 [0,63] indicating [1,64] 16B units Specifies the size of each vertex stored in the GS output entry (following any Control Header data) as a number of 128-bit units (minus one). **Programming Notes** Programming Restrictions: The vertex size must be programmed as a multiple of 32B units with the following exception: Rendering is disabled (as per SOL stage state) and the vertex size output by the GS thread is 16B.If rendering is enabled (as per SOL state) the vertex size must be

3DSTATE_GS_BODY

		programmed as a multiple of 32B units. In other words, the only time software can program a vertex size with an odd number of 16B units is when rendering is disabled.
	22:17	**Output Topology** Format: 3D_Prim_Topo_Type This field specifies the topology type (3DPrimType) to be associated with GS-thread output vertices (if any).
	16:11	**Vertex URB Entry Read Length** Specifies the amount of URB data read and passed in the thread payload for each Vertex URB entry, in 256-bit register increments. **Programming Notes** Programming Restriction: This field must be a non-zero value if Include Vertex Handles is cleared to zero.
	10	**Include Vertex Handles** Format: Boolean If set, all the input Vertex URB handles are included in the payload. These are referred to as "pull model" URB handles, as the thread will use them to read from the URB. **Programming Notes** Programming Restriction: This field must be set if Vertex URB Entry Read Length is cleared to zero.
	9:4	**Vertex URB Entry Read Offset** Format: U6 Specifies the offset (in 256-bit units) at which Vertex URB data is to be read from the URB before being included in the thread payload. This offset applies to all Vertex URB entries passed to the thread.
	3:0	**Dispatch GRF Start Register For URB Data** Format: U4 Specifies the starting GRF register number for the URB portion (Constant + Vertices) of the thread payload. The Dispatch GRF Start Register for URB Data [5:4] field is used to extend the range of the starting GRF register number to [0,63]. **Value** / **Name** / **Description** [0,15] / / indicating bits [3:0] of the GRF number **Programming Notes** If Include Vertex Handles is enabled (pull or hybrid handles case), then For simd4x2: For DUAL_OBJECT dispatch mode this field should be:

3DSTATE_GS_BODY

		(((2*numVerticesPerObject) + 8 - 1)/8) + 1 For SINGLE and DUAL_INSTANCE dispatch modes this field should be: ((numVerticesPerObject +8 - 1)/8) + 1 If Include Primitive ID is set, then add 1 to the value obtained by using the above
		If Include Vertex Handles is enabled (pull or hybrid handles case), then. simd8:For InstanceCount == 1:numVerticesPerObject 2For InstanceCount > 1:((numVerticesPerObject 8 - 1)/8) 2If Include Primitive ID is set, then add 1 to the value obtained by using the above
6	31:24	**Reserved** Access: RO Format: MBZ
	23:20	**Control Data Header Size** Format: U4 Specifies the number of 32B units of control data header located at the start of the GS URB entry. The value 0 indicates there is no control data header, and Control Data Format is ignored and neither Cut nor StreamID bits are defined. Software must ensure that the Control Data Header Size is sufficient to accommodate the maximum number of vertices possibly output by the GS thread. It is UNDEFINED for a GS thread to report more output vertices than can be accommodated in a non-zero-sized header.

Value	Name
[0,8]	32B Units

	19:15	**Instance Control** Format: U5-1 Specifies the number of instances (minus one) for each input object. To avoid confusion, this document uses the term **"InstanceCount"** to refer to InstanceControl+1, with a range of [1,32]If **InstanceCount>1**, DUAL_OBJECT mode is invalid. Software will likely want to use DUAL_INSTANCE mode for higher performance, but SINGLE mode is also supported. When **InstanceCount=1** (one instance per object), software can decide which dispatch mode to use. DUAL_OBJECT mode would likely be the best choice for performance, followed by SINGLE mode. DUAL_INSTANCE mode is not recommended but is supported.

Value	Name	Description
[0,31]		Indicating [1,31] instances

	14:13	**Default Stream Id** Format: U2 When the GS is enabled, unless the GS output entry contains StreamID bits in the control header, this field specifies the default StreamID associated with any GS-thread output vertices. When the GS is disabled, StreamID will be output as 0.
	12:11	**Dispatch Mode** Format: U2 This field specifies how the GS unit dispatches multiple instances and/or multiple objects.

Value	Name	Description	Programming Notes
3h	SIMD8	Each thread shades up to 8	The driver must send pipe control with a cs

3DSTATE_GS_BODY

		different objects or (if InstanceCount >1) 8 instances of a single object.	stall after a 3dstate_gs state change and the Dispatch Mode is simd8 and the number of handles allocated to gs is less than 16.

Programming Notes
The GS must be allocated at least two URB handles or behavior is UNDEFINED for Dual Instance or Dual Object mode.
The only valid Dispatch Mode is SIMD8.

10 **Statistics Enable**

Format:	Enable

This bit controls whether GS-unit-specific statistics register(s) can be incremented.

Value	Name	Description
0h	Disable	GS_INVOCATIONS_COUNT and GS_PRIMITIVES_COUNT cannot increment
1h	Enable	GS_INVOCATIONS_COUNT and GS_PRIMITIVES_COUNT can increment

9:5 **Invocations Increment Value**

Format:	U5

Specifies how much to increment the GS_INVOCATIONS_COUNT for each instance of each object. This control is provided to allow software to process multiple instances (from an API POV) in a single kernel invocation. In SINGLE dispatch mode, the counter will increment by this value for each dispatch (as it's only one instance of one object). In DUAL_INSTANCE mode, the counter will be incremented by the value if only one instance is included in the dispatch (i.e., the last odd instance), otherwise the counter will be incremented by twice this value.In DUAL_OBJECT dispatch mode, the counter will be incremented by the value if only one object is included in the dispatch (i.e., a forced dispatch of one object), otherwise the counter will be incremented by twice this value.

Value	Name	Description
[0,31]		indicating an increment of [1,32]

4 **Include Primitive ID**

Format:	Boolean

If set, R1 of the payload is written with Primitive ID value(s).If clear, these Primitive ID values are not included in the payload R1.

3 **Hint**

Format:	U1

This state bit is simply passed in GS thread payloads for use by the GS kernel - it has no other impact on hardware operation.

2 **Reorder Mode**

This bit controls how vertices of triangle objects resulting from TRISTRIP[_ADJ][_REV] topologies are [re]ordered when passed in the GS thread payload See Object Vertex Ordering table (below).

Value	Name	Description
0h	LEADING	Reorder the vertices of alternating triangles of a TRISTRIP[_ADJ] such that the leading (first) vertices are in consecutive order starting at v0. A similar

3DSTATE_GS_BODY

				reordering is performed on alternating triangles in a TRISTRIP_REV.
		1h	TRAILING	Reorder the vertices of alternating triangles of a TRISTRIP[_ADJ] such that the trailing (last) vertices are in consecutive order starting at v2. A similar reordering is performed on alternating triangles in a TRISTRIP_REV.

	1	**Discard Adjacency** Format: Enable When set, adjacent vertices will not be passed in the GS payload when objects with adjacency are processed. Instead, only the non-adjacent vertices will be passed inthe same fashion as the without-adjacency form of the primitive. Software should set this bit whenever a GS kernel is used that does not expect adjacent vertices. This allows both with-adjacency/without-adjacency variants of the primitive to be submitted to the pipeline (via 3DPRIMITIVE) - the GS unit will silently discard any adjacent vertices and present the GS thread with only the internal object. When clear, adjacent vertices will be passed to the GS thread, as dictated by the incoming primitive type. Software should only clear this bit when a GS kernel is used that does expect adjacent vertices. E.g., if the GS kernel is compiled to expect a TRIANGLE_ADJ object, software must clear this bit. Software should also clear this bit if the GS kernel expects a POINT or PATCHLIST_n object (which don't have with-adjacency variants). The only hardware assistance is to allow the submission of a with-adjacency variant of a primitive when operating with a GS kernel that expects the without-adjacency variant of the object. (E.g., when the GS kernel is compiled to expect a TRIANGLE object, software should set this bit just in case a TRILIST_ADJ is submitted to the pipeline.) Note that the GS unit is otherwise not aware of the object type that is expected by the GS kernel. It is up to software to ensure that the submitted primitive type (in 3DPRIMITIVE) is otherwise compatible with the object type expected by the GS kernel. (E.g., if the GS kernel expects a LINE_ADJ object, only LINELIST_ADJ or LINESTRIP_ADJ should be submitted, otherwise the GS kernel will produce unpredictable results.) Also note that it is possible to craft a GS kernel which can accept any object type that's thrown at it by first examining the PrimType passed in the payload and then using this info to correctly interpret the number of vertices passed in the payload.
	0	**Enable** Format: Enable Specifies whether the GS stage is enabled or disabled (pass-through).
7	31	**Control Data Format** Format: U1 This field specifies the format of the control data header (if any).

Value	Name	Description
0h	CUT	The control data header contains Cut bits.
1h	SID	The control data header contains StreamID bits. Output Topology must be set to POINTLIST, or behavior is UNDEFINED.

3DSTATE_GS_BODY

	30	**Static Output** Format: Enable Specifies whether the GS shader outputs a static number of vertices per invocation. If this bit is clear, the number of vertices output by each GS shader invocation is stored by the GS thread in Vertex Count at the very beginning of the output URB entry (see GS URB Entry description).
	29:27	**Reserved** Access: RO Format: MBZ
	26:16	**Static Output Vertex Count** Format: U11 If **GSEnable** is ENABLED and **StaticOutput** is ENABLED, this field specifies the total number of vertices output each GS shader invocation. If **GSEnable** is ENABLED and **StaticOutput** is DISABLED (i.e., variable GS output), the total number of vertices output by a GS shader invocation is stored by the thread at the very beginning of the output URB entry, and this field is ignored. (See GS URB Entry description). **Value** / **Name** [0,1024]
	15:9	**Reserved** Access: RO Format: MBZ
	8:0	**Maximum Number of Threads** Format: U9-1 Specifies the maximum number of simultaneous threads allowed to be active. Used to avoid using up the scratch space. Programming the value of the max threads over the number of threads based off number of threads supported in the execution units may improve performance since the architecture allows threads to be buffered between the check for max threads and the actual dispatch into the EU. Programming the max values to a number less than the number of threads supported in the execution units may reduce performance. **Value** / **Name** / **Description** [0,335] / / indicating thread count of [1,336]
8	31:27	**Reserved** Access: RO Format: MBZ
	26:21	**Vertex URB Entry Output Read Offset** Format: U6 Specifies the offset (in 256-bit units) at which Vertex URB data is to be read from the URB by SBE. **Value** / **Name** [0,63]

3DSTATE_GS_BODY

	20:16	**Vertex URB Entry Output Length** Format: U5 Specifies the amount of URB data written for each Vertex URB entry, in 256-bit register increments. **Value** [1,16] **Name** **Programming Notes** This length does not include the vertex header.
	15:8	**User Clip Distance Clip Test Enable Bitmask** Format: Enable[8] This 8 bit mask field selects which of the 8 user clip distances against which trivial reject / trivial accept / must clip determination needs to be made.DX10 allows simultaneous use of ClipDistance and Cull Distance test of up to 8 distances.
	7:0	**User Clip Distance Cull Test Enable Bitmask** Format: Enable[8] This 8 bit mask field selects which of the 8 user clip distances against which trivial reject / trivial accept determination needs to be made (does not cause a must clip).DX10 allows simultaneous use of ClipDistance and Cull Distance test of up to 8 distances.

3DSTATE_HIER_DEPTH_BUFFER_BODY

3DSTATE_HIER_DEPTH_BUFFER_BODY

Source: RenderCS
Size (in bits): 128
Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:25	**Hierarchical Depth Buffer Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for the hierarchical depth buffer.
	24	**Reserved** Access: RO Format: MBZ
	23:22	**Tiled Mode** **Description** This field specifies the tiled resource mode. **Value** / **Name** / **Description** 0h / No tiled resource (Tile Y Mode) / 1h / TILEYF / 4KB tiled resources 2h / TILEYS / 64KB tiled resources 3h / Reserved / **Programming Notes** HZ buffer only supports Tile Y mode.
	21	**Reserved** Access: RO Format: MBZ
	20	**Write thru enable for Texture** **Description** This bit must be set if the Depth buffer is used as a texture. If this bit is set, HZ will force a write of non-clear values to the Depth buffer avoiding the need of a Depth resolve. This means that HZ will not write planes to the HZ$. **Value** / **Name** 1h / Enable 0h / Disable **[Default]**

3DSTATE_HIER_DEPTH_BUFFER_BODY

		Programming Notes For MSAA with multiple resolves, SW may set this bit after the first resolve with compression disabled to enforce HW to write-through to depth buffer as uncompressed data to save on future resolves. However the clears on a surface must be done with this bit reset for MSAA.
	19:18	**Reserved** Access: RO Format: MBZ
	17	**Reserved** Access: RO Format: MBZ
	16:0	**Surface Pitch** Format: U17-1 This field specifies the pitch of the hierarchical depth buffer in (#Bytes - 1). **Value** / **Name** [7Fh,1FFFFh] corresponding to [128B, 512KB] also restricted to a multiple of 128B **Programming Notes** Since this surface is tiled, the pitch specified must be a multiple of the tile pitch, in the range [128B, 128KB].
1..2	63:0	**Surface Base Address** Format: GraphicsAddress[63:0] This field specifies the address of the buffer in Graphics Memory. **Programming Notes** The Hierarchical Depth Buffer can only be mapped to Main Memory (uncached). If the surface is tiled, the base address must conform to the Per-Surface Tiling Alignment Rules as documented in TBD.
3	31:16	**Reserved** Access: RO Format: MBZ
	15	**Reserved** Access: RO Format: MBZ
	14:0	**Surface QPitch** Format: U17[16:2] The interpretation of this field is dependent on **Surface Type** as follows: • SURFTYPE_2D/CUBE: distance in *rows* between array slices

3DSTATE_HIER_DEPTH_BUFFER_BODY

Format:
`QPitch[16:2]`

Value	Name	Description
[1h,7FFFh]		in multiples of 4 (low 2 bits missing)

Programming Notes

This field must be set to an integer multiple of 16 (QPitch[3] MBZ) Software must ensure that this field is set to a value sufficiently large such that the array slices in the surface do not overlap. Refer to the Memory Data Formats section for information on how surfaces are stored in memory.

3DSTATE_HS_BODY

3DSTATE_HS_BODY

Source: RenderCS

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:30	**Reserved** Access: RO Format: MBZ
	29:27	**Sampler Count** Format: U3 Specifies how many samplers (in multiples of 4) the HS kernels use. Used only for prefetching the associated sampler state entries. Value / Name / Description: 0h / No Samplers / no samplers used 1h / 1-4 Samplers / between 1 and 4 samplers used 2h / 5-8 Samplers / between 5 and 8 samplers used 3h / 9-12 Samplers / between 9 and 12 samplers used 4h / 13-16 Samplers / between 13 and 16 samplers used 5h-7h / Reserved / Reserved
	26	**Reserved** Access: RO Format: MBZ
	25:18	**Binding Table Entry Count** Format: U8 When HW Generated Binding Table is disabled: Specifies how many binding table entries the kernel uses. Used only for prefetching of the binding table entries and associated surface state. Note: For kernels using a large number of binding table entries, it may be wise to set this field to zero to avoid prefetching too many entries and thrashing the state cache. **Programming Notes** When HW binding table bit is set, it is assumed that the Binding Table Entry Count field will be generated at JIT time.
	17	**Thread Dispatch Priority** Specifies the priority of the thread for dispatch Value / Name / Description: 0h / Normal / Normal Priority 1h / High / High Priority

3DSTATE_HS_BODY

16 — Floating Point Mode

Specifies the initial floating point mode used by the dispatched thread.

Value	Name	Description
0h	IEEE-754	Use IEEE-754 Rules
1h	alternate	Use alternate rules

15:14 — Reserved

Access:	RO
Format:	MBZ

13 — Illegal Opcode Exception Enable

Format:	Enable

This bit gets loaded into EU CR0.1[12] (note the bit # difference). See Exceptions and ISA Execution Environment.

12 — Software Exception Enable

Format:	Enable

This bit gets loaded into EU CRO1[13] (note the bit # difference). See Exceptions and ISA Execution Environment.

11 — Reserved

Access:	RO
Format:	MBZ

10:8 — Reserved

Format:	MBZ

7:4 — Reserved

Access:	RO
Format:	MBZ

3:0 — Reserved

Access:	RO
Format:	MBZ

1 — 31 — Enable

Format:	Enable

Specifies whether the HS function is enabled or disabled (pass-through). If ENABLED MI_TOPOLOGY_FILTER must be used to silently discard any topologies that the HS kernel is not expecting. E.g., if the HS kernel is expecting PATCHLIST_32 topologies, MI_TOPOLOGY_FILTER must be set to PATCHLIST_32 so only those topologies can reach the enabled HS.

Programming Notes
The tessellation stages (HS, TE and DS) must be enabled/disabled as a group. I.e., draw commands can only be issued if all three stages are enabled or all three stages are disabled, otherwise the behavior is UNDEFINED.

3DSTATE_HS_BODY

30 Reserved

Access:	RO
Format:	MBZ

29 Statistics Enable

Format:	Enable

This bit controls whether HS-unit-specific statistics register(s) will increment (for each patch).

28:17 Reserved

Access:	RO
Format:	MBZ

16:8 Maximum Number of Threads

Format:	U9-1

Specifies the maximum number of simultaneous threads allowed to be active. Used to avoid using up the scratch space. Programming the value of the max threads over the number of threads based off number of threads supported in the execution units may improve performance since the architecture allows threads to be buffered between the check for max threads and the actual dispatch into the EU. Programming the max values to a number less than the number of threads supported in the execution units may reduce performance.

Value	Name	Description
[0,335]		indicating thread count of [1,336]

Programming Notes
The Maximum Number of Threads must be set to at least twice the setting of 3DSTATE_HS::**Instance Count.**

7:5 Reserved

Access:	RO
Format:	MBZ

4:0 Instance Count

Format:	U5-1

This field determines the number of threads (minus one) spawned per input patch. If the HS kernel uses a barrier function, software must restrict the **Instance Count** to the number of threads that can be simultaneously active within a subslice. Factors which must be considered includes scratch memory availability.

Value	Name	Description
[0,31]		representing [1,32] instances

2..3 63:6 Kernel Start Pointer

Format:	InstructionBaseOffset[63:6]

This field specifies the starting location of the kernel program run by threads spawned by this FF unit. It is specified as a 64-byte-granular offset from the Instruction Base Address.

3DSTATE_HS_BODY

	5:0	**Reserved** Access: RO Format: MBZ
4..5	63:32	**Reserved** Access: RO Format: MBZ
	31:10	**Scratch Space Base Pointer** Format: GeneralStateOffset[31:10] **Value** / **Name** / **Description** [0,31] / / Specifies the location of the scratch space area allocated to this FF unit, specified as a 1KB-granular offset from the General State Base Address. If required, each thread spawned by this FF unit will be allocated some portion of this space, as specified by Per-Thread Scratch Space.
	9:4	**Reserved** Access: RO Format: MBZ
	3:0	**Per-Thread Scratch Space** Format: U4 Specifies the amount of scratch space to be allocated to each thread spawned by this FF unit.The driver must allocate enough contiguous scratch space, starting at the Scratch Space Base Pointer, to ensure that the Maximum Number of Threads can each get Per-Thread Scratch Space size without exceeding the driver-allocated scratch space. **Value** / **Name** / **Description** [0,11] / / Indicating[1K Bytes, 2M Bytes
6	31:29	**Reserved** Access: RO Format: MBZ
	28	**Dispatch GRF Start Register For URB Data [5]** Format: U1 Specifies bit [5] of the starting GRF register number for the URB portion (Constant + Vertices) of the thread payload. The **Dispatch GRF Start Register For URB Data [4:0]** field is used to specify bits [4:0] of the starting GRF register number.
	27	**Single Program Flow** Format: Enable Specifies the initial condition of the kernel program as either a single program flow (SIMDnxm with m = 1) or as multiple program flows (SIMDnxm with m > 1). See CR0 description in *ISA Execution Environment*.

3DSTATE_HS_BODY

Value	Name	Description
0h	Reserved	
1h	Enable	Single Program Flow Enabled

26 **Vector Mask Enable**

Format:	Enable

Upon subsequent HS thread dispatches, this bit is loaded into the EU's Vector Mask Enable (VME, cr0.0[3]) thread state. Refer to the EU documentation for the definition and use of VME state.

Value	Name	Description
0h	Dmask	The EU will use the Dispatch Mask (supplied by the HS stage) for instruction execution.
1h	Vmask	The EU will use the Vector Mask (derived from the Dispatch Mask) for instruction execution.

Programming Notes
Under normal conditions SW shall specify DMask, as the HS stage will provide a Dispatch Mask appropriate to SIMD4x2 or SIMD8 thread execution (as a function of dispatch mode). E.g., for SIMD4x2 thread execution, the HS state will generate a Dispatch Mask that is equal to what the EU would use as a Vector Mask. For SIMD8 execution there is no known usage model for use of Vector Mask (as there is for PS shaders).

25 **Accesses UAV**

Format:	Enable

This field must be set when HS has a UAV access

Programming Notes
This field must not be set when HS Function Enable is disabled.

24 **Include Vertex Handles**

Format:	Boolean

If set, all the input Vertex URB handles are included in payloads. This field is ignored if **HS Function Enable** is DISABLED.

Programming Notes
Programming Restriction: This field must be set if value if **Vertex URB Entry Read Length** is cleared to zero.

23:19 **Dispatch GRF Start Register For URB Data**

Format:	U5

Specifies the starting GRF register number for the URB portion (Constant + Vertices) of the thread payload. This field is ignored if **HS Function Enable** is DISABLED.
The Dispatch GRF Start Register for URB Data [5] field is used to extend the range of the starting GRF register number to [0,63].

3DSTATE_HS_BODY

Value	Name	Description
[0,31]		indicating bits [4:0] of the GRF number

Programming Notes
When Include Vertex Handles is set for non-instanced 8_PATCH dispatch of PATCHLIST_30..32 objects, pushed vertex data and/or pushed constants cannot be used as they would need to start in the payload beyond the range of this field (i.e., beyond R31). When Include PrimitiveID is also set, this issue extends to non-instanced 8_PATCH dispatch of PATCHLIST_29..32 objects.

18:17 **Dispatch Mode**

Format:	U2

This field is unused to set the current thread dispatch mode for the HS stage.

Value	Name	Description
2h	8_PATCH	HS threads are passed inputs and an output handle associated with (up to) 8 patches in SIMD8 fashion. See 8_PATCH Thread Payload.
3h	Reserved	

Programming Notes
DUAL_PATCH is not supported.

16:11 **Vertex URB Entry Read Length**

Format:	U6

Specifies the amount of URB data read and passed in the thread payload for each Vertex URB entry, in 256-bit register increments. This field is ignored if HS Function Enable is DISABLED.

Value	Name
[0,63]	

Programming Notes
Programming Restriction: This field must be a non-zero value if **Include Vertex Handles** is cleared to zero.

10 **Reserved**

Access:	RO
Format:	MBZ

9:4 **Vertex URB Entry Read Offset**

Format:	U6

Specifies the offset (in 256-bit units) at which Vertex URB data is to be read from the URB before being included in the thread payload. This offset applies to all Vertex URB entries passed to the thread. This field is ignored if HS Function Enable is DISABLED.

Value	Name
[0,63]	

3DSTATE_HS_BODY

	3:1	**Patch Count Threshold** Format: U3 Specifies the maximum number of patches that will be accumulated before a thread is dispatched. The dispatch of threads can (optionally) be forced before a full complement of eight patches have been accumulated. **Value** / **Name** / **Description** [1,7] / / This specifies the maximum number of patches that will be accumulated before a thread dispatch is forced. 0 / **[Default]** / Early thread dispatch due to the Patch Count Threshold is disabled. A full complement of 8 patches can be accumulated before a thread is dispatched.
	0	**Include Primitive ID** Format: Enable If set, R1 of the payload is written with Primitive ID value(s).If clear, these Primitive IDs are not included in the payload R1. **Programming Notes** This field is only used when DUAL_PATCH DispatchMode is specified. In SINGLE_PATCH, the single Primitive ID is always passed in R0.
7	31:0	**Reserved** Access: RO Format: MBZ

3DSTATE_INDEX_BUFFER_BODY

3DSTATE_INDEX_BUFFER_BODY

Source: RenderCS

Size (in bits): 128

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:12	**Reserved** Access: RO Format: MBZ
	11	**L3 Bypass Disable** Format: Disable **Description** When set, index data fetches are routed through the L3 and therefore that index data may be coherent with the L3 cache, as controlled by the Memory Object Control State (MOCS) value. I.e., if portions of the index buffer already reside in the L3 (e.g., were written or read by another L3 agent), reads from VF may hit in the L3 with the cached data returned to VF. If reads from VF miss in the L3 cache, the reads are directed to the next higher in the memory hierarchy, but the data returned is not placed in the L3 cache. The MOCS value must not be set to cache the data in L3. When clear, index data fetches bypass the L3 logic, therefore precluding the coherency of that data in the L3 cache. If the vertex buffer data can be cached in L3, the L3 cache must first be flushed to maintain vertex buffer data coherency.
	10	**Reserved** Access: RO Format: MBZ
	9:8	**Index Format** Format: U2 This field specifies the data format of the index buffer. All index values are UNSIGNED. Value / Name: 0h: BYTE 1h: WORD 2h: DWORD
	7	**Reserved** Access: RO Format: MBZ

3DSTATE_INDEX_BUFFER_BODY

	6:0	**Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this index buffer.
1..2	63:0	**Buffer Starting Address** Format: GraphicsAddress[63:0] **Description** VIRTUAL_ADDR[63:48] are ignored by the HW. This field contains the size-aligned (as specified by Index Format) Graphics Address of the first element of interest within the index buffer. Software must program this value with the combination (sum) of the base address of the memory resource and the byte offset from the base address to the starting structure within the buffer. **Programming Notes** Index Buffers can only be allocated in linear (not tiled) graphics memory.
3	31:0	**Buffer Size** Format: U32 This field specifies the size of the buffer in bytes. Index accesses which straddle or go past the end of the buffer will return 0..Note that BufferSize=0 indicates that there is no valid data in the buffer. **Value**: [0, FFFFFFFFh] **Name**:

3DSTATE_MULTISAMPLE_BODY

3DSTATE_MULTISAMPLE_BODY

Source: RenderCS
Size (in bits): 32
Default Value: 0x00000000

DWord	Bit	Description
0	31:6	**Reserved** Access: RO Format: MBZ
	5	**Pixel Position Offset Enable** Format: Enable Enables the device to offset pixel positions by 0.5 both in horizontal and vertical directions. **Programming Notes** Setting this field along with setting the Pixel Location to upper left and number of multisamples to greater than one will cause the device to offset pixel positions by 0.5 both in horizontal and vertical directions. It is to be noted this is done to adjust the pixel co-ordinate system to DX9 like, so any WM_HZ_OP screen space rectangles (eg: legacy HiZ Clear, Resolve etc) generated internally by driver in this mode needs to be aware of this offset adjustment and send the rectangles according to alignment restriction taking this offset adjustment into consideration. SW can choose to set this bit only for DX9 API. DX10/OGL API's should not have any effect by setting or not setting this bit.
	4	**Pixel Location** Format: U1 This field specifies where the device evaluates "pixel" (vs. centroid or sample) values/attributes. Value / Name / Description: 0h / CENTER / Use the pixel center (0.5, 0.5 offset) 1h / UL_CORNER / Use the pixel upper-left corner **Programming Notes** The programming of this field is assumed to be a function of the API being supported. Specifically, it is expected that OpenGL and DX10+ APIs require CENTER selection, while DX9- APIs require UL_CORNER selection. When 3DSTATE_RASTER::**ForcedSampleCount** is other than NUMRASTSAMPLES_0, this field must be 0h.
	3:1	**Number of Multisamples** Format: U3 This field specifies how many samples/pixel exist in all RTs and the Depth Buffer, as log2(#samples). This field is valid regardless of the setting of **Multisample Rasterization Mode**.

3DSTATE_MULTISAMPLE_BODY

Value	Name	Description
0h	1	1 sample/pixel
1h	2	2 samples/pixel
2h	4	4 samples/pixel
3h	8	8 samples/pixel
4h	16	16 samples/pixel
5h-7h	Reserved	

Programming Notes
The setting of this field must match the **Number of Multisamples** field in SURFACE_STATE of all bound render targets.

0 **Reserved**

Access:	RO
Format:	MBZ

3DSTATE_PRIMITIVE_REPLICATION_BODY

3DSTATE_PRIMITIVE_REPLICATION_BODY

Source: RenderCS

Size (in bits): 160

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:16	**Replica Mask** Specifies which replicas should be drawn. If bit *k* (0 <= *k* < 16) is clear, then replica *k* will not be rasterized. Bits *k* >= numReplicas are ignored
	15:4	**Reserved** Access: RO Format: MBZ
	3:0	**Replication Count** Format: U4 Specifies the number of replica positions produced by the last pre-raster shader. This value must match the SV_Position array length of the last pre-raster shader Value: [0h, Fh] Name:
1	31:28	**Viewport Offset[7]** Format: U4 Specifies an offset to add to SV_ViewportArrayIndex for each replica
	27:24	**Viewport Offset[6]** Format: U4 Specifies an offset to add to SV_ViewportArrayIndex for each replica
	23:20	**Viewport Offset[5]** Format: U4 Specifies an offset to add to SV_ViewportArrayIndex for each replica
	19:16	**Viewport Offset[4]** Format: U4 Specifies an offset to add to SV_ViewportArrayIndex for each replica
	15:12	**Viewport Offset[3]** Format: U4 Specifies an offset to add to SV_ViewportArrayIndex for each replica

3DSTATE_PRIMITIVE_REPLICATION_BODY

	11:8	**Viewport Offset[2]** Format: U4 Specifies an offset to add to SV_ViewportArrayIndex for each replica
	7:4	**Viewport Offset[1]** Format: U4 Specifies an offset to add to SV_ViewportArrayIndex for each replica
	3:0	**Viewport Offset[0]** Format: U4 Specifies an offset to add to SV_ViewportArrayIndex for each replica
2	31:28	**Viewport Offset[15]** Format: U4 Specifies an offset to add to SV_ViewportArrayIndex for each replica
	27:24	**Viewport Offset[14]** Format: U4 Specifies an offset to add to SV_ViewportArrayIndex for each replica
	23:20	**Viewport Offset[13]** Format: U4 Specifies an offset to add to SV_ViewportArrayIndex for each replica
	19:16	**Viewport Offset[12]** Format: U4 Specifies an offset to add to SV_ViewportArrayIndex for each replica
	15:12	**Viewport Offset[11]** Format: U4 Specifies an offset to add to SV_ViewportArrayIndex for each replica
	11:8	**Viewport Offset[10]** Format: U4 Specifies an offset to add to SV_ViewportArrayIndex for each replica
	7:4	**Viewport Offset[9]** Format: U4 Specifies an offset to add to SV_ViewportArrayIndex for each replica

3DSTATE_PRIMITIVE_REPLICATION_BODY		
	3:0	**Viewport Offset[8]** Format: U4 Specifies an offset to add to SV_ViewportArrayIndex for each replica
3	31:28	**RTAI Offset[7]** Format: U4 Specifies an offset to add to SV_RenderTargetArrayIndex for each replica
	27:24	**RTAI Offset[6]** Format: U4 Specifies an offset to add to SV_RenderTargetArrayIndex for each replica
	23:20	**RTAI Offset[5]** Format: U4 Specifies an offset to add to SV_RenderTargetArrayIndex for each replica
	19:16	**RTAI Offset[4]** Format: U4 Specifies an offset to add to SV_RenderTargetArrayIndex for each replica
	15:12	**RTAI Offset[3]** Format: U4 Specifies an offset to add to SV_RenderTargetArrayIndex for each replica
	11:8	**RTAI Offset[2]** Format: U4 Specifies an offset to add to SV_RenderTargetArrayIndex for each replica
	7:4	**RTAI Offset[1]** Format: U4 Specifies an offset to add to SV_RenderTargetArrayIndex for each replica
	3:0	**RTAI Offset[0]** Format: U4 Specifies an offset to add to SV_RenderTargetArrayIndex for each replica
4	31:28	**RTAI Offset[15]** Format: U4 Specifies an offset to add to SV_RenderTargetArrayIndex for each replica

3DSTATE_PRIMITIVE_REPLICATION_BODY		
	27:24	**RTAI Offset[14]** Format: U4 Specifies an offset to add to SV_RenderTargetArrayIndex for each replica
	23:20	**RTAI Offset[13]** Format: U4 Specifies an offset to add to SV_RenderTargetArrayIndex for each replica
	19:16	**RTAI Offset[12]** Format: U4 Specifies an offset to add to SV_RenderTargetArrayIndex for each replica
	15:12	**RTAI Offset[11]** Format: U4 Specifies an offset to add to SV_RenderTargetArrayIndex for each replica
	11:8	**RTAI Offset[10]** Format: U4 Specifies an offset to add to SV_RenderTargetArrayIndex for each replica
	7:4	**RTAI Offset[9]** Format: U4 Specifies an offset to add to SV_RenderTargetArrayIndex for each replica
	3:0	**RTAI Offset[8]** Format: U4 Specifies an offset to add to SV_RenderTargetArrayIndex for each replica

3DSTATE_PS_BLEND_BODY

3DSTATE_PS_BLEND_BODY		
Source:	RenderCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31	**Alpha To Coverage Enable** Format: Enable If set, indicates that AlphaToCoverage is on RT[0], since this bit must be set the same for all RTs in the MRT case.
	30	**Has Writeable RT** Format: Enable When set indicates the there is at least one non-null RT w/ at least one channel write enabled
	29	**Color Buffer Blend Enable** Format: Enable When set indicates that RT[0] has color buffer blend enabled.
	28:24	**Source Alpha Blend Factor** Format: 3D_Color_Buffer_Blend_Factor Indicates the "source factor" in alpha Color Buffer Blending stage for RT[0]
	23:19	**Destination Alpha Blend Factor** Format: 3D_Color_Buffer_Blend_Factor Indicates the "destination factor" in alpha Color Buffer Blending stage for RT[0]
	18:14	**Source Blend Factor** Format: 3D_Color_Buffer_Blend_Factor Indicates the "source factor" in Color Buffer Blending stage for RT[0]
	13:9	**Destination Blend Factor** Format: 3D_Color_Buffer_Blend_Factor Indicates the "destination factor" in Color Buffer Blending stage for RT[0]
	8	**Alpha Test Enable** Format: Enable Indicates the AlphaTestEnable for RT[0]

3DSTATE_PS_BLEND_BODY		
	7	**Independent Alpha Blend Enable** Format: Enable Indicates the Independent Alpha Blend Enable for RT[0] When enabled, the other fields in this instruction control the combination of the alpha components in the Color Buffer Blend stage. When disabled, the alpha components are combined in the same fashion as the color components.
	6:0	**Reserved** Access: RO Format: MBZ

3DSTATE_PS_BODY

3DSTATE_PS_BODY

Source: RenderCS

Size (in bits): 352

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0..1	63:6	**Kernel Start Pointer 0** Format: InstructionBaseOffset[63:6] Specifies the 64-byte aligned address offset of the first instruction in the kernel[0]. This pointer is relative to the **Instruction Base Address.**
	5:0	**Reserved** Access: RO Format: MBZ
2	31	**Single Program Flow** Single Program Flow (SPF) specifies the initial condition of the kernel program as either a single program flow (SIMDnxm with m = 1) or as multiple program flows (SIMDnxm with m > 1). See CR0 description in ISA Execution Environment. Value / Name / Description 0h / Multiple / Multiple Program Flows 1h / Single / Single Program Flows
	30	**Vector Mask Enable** Format: Enable When SPF=0, Vector Mask Enable (VME) specifies which mask to use to initialize the initial channel enables. When SPF=1, VME specifies which mask to use to generate execution channel enables. Value / Name / Description 0h / Dmask / Channels are enabled based on the dispatch mask 1h / Vmask / Channels are enabled based on the vector mask
	29:27	**Sampler Count** Specifies how many samplers (in multiples of 4) the vertex shader 0 kernel uses. Used only for prefetching the associated sampler state entries. Value / Name / Description [0,4] / / 0h / No Samplers / no samplers used 1h / 1-4 Samplers / between 1 and 4 samplers used 2h / 5-8 Samplers / between 5 and 8 samplers used

3h	9-12 Samplers	between 9 and 12 samplers used
4h	13-16 Samplers	between 13 and 16 samplers used
5h-7h		Reserved

26 **Single Precision Denormal Mode**

Specifies the single precision denormal mode used by the dispatched thread.

Value	Name	Description
0h	Flushed to Zero	Single Precision denormals are flushed to zero
1h	Retained	Single Precision denormals are retained

25:18 **Binding Table Entry Count**

Specifies how many binding table entries the kernel uses. Used only for prefetching of the binding table entries and associated surface state. **Note:** For kernels using a large number of binding table entries, it may be advantageous to set this field to zero to avoid prefetching too many entries and thrashing the state cache. This field is ignored if [PS Function Enable] is DISABLED.

When [HW Generated Binding Table] bit is enabled: This field indicates which cache lines (512bit units - 32 Binding Table Entry section) should be fetched. Each bit in this field corresponds to a cache line. Only the 1st 4 non-zero Binding Table entries of each 32 Binding Table entry section prefetched will have its surface state prefetched. See 3D Pipeline for more information.

Programming Notes
When HW binding table bit is set, it is assumed that the Binding Table Entry Count field will be generated at JIT time.

17 **Thread Dispatch Priority**

Specifies the priority of the thread for dispatch.

Value	Name	Description
0h	Normal	Normal Priority
1h	High	High Priority

16 **Floating Point Mode**

Specifies the floating point mode used by the dispatched thread.

Value	Name	Description
0h	IEEE-754	Use IEEE-754 rules
1h	Alternate	Use alternate rules

15:14 **Rounding Mode**

Specifies the rounding mode used by the dispatched thread.

Value	Name	Description
0h	RTNE	Round to Nearest Even
1h	RU	Round toward +infinity
2h	RD	Round toward -infinity

3DSTATE_PS_BODY

3h	RTZ	Round toward zero

13 Illegal Opcode Exception Enable

Format:	Enable

This bit gets loaded into EU CR0.1[12] (note the bit # difference). See Exceptions and ISA Execution Environment.

12 Reserved

Access:	RO
Format:	MBZ

11 Mask Stack Exception Enable

Format:	Enable

This bit gets loaded into EU CR0.1[12] (note the bit # difference). See Exceptions and ISA Execution Environment.

10:8 Reserved

Format:	MBZ

7 Software Exception Enable

Format:	Enable

This bit gets loaded into EU CR0.1[13] (note the bit # difference). See Exceptions and ISA Execution Environment.

6:0 Reserved

Access:	RO
Format:	MBZ

3..4 63:32 Reserved

Access:	RO
Format:	MBZ

31:10 Scratch Space Base Pointer

Format:	GeneralStateOffset[31:10]

Specifies the 1k-byte aligned address offset to scratch space for use by the kernel. This pointer is relative to the General State Base Address.

Programming Notes

Scratch Space per slice is computed based on 4 sub-slices. SW must allocate scratch space enough so that each slice has 4 slices allowed.

9:4 Reserved

Access:	RO
Format:	MBZ

3DSTATE_PS_BODY

	3:0	**Per Thread Scratch Space** Format: U4 Specifies the amount of scratch space allowed to be used by each thread. The driver must allocate enough contiguous scratch space, pointed to by the Scratch Space Pointer, to ensure that the Maximum Number of Threads each get Per Thread Scratch Space size without exceeding the driver-allocated scratch space. **Value** / **Name** [0,11] indicating [1k bytes, 2M bytes] in powers of two
5	31:23	**Maximum Number of Fused Threads Per PSD** Format: U9-1 **Description** Specifies the maximum number of simultaneous virtual fused threads allowed to be active per Pixel Shader Dispatch(PSD). PSD serves a pair of subslices. This bit-field can be programmed in the range: [0,63] each integer in the range linearly maps to maximum number of virtual fused threads in the range: [1, 64]. The allowable range is larger than the maximum number of fused physical threads per PSD. It is advantageous for performance reasons to allow more virtual threads than physical threads to ensure maximum usage of compute resources. Each fused thread represents 2 threads. **Programming Notes** If this field is changed between 3DPRIMITIVE commands, a PIPE_CONTROL command with Stall at Pixel Scoreboard set is required to be issued. This note only applies to 2 pass AMFS approach where AMFS unit launches Texel Shaders. This deadlock workaround is not needed for 3 pass approach where evaluate message does not cause AMFS unit to spawn Texel Shaders. When Pixel Shader contains one or more evaluate message for Procedural Texture, and AMFS is expected to dispatch Texel Shaders, the maximum number of fused virtual threads must be programmed to be less than maximum number of fused physical threads possible per PSD. Maximum number of fused physical threads is device specific. (see device specific programming notes for this field)This ensures that AMFS unit never gets deadlocked by restricting PSD from using all available compute resources. For typical Procedural Texture usage model we can program one less than maximum physical fused threads.
	22	**Reserved** Access: RO Format: MBZ
	21	**Pixel Scoreboard Disable** Setting this bit disables the pixel shader scoreboard for ordering the RTs and ROVs on the same screen space coordinates. **Programming Notes** There are cases when ordering the render target outputs or ROV outputs from the shader, for example:

3DSTATE_PS_BODY

1) all blend functions are commutative, here are the most common cases:
BLEND_OP = ADD or MIN or MAXand both src and dst blend factors are constants= 1.0.
2) There is no over-draw in the render pass (for example full screen 3D PASS which accesses a pixel in the color buffer just once).
When HW detects the change in this bit, it implicitly performs the PS scoreboard stall before allowing the subsequent group of pixel shader threads.

When SIMD32 is enabled, this bit **must not be set**. Pixel Scoreboard must always be enabled when SIMD32 is enabled.

20 **Reserved**

Access:	RO
Format:	MBZ

19:12 **Clear/Resolve BTI for Render Target**

Description
When Color Clear or Resolve bit is set in this state packet, this bit field indicates BTI to be used to access the Render Target Surface that's being cleared/resolved.

11 **Push Constant Enable**

Format:	Enable

This field must be enabled if the sum of the **PS Constant Buffer [3:0] Read Length** fields in 3DSTATE_CONSTANT_PS is nonzero, and must be disabled if the sum is zero.

10 **3D Scoreboard Address Size select**

Select the granularity use for scoreboard address calculation.

Value	Name
0h	2x2
1h	4x2

Programming Notes
This bit field allows programmable pixel scoreboard granularity: 2X2 pixel block(value = 0) or 4X2 pixel block(value = 1). When the value of this bit field changes, HW detects the change and takes the action to either force thread-group dependency or stalls at the scoreboard (based on the MMIO(PSS_MODE2):Thread Group Dependency Control). When enabling fused-SIMD32 dispatch mode, HW implicitly sets the scoreboard size to 4X2 independent of the value of this bit-field.

9 **Overlapping Subspans Enable**

Format:	Enable

This field indicates if two subspans (from two objects) rasterized to same screen-space XY coordinates can be packed into a single EU thread payload or across fused threads.
The shader compiler must set this field to DISABLED when pixel shader code requires serialized execution on per-pixel basis; examples include pixel shader using RT reads or pixel sync.
This field must be set to DISABLED if kernel has any coding structures that can create possibility

of younger object (e.g. upper fused thread or upper SIMD8 of dual-SIMD8 pair) to issue message before older object in fused thread.
Examples include:

- SIMD16/dual-SIMD8 thread issuing 2 SIMD8 messages
- A message issued from within if-else. (if message else different message)
- 3DSTATE_PS_EXTRA:killpix is set and depth/stencil write is enabled
- 3DSTATE_PS_EXTRA:computed depth/stencil is set and depth/stencil write is enabled
- Read and write to same UAV or RT

Programming Notes
When 3DSTATE_PS:Pixel Scoreboard Disable is set, this field **must be set**. The intent of scoreboard disable is to allow overlapping primitives in the shader stage and therefore disabling overlapping pixels defeats the purpose of this optimization.

8 **Render Target Fast Clear Enable**

Format:	Enable

This field is set to enable fast clear of the bound render targets. See "**Render Target Fast Clear**" and "**MCS/CCS Buffers for Render Targets**" for restrictions on enabling this field.
A general programming sequence for doing the Render Target Fast Clear requires:
-- setting up the Render Target State with RT that needs to be cleared as well as clear value is stored at the Clear Value Address in the RT State
-- Provide the BTI for that RT in this state packet
-- set this bit in the state packet provided the fast clear guidelines described in the Fast Clear section
-- DRAW command with a rectangle (scaled appropriately) as a primitive

Programming Notes
For PoSH based Tiled Rendering, Color Fast clear operation is recommended to be performed outside of tile pass, for performance reasons. After Fast clear, render cache flush is required.
For PoSH based Tiled Rendering, Color Fast clear can be inside the tile pass without significant performance penalty and it does not require render cache flush after fast clear of color.
When this bit is set, corresponding BTI for the render target that is being cleared must be equal to 0.
When this bit is set, RENDER_SURFACE_STATE type must not be NULL.
3D/Volumetric surfaces do not support Fast Clear operation.

7:6 **Render Target Resolve Type**

Format:	U2

Specifies what type of Render Target Resolve is needed for the surface to be consumed properly by the end Client. Programming notes below.

Value	Name	Description	Programming Notes
0h	RESOLVE_DISABLED	No Resolve Needed	
1h	RESOLVE_PARTIAL	Partial resolve is for resolving RT for clear values i.e. it leaves	Display engine does not support unresolved clear values in the

3DSTATE_PS_BODY

		no cache lines at implied clear value.	display buffer, hence this resolve is required before binding any compressed RT to the display via flip commands.
2h	FAST_CLEAR_0	Fast Clear to 0 during Clear Pass; Used to Initialize CCS Buffer with 0s to support Lossless Compressed Without Clear.	This state has to be programmed only with Render Target Fast Clear Enable described above. If the Render Target Fast Clear = 0, this Field Cannot be programmed to 2h.
3h	RESOLVE_FULL	Full Resolve is for Resolving RT for Clear/Compressed to Uncompressed State	

Programming Notes
When this bit is set, corresponding BTI for the render target that is being resolved must be equal to 0.
When this bit is set, RENDER_SURFACE_STATE type must not be NULL.
For multisample render target, this field must beRESOLVE_DISABLED.
3D/Volumetric surfaces do not support in-place resolve pass operation.

5 **Dual-SIMD8 Dispatch Enabled**

Format:	Enable

This field determines type of pixel shader enabled by**8 Pixel Dispatch or Dual-8 Pixel Dispatch Enable** field.
If DISABLED, the pixel shader kernel receives SIMD8 payload (8 pixels from 1 object).
If ENABLED, the pixel shader kernel receives dual-SIMD8 payload (8 pixels from 1st object and 8 pixels from 2nd object)

4:3 **Position XY Offset Select**

Format:	U2

This field specifies if/what Position XY Offset values are passed in the PS payload. Note that these are per-slot (pixel|sample) offsets, and therefore separate from the subspan XY coordinates passed in R1.

Value	Name	Description
0h	POSOFFSET_NONE	No Position XY Offsets are included in the PS payload.
1h	Reserved	
2h	POSOFFSET_CENTROID	Position XY Offsets will be passed in the PS payload, and these will reflect the Centroid position(s).
3h	POSOFFSET_SAMPLE	Position XY Offsets will be passed in the PS payload, and these will reflect the multisample position(s).

		Programming Notes SW Recommendation: If the PS kernel needs the Position Offsets to compute a Position XY value, this field should match Position ZW Interpolation Mode to ensure a consistent position.xyzw computation If the PS kernel does not need the Position XY Offsets to compute a Position Value, then this field should be programmed to POSOFFSET_NONE, as the PS kernel should be using the various barycentric inputs to evaluate other-than-position attributes. However, this field can be used to pass Centroid or Sample offsets in the payload for special test modes (e.g., where barycentric coordinates are computed in the PS vs. being HW-generated and passed in the payload). MSDISPMODE_PERSAMPLE is required in order to select POSOFFSET_SAMPLE.
	2	**32 Pixel Dispatch Enable** Format: Enable Enables the Windower to dispatch 8 subspans in one payload. Variable Pixel Dispatch in Section: Pixel Grouping (Dispatch size) control for valid pixel dispatch combinations. **Programming Notes** When NUM_MULTISAMPLES = 16 or FORCE_SAMPLE_COUNT = 16, SIMD32 Dispatch must not be enabled for PER_PIXEL dispatch mode. Must not be enabled when dispatch rate is sample AND NUM_MULTISAMPLES > 1. SIMD32 may only be enabled if SIMD16 or (dual)SIMD8 is also enabled. Must not be enabled when dispatch rate is coarse.
	1	**16 Pixel Dispatch Enable** Format: Enable Enables the Windower to dispatch 4 subspans in one payload. Variable Pixel Dispatch in Section: Pixel Grouping (Dispatch size) control for valid pixel dispatch combinations.
	0	**8 Pixel Dispatch Enable** Format: Enable **Description** Enables the Windower to dispatch 2 subspans from 1 object (polygon) in one payload. Variable Pixel Dispatch in Section: Pixel Grouping (Dispatch size) control for valid pixel dispatch combinations. If Dual-SIMD8 Dispatch Enabled, kernel pointer referenced by this field isDual-SIMD8 kernel pointer instead of SIMD8 kernel pointer.Dual-SIMD8 and SIMD8 modes are mutually exclusive and use the same kernel pointer entry. If Dual-SIMD8 Dispatch Enabled, the Widower packs 2 subspans from one object followed by 2 subspans from another object into one PS thread payload. **Programming Notes** When Render Target Fast Clear Enable is ENABLED or Render Target Resolve Type =

3DSTATE_PS_BODY

		RESOLVE_PARTIAL or RESOLVE_FULL, this bit must be DISABLED.
6	31:23	**Reserved** Access: RO Format: MBZ
	22:16	**Dispatch GRF Start Register For Constant/Setup Data 0** Format: U7 Specifies the starting GRF register number for the Constant/Setup portion of the thread payload for kernel[0]. Value: [0,127] — Name:
	15	**Reserved** Access: RO Format: MBZ
	14:8	**Dispatch GRF Start Register For Constant/Setup Data 1** Format: U7 Specifies the starting GRF register number for the Constant/Setup portion of the thread payload for kernel[1]. Value: [0,127] — Name:
	7	**Reserved** Access: RO Format: MBZ
	6:0	**Dispatch GRF Start Register For Constant/Setup Data 2** Format: U7 Specifies the starting GRF register number for the Constant/Setup portion of the thread payload for kernel[2]. Value: [0,127] — Name:
7..8	63:6	**Kernel Start Pointer 1** Format: InstructionBaseOffset[63:6] Specifies the 64-byte aligned address offset of the first instruction in kernel[1]. This pointer is relative to the Instruction Base Address.
	5:0	**Reserved** Access: RO Format: MBZ

3DSTATE_PS_BODY		
9..10	63:6	**Kernel Start Pointer 2** Format: InstructionBaseOffset[63:6] Specifies the 64-byte aligned address offset of the first instruction in kernel[2]. This pointer is relative to the **Instruction Base Address**.
	5:0	**Reserved** Access: RO Format: MBZ

3DSTATE_PS_EXTRA_BODY

3DSTATE_PS_EXTRA_BODY

Source: RenderCS
Size (in bits): 32
Default Value: 0x00000000

DWord	Bit	Description
0	31	**Pixel Shader Valid** Format: Enable When set indicates a valid pixel shader. When this bit clear the rest of this command should also be clear.
	30	**Pixel Shader Does not write to RT** Format: Enable When set indicates the pixel shader does not write to render target. **Programming Notes** When Pixel Shader writes to UAV but does not write to RT, a dummy render target write is required to convey EOT to the PS dispatch function. Hence, this bit must be reset in this case. If there is no RT or a NULL RT, Pixel Shader Kills Pixel is reset, and there is no UAV output from PS, SW must set this bit. When Pixel Shader has evaluate message present, i.e. '3DSTATE_PS_EXTRA:PS has Evaluate Message' is enabled, this bit field must be reset.
	29	**oMask Present to Render Target** Format: Enable This bit is inserted in the PS payload header and made available to the DataPort (either via the message header or via header bypass) to indicate that oMask data from the shader (one or two phases) is included in Render Target Write messages. If present, the oMask data is used to mask off samples.
	28	**Pixel Shader Kills Pixel** Format: Enable This bit, if ENABLED, indicates that the PS kernel has the ability to kill (discard) pixels or samples, other than due to depth or stencil testing. This bit is required to be ENABLED in the following situations: • The API pixel shader program contains "killpix" or "discard" instructions, or other code in the pixel shader kernel that can cause the final pixel mask to differ from the pixel mask received on dispatch.
	27:26	**Pixel Shader Computed Depth Mode** Format: U2 This field specifies the computed depth mode for the pixel shader.

3DSTATE_PS_EXTRA_BODY

Value	Name	Description
0h	PSCDEPTH_OFF	Pixel shader does not compute depth
1h	PSCDEPTH_ON	Pixel shader computes depth with no guarantee as to its value
2h	PSCDEPTH_ON_GE	
3h	PSCDEPTH_ON_LE	Pixel shader computes depth and guarantees that oDepth <= SourceDepth If the Position ZW interpolation mode in 3DSTATE_WM does not match the DX Multisample Rasterization mode in 3DSTATE_RASTER, HW will internally convert to PSCDEPTH_ON.

Programming Notes
If this field is set to any value other than PSCDEPTH_OFF, a multi-phase shader (i.e. dispatch RATE_COARSE or RATE_PIXEL with pixel/sample loops or sample loop respectively) must output depth and render targets only at the last phase.
When PS dispatch rate is COARSE_RATE, this field must be programmed to PSCDEPTH_OFF.

25 **Force Computed Depth**

Format:	Enable

Programming Notes
This field should be left DISABLED. This field should not be tested for functional validation.

24 **Pixel Shader Uses Source Depth**

Format:	Enable

This bit, if ENABLED, indicates that the PS kernel requires the source depth value (vPos.z) to be passed in the payload. The source depth value is interpolated according to the Position ZW Interpolation Mode state.

Programming Notes
This bit cannot be enabled when dispatch rate is RATE_COARSE

23 **Pixel Shader Uses Source W**

Format:	Enable

This bit, if ENABLED, indicates that the PS kernel requires the interpolated source W value (vPos.w) to be passed in the payload. The W value is interpolated according to the Position ZW Interpolation Mode state.

22 **Pixel Shader Requires Requested Coarse Pixel Shading Size**

Format:	Enable

This bit, if ENABLED, indicates that the PS kernel requires values of requested coarse pixel shading size to be passed in the payload for each 2x2 coarse pixel quad. Note: Actual coarse pixel shading rate is always delivered (constant across thread slot). This bit can only be set when dispatch rate is RATE_COARSE.

3DSTATE_PS_EXTRA_BODY

	21	**Pixel Shader Requires Source Depth and/or W Plane Coefficients** Format: Enable This bit, if ENABLED, indicates that the PS kernel requires the source depth and/or W plane equations to be passed in the payload. Note: both attributes are always delivered in same message phase, even if only one is used.
	20	**Pixel Shader Requires Perspective Bary Plane Coefficients** Format: Enable This bit, if ENABLED, indicates that the PS kernel requires the perspective plane coefficients to be passed in the payload.
	19	**Pixel Shader Requires Non-Perspective Bary Plane Coefficients** Format: Enable This bit, if ENABLED, indicates that the PS kernel requires the non-perspective plane coefficients to be passed in the payload.
	18	**Pixel Shader Requires Subpixel Sample Offsets** Format: Enable This bit, if ENABLED, indicates that the PS kernel requires the sub-pixel sample offsets to be passed in the payload.
	17	**Reserved** Format: MBZ
	16:12	**Reserved** Format: MBZ
	11	**PS has Evaluate Message** Format: Enable **Description** This bit indicates if Pixel Shader has Evaluate Message typically used in conjunction with AMFS. **Value** / **Name** 0 – Disable **[Default]** 1 – Enable **Programming Notes** This bit must be reset if 3DSTATE_PS_EXTRA:PS Valid bit is not set. This bit must be reset if Cache Mode Register 1 [6] is set , ie Shader Independent AMFS is enabled

3DSTATE_PS_EXTRA_BODY

10:9	**Reserved** Access: RO Format: MBZ
8	**Attribute Enable** Format: Enable **Description** This field must be enabled if the Number of SF Output Attributes field in 3DSTATE_SBE is nonzero, and must be disabled if that field is zero.
7	**Pixel Shader Disables Alpha To Coverage** Format: Enable When set indicates the pixel shader AlphaToCoverage should be disabled due to oMask output. The setting of this bit is API dependent.
6	**Pixel Shader Is Per Sample** Format: Enable This bit, when ENABLED, indicates that the pixel shader is dispatched at the per sample shading rate. If this bit is DISABLED, the dispatch rate is determined by the value of Pixel Shader Is Per Coarse Pixel. If this bit is ENABLED, Pixel Shader Is Per Coarse Pixel bit must be DISABLED.
5	**Pixel Shader Computes Stencil** Format: Enable This field when set indicates that the pixel shader computes the stencil reference value. **Programming Notes** If this field is ENABLED,a multi-phase shader (i.e. dispatch RATE_COARSE or RATE_PIXEL with pixel/sample loops or sample loop respectively) must output stencil and render targets only at the last phase. WhenPixel Shader is at COARSE_RATE, this field must not be set.
4	**Pixel Shader Is Per Coarse Pixel** Format: Enable If Pixel Shader Is Per Sample is DISABLED and this bit is ENABLED, the pixel shader is dispatched at the per coarse pixel shading rate. If Pixel Shader Is Per Sample is DISABLED and this bit is DISABLED, the pixel shader is dispatched at the per pixel shading rate. If Pixel Shader Is Per Sample is ENABLED, this bit must be DISABLED. **Restriction** SIMD32 kernel version cannot be configured when this bit is ENABLED.
3	**Pixel Shader Pulls Bary** Format: Enable This bit indicates if Pixel Shader uses Pull Bary i.e. PI message. If this bit is reset, PS does not do Pull Bary.

3DSTATE_PS_EXTRA_BODY

2 **Pixel Shader Has UAV**

Format:	Enable
Format:	U1 Enumerated Type

This field when set indicates that the pixel shader has a UAV attached to it.

1:0 **Input Coverage Mask State**

Format:	U2

This field indicates the type of input coverage mask that the PS kernel requires to be passed in the payload.

Value	Name	Description	Programming Notes
0h	NONE	Pixel shader does not use input coverage masks.	
1h	NORMAL	Input Coverage masks based on outer conservatism and factors in SAMPLE_MASKs. If Pixel is conservatively covered, all samples are enabled.	
2h	INNER_CONSERVATIVE	Input Coverage masks based on inner conservatism. If Pixel is conservatively fully covered all samples are enabled else none of the samples are covered.	When PS Dispatch Rate is at Coarse Pixel, requesting this Input Coverage Mask mode is illegal and not supported by HW. Input converage masks based on inner consevatism incorrectly ANDs SAMPLE_MASK in HW. Therefore, PS must retrieve the INNER coverage mask per pixel by bit-wise OR operation. [] Input converage masks based on inner consevatism incorrectly ANDs SAMPLE_MASK in HW. Therefore, PS must retrieve the INNER coverage mask per pixel by bit-wise OR operation. [] Input converage masks based on inner consevatism incorrectly ANDs SAMPLE_MASK in HW. Therefore, PS must retrieve the INNER coverage mask

3DSTATE_PS_EXTRA_BODY					
					per pixel by bit-wise OR operation.
		3h	DEPTH_COVERAGE	Input coverage masks are computed after factoring depth/stencil test results, only if Early Depth Stencil Test is enabled. If Early Depth Stencil Test is not enabled, HW uses NORMAL Input Coverage Masks.	

3DSTATE_PTBR_MARKER_BODY

3DSTATE_PTBR_MARKER_BODY		
Source:	RenderCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:2	**Reserved** Access: RO Format: MBZ
	1	**End of Tile** Format: Enable When set, indicates marker stating End of Tile in the command sequence.
	0	**Start of Tile** Format: Enable When set, indicates marker stating Start of Tile in the command sequence.

3DSTATE_PTBR_TILE_SELECT_BODY

3DSTATE_PTBR_TILE_SELECT_BODY

Source: RenderCS
Size (in bits): 32
Default Value: 0x00000000

DWord	Bit	Description
0	31	**Free Render List Disable** Format: Disable This bit controls the recycling (Freeing up, add back to the free pool) of the visibility data pages by render pipe. Value / Name / Description 0 / / Render pipe will free the pages to be recycled after consuming the visibility data for the current tile. 1 / / Render pipe will not free the pages to be recycled after consuming the visibility data for the current tile.
	30	**Geometry Statistics Disable** Format: Disable This bit controls the incrementing statistics counters in geometry units (VF, VS, HS, TE, TDS, GS, SOL, CL, SF). Value / Name / Description 0 / **[Default]** / Geometry units (VF, VS, HS, TE, TDS, GS, SOL, CL, SF)will increment their pipeline statistics counters. 1 / / Geometry units (VF, VS, HS, TE, TDS, GS, SOL, CL, SF)will not increment their pipeline statistics counters.
	29:24	**Reserved** Access: RO Format: MBZ
	23:16	**Render List Index** Format: U8 Specifies the index in to the Render-List for the current Tile. Range [0..127]. HW will fetch the starting page offset for the visibility data of the current tile from below memory location [{render_list_base_address[47:12], 12b0} + {render_list_pointer[31:6], 6b0} + (Render List Index «1)] **Programming Notes** Render List Index must be set to 0 when 3DSTATE_TILE_PASS_INFO:Tile Count is 0x0 for the corresponding Tile Pass.

3DSTATE_PTBR_TILE_SELECT_BODY		
	15:10	**Reserved** Access: RO Format: MBZ
	9:0	**Tile Rect Array Index** Format: U10 Specifies the index in to the Tile Rect Array of the current Tile Pass. Rang [0..1023].HW will fetch the RECT_STATE of the current tile from below memory location [{dynamic_state_base_addres[47:12], 12b0} + {Tile Rect Array Pointer[31:6], 6b0} + (Tile Index«3)] **Programming Notes** Tile Rect Array Index must be set to 0 when 3DSTATE_TILE_PASS_INFO:Tile Count is 0x0 for the corresponding Tile Pass.

3DSTATE_RASTER_BODY

3DSTATE_RASTER_BODY

Source: RenderCS

Size (in bits): 128

Default Value: 0x00210000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:27	**Reserved** Access: RO Format: MBZ
	26	**Viewport Z Far Clip Test Enable** Format: Enable This field is used to control whether the Viewport Z Far extent is considered in VertexClipTest.
	25	**Reserved** Access: RO Format: MBZ
	24	**Conservative Rasterization Enable** Format: Enable This field when set enables conservative rasterization rules for all primitives except rectangles, points and lines. For rectangle, points and lines, setting this bit is ignored by hardware. **Programming Notes** This bit must not be set for primitives with poly-stippling enabled or native rectlist. When this bit is set, sampling mode must be set to "Centre" sampling i.e 3DSTATE_MULTISAMPLE::Pixel Location set to CENTER
	23:22	**API Mode** Software sets this field according to the API's version. These bits are set for DX9 or OGL/DX10.0/DX10.1+/DX11.1 per the following values. Value / Name / Description 0h / DX9/OGL / 1h / DX10.0 / 2h / DX10.1+ / This is used for DX10.1+ and Vulkan API 3h / Reserved /
	21	**Front Winding** Determines whether a triangle object is considered "front facing" if the screen space vertex positions, when traversed in the order, result in a clockwise (CW) or counter-clockwise (CCW)

winding order. Does not apply to points or lines.

Value	Name	Description
0h	Clockwise	FRONTWINDING_CW
1h	Counter Clockwise **[Default]**	FRONTWINDING_CCW

20:18 **Forced Sample Count**

Format:	U3

This field specifies how many samples/pixel exist for RT Independent Rasterization

Value	Name	Description
0h	NUMRASTSAMPLES_0	No RT Independent Rasterization
1h	NUMRASTSAMPLES_1	1 rast-sample/pixel
2h	NUMRASTSAMPLES_2	2 rast-samples/pixel
3h	NUMRASTSAMPLES_4	4 rast-samples/pixel
4h	NUMRASTSAMPLES_8	8 rast-samples/pixel
5h	NUMRASTSAMPLES_16	16 rast-samples/pixel
6h-7h	Reserved	

Programming Notes
When 3DSTATE_MULTISAMPLE::Number of Multisamples !=NUMSAMPLES_1, this field must be either NUMRASTSAMPLES_0 or NUMRASTSAMPLES_1. When 3DSTATE_MULTISAMPLE::Number of Multisamples == NUMSAMPLES_1, this field must not be NUMRASTSAMPLES_1.

17:16 **Cull Mode**

Controls removal (culling) of triangle objects based on orientation. The cull mode only applies to triangle objects and does not apply to lines, points or rectangles.

Value	Name	Description
0h	CULLMODE_BOTH	All triangles are discarded (i.e., no triangle objects are drawn)
1h	CULLMODE_NONE **[Default]**	No triangles are discarded due to orientation
2h	CULLMODE_FRONT	Triangles with a front-facing orientation are discarded
3h	CULLMODE_BACK	Triangles with a back-facing orientation are discarded

Programming Notes
Orientation determination is based on the setting of the Front Winding state.

15 **Reserved**

Access:	RO
Format:	MBZ

3DSTATE_RASTER_BODY

14 Force Multisampling

This field provides a work around override for the computation of SF_INT::Multisample Rasterization Mode and WM_INT::Multisample Rasterization Mode.

Value	Name	Description
0h	Normal	Multisampling mode is computed by HW according to formula for signal SF_INT::Multisample Rasterization Mode and WM_INT::Multisample Rasterization Mode in vol2a.11 3D Pipeline Windower > Windower Pipelined State > 3DSTATE_WM > 3DSTATE_WM.
1h	Force	Forces the DX Multisampling mode to be used directly

13 Smooth Point Enable

Format:	Enable

Software sets this according to API. When OGL and smooth point rasterization is required, this bit must be set. HW ignores this bit for primitives other than points.

12 DX Multisample Rasterization Enable

Format:	Enable

Software sets this according to the API's multisample enable

Programming Notes
This state only effects how the SF_INT/WM_INT::Multisample Rasterization Mode are set depending on some other states. This state mainly modifies the how the line rendering is done by setting SF_INT/WM_INT::Multisample Rasterization Mode to either OFF* or ON* . Please refer to table under SF_INT::Multisample Rasterization Mode.

11:10 DX Multisample Rasterization Mode

Format:	U2

This field determines whether multisample rasterization is turned on/off, and how the pixel sample point(s) are defined. Software sets this according to the API's multisample state setting (if any)

Value	Name
0h	MSRASTMODE_ OFF_PIXEL
1h	MSRASTMODE_ OFF_PATTERN
2h	MSRASTMODE_ ON_PIXEL
3h	MSRASTMODE_ ON_PATTERN

Programming Notes
This field is used to directly set the SF_INT/WM_INT::Multisample Rasterization Mode when DX Multisample Rasterization Enable is set. Please refer to equation of SF_INT::Multisample Rasterization Mode.

9 Global Depth Offset Enable Solid

Format:	Enable

Enables computation and application of Global Depth Offset for SOLID objects.

3DSTATE_RASTER_BODY

	8	**Global Depth Offset Enable Wireframe** Format: Enable Enables computation and application of Global Depth Offset when triangles are rendered in WIREFRAME mode.
	7	**Global Depth Offset Enable Point** Format: Enable Enables computation and application of Global Depth Offset when triangles are rendered in POINT mode.
	6:5	**Front Face Fill Mode** Format: U2 This state controls how front-facing triangle and rectangle objects are rendered.

Value	Name	Description
0h	SOLID	Any triangle or rectangle object found to be front-facing is rendered as a solid object. This setting is required when rendering rectangle (RECTLIST) objects.
1h	WIREFRAME	Any triangle object found to be front-facing is rendered as a series of lines along the triangle boundaries (as determined by the topology type and controlled by the vertex EdgeFlags).
2h	POINT	Any triangle object found to be front-facing is rendered as a set of point primitives at the triangle vertices (as determined by the topology type and controlled by the vertex EdgeFlags).
3h	Reserved	

	4:3	**Back Face Fill Mode** Format: U2 This state controls how back-facing triangle and rectangle objects are rendered.

Value	Name	Description
0h	SOLID	Any triangle or rectangle object found to be back-facing is rendered as a solid object. This setting is required when rendering rectangle (RECTLIST) objects.
1h	WIREFRAME	Any triangle object found to be back-facing is rendered as a series of lines along the triangle boundaries (as determined by the topology type and controlled by the vertex EdgeFlags).
2h	POINT	Any triangle object found to be back-facing is rendered as a set of point primitives at the triangle vertices (as determined by the topology type and controlled by the vertex EdgeFlags).
3h	Reserved	

3DSTATE_RASTER_BODY

	2	**Antialiasing Enable** Format: Enable This field enables "alpha-based" line antialiasing. **Programming Notes** This field must be disabled if any of the render targets have integer (UINT or SINT) surface format.
	1	**Scissor Rectangle Enable** Format: Enable Enables operation of Scissor Rectangle.
	0	**Viewport Z Near Clip Test Enable** Format: Enable This field is used to control whether the Viewport Z Near extent is considered in VertexClipTest.
1	31:0	**Global Depth Offset Constant** Format: IEEE_FLOAT Specifies the constant term in the Global Depth Offset function.
2	31:0	**Global Depth Offset Scale** Format: IEEE_FLOAT Specifies the scale term used in the Global Depth Offset function.
3	31:0	**Global Depth Offset Clamp** Format: IEEE_FLOAT Specifies the clamp term used in the Global Depth Offset function.

3DSTATE_SAMPLE_MASK_BODY

3DSTATE_SAMPLE_MASK_BODY

Source: RenderCS

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Sample Mask** Format: Enable[16] A per-multisample-position mask state variable that is immediately and unconditionally ANDed with the sample coverage mask as part of the rasterization process. This mask is applied prior to centroid selection. This mask must be ignored for centroid selection when RTIR is enabled i.e. Forced_Sample_Count > 0. **Programming Notes** • If **Number of Multisamples** is NUMSAMPLES_1, bits 15:1 of this field will be zeroed by HW. • If **Number of Multisamples** is NUMSAMPLES_2, bits 15:2 of this field will be zeroed by HW. • If **Number of Multisamples** is NUMSAMPLES_4, bits 15:4 of this field will be zeroed by HW. • If **Number of Multisamples** is NUMSAMPLES_8, bits 15:8 of this field will be zeroed by HW. When pixel shader writes to UAV but does not have actual render target write (i.e. no RT is bound to pixel shader, even though, RT write message is sent for EOT), appropriate SAMPLE_MASK must be all set depending on Number of Multisamples.

3DSTATE_SAMPLER_STATE_POINTERS_BODY

3DSTATE_SAMPLER_STATE_POINTERS_BODY		
Source:	RenderCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:5	**Pointer to Sampler State** Format: DynamicStateOffset[31:5]SAMPLER_STATE*16 Specifies the 32-byte aligned address offset of the function's SAMPLER_STATE table. This offset is relative to the Dynamic State Base Address.
	4:0	**Reserved** Access: RO Format: MBZ

3DSTATE_SBE_BODY

3DSTATE_SBE_BODY

Source: RenderCS

Size (in bits): 160

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Reserved** Access: RO Format: MBZ
	29	**Force Vertex URB Entry Read Length** Format: Enable This field provides a work around override for the computation of SBE_INT::Vertex URB Entry Read Length. If asserted, 3DSTATE_SBE::Vertex URB Entry Read Length is be used directly. Otherwise, SBE_INT::Vertex URB Entry Read Length is computed normally.
	28	**Force Vertex URB Entry Read Offset** Format: Enable This field provides a work around override for the computation of SBE_INT::Vertex URB Entry Read Offset. If asserted, 3DSTATE_SBE::Vertex URB Entry Read Offset is be used directly. Otherwise, SBE_INT::Vertex URB Entry Read Offset is computed normally.
	27:22	**Number of SF Output Attributes** Format: U6 Specifies the number of vertex attributes passed from the SF stage to the WM stage (does not include Position). Value: [0,32] — Name:
	21	**Attribute Swizzle Enable** Format: Enable Enables the SF to perform swizzling on (up to the first 16) vertex attributes. If DISABLED, all vertex attributes are passed through.
	20	**Point Sprite Texture Coordinate Origin** This state controls how Point Sprite Texture Coordinates are generated (when enabled on a per-attribute basis by Point Sprite Texture Coordinate Enable).

3DSTATE_SBE_BODY

Value	Name	Description
0h	UPPERLEFT	Top Left = (0,0,0,1)Bottom Left = (0,1,0,1)Bottom Right = (1,1,0,1)
1h	LOWERLEFT	Top Left = (0,1,0,1)Bottom Left = (0,0,0,1)Bottom Right = (1,0,0,1)

19 **Primitive ID Override Component W**

Format:	Enable

If set, the W component of output attribute selected by Primitive ID Override Attribute Select is overridden with the Primitive ID.

18 **Primitive ID Override Component Z**

Format:	Enable

If set, the Z component of output attribute selected by Primitive ID Override Attribute Select is overridden with the Primitive ID.

17 **Primitive ID Override Component Y**

Format:	Enable

If set, the Y component of output attribute selected by Primitive ID Override Attribute Select is overridden with the Primitive ID.

16 **Primitive ID Override Component X**

Format:	Enable

If set, the X component of output attribute selected by Primitive ID Override Attribute Select is overridden with the Primitive ID.

15:11 **Vertex URB Entry Read Length**

Format:	U5

Specifies the amount of URB data read for each Vertex URB entry, in 256-bit register increments.

Value	Name
[1,16]	

Programming Notes

It is UNDEFINED to set this field to 0 indicating no Vertex URB data to be read. This field should be set to the minimum length required to read the maximum source attribute. The maximum source attribute is indicated by the maximum value of the enabled Attribute # Source Attribute if Attribute Swizzle Enable is set, Number of Output Attributes-1 if enable is not set.
read_length = ceiling((max_source_attr+1)/2)

10:5 **Vertex URB Entry Read Offset**

Specifies the offset (in 256-bit units) at which Vertex URB data is to be read from the URB.

4:0 **Primitive ID Override Attribute Select**

Specifies which attribute is overridden w/ the Primitive ID

3DSTATE_SBE_BODY

		Programming Notes Set all Primitive ID Override Component Select X/Y/Z/W to 0 to indicate there is no Primitive ID override.
1	31:0	**Point Sprite Texture Coordinate Enable** Format: Enable[32] When processing point primitives, the attributes from the incoming point vertex are typically copied to the point object corner vertices. However, if a bit is set in this field, the corresponding Attribute is selected as a Point Sprite Texture Coordinate, in which case each corner vertex is assigned a pre-defined texture coordinate as defined by the Point Sprite Texture Coordinate Origin state bit. Bit 0 corresponds to output Attribute 0.
2	31:0	**Constant Interpolation Enable** Format: Enable[32] This field is a bitmask containing a Constant Interpolation Enable bit for each corresponding attribute. If a bit is set, that attribute will undergo constant interpolation, and the corresponding WrapShortest Enable bits (if defined) will be ignored. If a bit is clear, components which are not enabled for WrapShortest interpolation (if defined) will be linearly interpolated.
3	31:30	**Attribute 15 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 15 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	29:28	**Attribute 14 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 14 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	27:26	**Attribute 13 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 13 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	25:24	**Attribute 12 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 12 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.

3DSTATE_SBE_BODY

	23:22	**Attribute 11 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 11 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	21:20	**Attribute 10 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 10 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	19:18	**Attribute 9 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 9 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	17:16	**Attribute 8 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 8 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	15:14	**Attribute 7 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 7 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	13:12	**Attribute 6 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 6 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	11:10	**Attribute 5 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 5 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.

3DSTATE_SBE_BODY		
	9:8	**Attribute 4 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 4 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	7:6	**Attribute 3 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 3 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	5:4	**Attribute 2 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 2 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	3:2	**Attribute 1 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 1 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	1:0	**Attribute 0 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 0 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
4	31:30	**Attribute 31 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 31 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	29:28	**Attribute 30 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 30 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.

3DSTATE_SBE_BODY

	27:26	**Attribute 29 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 29 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	25:24	**Attribute 28 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 28 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	23:22	**Attribute 27 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 27 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	21:20	**Attribute 26 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 26 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	19:18	**Attribute 25 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 25 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	17:16	**Attribute 24 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 24 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	15:14	**Attribute 23 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 23 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.

3DSTATE_SBE_BODY

	13:12	**Attribute 22 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 22 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	11:10	**Attribute 21 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 21 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	9:8	**Attribute 20 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 20 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	7:6	**Attribute 19 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 19 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	5:4	**Attribute 18 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 18 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	3:2	**Attribute 17 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 17 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.
	1:0	**Attribute 16 Active Component Format** Format: **Attribute_Component_Format** This state indicates which components of Attribute 16 are being used by the pixel shader Kernel. SBE will not perform attribute delta calculations for any disabled components. Operation is UNDEFINED if kernel uses attribute vertex delta for any disabled component.

3DSTATE_SBE_SWIZ_BODY

3DSTATE_SBE_SWIZ_BODY		
Source:	RenderCS	
Size (in bits):	320	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0..7	255:0	**Attribute** Format: SF_OUTPUT_ATTRIBUTE_DETAIL[16]
8..9	63:60	**Attribute 15 Wrap Shortest Enables** Format: WRAP_SHORTEST_ENABLE
	59:56	**Attribute 14 Wrap Shortest Enables** Format: WRAP_SHORTEST_ENABLE
	55:52	**Attribute 13 Wrap Shortest Enables** Format: WRAP_SHORTEST_ENABLE
	51:48	**Attribute 12 Wrap Shortest Enables** Format: WRAP_SHORTEST_ENABLE
	47:44	**Attribute 11 Wrap Shortest Enables** Format: WRAP_SHORTEST_ENABLE
	43:40	**Attribute 10 Wrap Shortest Enables** Format: WRAP_SHORTEST_ENABLE
	39:36	**Attribute 09 Wrap Shortest Enables** Format: WRAP_SHORTEST_ENABLE
	35:32	**Attribute 08 Wrap Shortest Enables** Format: WRAP_SHORTEST_ENABLE
	31:28	**Attribute 07 Wrap Shortest Enables** Format: WRAP_SHORTEST_ENABLE
	27:24	**Attribute 06 Wrap Shortest Enables** Format: WRAP_SHORTEST_ENABLE
	23:20	**Attribute 05 Wrap Shortest Enables** Format: WRAP_SHORTEST_ENABLE
	19:16	**Attribute 04 Wrap Shortest Enables** Format: WRAP_SHORTEST_ENABLE
	15:12	**Attribute 03 Wrap Shortest Enables** Format: WRAP_SHORTEST_ENABLE

3DSTATE_SBE_SWIZ_BODY		
	11:8	**Attribute 02 Wrap Shortest Enables** Format: WRAP_SHORTEST_ENABLE
	7:4	**Attribute 01 Wrap Shortest Enables** Format: WRAP_SHORTEST_ENABLE
	3:0	**Attribute 00 Wrap Shortest Enables** Format: WRAP_SHORTEST_ENABLE

3DSTATE_SCISSOR_STATE_POINTERS_BODY

3DSTATE_SCISSOR_STATE_POINTERS_BODY		
Source:	RenderCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:5	**Scissor Rect Pointer** Format: DynamicStateOffset[31:5]SCISSOR_RECT*16 Specifies the 32-byte aligned address offset of the SCISSOR_RECT state. This offset is relative to the **Dynamic State Base Address**.
	4:0	**Reserved** Access: RO Format: MBZ

3DSTATE_SF_BODY

3DSTATE_SF_BODY		
Source:	RenderCS	
Size (in bits):	96	
Default Value:	0x00000000, 0x00000000, 0x00000800	
DWord	**Bit**	**Description**
0	31:30	**Reserved** Access: RO Format: MBZ
	29:12	**Line Width** Format: U11.7 Range: [0.0, 2047.9921875] Controls width of line primitives. Setting a Line Width of 0.0 specifies the rasterization of the "thinnest" (one-pixel-wide), non-antialiased lines. Note that this effectively overrides the effect of AAEnable (though the AAEnable state variable is not modified). **Programming Notes** Software must not program a value of 0.0 when running in MSRASTMODE_ON_xxx modes - zero-width lines are not available when multisampling rasterization is enabled.
	11	**Legacy Global Depth Bias Enable** Format: Enable Enables the SF to use the Global Depth Offset Constant state unmodified. If this bit is not set, the SF will scale the Global Depth Offset Constant as described in section Error! Reference source not found. of this document. **Programming Notes** This bit should be set whenever non zero depth bias (Slope, Bias) values are used. Setting this bit may have some degradation of performance for some workloads.
	10	**Statistics Enable** Format: Enable If ENABLED, this FF unit will increment CL_PRIMITIVES_COUNT on behalf of the CLIP stage. If DISABLED, CL_PRIMITIVES_COUNT will be left unchanged. **Programming Notes** This bit should be set whenever clipping is enabled and the Statistics Enable bit is set in CLIP_STATE. It should be cleared if clipping is disabled or Statistics Enable in CLIP_STATE is clear.
	9:2	**Reserved** Access: RO Format: MBZ

3DSTATE_SF_BODY

	1	**Viewport Transform Enable** Format: Enable This bit controls the Viewport Transform function.
	0	**Reserved** Access: RO Format: MBZ
1	31	**Reserved** Access: RO Format: MBZ
	30:29	**Deref Block Size** Format: U2 Set the EODB batch size

Value	Name	Description
0h	Default Mode. Block deref size 32 **[Default]**	Block deref size of 32
1h	Per Poly Deref Mode when VS/DS have minimal handles.	Per Poly. Deref mode when VS/DS have minimal handles.
2h	Block deref size 8	8 handles mode, block size.
3h	Reserved	

Programming Notes

Deref Block size depends on the last enabled shader and number of handles programmed for that shader
1) For GS last shader enabled cases, the deref block is always set to a per poly(within hardware) If the last enabled shader is VS or DS.
1) If DS is last enabled shader then if the number of DS handles is less than 324, need to set per poly deref.
2) If VS is last enabled shader then if the number of VS handles is less than 192, need to set per poly deref

	28	**Reserved**
	27:18	**Reserved** Access: RO Format: MBZ
	17:16	**Line End Cap Antialiasing Region Width** Format: U2 This field specifies the distances over which the coverage of anti-aliased line end caps are computed.

3DSTATE_SF_BODY

		Value / **Name** / **Description** 0h / 0.5 pixels / 0.5 pixels 1h / 1.0 pixels / 1.0 pixels 2h / 2.0 pixels / 2.0 pixels 3h / 4.0 pixels / 4.0 pixels
	15:14	**Reserved** Access: RO Format: MBZ
	13	**Reserved**
	12	**Reserved**
	11:0	**Reserved** Access: RO Format: MBZ
2	31	**Last Pixel Enable** Format: Enable If ENABLED, the last pixel of a diamond line will be lit. This state will only affect the rasterization of Diamond lines (will not affect wide lines or anti-aliased lines). **Programming Notes** Last pixel is applied to all lines of a LINELIST, and only the last line of a LINESTRIP.
	30:29	**Triangle Strip/List Provoking Vertex Select** Format: U2 Selects which vertex of a triangle (in a triangle strip or list primitive) is considered the "provoking vertex". Used for flat shading of primitives. Does current implementation send provoking vertex first? **Value** / **Name** 0h / 0 1h / 1 2h / 2 3h / Reserved
	28:27	**Line Strip/List Provoking Vertex Select** Format: U2 Selects which vertex of a line (in a line strip or list primitive) is considered the "provoking vertex". **Value** / **Name** / **Description** 0h / 0 / Vertex 0 1h / 1 / Vertex 1 2h / Reserved / Reserved 3h / Reserved / Reserved

3DSTATE_SF_BODY

26:25 Triangle Fan Provoking Vertex Select

Format:	U2

Selects which vertex of a triangle (in a triangle fan primitive) is considered the "provoking vertex".

Value	Name
0h	0
1h	1
2h	2
3h	Reserved

24:15 Reserved

Access:	RO
Format:	MBZ

14 AA Line Distance Mode

Format:	U1

This bit controls the distance computation for antialiased lines.

Value	Name	Description
1h	AALINEDISTANCE_TRUE	True distance computation. This is the normal setting which should yield WHQL compliance.

13 Smooth Point Enable

Format:	Enable
_Custom_Display_DoubleBufferArmedBy:	Enables logic to draw smooth OGL Points

Programming Notes
If Enabled, SF will treat points in the same fashion that AA lines are processed

12 Vertex Sub Pixel Precision Select

Format:	U1

Selects the number of fractional bits maintained in the vertex data

Value	Name	Description
0h	8	8 sub pixel precision bits maintained
1h	4	4 sub pixel precision bits maintained

Programming Notes
When Conservative Rasterization is enabled, this bit must be programmed to 0.

11 Point Width Source

Controls whether the point width passed on the vertex or from state is used for rendering point primitives.

Value	Name	Description
0h	Vertex	Use Point Width on Vertex

3DSTATE_SF_BODY				
		1h	State **[Default]**	Use Point Width from State
	10:0	**Point Width** Format: U8.3 Range: [0.125, 255.875] pixels This field specifies the size (width) of point primitives in pixels. This field is overridden (though not overwritten) whenever point width information is passed in the FVF		

3DSTATE_SO_BUFFER_INDEX_BODY

3DSTATE_SO_BUFFER_INDEX_BODY

Source: RenderCS

Size (in bits): 224

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31	**SO Buffer Enable** Format: Enable If set, stream output to SO Buffer is enabled, , if 3DSTATE_STREAMOUT::SO Function ENABLE is also enabled..If clear, the SO Buffer is considered "not bound" and effectively treated as a zero-length buffer for the purposes of SO output and overflow detection. If an enabled stream's Stream to Buffer Selects includes this buffer it is by definition an overflow condition. That stream will cause no writes to occur, and only SO_PRIM_STORAGE_NEEDED[<stream>] will increment.
	30:29	**Reserved** Access: RO Format: MBZ
	28:22	**SO Buffer Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for the SO buffer.
	21	**Stream Offset Write Enable** Format: Enable When set, this field allows the hardware to write SO_WRITE_OFFSET[n] as specified in the Stream Offset field. **Programming Notes** The field is operates irrespective of whether SO Buffer Enable is set or clear.
	20	**Stream Output Buffer Offset Address Enable** Format: Enable When set, this field allows the hardware to read/write the stream output buffer offset as specified in the "Stream Output Buffer Offset Address" field. **Programming Notes** The field is operates irrespective of whether SO Buffer Enable is set or clear.
	19:0	**Reserved** Access: RO Format: MBZ

3DSTATE_SO_BUFFER_INDEX_BODY

1..2	63:2	**Surface Base Address** Format: VIRTUAL_ADDR[63:2] **Description** VIRTUAL_ADDR[63:48] are ignored by the HW This field specifies the starting address of the buffer in Graphics Memory.
	1:0	**Reserved** Access: RO Format: MBZ
3	31:30	**Reserved** Access: RO Format: MBZ
	29:0	**Surface Size** Format: U30-1 This field specifies the size of buffer in number DWords minus 1 of the buffer in Graphics Memory.
4..5	63:2	**Stream Output Buffer Offset Address** Format: VIRTUAL_ADDR[63:2] **Description** VIRTUAL_ADDR[63:48] are ignored by the HW This field specifies the starting address of the buffer in Graphics Memory where the Stream Output Buffer Offset is stored when all the data has been written. It is also used to fetch the stream Output buffer Offset when needed.
	1:0	**Reserved** Access: RO Format: MBZ
6	31:0	**Stream Offset** This field specifies the Offset in stream output buffer to start at, or whether to append to the end of an existing buffer. The Offset must be DWORD aligned. If Stream Offset is equal to 0xFFFFFFFF then load the value at the Stream Output Buffer Offset address into SO_WRITE_OFFSET[Buffer#]. Otherwise, SO_WRITE_OFFSET[n] = Stream Offset.

3DSTATE_STENCIL_BUFFER_BODY

3DSTATE_STENCIL_BUFFER_BODY

Source: RenderCS

Size (in bits): 224

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:29	**Surface Type** 0h Reserved 1h SURFTYPE_2D — Defines a 2-dimensional map or array of maps 2h Reserved 3h SURFTYPE_CUBE — Defines a cube map 4h-6h Reserved 7h SURFTYPE_NULL — Defines a null surface **Programming Notes** The Surface Type of the Stencil buffer must be the same as the Surface Type of the 1. Render target(s) (defined in SURFACE_STATE), unless either the stencil buffer or render targets are SURFTYPE_NULL 2. Depth buffer (defined in 3DSTATE_DEPTH_BUFFER) unless either the depth buffer or Stencil buffer surf_type are SURFTYPE_NULL. If stencil is enabled with 1D render target, stencil surface type needs to be set to 2D surface type and height set to 1. For this case only, the Surface Type of the stencil buffer can be 2D while the Surface Type of the render target(s) are 1D, representing an exception to a programming note above. Issue Semi pipelined flush not back pressuring when stencil buffer state is enabling thread dispatch. Workaround **Astep only** An **additional pipe control with post-sync = store dword** operation would be required.(w/a is to have an **additional** pipe control after the stencil state whenever the surface state bits of this state is changing).
	28	**Stencil Write Enable** Format: Enable This field enables stencil writes to the Stencil buffer surface. Both this field and the Stencil Buffer Write Enable field in DEPTH_STENCIL_STATE must be enabled in order for stencil writes to occur.
	27	**Null Page Coherency Enable** Format: Enable This field is used for enabling NULL coherency as defined under Tiled Resources.

3DSTATE_STENCIL_BUFFER_BODY

Value	Name
1	Enable
0	Disable **[Default]**

Programming Notes
SW must enable this bit only if Tiled Resource is enabled

26 **Reserved**

Access:	RO
Format:	MBZ

25 **Stencil Compression Enable**

if enabled, indicates that Stencil Buffer Compression is Enabled
When enabled, Stencil Buffer needs to be initialized via stencil clear (HZ_OP) before any renderpass.

Programming Notes
SW must set this bit if the Stencil Control surface enable is also set. The Stencil surface control enable is in Bit[24] of this DWORD.

24 **Control Surface Enable**

If set to 1, it indicates if the common control surface is present. The read and write transaction opcodes sent by the clients (HZ, Z, STC) to the fabric are different depending on the control surface. If the control surface is not present, the reads and writes are in legacy mode. If the control surface is present, the reads and write opcodes will be either UNCOMPRESSED_TYP for uncompressible transactions or COMPRESSED_TYP for compressible transactions.

Programming Notes
SW must set this bit to "1", if the common control surface is present in the system.

23 **Corner Texel Mode**

Format:	Enable

This field, when ENABLED, indicates when a surface is using corner texel-mode for stencil surface. This bit changes how the size of each MIP when calculating the offset within a surface.

Value	Name	Description
0h	Disable **[Default]**	Corner Texel mode is not enabled.
1h	Enable	Corner Texel Mode is enabled.

Programming Notes
Corner texel mode for the stencil buffer must be the same as the Corner texel mode of the 1. Render target(s) (defined in SURFACE_STATE), unless either the stencil buffer or render targets are SURFTYPE_NULL 2. Depth buffer (defined in 3DSTATE_DEPTH_BUFFER) unless either the depth buffer or Stencil buffer surf_type are SURFTYPE_NULL

3DSTATE_STENCIL_BUFFER_BODY

22:18 Reserved

Access:	RO
Format:	MBZ

17 Reserved

Access:	RO
Format:	MBZ

16:0 Surface Pitch

Format:	U17-1

For TileYF and TileYS surfaces, the range is dependent on the Cu parameter (refer to *Memory Data Formats* section for the definition of the Cu parameter depending on the case). The range in bytes is [2^{Cu}-1,262143] -> [(2^{Cu})B,256KB] = [1 tile, 256KB/(2^{Cu}) tiles]

Value	Name	Description
[7Fh,1FFFFh]		corresponding to [128B, 256KB] also restricted to a multiple of 128B

Programming Notes

The pitch specified must be a multiple of the tile pitch, in the range [128B, 128KB].
The minimum pitch should be calculated as per the formula given below.
The minimum pitch should be calculated based on Cu, Cv, W_L.
The Cu, Cv are the tile constants and $W_L$is the aligned width adjusted for MSAA.
use this for pitch formula :
Minimum_pitch =(ceiling((W_0 pixel_size) / (1 « Cu)) *(1 « Cu)) 1 ; //$W_0$is the aligned width for the largest LOD (i.e LOD 0)*
(1 « Cu) = tile width in bytes
(1 « Cv) = tile height in lines
Pixel_size = 1 (for STC buffer)

1..2 63:0 Surface Base Address

Format:	GraphicsAddress[63:0]

This field specifies address of the buffer in mapped Graphics Memory. Graphics Address [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] = [47].

Programming Notes

The stencil Buffer can only be mapped to Main Memory (uncached). If the surface is tiled, the base address must conform to the Per-Surface Tiling Alignment. If the buffer is linear, the surface must be 64-byte aligned.

If the buffer is linear, the surface must be 64-byte aligned.

3 31 Reserved

Access:	RO
Format:	MBZ

3DSTATE_STENCIL_BUFFER_BODY

30:17 **Height**

Format:	U14-1

This field specifies the height of the surface. If the surface is MIP-mapped, this field contains the height of the base MIP level.

Value	Name	Description	Exists If
[0,16383]	Legal Range	Height of surface - 1 (y/v dimension)	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_2D')
[0,16383]	Legal Range	y/v dimension	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_CUBE')

Programming Notes

The Height of the stencil buffer must be the same as the
1. Height of the render target(s) (defined in SURFACE_STATE), unless Surface Type is SURFTYPE_2D with Depth = 0 (non-array) and LOD = 0 (non-mip mapped).
2.Depth buffer (defined in 3DSTATE_DEPTH_BUFFER) unless either the depth buffer or Stencil buffer surf_typeare SURFTYPE_NULL

16 **Reserved**

Access:	RO
Format:	MBZ

15 **Reserved**

Access:	RO
Format:	MBZ

14:1 **Width**

Format:	U14-1

This field specifies the width of the surface. If the surface is MIP-mapped, this field specifies the width of the base MIP level. The width is specified in units of pixels.

Value	Name	Description	Exists If
[0,16383]	Legal Range	Width of surface - 1 (x/u dimension)	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_2D')
[0,16383]	Legal Range	Width of surface - 1 (x/u dimension)	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_CUBE')

Programming Notes

The Width specified by this field must be less than or equal to the surface pitch (specified in bytes via the Surface Pitch field). For cube maps, Width must be set equal to Height.
1. The Width of the stencil buffer must be the same as the Width of the render target(s) (defined in SURFACE_STATE), unless Surface Type is SURFTYPE_1D or SURFTYPE_2D with Depth = 0 (non-array) and LOD = 0 (non-mip mapped).
2.Depth buffer (defined in 3DSTATE_DEPTH_BUFFER) unless either the depth buffer or Stencil buffer surf_typeare SURFTYPE_NULL

3DSTATE_STENCIL_BUFFER_BODY

0 — Reserved

Access:	RO
Format:	MBZ

4 — 31 — Reserved

Access:	RO
Format:	MBZ

4 — 30:20 — Depth

Format:	U11-1

This field specifies the total number of levels for a volume texture or the number of array elements allowed to be accessed starting at the Minimum Array Element for arrayed surfaces. If the volume texture is MIP-mapped, this field specifies the depth of the base MIP level.

Value	Name	Description	Exists If
[0,2047]	Legal Range	Number of array elements - 1	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_2D')
[0,0]	Legal Range	Must be zero	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_CUBE')

Programming Notes

The Depth of the Stencil buffer must be the same as
1. The Depth of the render target(s) (defined in SURFACE_STATE).
2.Depth buffer (defined in 3DSTATE_DEPTH_BUFFER) unless Depth buffer surf_type is SURFTYPE_NULL

4 — 19 — Reserved

Access:	RO
Format:	MBZ

4 — 18:8 — Minimum Array Element

Format:	U11

For 2D Surfaces: This field indicates the minimum array element that can be accessed as part of this surface. The delivered array index is added to this field before being used to address the surface. For Other Surfaces This field is ignored

Value	Name	Exists If
[0,2047]	SURFTYPE_2D	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_1D'|Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_2D')

Programming Notes

Minimum array element of the Stencil buffer must be the same as the Surface Type of the
1. Render target(s) (defined in SURFACE_STATE), unless either the Stencil buffer or render targets are SURFTYPE_NULL
2. Depth buffer (defined in 3DSTATE_DEPTH_BUFFER) unless either the depth buffer or Stencil

3DSTATE_STENCIL_BUFFER_BODY

buffer surf_type are SURFTYPE_NULL

7 Reserved

Access:	RO
Format:	MBZ

6:0 Stencil Buffer Object Control State

Format:	MEMORY_OBJECT_CONTROL_STATE

Specifies the memory object control state for the stencil buffer.

5 31:30 Tiled Mode

For stencil Buffer Surfaces: This field specifies the tiled mode. For other surfaces: This field is ignored.

Value	Name	Description
0h	NONE	No tiled modes (TileYf, TileYs). Use Standard TileY
1h	TILEYF	4KB tiled mode
2h	TILEYS	64KB tiled mode
3h	Reserved	

Programming Notes

If **Tile Mode** is not set to TILEMODE_YMAJOR, this field must be set to TRMODE_NONE.

29:26 Mip Tail Start LOD

Format:	U4

For Sampling Engine, Render Target, and Typed Surfaces: This field indicates which LOD is the first one in the MIP tail if **Tiled Mode** is not TRMODE_NONE. The MIP tail has a different layout than the rest of the surface. Refer to the *Memory Data Formats* section for more details.
For other surfaces: This field is ignored.

Programming Notes

This field must be zero if the **Surface Format** is MONO8.
This field is ignored if **Tiled Mode** is TRMODE_NONE unless **Surface Type**is SURFTYPE_1D.

If Tiled Mode is not TRMODE_NONE, this field must be set to ensure that mips within the mip tail do not overlap given the storage algorithms given in the Memory Data Formats section. The following table indicates the maximum size of the mip that is set to be the Mip Tail Start LOD for various cases:

Tiling Mode	Slot Size in Bytes	8-bit Size
2D TIleYs 1x	32KB	(128, 256)
2D TileYf 1x	2KB	(32, 64)

25:6 Reserved

Access:	RO
Format:	MBZ

3DSTATE_STENCIL_BUFFER_BODY

	5:0	**Reserved** Access: RO Format: MBZ
6	31:21	**Render Target View Extent** Format: U11-1 **Value** / **Name** / **Description** / **Exists If** [0,2047] / Legal Range / Number of array elements- 1 / (Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_2D') [0,0] / Legal Range / Must be zero / (Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_CUBE') **Programming Notes** Render Target View Extof the stencil buffer must be the same as the Surface Type of the 1. Render target(s) (defined in SURFACE_STATE), unless either the depth buffer or render targets are SURFTYPE_NULL 2. Depth buffer (defined in 3DSTATE_DEPTH_BUFFER) unless either the depth buffer or Stencil buffer surf_type are SURFTYPE_NULL
	20	**Reserved** Access: RO Format: MBZ
	19:16	**Surf LOD** Format: U4 LOD units **Value** / **Name** [0-14] **Programming Notes** Surf LOD of the stencil buffer must be the same as the Surface Type of the 1. Render target(s) (defined in SURFACE_STATE), unless either the stencil buffer or render targets are SURFTYPE_NULL 2. Depth buffer (defined in 3DSTATE_DEPTH_BUFFER) unless either the depth buffer or Stencil buffer surf_type are SURFTYPE_NULL
	15	**Reserved** Access: RO Format: MBZ

3DSTATE_STENCIL_BUFFER_BODY

14:0 **Surface QPitch**

Format:	U17[16:2]

The interpretation of this field is dependent on **Surface Type** as follows:

- SURFTYPE_2D/CUBE: distance in *rows* between array slices.

Other surface types: field is ignored

Format:

```
QPitch[16:2]
```

Value	Name	Description
[1h,7FFFh]		in multiples of 4 (low 2 bits missing)

Programming Notes

For 2D Surfaces: This field must be set to an integer multiple of 8. Software must ensure that this field is set to a value sufficiently large that array slices in the surface do not overlap. Refer to the Memory Data Formats section for information on how surfaces are stored.
TYS/TYF QPitch is valid only for 2D array surfaces and represents the tile-padded total number of texels(lines) in a single array slice.
Height of each LOD:
HL = AlignToTileHeight(MSAA_height_factor * (**height**»L) > 0?**height**»L : 1), where AlignToTileHeight(x) is (ceiling((x) / (1 « Cv)) *(1 « Cv))
Height of all LODs is a sum:
H = H0+H1+..Hn,
N is number of mip levels.
If surface has MIP tail, equation stops at Hn where n=MipTailStartLOD. MipTail is single tile.
QPitch is multiple of tile height (1 « Cv) and should be equal or greater H computed above.

3DSTATE_STREAMOUT_BODY

3DSTATE_STREAMOUT_BODY		
Source:	RenderCS	
Size (in bits):	128	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0	31	**SO Function Enable** Format: U1 If set, the SO function is enabled. Vertex data will be streamed out to memory (subject to overflow detection) as controlled by the various SO-related state variables. If clear, the SO function is disabled, and therefore no vertex data will be streamed out to memory. However, the Rendering Disable and Render Stream Select fields will still be used to determine which vertices (if any) are forwarded down the pipeline for (possible) rendering.
	30	**API Rendering Disable** Format: U1 If set, Indicates the API wants the SO stage not to forward any topologies down the pipeline. If clear, Indicates the API wants the SO stage to forward topologies associated with **Render Stream Select** down the pipeline. This bit is used even if **SO Function Enable** is DISABLED. **Programming Notes** The SOL unit generates an SOL_INT::Render_Enable which ultimately controls whether rendering occurs or not.
	29	**Reserved** Access: RO Format: MBZ
	28:27	**Render Stream Select** Format: U2 This field specifies which stream has been selected to be forwarded down the pipeline for possible rendering. Topologies from other streams will not be passed down the pipeline. If Rendering Disable is set, this field is ignored, as no topologies are sent down the pipeline. SO Function Enable must also be ENABLED in order for this field to select a stream for rendering. When **SO Function Enable** is DISABLED and Rendering Disable is cleared (i.e., rendering is enabled), StreamID is ignored downstream of the SO stage, allowing any stream to be rendered.
	26	**Reorder Mode** This bit controls how vertices of triangle objects in TRISTRIP[_ADJ] and TRISTRIP_REV are reordered for the purposes of stream-out only (does not impact rendering). See table in Input Buffering.

3DSTATE_STREAMOUT_BODY

Value	Name	Description
0h	LEADING	Reorder the vertices of alternating triangles of a TRISTRIP[_ADJ] such that the leading (first) vertices are in consecutive order starting at v0. A similar reordering is performed on alternating triangles in a TRISTRIP_REV.
1h	TRAILING	Reorder the vertices of alternating triangles of a TRISTRIP[_ADJ] such that the trailing (last) vertices are in consecutive order starting at v2. A similar reordering is performed on alternating triangles in a TRISTRIP_REV.

25 **SO Statistics Enable**

Format:	Enable

This bit controls whether StreamOutput statistics register(s) can be incremented.

Value	Name	Description
0h	Disable	SO_NUM_PRIMS_WRITTEN[0..3] and SO_PRIM_STORAGE_NEEDED[0..3] registers cannot increment.
1h	Enable	SO_NUM_PRIMS_WRITTEN[0..3] and SO_PRIM_STORAGE_NEEDED[0..3] registers can increment.

24:23 **Force Rendering**

This field provides a work around override for the computation of SOL_INT::Render_Enable

Value	Name	Description
0h	Normal	SOL_INT::Render_Enable is computed normally
1h	Resreved	
2h	Force_Off	Forces the rendering to be disabled.
3h	Force_on	Forces the rendering to be enabled.

22:0 **Reserved**

Access:	RO
Format:	MBZ

1 31:30 **Reserved**

Access:	RO
Format:	MBZ

29 **Stream 3 Vertex Read Offset**

Format:	U1

Specifies amount of data to skip over before reading back Stream 3 vertex data. (See **Stream 0 Vertex Read Offset**)

28:24 **Stream 3 Vertex Read Length**

Format:	U5-1

(See Stream 0 Vertex Read Length)

23:22 **Reserved**

Access:	RO
Format:	MBZ

3DSTATE_STREAMOUT_BODY

	21	**Stream 2 Vertex Read Offset** Format: U1 Specifies amount of data to skip over before reading back Stream 2 vertex data. (See Stream 0 Vertex Read Offset)
	20:16	**Stream 2 Vertex Read Length** Format: U5-1
	15:14	**Reserved** Access: RO Format: MBZ
	13	**Stream 1 Vertex Read Offset** Format: U1 Specifies amount of data to skip over before reading back Stream 1 vertex data. (See Stream 0 Vertex Read Offset)
	12:8	**Stream 1 Vertex Read Length** Format: U5-1 (See Stream 0 Vertex Read Length)
	7:6	**Reserved** Access: RO Format: MBZ
	5	**Stream 0 Vertex Read Offset** Format: U1 Specifies amount of data to skip over before reading back Stream 0 vertex data. Must be zero if the GS is enabled and the Output Vertex Size field in 3DSTATE_GS is programmed to 0 (i.e., one 16B unit).
	4:0	**Stream 0 Vertex Read Length** Format: U5-1 Specifies amount of vertex data to read back for Stream 0 vertices, starting at the Stream 0 Vertex Read Offset location. Maximum readback is 17 256-bit units (34 128-bit vertex attributes). Read data past the end of the valid vertex data has undefined contents, and therefore shouldn't be used to source stream out data. Must be zero (i.e., read length = 256b) if the GS is enabled and the Output Vertex Size field in 3DSTATE_GS is programmed to 0 (i.e., one 16B unit).
2	31:28	**Reserved** Access: RO Format: MBZ
	27:16	**Buffer 1 Surface Pitch**

3DSTATE_STREAMOUT_BODY

	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Buffer 0 Surface Pitch** Format: U12 This field specifies the pitch of the SO buffer in #Bytes. **Value** / **Name** [0,2048] Must be 0 or a multiple of 4 Bytes. **Programming Notes** A Surface Pitch of 0 indicates an un-bound buffer. No writes are performed. Surface Base Address is ignored.
3	31:28	**Reserved** Access: RO Format: MBZ
	27:16	**Buffer 3 Surface Pitch** Format: U12
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Buffer 2 Surface Pitch** Format: U12

3DSTATE_TE_BODY

3DSTATE_TE_BODY

Source: RenderCS

Size (in bits): 128

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:24</td><td>Reserved
Access: RO
Format: MBZ</td></tr>
<tr><td></td><td>23:22</td><td>Patch Header Layout
This field describes the layout of the tessellation factor DWORDS in the patch header. The layout depends on the value of this field and the TE Domain.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th><th>Programming Notes</th></tr>
<tr><td>0h</td><td>LEGACY</td><td>
<table>
<tr><th></th><th>DW7</th><th>DW6</th><th>DW5</th><th>DW4</th><th>DW3</th><th>DW2</th><th>DW1</th><th>DW0</th></tr>
<tr><td>QUAD</td><td>UEQ0</td><td>UEQ1</td><td>VEQ0</td><td>VEQ1</td><td>InsideU</td><td>InsideV</td><td>-</td><td>-</td></tr>
<tr><td>TRI</td><td>UEQ0</td><td>VEQ0</td><td>WEQ0</td><td>Inside</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>ISOLINE</td><td>LineDetail</td><td>LineDensity</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
</table>
</td><td></td></tr>
<tr><td>2h</td><td>REVERSED</td><td>
<table>
<tr><th></th><th>DW7</th><th>DW6</th><th>DW5</th><th>DW4</th><th>DW3</th><th>DW2</th><th>DW1</th><th>DW0</th></tr>
<tr><td>QUAD</td><td>-</td><td>-</td><td>InsideV</td><td>InsideU</td><td>VEQ1</td><td>VEQ0</td><td>UEQ1</td><td>UEQ0</td></tr>
<tr><td>TRI</td><td>-</td><td>-</td><td>-</td><td>-</td><td>Inside</td><td>WEQ0</td><td>VEQ0</td><td>UEQ0</td></tr>
<tr><td>ISOLINE</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>LineDensity</td><td>LineDetail</td></tr>
</table>
</td><td></td></tr>
<tr><td>3h</td><td>REVERSED_TRI_INSIDE_SEPARATE</td><td>
<table>
<tr><th></th><th>DW7</th><th>DW6</th><th>DW5</th><th>DW4</th><th>DW3</th><th>DW2</th><th>DW1</th><th>DW0</th></tr>
<tr><td>TRI</td><td>-</td><td>-</td><td>-</td><td>Inside</td><td>-</td><td>WEQ0</td><td>VEQ0</td><td>UEQ0</td></tr>
</table>
</td><td>This layout may only be used with a TE Domain of TRI.</td></tr>
</table>
</td></tr>
<tr><td></td><td>21</td><td>Reserved
Access: RO
Format: MBZ</td></tr>
</table>

3DSTATE_TE_BODY

20 Tessellation Factor Format

Value	Name	Description
0h	FLOAT32	The tessellation factors in the patch header are in a FLOAT32 format.
1h	FLOAT16	The tessellation factors in the patch header are in a FLOAT16 format. The tessellation factors still occupy the same DWORD as with the FLOAT32 format, but occupy only bits [15:0] of the DWORD.

19 Tessellation Scale Factor Enable

Format:	Enable

If ENABLED, the tessellation factors will be multiplied by the Tessellation Scale Factor.

Programming Notes
Note that if ENABLED, the Tessellation Factor Format must be FLOAT16.

18:17 Reserved

Access:	RO
Format:	MBZ

16 Reserved

Access:	RO
Format:	MBZ

15:14 Reserved

Access:	RO
Format:	MBZ

13:12 Partitioning

Format:	U2

This field specifies how edges are partitioned based on tessellation factor.

Value	Name	Description
0h	INTEGER	Outside/inside edges are divided into an integer number of equal-sized segments.
1h	ODD_FRACTIONAL	Outside/inside edges are divided into an odd number of possibly-unequal-sized segments.
2h	EVEN_FRACTIONAL	Outside/inside edges are divided into an even number of possibly-unequal-sized segments.
3h	POW2	Outside/inside edges are divided into a power of 2 number of equal-sized segments.

11:10 Reserved

Access:	RO
Format:	MBZ

3DSTATE_TE_BODY

9:8 Output Topology

Format:	U2

This field specifies which primitive types are to be output.

Value	Name	Description
0h	POINT	Points are output (as POINTLIST topologies)
1h	LINE	Lines are output (as LINESTRIP topologies). Only valid if ISOLINE domain is selected.
2h	TRI_CW	Clockwise-ordered triangles are output (either as TRISTRIP, TRISTRIP_REV or TRILIST topologies). Not valid if ISOLINE domain is selected.
3h	TRI_CCW	Count-clockwise-ordered triangles are output (either as TRISTRIP, TRISTRIP_REV or TRILIST topologies). Not valid if ISOLINE domain is selected.

7:6 Reserved

Access:	RO
Format:	MBZ

5:4 TE Domain

Format:	U2

This field specifies which type of domain is to be tessellated.

Value	Name	Description
0h	QUAD	2D (U, V) domain is tessellated
1h	TRI	Triangular (U, V, W) domain is tessellated
2h	ISOLINE	2D (U, V) domain is tessellated.

3 Reserved

Access:	RO
Format:	MBZ

2:1 TE Mode

Format:	U2

When TE Enable is ENABLED, this field specifies the overall operation of the TE stage. This field is ignored if TE Enable is DISABLED.

Value	Name	Description
0h	HW_TESS	Normal HW Tessellation Mode. The TessFactors are read from the patch URB entry, and are used to perform fixed-function hardware tessellation of the specified domain.

3DSTATE_TE_BODY

	0	**TE Enable** Format: Enable If ENABLED, the TE stage will perform tessellation processing on incoming patch primitives. The TE Mode field determines how this tessellation operation proceeds. If DISABLED, the TE goes into pass-through mode. All other state fields are ignored. **Programming Notes** The tessellation stages (HS, TE and DS) must be enabled/disabled as a group. I.e., draw commands can only be issued if all three stages are enabled or all three stages are disabled, otherwise the behavior is UNDEFINED.
1	31:0	**Maximum Tessellation Factor Odd** Format: IEEE_FLOAT This field specifies the maximum TessFactor for ODD_FRACTIONAL partitioning when in HW_TESS mode. **Value** / **Name** / **Description** [3F800000h,427C0000h] / [1,63] / Value can be set between [1,63]. Value must be a IEEE_Float representation of an odd integer. **Programming Notes** Note that ISOLINE's LineDensity TF is always subjected to INTEGER partitioning regardless of the Partitioning state.
2	31:0	**Maximum Tessellation Factor Not Odd** Format: IEEE_FLOAT This field specifies the maximum TessFactor for EVEN_FRACTIONAL, INTEGER or POW2 partitioning when in HW_TESS mode. **Value** / **Name** / **Description** [40000000h,42800000h] / [2,64] / Value can be set between [2,64]. Value must be a IEEE_Float representation of an even integer. **Programming Notes** Note that ISOLINE's LineDensity TF is always subjected to INTEGER partitioning regardless of the Partitioning state. If Partioning is set to POW2, this field must be programmed to a power of 2 number.
3	31:0	**Tessellation Scale Factor** Format: IEEE_FLOAT If Tessellation Scale Factor Enable is ENABLED, the tessellation factors in the patch header will be multiplied by this value. **Programming Notes** This FLOAT32 value will be converted to a FLOAT16 value by hardware prior to scaling. This FLOAT32 value must **NOT** be negative, zero, denormal, infinite, or a NaN.

3DSTATE_URB_ALLOC_DS_BODY

3DSTATE_URB_ALLOC_DS_BODY

Source: RenderCS

Size (in bits): 64

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:21	**DS URB Starting Address SliceN** Format: U8 This field specifies the offset (from the start of the URB memory in additional slices) of the DS URB allocation, specified in multiples of 8 KB. For each additional, enabled slice, HW will increase the DS URB Starting Address by the value specified, resulting in an DS URB Starting Address within the total URB space. Value: [0,127] Name: Programming Notes: This field is ignored by HW as there is only one slice in the device.
	20:18	**Reserved** Access: RO Format: MBZ
	17:10	**DS URB Starting Address Slice0** Format: U8 This field specifies the offset (from the start of Slice0 URB memory) of the DS URB Handle allocation for Slice0, specified in multiples of 8 KB. This address must account for any Push Constant allocations, as those allocations begin at offset 0 in URB memory. Value: [0,127] Programming Notes: The DS URB Starting Address Slice 0 must be greater than the render and posh push constant space allocated using3DSTATE_PUSH_CONSTANT_ALLOC_VS,3DSTATE_PUSH_CONSTANT_ALLOC_HS,3DSTATE_PUSH_CONSTANT_ALLOC_DS,3DSTATE_PUSH_CONSTANT_ALLOC_GS,3DSTATE_PUSH_CONSTANT_ALLOC_PS and VSR_PUSH_CONSTANT_BASE.
	9:0	**DS URB Entry Allocation Size** Format: U10-1 Specifies the size, count of 512-bit units, of each URB entry owned by DS.

3DSTATE_URB_ALLOC_DS_BODY

		Value / **Name** [0,511]
1	31:16	**DS Number of URB Entries SliceN** Format: U16 This field specifies the number of URB entries in slices beyond Slice0 to be allocated to DS. **Value** / **Name** [0,3576] **Programming Notes** This field is ignored by HW as there is only one slice in the device. DS Number of URB Entries must be divisible by 8 if the DS URB Entry Allocation Size is less than 9 512-bit URB entries."2:0" = reserved "000b"
	15:0	**DS Number of URB Entries Slice0** Format: U16 This field specifies the number of URB entries in Slice0 URB memory to be allocated to DS. **Value** / **Name** [0,3576] **Programming Notes** DS Number of URB Entries must be divisible by 8 if the DS URB Entry Allocation Size is less than 9 512-bit URB entries."2:0" = reserved "000b"

3DSTATE_URB_ALLOC_GS_BODY

3DSTATE_URB_ALLOC_GS_BODY

Source: RenderCS

Size (in bits): 64

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:21	**GS URB Starting Address SliceN** Format: U8 This field specifies the offset (from the start of the URB memory in slices beyond Slice0) of the GS URB allocation, specified in multiples of 8 KB. Value: [0,127] Programming Notes: This field is ignored by HW as there is only one slice in the device.
	20:18	**Reserved** Access: RO Format: MBZ
	17:10	**GS URB Starting Address Slice0** Format: U8 This field specifies the offset (from the start of Slice0 URB memory) of the GS URB allocation, specified in multiples of 8 KB. This address must account for any Push Constant allocations, as those allocations begin at offset 0 in Slice0 URB memory. Value: [0,127] Programming Notes: The GS URB Starting Address Slice 0 must be greater than the render and posh push constant space allocated using3DSTATE_PUSH_CONSTANT_ALLOC_VS,3DSTATE_PUSH_CONSTANT_ALLOC_HS,3DSTATE_PUSH_CONSTANT_ALLOC_DS,3DSTATE_PUSH_CONSTANT_ALLOC_GS,3DSTATE_PUSH_CONSTANT_ALLOC_PS andVSR_PUSH_CONSTANT_BASE.
	9:0	**GS URB Entry Allocation Size** Format: U10-1 Specifies the size,count of 512-bit units, of each URB entry owned by GS in units of 64B. Value: [0,511]

3DSTATE_URB_ALLOC_GS_BODY

1	31:16	**GS Number of URB Entries SliceN** Format: U16 This field specifies the number of URB entries in slices beyond Slice0 to be allocated to GS. SW shall ensure that the GS Number of URB Entries does not exceed the relevant ValidValue range listed below. **Value** / **Name** [0,1548] **Programming Notes** This field is ignored by single-slice devices. GS Number of URB Entries must be divisible by 8 if the GS URB Entry Allocation Size is less than 9 512-bit URB entries."2:0" = reserved "000b" This field is ignored by HW as there is only one slice in the device.
	15:0	**GS Number of URB Entries Slice0** Format: U16 This field specifies the number of URB entries in Slice0 URB memory to be allocated to GS. SW shall ensure that the GS Number of Entries does not exceed the relevant ValidValue range listed below. **Value** / **Name** [0,1548] **Programming Notes** GS Number of URB Entries must be divisible by 8 if the GS URB Entry Allocation Size is less than 9 512-bit URB entries."2:0" = reserved "000b"

3DSTATE_URB_ALLOC_HS_BODY

3DSTATE_URB_ALLOC_HS_BODY

Source: RenderCS
Size (in bits): 64
Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:21	**HS URB Starting Address SliceN** Format: U8 This field specifies the offset (from the start of the URB memory in slices beyond Slice0) of the HS URB allocation, specified in multiples of 8 KB. Value: [0,127] — Name: **Programming Notes** This field is ignored by HW as there is only one slice in the device.
	20:18	**Reserved** Access: RO Format: MBZ
	17:10	**HS URB Starting Address Slice0** Format: U8 This field specifies the offset (from the start of Slice0 URB memory) of the HS URB allocation, specified in multiples of 8 KB. This address must account for any Push Constant allocations, as those allocations begin at offset 0 in Slice0 URB memory. Value: [0,127] — Programming Notes: The VS URB Starting Address Slice 0 must be greater than the render and posh push constant space allocated using3DSTATE_PUSH_CONSTANT_ALLOC_VS,3DSTATE_PUSH_CONSTANT_ALLOC_HS,3DSTATE_PUSH_CONSTANT_ALLOC_DS,3DSTATE_PUSH_CONSTANT_ALLOC_GS,3DSTATE_PUSH_CONSTANT_ALLOC_PS and VSR_PUSH_CONSTANT_BASE.
	9:0	**HS URB Entry Allocation Size** Format: U10-1 Specifies the size,count of 512-bit units, of each URB entry owned by HS. Value: [0,511] — Name:

3DSTATE_URB_ALLOC_HS_BODY

1	31:16	**HS Number of URB Entries SliceN** Format: U16 This field specifies the number of URB entries in slices beyond Slice0 to be allocated to HS. SW shall ensure that the total HS Number of URB Entries does not exceed the relevant ValidValue range listed below. **Value** / **Name** [0,1548] **Programming Notes** This field is ignored by HW as there is only one slice in the device. HS Number of URB Entries must be divisible by 8 if the HS URB Entry Allocation Size is less than 9 512-bit URB entries."2:0" = reserved "000b"
	15:0	**HS Number of URB Entries Slice0** Format: U16 This field specifies the number of URB entries in Slice0 URB memory to be allocated to HS. SW shall ensure that the total HS Number of Entries does not exceed the relevant ValidValue range listed below. **Value** / **Name** [0,1548] **Programming Notes** HS Number of URB Entries must be divisible by 8 if the HS URB Entry Allocation Size is less than 9 512-bit URB entries."2:0" = reserved "000b"

3DSTATE_URB_ALLOC_VS_BODY

3DSTATE_URB_ALLOC_VS_BODY

Source: RenderCS

Size (in bits): 64

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:21	**VS URB Starting Address SliceN** Format: U8 This field specifies the offset (from the start of the URB memory in additional slices) of the VS URB allocation, specified in multiples of 8 KB. For each additional, enabled slice, HW will increase the VS URB Starting Address by the value specified, resulting in a Starting Address within the total URB space. Value: [0,127] Name: **Programming Notes** This field is ignored by HW as there is only one slice in the device.
	20:18	**Reserved** Access: RO Format: MBZ
	17:10	**VS URB Starting Address Slice0** Format: U8 This field specifies the offset (from the start of Slice0 URB memory) of the VS URB allocation, specified in multiples of 8 KB. This allocation must account for any Push Constant allocations, as those allocations begin at offset 0 in Slice0 URB memory. Value: [0,127] Programming Notes: The VS URB Starting Address Slice 0 must be greater than the render and posh push constant space allocated using3DSTATE_PUSH_CONSTANT_ALLOC_VS,3DSTATE_PUSH_CONSTANT_ALLOC_HS,3DSTATE_PUSH_CONSTANT_ALLOC_DS,3DSTATE_PUSH_CONSTANT_ALLOC_GS,3DSTATE_PUSH_CONSTANT_ALLOC_PS and VSR_PUSH_CONSTANT_BASE.
	9:0	**VS URB Entry Allocation Size** Format: U10-1 Specifies the length, count of 512-bit units, of each URB entry owned by VS. This field is always used (even if VS Function Enable is DISABLED).

3DSTATE_URB_ALLOC_VS_BODY

Value	Name
[0,511]	

Programming Notes
As the VS URB entry serves as both the per-vertex input and output of the VS shader, the VS URB Allocation Size shall be sized to the maximum of the vertex input and output structures.

1 — 31:16 — VS Number of URB Entries SliceN

Format:	U16

This field specifies the number of URB entries in slices beyond Slice0 to be allocated to VS. SW shall ensure that the total Number of Entries does not exceed the relevant ValidValue range listed below.

Value	Name	
[64,3576]		RenderCS
[64,1280]		PositionCS

Programming Notes
This field is ignored by HW as there is only one slice in the device.
VS URB entries shall be allocated even if VS Function Enable is DISABLED.
VS Number of URB Entries must be divisible by 8 if the VS URB Entry Allocation Size is less than 9 512-bit URB entries."2:0" = reserved "000b"

15:0 — VS Number of URB Entries Slice0

Format:	U16

This field specifies the number of URB entries in Slice0 URB memory to be allocated to VS. SW shall ensure that the total Number of Entries does not exceed the relevant ValidValue range listed below.

Value	Name	
[64,3576]		RenderCS
[64,1280]		PositionCS

Programming Notes
VS URB entries shall be allocated even if VS Function Enable is DISABLED.
VS Number of URB Entries must be divisible by 8 if the VS URB Entry Allocation Size is less than 9 512-bit URB entries."2:0" = reserved "000b"

3DSTATE_URB_DS_BODY

3DSTATE_URB_DS_BODY

Source: RenderCS
Size (in bits): 32
Default Value: 0x00000000

DWord	Bit	Description
0	31:25	**DS URB Starting Address** Format: U7 Offset from the start of the URB memory where DS starts its allocation, specified in multiples of 8 KB. Value: [0,127] — Name: **Programming Notes** If CTXT_SR_CTL::POSH_Enable is set and Push Constants are required or Device[SliceCount] GT 1, the lower limit is 8. If CTXT_SR_CTL::POSH_Enable is clear and Push Constants are required or Device[SliceCount] GT 1, the lower limit is 4. If Push Constants are not required and Device[SliceCount] == 1, the lower limit is 0.
	24:16	**DS URB Entry Allocation Size** Format: U9-1 Specifies the length, count of 512-bit units, of each URB entry owned by DS. This field is always used (even if DS Function Enable is DISABLED).
	15:0	**DS Number of URB Entries** Specifies the number of URB entries that are used by DS, based on only 1 slice enabled. When multiple slices are enabled, HW will multiply the value programmed by the number of slices in order to determine the total number of entries. SW shall ensure that the total number of entries does not exceed the relevant ValidValue range listed below. This field is always used (even if DS Function Enable is DISABLED). If Domain Shader Thread Dispatch is Enabled then the minimum number of handles that must be allocated is 34 URB entries. Value: [0,3576] — Name: **Programming Notes** DS Number of URB Entries must be divisible by 8 if the DS URB Entry Allocation Size is programmed to a value less than 9, which is 10 512-bit URB entries. "2:0" = reserved "000"

3DSTATE_URB_GS_BODY

3DSTATE_URB_GS_BODY

Source: RenderCS
Size (in bits): 32
Default Value: 0x00000000

DWord	Bit	Description
0	31:25	**GS URB Starting Address** Format: U7 Offset from the start of the URB memory where GS starts its allocation, specified in multiples of 8 KB. **Value** [0,127] — **Name** **Programming Notes** If CTXT_SR_CTL::POSH_Enable is set and Push Constants are required or Device[SliceCount] GT 1, the lower limit is 8. If CTXT_SR_CTL::POSH_Enable is clear and Push Constants are required or Device[SliceCount] GT 1, the lower limit is 4. If Push Constants are not required and Device[SliceCount] == 1, the lower limit is 0.
	24:16	**GS URB Entry Allocation Size** Format: U9-1 Specifies the length, count of 512-bit units, of each URB entry owned by GS. This field is always used (even if GS Function Enable is DISABLED).
	15:0	**GS Number of URB Entries** Specifies the number of URB entries that are used by GS, based on only 1 slice enabled. When multiple slices are enabled, HW will multiply the value programmed by the number of slices in order to determine the total number of entries. SW shall ensure that the total number of entries does not exceed the relevant ValidValue range listed below. This field is always used (even if GS Function Enable is DISABLED). **Value** [0,1548] — **Name** **Programming Notes** Only if GS is disabled can this field be programmed to 0. If GS is enabled this field shall be programmed to a value greater than 0. For GS Dispatch Mode "Single", this field shall be programmed to a value greater than or equal to 1. For other GS Dispatch Modes, refer to the definition of Dispatch Mode (3DSTATE_GS) for minimum values of this field. GS Number of URB Entries must be divisible by 8 if the GS URB Entry Allocation Size is less than 9 512-bit URB entries."2:0" = reserved "000" When 3DSTATE_GS:Enable is true, the minimum number of GS Number of URB Entries must be set to 2.

3DSTATE_URB_HS_BODY

3DSTATE_URB_HS_BODY

Source: RenderCS
Size (in bits): 32
Default Value: 0x00000000

DWord	Bit	Description
0	31:25	**HS URB Starting Address** Format: U7 Offset from the start of the URB memory where HS starts its allocation, specified in multiples of 8 KB. **Value** [0,127] **Name** **Programming Notes** If CTXT_SR_CTL::POSH_Enable is set and Push Constants are required or Device[SliceCount] GT 1, the lower limit is 8. If CTXT_SR_CTL::POSH_Enable is clear and Push Constants are required or Device[SliceCount] GT 1, the lower limit is 4. If Push Constants are not required and Device[SliceCount] == 1, the lower limit is 0.
	24:16	**HS URB Entry Allocation Size** Format: U9-1 Specifies the length, count of 512-bit units, of each URB entry owned by HS. This field is always used (even if HS Function Enable is DISABLED).
	15:0	**HS Number of URB Entries** Specifies the number of URB entries that are used by HS, based on only 1 slice enabled. When multiple slices are enabled, HW will multiply the value programmed by the number of slices in order to determine the total number of entries. SW shall ensure that the total number of entries does not exceed the relevant ValidValue range listed below. This field is always used (even if HS Function Enable is DISABLED). Programming Restriction:HS Number of URB Entries must be divisible by 8 if the HS URB Entry Allocation Size is less than 9 512-bit URB entries."2:0" = reserved "000" **Value** [0,1548] **Name**

3DSTATE_URB_VS_BODY

3DSTATE_URB_VS_BODY

Source: RenderCS
Size (in bits): 32
Default Value: 0x00000000

DWord	Bit	Description
0	31:25	**VS URB Starting Address** Format: U7 Offset from the start of the URB memory where VS starts its allocation, specified in multiples of 8 KB. **Value** [0,127] — **Name** **Programming Notes** If CTXT_SR_CTL::POSH_Enable is set and Push Constants are required or Device[SliceCount] GT 1, the lower limit is 8. If CTXT_SR_CTL::POSH_Enable is clear and Push Constants are required or Device[SliceCount] GT 1, the lower limit is 4. If Push Constants are not required and Device[SliceCount] == 1, the lower limit is 0.
	24:16	**VS URB Entry Allocation Size** Format: U9-1 Specifies the length, count of 512-bit units, of each URB entry owned by VS. This field is always used (even if VS Function Enable is DISABLED). **Programming Notes** Programming Restriction: As the VS URB entry serves as both the per-vertex input and output of the VS shader,the VS URB Allocation Size must be sized to the maximum of the vertex input and output structures.
	15:0	**VS Number of URB Entries** Format: U16 Specifies the number of URB entries that are used by VS, based on only 1 slice enabled. When multiple slices are enabled, HW will multiply the value programmed by the number of slices in order to determine the total number of entries. SW shall ensure that the total number of entries does not exceed the relevant ValidValue range listed below. This field is always used (even if VS Function Enable is DISABLED). **Value** — **Name** [64,3576] — RenderCS [64,1280] — PositionCS

3DSTATE_URB_VS_BODY		
		Programming Notes Programming Restriction: VS Number of URB Entries must be divisible by 8 if the VS URB Entry Allocation Size is less than 9 512-bit URB entries."2:0" = reserved "000b"

3DSTATE_VF_BODY

3DSTATE_VF_BODY		
Source:	RenderCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:0	**Cut Index** This field specifies the index value that is considered the "cut index" which vertex indices are compared to if a Cut Index Enable is set. The Cut Index is compared to the fetched (and possibly-sign-extended) vertex index, and if these values are equal, the current primitive topology is terminated. Note that, for index buffers less than 32bpp, it is possible to set the Cut Index to a (large) value that will never match a sign-extended vertex index.

3DSTATE_VF_COMPONENT_PACKING_BODY

3DSTATE_VF_COMPONENT_PACKING_BODY		
Source:	RenderCS	
Size (in bits):	128	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0..3	127:0	**Vertex Elements Enables** Format: COMPONENT_ENABLES[32]

3DSTATE_VF_INSTANCING_BODY

3DSTATE_VF_INSTANCING_BODY

Source: RenderCS

Size (in bits): 64

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0	31:10	**Reserved** Access: RO Format: MBZ
	9	**Instance Stride Enable** Format: Enable **Value / Name / Description** 0h / Disabled / The Instance Stride value is neither defined nor used. For this vertex element, VF will access the Vertex Buffer as a simple 1D array using the Vertex Buffer Pitch as a stride. 1h / Enable / For this vertex element, VF will access the Vertex Buffer as a 2D array. The Instance Advancement State field defines the stride between instances. The Vertex Buffer Pitch defines the stride between vertices within a given instance. **Programming Notes** This field must be DISABLED when Instancing Enable is ENABLED.
	8	**Instancing Enable** Format: Enable **Value / Name / Description / Programming Notes** 0h / Disabled / This vertex element is not instanced and therefore vertices within instances can each receive different data for this vertex element. Within each instance, the source vertex data for this vertex element is determined according the Vertex Access Type of the 3DPRIMITIVE command. There is no Instance Data Step Rate state defined for this vertex element. / 1h / Enabled / This vertex element is instanced and therefore vertices within instances will receive the same data for this vertex element. The source pointer for this particular vertex element will be (a) initialized at the start of 3DPRIMITIVE processing, (b) held constant for all vertices within an instance, and (c) advanced between instances as a function of Instance Data Step Rate. / The Instance Advancement State field provides the Instance Data Step Rate.

3DSTATE_VF_INSTANCING_BODY

		Programming Notes This field must be DISABLED when Instance Stride Enable is ENABLED.
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**Vertex Element Index** Format: U6 This field identifies which vertex element state is to be modified by this command. **Value** [0,33] — **Name**
1	31:0	**Instance Advancement State** If Instancing Enable is ENABLED, this field determines the rate at which data for this particular vertex element is changed between instances. Only after the number of instances specified by this field is generated is new (sequential) vertex element data provided. This process continues for each group of instances defined in the 3DPRIMTIVE command. For example, a value of 1 in this field causes new data to be supplied for this vertex element with each sequential (instance) group of vertices. A value of 2 causes every other instance group of vertices to be provided with new vertex element data. The special value of 0 causes all vertices of all instances generated by the 3DPRIMITIVE command to be provided with the same data for this vertex element. (The same effect can be achieved by setting this field to its maximum value.) If Instance Stride Enable is ENABLED, this field determines the stride in BYTES from one instance to the next. An InstanceStride of 0 means there is only one set of instance data (a degenerate 2D array with an instance dimension of 1). For this vertex element, VF will access the vertex buffer data sequentially for each vertex within first instance, and then return to the start of the vertex buffer for the vertices in the next instance, and so on. **Programming Notes** The interpretation/use of this field depends on whether the Instancing Enable or Instance Stride Enable state bit is ENABLED. Software shall not concurrently ENABLE both of those state bits. If both state bits are DISABLED, this field is ignored.

3DSTATE_VF_SGVS_2_BODY

3DSTATE_VF_SGVS_2_BODY

Source: RenderCS
Size (in bits): 64
Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0	31	**XP1 Enable** Format: Boolean Value / Name / Description: 0h / Disabled / XP1 is not inserted 1h / Enabled / XP1 (as defined by XP1 Source Select) is inserted.
	30:29	**XP1 Component Number** If XP1 Enable is ENABLED, this field specifies the 32-bit component location (within the 4-component VUE) where it is inserted. If XP1 Enable is DISABLED, this field is ignored. Value / Name / Description: 0 / COMP_0 / If enabled, XP1 is inserted in component 0 (.x) 1 / COMP_1 / If enabled, XP1 is inserted in component 1 (.y) 2 / COMP_2 / If enabled, XP1 is inserted in component 2 (.z) 3 / COMP_3 / If enabled, XP1 is inserted in component 3 (.w)
	28	**XP1 Source Select** If XP1 Enable is ENABLED, this field selects between the available sources for the XP1 SGV to be inserted. If XP1 Enable is DISABLED, this field is ignored. Value / Name / Description / Programming Notes: 1h / Start Instance Location / The XP1 value is sourced from the Start Instance Location Parameter. / Start Instance Location is the only valid value if 3DSTATE_VF::InstanceIDOffsetEnable is set. 0h / XP1_PARAMETER / The XP1 value is sourced from the XP1 parameter as defined by 3DPRIMITIVE. /
	27:22	**Reserved** Access: RO Format: MBZ

3DSTATE_VF_SGVS_2_BODY

21:16 XP1 Element Offset

Format:	U6

If XP1 Enable is ENABLED, this field specifies the VUE element offset of the 128-bit element where it is to be inserted. The XP1 Component Number specifies where in the specified element it is to be inserted.

Value	Name
[0,33]	

15 XP0 Enable

Format:	Boolean

Value	Name	Description
0h	Disabled	XP0 is not inserted
1h	Enabled	XP0 (as defined by XP0 Source Select) is inserted

14:13 XP0 Component Number

If XP0 Enable is ENABLED, this field specifies the 32-bit component location (within the 4-component VUE) where it is inserted. If XP0 Enable is DISABLED, this field is ignored.

Value	Name	Description
0	COMP_0	If enabled, XP0 is inserted in component 0 (.x)
1	COMP_1	If enabled, XP0 is inserted in component 1 (.y)
2	COMP_2	If enabled, XP0 is inserted in component 2 (.z)
3	COMP_3	If enabled, XP0 is inserted in component 3 (.w)

12 XP0 Source Select

If XP0 Enable is ENABLED, this field selects between the available sources for the XP0 SGV to be inserted.

If XP0 Enable is DISABLED, this field is ignored.

Value	Name	Description
1h	VERTEX_LOCATION	The XP0 value is sourced from one of the two Vertex Location parameters passed in 3DPRIMITIVE. If Vertex Access Mode is SEQUENTIAL, the Start Vertex Location value is used. If Vertex Access Mode is RANDOM, the Base Vertex Location value is used.
0h	XP0_PARAMETER	The XP0 value is sourced from the XP0 parameter as defined by 3DPRIMITIVE.

11:6 Reserved

Access:	RO
Format:	MBZ

3DSTATE_VF_SGVS_2_BODY

5:0 XP0 Element Offset

Format:	U6

If XP0 Enable is ENABLED, this field specifies the VUE element offset of the 128-bit element where it is to be inserted. The XP0 Component Number specifies where in the specified element it is to be inserted. If XP0 Enable is DISABLED, this field is ignored.

Value	Name
[0,33]	

1 31:16 Reserved

Access:	RO
Format:	MBZ

15 XP2 Enable

Format:	Boolean

Value	Name	Description
0h	Disabled	XP2 is not inserted
1h	Enabled	XP2 is inserted, sourced from the XP2 parameter as defined by 3DPRIMITIVE.

14:13 XP2 Component Number

If XP2 Enable is ENABLED, this field specifies the 32-bit component location (within the 4-component VUE) where it is to be inserted. If XP2 Enable is DISABLED, this field is ignored.

Value	Name	Description
0	COMP_0	If enabled, XP2 is inserted in component 0 (.x)
1	COMP_1	If enabled, XP2 is inserted in component 1 (.y)
2	COMP_2	If enabled, XP2 is inserted in component 2 (.z)
3	COMP_3	If enabled, XP2 is inserted in component 3 (.w)

12:6 Reserved

Access:	RO
Format:	MBZ

5:0 XP2 Element Offset

Format:	U6

If XP2 Enable is ENABLED, this field specifies the VUE element offset of the 128-bit element where it is to be inserted. The XP2 Component Number specifies where in the specified element it is to be inserted. If XP2 Enable is DISABLED, this field is ignored.

Value	Name
[0,33]	

3DSTATE_VF_SGVS_BODY

3DSTATE_VF_SGVS_BODY

Source: RenderCS
Size (in bits): 32
Default Value: 0x00000000

DWord	Bit	Description
0	31	**InstanceID Enable** Format: Enable Value / Name / Description 0h / Disabled / InstanceID is not inserted 1h / Enabled / InstanceID is inserted
	30:29	**InstanceID Component Number** If InstanceID Enable is ENABLED, this field specifies the 32-bit component location (within the 4-component VUE) where it is inserted. If InstanceID Enable is DISABLED, this field is ignored. Value / Name / Description 0 / COMP_0 / If enabled, InstanceID is inserted in component 0 (.x) 1 / COMP_1 / If enabled, InstanceID is inserted in component 1 (.y) 2 / COMP_2 / If enabled, InstanceID is inserted in component 2 (.z) 3 / COMP_3 / If enabled, InstanceID is inserted in component 3 (.w)
	28:22	**Reserved** Access: RO Format: MBZ
	21:16	**InstanceID Element Offset** Format: U6 If InstanceID Enable is ENABLED, this field specifies the VUE element offset of the 128-bit element where it is to be inserted. The InstanceID Component Number specifies where in the specified element it is inserted. Value / Name [0,33] /
	15	**VertexID Enable** Format: Enable Value / Name / Description 0h / Disabled / VertexID is not inserted 1h / Enabled / VertexID is inserted

3DSTATE_VF_SGVS_BODY

14:13 VertexID Component Number

If VertexID Enable is ENABLED, this field specifies the 32-bit component location (within the 4-component VUE) where it is inserted. If VertexID Enable is DISABLED, this field is ignored.

Value	Name	Description
0	COMP_0	If enabled, VertexID is inserted in component 0 (.x)
1	COMP_1	If enabled, VertexID is inserted in component 1 (.y)
2	COMP_2	If enabled, VertexID is inserted in component 2 (.z)
3	COMP_3	If enabled, VertexID is inserted in component 3 (.w)

12:6 Reserved

Access:	RO
Format:	MBZ

5:0 VertexID Element Offset

Format:	U6

If VertexID Enable is ENABLED, this field specifies the VUE element offset of the 128-bit element where it is to be inserted. The VertexID Component Number specifies where in the specified element it is inserted. This is also the vertex element index. If VertexID Enable is DISABLED, this field is ignored.

Value	Name
[0,33]	

3DSTATE_VF_TOPOLOGY_BODY

3DSTATE_VF_TOPOLOGY_BODY		
Source:	RenderCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:6	**Reserved** Access: RO Format: MBZ
	5:0	**Primitive Topology Type** Format: 3D_Prim_Topo_Type This field specifies the VF stage's Topology state.

3DSTATE_VIEWPORT_STATE_POINTERS_CC_BODY

3DSTATE_VIEWPORT_STATE_POINTERS_CC_BODY

Source: RenderCS
Size (in bits): 32
Default Value: 0x00000000

DWord	Bit	Description
0	31:5	**CC Viewport Pointer** Format: DynamicStateOffset[31:5]CC_VIEWPORT*16 Specifies the 32-byte aligned address offset of the CC_VIEWPORT state. This offset is relative to the Dynamic State Base Address.
	4:0	**Reserved** Access: RO Format: MBZ

3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP_BODY

3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP_BODY

Source: RenderCS

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:6	**SF Clip Viewport Pointer** Format: DynamicStateOffset[31:6]SF_CLIP_VIEWPORT*16 Specifies the 64-byte aligned address offset of the SF_CLIP_VIEWPORT state. This offset is relative to the Dynamic State Base Address.
	5:0	**Reserved** Access: RO Format: MBZ

3DSTATE_VS_BODY

3DSTATE_VS_BODY

Source: RenderCS

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0..1	63:6	**Kernel Start Pointer** Format: InstructionBaseOffset[63:6] This field specifies the starting location of the kernel program run by threads spawned by the VS pipeline stage. It is specified as a 64-byte-granular offset from the Instruction Base Address. This field is ignored if VS Function Enable is DISABLED.
	5:0	**Reserved** Access: RO Format: MBZ
2	31	**Reserved** Access: RO Format: MBZ
	30	**Vector Mask Enable** Format: Enable Upon subsequent VS thread dispatches, this bit is loaded into the EUs **Vector Mask Enable** (VME, cr0.0[3]) thread state. Refer to EU documentation for the definition and use of VME state. **Value / Name / Description** 0h / Dmask / The EU will use the Dispatch Mask (supplied by the VS stage) for instruction execution. 1h / Vmask / The EU will use the Vector Mask (derived from the Dispatch Mask) for instruction execution. **Programming Notes** Under normal conditions SW shall specify DMask, as the VS stage will provide a Dispatch Mask appropriate to SIMD4x2 or SIMD8 thread execution (as a function of SIMD8 Dispatch Enable). E.g., for SIMD4x2 thread execution, the VS stage will generate a Dispatch Mask that is equal to what the EU would use as the Vector Mask. For SIMD8 execution there is no known usage model for use of Vector Mask (as there is for PS shaders).

3DSTATE_VS_BODY

29:27 **Sampler Count**

Format:	U3

This field specifies (in multiples of 4)the number of sets of sampler state that will be prefetched for use by the VS kernel. While the prefetching of sampler state is optional and does not impact functionality, it may improve performance.
This field is ignored if the Function Enable state is set to DISABLED.

Value	Name	Description
0h	No Samplers	no samplers used
1h	1-4 Samplers	between 1 and 4 samplers used
2h	5-8 Samplers	between 5 and 8 samplers used
3h	9-12 Samplers	between 9 and 12 samplers used
4h	13-16 Samplers	between 13 and 16 samplers used

26 **Reserved**

Access:	RO
Format:	MBZ

25:18 **Binding Table Entry Count**

Format:	U8

Specifies how many binding table entries the kernel uses. Used only for prefetching of the binding table entries and associated surface state. Note: For kernels using a large number of binding table entries, it may be wise to set this field to zero to avoid prefetching too many entries and thrashing the state cache. This field is ignored if VS Function Enable is DISABLED.

When HW Generated Binding Table bit is enabled: This field indicates which cache lines (512bit units - 32 Binding Table Entry section) should be fetched. Each bit in this field corresponds to a cache line. Only the 1st 4 non-zero Binding Table entries of each 32 Binding Table entry section prefetched will have its surface state prefetched.

Value	Name
[0,255]	

Programming Notes
When HW binding table bit is set, it is assumed that the Binding Table Entry Count field will be generated at JIT time.

17 **Thread Dispatch Priority**

Format:	U1

Specifies the priority of the thread for dispatch: This field is ignored if VS Function Enable is DISABLED.

Value	Name	Description
0h	Normal	Normal Priority
1h	High	High Priority

3DSTATE_VS_BODY

16 Floating Point Mode

Format:	U1

Specifies the initial floating point mode used by the dispatched thread. This field is ignored if VS Function Enable is DISABLED.

Value	Name	Description
0h	IEEE-754	Use IEEE-754 Rules
1h	Alternate	Use Alternate Rules

15 Reserved

Access:	RO
Format:	MBZ

14 Reserved

Access:	RO
Format:	MBZ

13 Illegal Opcode Exception Enable

Format:	Enable

This bit gets loaded into EU CR0.1[12] (note the bit # difference). See Exceptions and ISA Execution Environment. This field is ignored if VS Function Enable is DISABLED.

12 Accesses UAV

Format:	Enable

This field must be set when VS has a UAV access.

Programming Notes
This field must not be set when VS Function Enable is disabled.
This bit shall not be set when the command is executed in the PCS pipeline.

11 Reserved

Access:	RO
Format:	MBZ

10:8 Reserved

Format:	MBZ

7 Software Exception Enable

Format:	Enable

This bit gets loaded into EU CR0.1[13] (note the bit # difference). See Exceptions and ISA Execution Environment. This field is ignored if VS Function Enable is DISABLED.

6:0 Reserved

Access:	RO
Format:	MBZ

3DSTATE_VS_BODY

3..4 | 63:32 | Reserved

Access:	RO
Format:	MBZ

31:10 | Scratch Space Base Pointer

Format:	GeneralStateOffset[31:10]

Specifies the starting location of the scratch space area allocated to this FF unit as a 1K-byte aligned offset from the General State Base Address. If required, each thread spawned by this FF unit will be allocated some portion of this space, as specified by Per-Thread Scratch Space. The computed offset of the thread-specific portion will be passed in the thread payload as Scratch Space Offset. The thread is expected to utilize "stateless" DataPort read/write requests to access scratch space, where the DataPort will cause the General State Base Address to be added to the offset passed in the request header. This field is ignored if VS Function Enable is DISABLED. In 64b OS all pointers need to be seen by SW as 48b. HW does not support a Scratch Space Base Pointer larger than 32b, therefore SW must ensure Bits<63:32> are set to 0's.

Programming Notes

The scratch spaces allocated to the POCS VSR stage and RCS VS stage shall not overlap with each other or the scratch space allocations of any other enabled stage in the RCS pipeline.

9:4 | Reserved

Access:	RO
Format:	MBZ

3:0 | Per-Thread Scratch Space

Format:	U4

Specifies the amount of scratch space to be allocated to each thread spawned by this FF unit.The driver must allocate enough contiguous scratch space, starting at the Scratch Space Base Pointer, to ensure that the Maximum Number of Threads can each get Per-Thread Scratch Space size without exceeding the driver-allocated scratch space. This field is ignored if VS Function Enable is DISABLED.

Value	Name	Description
[0,11]		Indicating [1K Bytes, 2M Bytes]

Programming Notes

This amount is available to the kernel for information only. It will be passed verbatim (if not altered by the kernel) to the Data Port in any scratch space access messages, but the Data Port will ignore it.

5 | 31:25 | Reserved

Access:	RO
Format:	MBZ

3DSTATE_VS_BODY

24:20 Dispatch GRF Start Register For URB Data

Format:	U5

Specifies the starting GRF number for the URB portion (URB constants and vertices) of the thread payload.
This field is ignored if VS Function Enable is DISABLED.

Value	Name	Description
[0,31]		indicating GRF [R0, R31]

19:17 Reserved

Access:	RO
Format:	MBZ

16:11 Vertex URB Entry Read Length

Format:	U6

Specifies the number of pairs of 128-bit vertex elements to be passed into the payload for each vertex. This field is ignored if VS Function Enable is DISABLED. For SIMD4x2 dispatch, each vertex element requires one GRF of payload data, therefore the number of GRFs with vertex data will be double the value programmed in this field. For SIMD8 dispatch, each vertex element requires 4 GRFs of payload data, therefore the number of GRFs with vertex data will be 8 times the value programmed in this field. The EU limit of 128 GRFs imposes a maximum limit of 30 elements per vertex pushed into the payload, though the practical limit may be lower. If input vertices exceed the practical limit, software must decide between resorting to pulling elements during thread execution or dropping back to SIMD4x2 dispatch. Note that the VUE is used for both input and output, so when using the pull-model software must ensure inputs are not overwritten before last use.

Value	Name	Description
[1,63]		if SIMD8 dispatch disabled
[0,15]		if SIMD8 dispatch enabled

10 Reserved

Access:	RO
Format:	MBZ

9:4 Vertex URB Entry Read Offset

Format:	U6

Specifies the offset (in 256-bit units) at which Vertex URB data is to be read from the URB before being included in the thread payload. This offset applies to all Vertex URB entries passed to the thread. This field is ignored if VS Function Enable is DISABLED.

Value	Name
[0,63]	

3:0 Reserved

Access:	RO
Format:	MBZ

3DSTATE_VS_BODY

6 — 31:22 — Maximum Number of Threads

Format:	U10-1

Specifies the maximum number of simultaneous threads allowed to be active. Used to avoid using up the scratch space. Programming the value of the max threads over the number of threads based off number of threads supported in the execution units may improve performance since the architecture allows threads to be buffered between the check for max threads and the actual dispatch into the EU. Programming the max values to a number less than the number of threads supported in the execution units may reduce performance. This field is ignored if VS Function Enable is DISABLED.

Value	Name	Description
[0,191]		indicating thread count of [1,192]
[0,545]		indicating thread count of [1,546]

21:11 — Reserved

Access:	RO
Format:	MBZ

10 — Statistics Enable

Format:	Enable

If ENABLED, the VS stage will perform statistics gathering. See the Statistics Gathering subsection.
If DISABLED, statistics information associated with the VS stage will be left unchanged.

Programming Notes
When a 3DPRIMITIVE command with POSH Enable set is executed from the RCS command stream, VS statistics gathering is inhibited for that command.

9 — SIMD8 Single Instance Dispatch Enable

Format:	Enable

This field is used to specify whether vertices from different instances can be combined in a single SIMD8 dispatch. This bit is ignored if SIMD4x2 dispatches are enabled (i.e., SIMD8 Dispatch Enable is DISABLED).
If ENABLED, SIMD8 VS thread dispatches will not combine vertices from different instances. This allows the VS kernel to handle instance-specific operations (e.g., read constants indexed by the InstanceID) in a global fashion, as these operations pertain to all vertices of the dispatch.
If DISABLED, SIMD8 VS thread dispatches can combine vertices from different instances. The VS kernel must determine if instance-specific operations can be handled globally (vs. per-vertex). E.g., it can examine the Single Instance payload bit.

Programming Notes
SIMD8 Single Instance Dispatch Enable is not supported for HPCXTs.

8:3 — Reserved

Access:	RO
Format:	MBZ

3DSTATE_VS_BODY

	2	**SIMD8 Dispatch Enable** Format: Enable This field determines how VS threads are dispatched and how the thread payloads are generated. The setting of this field must agree with how the VS kernel was compiled. If ENABLED, SIMD8 VS thread dispatches are performed. The **Single Vertex Dispatch** field is ignored. If DISABLED, SIMD4x2 thread dispatches are performed. The **Single Vertex Dispatch** field can be used to force single-vertex dispatches. **Programming Notes** The only supported mode is SIMD8 Dispatch Enable set to Enable (1).
	1	**Vertex Cache Disable** Format: Disable This bit controls the operation of the Vertex Cache. This field is always used. If the Vertex Cache is DISABLED and the VS Function is ENABLED, the Vertex Cache is not used and all incoming vertices will be passed to VS threads. If the Vertex Cache is ENABLED and the VS Function is ENABLED, only incoming vertices that do not hit in the Vertex Cache will be passed to VS threads. If the Vertex Cache is ENABLED and the VS Function is DISABLED, input vertices that miss in the Vertex Cache will be assembled and written to the URB (by the VF stage), and subsequently passed through the VS stage unmodified (i.e, no VS threads are spawned). The Vertex Cache is invalidated whenever the Vertex Cache becomes DISABLED, whenever the VS Function Enable toggles, between 3DPRIMITIVE commands and between instances within a 3DPRIMITIVE command. **Programming Notes** See the Vertex Caching subsection for details on implicit Vertex Cache disabling.
	0	**Function Enable** Format: Enable This bit determines whether or not the VS stage spawns VS threads, which comprises the bulk of the VS stage functionality. If ENABLED, VS threads may be spawned to process VF-generated vertices before the resulting vertices are passed down the pipeline. If DISABLED, VF-generated vertices will pass thru the VS function and are sent down the pipeline unmodified. The Vertex Cache (if enabled) is still available.
7	31:27	**Reserved** Access: RO Format: MBZ

3DSTATE_VS_BODY

26:21 Vertex URB Entry Output Read Offset

Format:	U6

Specifies the offset (in 256-bit units) at which Vertex URB data is to be read from the URB by the Setup Back-End (SBE) function. The offset programmed will specify the start of Attribute 0 to be passed in subsequent Pixel Shader thread payloads. Refer to the Attribute Interpolator Setup documentation.

Value	Name
[0,63]	

Programming Notes
As the vertex header data located at the start of the Vertex URB entry is typically only used by 3D pipeline FFs (i.e., Clipper, Setup FrontEnd) and not required as interpolated attributes in PIxel Shader threads, it is expected that SW will program this Start Offset skip over the vertex header. This offset value is ignored if SBE's Number of SF Attributes state is programmed to 0 (i.e., no attributes are defined beyond the position read from the Vertex Header)

20:16 Vertex URB Entry Output Length

Format:	U5

Specifies the amount of Vertex Attribute URB data to be read by the Setup Back-End function for each Vertex URB entry, in 256-bit units. The attribute data will be read starting at the offset specified by the Vertex URB Entry Output Read Offset state.

Value	Name
[1,16]	

Programming Notes
This length value is ignored if SBE's Number of SF Attributes state is programmed to 0 (i.e., no attributes are defined beyond the position read from the Vertex Header).

15:8 User Clip Distance Clip Test Enable Bitmask

Format:	U8

This 8 bit mask field selects which of the 8 Clip Distance Values (if any) are to be included in the Clip stage's trivial reject / trivial accept / must clip determination function.
The ClipDistance Values (if present) are located in DW8-15 of the VUE Vertex Header located at the beginning of VUE URB entries. Bit 0 of this field corresponds to Clip Distance Value 0.

7:0 User Clip Distance Cull Test Enable Bitmask

Format:	U8

This 8 bit mask field selects which of the 8 Clip Distance Values (if any) are to be included in the Clip stage's trivial reject / trivial accept determination function. Note that must clip determination is not included in this function.
The ClipDistance Values (if present) are located in DW8-15 of the VUE Vertex Header located at the beginning of VUE URB entries. Bit 0 of this field corresponds to Clip Distance Value 0.

3DSTATE_WM_BODY

3DSTATE_WM_BODY

Source: RenderCS
Size (in bits): 32
Default Value: 0x00000000

DWord	Bit	Description
0	31	**Statistics Enable** Format: Enable If ENABLED, the Windower and pixel pipeline will engage in statistics gathering. If DISABLED, statistics information associated with this FF stage will be left unchanged. See Statistics Gathering. **Programming Notes** This bit must be disabled if any of these bits is set: 3DSTATE_WM::**Legacy Depth Buffer Clear**, 3DSTATE_WM::**Legacy Hierarchical Depth Buffer Resolve Enable** or 3DSTATE_WM::**Legacy Depth Buffer Resolve Enable**.
	30:27	**Reserved** Access: RO Format: MBZ
	26	**Legacy Diamond Line Rasterization** Format: Enable This bit, if ENABLED, indicates that the Windower will rasterize zero width lines using the DX9 rasterization rules. If DISABLED, the Windower will rasterize zero width lines using the DX10 rasterization rules (see Strips Fans chapter).
	25:23	**Reserved** Access: RO Format: MBZ
	22:21	**Early Depth/Stencil Control** Format: U2 This field specifies the behavior of early depth/stencil test.

Value	Name	Description
0h	NORMAL	Depth/Stencil Test/Write behaves as if it happens post-shader, however the pixel shader is not necessarily executed if the pixel fails depth or stencil test (this is the legacy behavior)
1h	PSEXEC	Depth/Stencil Test/Write behaves as if it happens post-shader, and the pixel shader is executed if the pixel fails depth or stencil test (although pre-shader actions such as primitive inclusion, stipple, etc. will still cause the shader not to execute)
2h	PREPS	Depth/Stencil Test/Write behaves as if it happens pre-shader. The pixel shader is not executed if the pixel fails depth or stencil test. Depth and stencil

		writes occur even if the pixel is killed by the shader or post-shader by alpha test, etc. Depth output by the pixel shader is ignored.
3h	Reserved	

Programming Notes
The Early Depth/Stencil Control field cannot be set to PREPS (value = 2h) if ForceKillpix = ForceON or Forced Thread Dispatch = ForceON

20:19 **Force Thread Dispatch Enable**

Value	Name	Description
0h	Normal	WM_INT::ThreadDispatchEnable is computed normally
1h	ForceOff	Forces WM_INT::ThreadDispatchEnable Off
2h	ForceON	Forces WM_INT::ThreadDispatchEnable On
3h	Reserved	

Programming Notes
This should must always be set to Normal. This field should not be tested for functional validation

18:17 **Position ZW Interpolation Mode**

Format:	U2

This field elects "interpolation mode" associated with the Position Z (source depth) and W coordinates passed in the PS payload when the PS requires Position as input. This field does not determine whether these coordinates are actually included in the payload (see Pixel Shader Requires Depth, Pixel Shader Requires W).

Value	Name	Description
0h	INTERP_PIXEL	`Evaluate Z & W at the pixel center or UL corner (as specified by Pixel Location of 3DSTATE_MULTISAMPLE)`
1h	Reserved	
2h	INTERP_CENTROID	
3h	INTERP_SAMPLE	

Programming Notes
WM_INT::RT Independent Rasterization Enable must be disabled in order to select INTERP_SAMPLE.
MSDISPMODE_PERSAMPLE is required in order to select INTERP_SAMPLE.

16:11 **Barycentric Interpolation Mode**

Format:	Enable[6]

Controls which barycentric interpolation terms must be passed into the pixel shader kernel.
Bit 0: Perspective Pixel Location barycentric is required
Bit 1: Perspective Centroid barycentric is required
Bit 2: Perspective Sample barycentric is required

3DSTATE_WM_BODY

Bit 3: Non-perspective Pixel Location barycentric is required
Bit 4: Non-perspective Centroid barycentric is required
Bit 5: Non-perspective Sample barycentric is required

Programming Notes

If contiguous dispatch modes are enabled, only bit 3 (non-perspective pixel location) can be set, all other bits in this field must be zero. Pixel Location below refers to either the upper left corner or pixel center depending on the **Pixel Location** state of 3DSTATE_MULTISAMPLING).MSDISPMODE_PERSAMPLE is required in order to select Perspective Sample or Non-perspective Sample barycentric coordinates.

10 **Reserved**

Access:	RO
Format:	MBZ

9:8 **Line End Cap Antialiasing Region Width**

Format:	U2

This field specifies the distances over which the coverage of anti-aliased line end caps are computed.

Value	Name	Description
0h	0.5 pixels	0.5 pixels
1h	1.0 pixels	1.0 pixels
2h	2.0 pixels	2.0 pixels
3h	4.0 pixels	4.0 pixels

7:6 **Line Antialiasing Region Width**

Format:	U2

This field specifies the distance over which the anti-aliased line coverage is computed.

Value	Name	Description
0h	0.5 pixels	0.5 pixels
1h	1.0 pixels	1.0 pixels
2h	2.0 pixels	2.0 pixels
3h	4.0 pixels	4.0 pixels

5 **Reserved**

Access:	RO
Format:	MBZ

4 **Polygon Stipple Enable**

Format:	Enable

Enables the Polygon Stipple function.

3 **Line Stipple Enable**

Format:	Enable

Enables the Line Stipple function.

3DSTATE_WM_BODY

2 **Point Rasterization Rule**

This field specifies the rasterization rules to be applied whenever the edges of a point primitive fall exactly on a pixel sampling point.

Value	Name	Description
0h	RASTRULE_UPPER_LEFT	To match "normal" upper left rules for surface primitives
1h	RASTRULE_UPPER_RIGHT	To match OpenGL point rasterization rules (round to + infinity, where this is the upper right direction wrt OpenGL screen origin of lower left).

1:0 **Force Kill Pixel Enable**

Value	Name	Description
0h	Normal	WM_INT:: Pixel Shader Kill Pixel is computed normally
1h	ForceOff	Forces WM_INT:: Pixel Shader Kill Pixel Off
2h	ForceON	Forces WM_INT:: Pixel Shader Kill Pixel On
3h	Reserved	

Programming Notes
This should must always be set to Normal. This field should not be tested for functional validation

3DSTATE_WM_CHROMAKEY_BODY

3DSTATE_WM_CHROMAKEY_BODY		
Source:	RenderCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31	**ChromaKey Kill Enable** Format: Enable If ENABLED, indicates that at least one of the attached samplers has ChromaKeyKill enabled.
	30:0	**Reserved** Access: RO Format: MBZ

3DSTATE_WM_DEPTH_STENCIL_BODY

3DSTATE_WM_DEPTH_STENCIL_BODY		
Source:	RenderCS	
Size (in bits):	96	
Default Value:	0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0	31:29	**Stencil Fail Op** Format: **3D_Stencil_Operation** This field specifies the operation to perform on the Stencil Buffer when the (front face) stencil test fails. **Programming Notes** if all three stencil ops (Stencil Fail, Stencil Pass Depth Fail, and Stencil Pass Depth Pass) are KEEP, ZERO, or REPLACE, the stencil buffer is not read.
	28:26	**Stencil Pass Depth Fail Op** Format: **3D_Stencil_Operation** This field specifies the operation to perform on the Stencil Buffer when the (front face) stencil test passes but the depth pass fails.
	25:23	**Stencil Pass Depth Pass Op** Format: **3D_Stencil_Operation** This field specifies the operation to perform on the Stencil Buffer when the (front face) stencil test passes but the depth test passes.
	22:20	**Backface Stencil Test Function** Format: **3D_Compare_Function**
	19:17	**Backface Stencil Fail Op** Format: **3D_Stencil_Operation**
	16:14	**Backface Stencil Pass Depth Fail Op** Format: **3D_Stencil_Operation** This field specifies the operation to perform on the Stencil Buffer when the stencil test passes but the depth pass fails.
	13:11	**Backface Stencil Pass Depth Pass Op** Format: **3D_Stencil_Operation** This field specifies the operation to perform on the Stencil Buffer when the stencil test passes and the depth pass passes (or is disabled).
	10:8	**Stencil Test Function** Format: **3D_Compare_Function** This field specifies the comparison function used in the (front face) StencilTest function.

3DSTATE_WM_DEPTH_STENCIL_BODY

7:5 Depth Test Function

Format:	3D_Compare_Function

Specifies the comparison function used in DepthTest function.

Programming Notes
If the Depth Test Function is ALWAYS or NEVER, the depth buffer is not read.

4 Double Sided Stencil Enable

Format:	Enable

Enable doubled sided stencil operations.

Value	Name	Description
0h	False	Double Sided Stencil Disabled
1h	True	Double Sided Stencil Enabled

Programming Notes
• Back-facing primitives have a vertex winding order opposite to the currently selected Front Winding state. • Culling of primitives is not affected by the double sided stencil state • Back-facing primitives will be rendered, honoring all current device state, as though it were a front-facing primitive with no implicitly overloaded state.

3 Stencil Test Enable

Format:	Enable

Enables StencilTest function of the Pixel Processing pipeline.

Programming Notes
If any of the render targets are YUV format, this field must be disabled.

2 Stencil Buffer Write Enable

Format:	Enable

Enables writes to the Stencil Buffer.

Programming Notes
If this field is enabled, Stencil Test Enable must also be enabled.

1 Depth Test Enable

Format:	Enable

Enables the DepthTest function of the Pixel Processing pipeline.

Value	Name
0h	Disable
1h	Enable

Programming Notes
If any of the render targets are YUV format, this field must be disabled.

3DSTATE_WM_DEPTH_STENCIL_BODY

	0	**Depth Buffer Write Enable** Format: Enable Enables writes to the Depth Buffer. **Programming Notes** A Depth Buffer must be defined before enabling writes to it, or operation is UNDEFINED. This bit must not be set when WM_INT::RT Independent Rasterization Enable is true.
1	31:24	**Stencil Test Mask** Format: U8 This field specifies a bit mask applied to stencil test values. Both the stencil reference value and value read from the stencil buffer will be logically ANDed with this mask before the stencil comparison test is performed.
	23:16	**Stencil Write Mask** Format: U8 This field specifies a bit mask applied to stencil buffer writes. Only those stencil buffer bits corresponding to bits set in this mask will be modified.
	15:8	**Backface Stencil Test Mask** Format: U8 This field specifies a bit mask applied to backface stencil test values. Both the stencil reference value and value read from the stencil buffer will be logically ANDed with this mask before the stencil comparison test is performed.
	7:0	**Backface Stencil Write Mask** Format: U8 This field specifies a bit mask applied to backface stencil buffer writes. Only those stencil buffer bits corresponding to bits set in this mask will be modified.
2	31:16	**Reserved** Access: RO Format: MBZ
	15:8	**Stencil Reference Value** Format: U8 This field specifies the stencil reference value to compare against in the (front face) StencilTest function.
	7:0	**Backface Stencil Reference Value** Format: U8 This field specifies the stencil reference value to compare against in the StencilTest function.

3DSTATE_WM_HZ_OP_BODY

3DSTATE_WM_HZ_OP_BODY

Source: RenderCS

Size (in bits): 128

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31	**Stencil Buffer Clear Enable** Format: Enable When set, the stencil buffer is initialized. **Programming Notes** If this field is enabled, 1. the **Depth Buffer Resolve Enable (full or partial)** and **Hierarchical Depth Buffer Resolve Enable** fields must both be disabled. 2. 3DSTATE_DEPTH_BUFFER::Stencil Write Enable must be set if 3DSTATE_STENCIL_BUFFER::Stencil buffer enable is set. When this field is enabled, Stencil Buffer Resolve Enable should be disabled
	30	**Depth Buffer Clear Enable** Format: Enable When set, the depth buffer is initialized. **Programming Notes** If this field is enabled, 1. the **Depth Buffer Resolve Enable**(full or partial) and **Hierarchical Depth Buffer Resolve Enable** fields must both be disabled. 2. 3DSTATE_DEPTH_BUFFER::Depth Write Enable must be set.
	29	**Scissor Rectangle Enable** Format: Enable Enables operation of Scissor Rectangle. **Programming Notes** In order get the functionality right if this bit is disabled, driver must clip the clear rectangle to scissor rectangle if scissor test is enabled before clearing.

3DSTATE_WM_HZ_OP_BODY

28 **Depth Buffer Resolve Enable**

Format:	Enable

When set, the depth buffer is made to be consistent with the hierarchical depth buffer as a side-effect of rendering pixels. This is intended to be used when the depth buffer is to be used as a surface outside of the 3D rendering operation. The Depth buffer will be in uncompressed state after this operation.

Programming Notes

If this field is enabled,

1. the **Depth Buffer Clear** and **Hierarchical Depth Buffer Resolve Enable** fields must both be disabled.
2. 3DSTATE_DEPTH_BUFFER::Depth Write Enable must be set.
3. Depth Buffer Partial Resolve Enable field should be disabled.

For validation reasons, the need to resolve an area smaller than the whole depth buffer can occur. See the programming notes for X/Y Min and X/Y Max

27 **Hierarchical Depth Buffer Resolve Enable**

Format:	Enable

When set, the hierarchical depth buffer is made to be consistent with the depth buffer as a side-effect of rendering pixels. This is intended to be used when the depth buffer has been modified outside of the 3D rendering operation.

Programming Notes

If this field is enabled,

1. the **Depth Buffer Clear** and **Depth Buffer Resolve Enable (full or partial)** fields must both be disabled.
2. 3DSTATE_DEPTH_BUFFER::Depth Write Enable must be set.
3. Stencil Buffer Resolve Enable must be disabled.

Doing a Hierarchical Depth Buffer resolve (HZ resolve) on a partial HZ buffer is not permitted. The HZ resolve operation must be done on the entire HZ buffer.
Performance Note: expect the hierarchical depth buffer's impact on performance to be reduced for some period of time after this operation is performed, as the hierarchical depth buffer is initialized to a state that makes it ineffective. Further rendering will tend to bring the hierarchical depth buffer back to a more effective state.

3DSTATE_WM_HZ_OP_BODY

26	**Pixel Position Offset Enable** Format: Enable Enables the device to offset pixel positions by 0.5 both in horizontal and vertical directions. **Programming Notes** Setting this field along with setting the Pixel Location to upper left and number of multisamples to greater than one will cause the device to offset pixel positions by 0.5 both in horizontal and vertical directions. It is to be noted this is done to adjust the pixel co-ordinate system to DX9 like, so any WM_HZ_OP screen space rectangles (eg: legacy HiZ Clear, Resolve etc) generated internally by driver in this mode needs to be aware of this offset adjustment and send the rectangles according to alignment restriction taking this offset adjustment into consideration. SW can choose to set this bit only for DX9 API. DX10/OGL API's should not have any effect by setting or not setting this bit.
25	**Full Surface Depth and Stencil Clear** Format: Enable **Programming Notes** Setting this field to "1" along with "Depth buffer clear" will cause all the pixels/samples in an the HZ and Stencil CLs to be cleared. Software must set this only when the APP requires the entire Depth surface to be cleared. Setting this field to "1" for STC-buffer only clear without "depth buffer clear" will cause all the pixels/samples in the STC-CL to get the stc-ref value.
24	**Stencil Buffer Resolve Enable** When set, the stencil buffer is filled with the true stencil values. This is intended to be used when the stencil buffer is to be used as a surface outside of the 3D rendering operation. When this is enabled, Stencil Buffer Clear Enable field should be disabled. **Programming Notes** The STC buffer is required to be done only if the "compression bit" in the 3DSTATE_STENCIL_BUFFER is enabled. If STC buffer compression is enabled, then the STC buffer will hold compressed data. To get the true stencil values for all the pixels, a stencil resolve operation is required. Issue The stencil resolve bit in WM_HZ_OP state is not being

3DSTATE_WM_HZ_OP_BODY

		considered by WM in the right pipeline stage Workaround **Astep only** W/a is to have a pipe control with thread_dispatch set to OFF before the WM_HZ_OP with stencil buffer resolve is required(bit 24 below). 1) Setting the force thread dispatch enable(bits 20:19) in the 3dstate_WM_body state to be set to Force_OFF (value of 1) before the first WM_HZ_OP state cycle 2) Before second WM_HZ_OP state which is required by programming sequencing to complete the HZ_OP operation, reprogram the 3dstate_WM_body to set to NORMAL(value of 0). 3DSTATE_WM.ForceThreadDispatchEnable = 1(ForceOff) PIPE_CONTROL commit the above state before HZ_OP 3DSTATE_WM_HZ_OP (stencil resolve bit set) PIPE_CONTROL 3DSTATE_WM.ForceThreadDispatchEnable = 0(Normal) 3DSTATE_WM_HZ_OP (empty, no bits set)
	23:16	**Stencil Clear Value** Format: U8.0 This field specifies the stencil clear value.
	15:13	**Number of Multisamples** Format: U3 This field specifies how many samples/pixel exist in the Depth Buffer and Stencil buffers, as log2(#samples).

Value	Name	Description
0h	1	1 sample/pixel
1h	2	2 samples/pixel
2h	4	4 samples/pixel
3h	8	8 samples/pixel
4h	16	16 samples/pixel
5h-7h	Reserved	

	12:9	**Reserved** Access: RO Format: MBZ
	8:0	**Reserved** Access: RO Format: MBZ

3DSTATE_WM_HZ_OP_BODY

1
Programming Notes:
The Clear/Resolve rectangle X and Y Min values must be shifted by the LOD level; i.e. the hardware does not include the LOD in this function. Hence to clear any particular X, Y from the base level, to clear the contents at level "LOD" use (X»LOD) and (Y»LOD).
The final X and Y Min values, after LOD adjustment described above, have to be manually 8x4 or 8x8 aligned for Depth and HZ Resolve passes only. For Clears see "Full Surface Depth and Stencil Clear" field in this command instead.
For multisample set to 1X together with Z format set to16 bpp:
resolve_aligned_y_min = (y_min & ~0x7) //round down to last multiple of 8
resolve_aligned_x_min = (x_min & ~0x7) //round down to last multiple of 8
All other cases:
resolve_aligned_y_min = (y_min & ~0x3) //round down to last multiple of 4
resolve_aligned_x_min = (x_min & ~0x7) //round down to last multiple of 8
	31:16	**Clear Rectangle Y Min**
		Format: U16
		Specifies Ymin value of (inclusive) of clear rectangle with the Depth Buffer, used for clipping. Pixels with Y coordinates less than Ymin will not be affected.

	15:0	**Clear Rectangle X Min** Format: U16 Specifies Xmin value of (inclusive) of clear rectangle with the Depth Buffer, used for clipping. Pixels with X coordinates less than or equal to Xmin will not be affected.
2 **Programming Notes:** See the programming note in the previous DWORD for the Min values. The Clear/Resolve rectangleX and YMax values must be shifted by the LOD level; i.e. the hardware does not include the LOD in this function. Hence to clear any particular X, Y from the base level, to clear the contents at level "LOD" use (X»LOD) and (Y»LOD). The final X and Y Max values, after LOD adjustment described above, have to be manually 8x4 or 8x8 aligned for Depth and HZ Resolve passes only. For	31:16	**Clear Rectangle Y Max** Format: U16 Specifies Ymax value of (exclusive) of clear rectangle with the Depth Buffer, used for clipping. Pixels with Y coordinates greater than Ymax will be not be cleared.
	15:0	**Clear Rectangle X Max** Format: U16 Specifies Xmax value of (exclusive) of clear rectangle with the Depth Buffer, used for clipping. Pixels with X coordinates greater than or equal to Xmax will be not be affected.

3DSTATE_WM_HZ_OP_BODY

Clears see "Full Surface Depth and Stencil Clear" field in this command instead.
For multisample set to 1X together with Z format set to16 bpp:
resolve_aligned_y_max= (y_max & ~0x7) + ((y_max & 0x7 == 0) ? 0 : 8) //round up to next multiple of 8
resolve_aligned_x_max= (x_max & ~0x7) + ((x_max & 0x7 == 0) ? 0 : 8) //round up to next multiple of 8
All other cases:
resolve_aligned_y_max= (y_max & ~0x3) + ((y_max & 0x3 == 0) ? 0 : 4) //round up to next multiple of 4
resolve_aligned_x_max= (x_max & ~0x7) + ((x_max & 0x7 == 0) ? 0 : 8) //round up to next multiple of 8		

3	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Sample Mask** Format: `Right-justified bitmask (Bit 0 = Sample0). Number of bits that are used is determined by Num Multisamples (3DSTATE_WM_HZ_OP)` A per-multisample-position mask state variable that is immediately and unconditionally ANDed with the sample coverage mask as part of the rasterization process. This mask is applied prior to centroid selection. **Programming Notes** If Number of Multisamples is NUMSAMPLES_1, bits 15:1 of this field will be zeroed by HW.If Number of Multisamples is NUMSAMPLES_2, bits 15:2 of this field will be zeroed by HW.If Number of Multisamples is NUMSAMPLES_4, bits 15:4 of this field will be zeroed by HW.If Number of Multisamples is NUMSAMPLES_8, bits 15:8 of this field will be zeroed by HW.

A32 Buffer Base Address Message Header Control

MHC_A32_BBA - A32 Buffer Base Address Message Header Control

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:10	**Buffer Base Address Offset** Format: GeneralStateOffset[31:10] Specifies the base address offset page [31:10] for A32 stateless messages. **Restriction** When using stateless A32 Data Port messages, General State Base Address[47:12] + Buffer Base Address[31:10] must be less than 2^48. It is illegal for this to be greater or equal than 2^48.
	9:0	**Reserved** Access: RO Format: MBZ

A32 Scaled Header Present Message Descriptor Control Field

MDC_A32_MHP - A32 Scaled Header Present Message Descriptor Control Field

Size (in bits): 1

Default Value: 0x00000000

DWord	Bit	Description
0	0	**Message Header Present** Format: Boolean Specifies if the message uses the optional message header to modify the A32 address calculation, in combination with the MDC_A32_SSO field. **Value / Name / Description** 0h / No **[Default]** / Message header is not present. 1h / Yes / Message header is present. **Programming Notes** The access is Out-of-bounds if the SideB and Offset is enabled when the Message Header is not present.

A32 Sideband Scale and Offset Enable Message Descriptor Control Field

MDC_A32_SBSO - A32 Sideband Scale and Offset Enable Message Descriptor Control Field

Size (in bits): 8

Default Value: 0x00000000

DWord	Bit	Description
0	7	**Sideband Offset Enable** Format: MBO Must be set for a scaled SLM access. The 16-bit offset from the Sideband is added to all the offsets in the Address Payload for the SLM access. The 16-bit Sideband Offset is specified in the extended function control field in the SEND instruction.
	6:0	**Scale** Format: U7 Specifies the scale pitch to be used for SLM messages as (#bytes-1).

A64 Data Size Message Descriptor Control Field

MDC_A64_DS - A64 Data Size Message Descriptor Control Field

Size (in bits): 2

Default Value: 0x00000000

DWord	Bit	Description
0	1:0	**Data Size** Specifies the number of data elements to be read or written

Value	Name	Description
00h	DE1	1 data element (B, DW, QW)
01h	DE2	2 data elements (B, DW, QW)
02h	DE4	4 data elements (B, DW, QW)

Restriction
The number of elements is constrained by SIMD Mode and Data Width. The max data payload limit is 256B: 2 elements SIMD16 QW, 4 elements SIMD16 DW, or 4 elements SIMD8 QW.

A64 Hword Block Message Header

MH_A64_HWB - A64 Hword Block Message Header

Source:	EuSubFunctionDataPort1
Size (in bits):	256
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0..1	63:0	**BlockOffset** Format: U64 Specifies the U64 byte offset of Oword block. **Programming Notes** If the BlockOffset is not in the 48-bit canonical address range, the access is Out-of-Bounds. **Restriction** The byte offset must be aligned to the message's data type. Dwords have [1:0] = 0, Qwords have [2:0] = 0, and Hwords have [4:0] = 0.
2..4	95:0	**Reserved** Access: RO Format: MBZ
5	31:0	**Reserved** Access: RO Format: MBZ
6..7	63:0	**Reserved** Access: RO Format: MBZ

A64 Hword Data Blocks Message Descriptor Control Field

MDC_A64_DB_HW - A64 Hword Data Blocks Message Descriptor Control Field

Size (in bits): 3

Default Value: 0x00000001

DWord	Bit	Description
0	2:0	**Data Blocks** Specifies the number of Hwords to be read or written

Value	Name	Description
01h	HW1 **[Default]**	1 Hword block
02h	HW2	2 Hword blocks
03h	HW4	4 Hword blocks
04h	HW8	8 Hword blocks
Others	Reserved	Ignored

A64 Oword Block Message Header

MH_A64_OWB - A64 Oword Block Message Header		
Source:	EuSubFunctionDataPort1	
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0..1	63:0	**BlockOffset** Format: U64 Specifies the U64 byte offset of Oword block. **Programming Notes** If the BlockOffset is not in the 48-bit canonical address range, the access is Out-of-Bounds. **Restriction** The byte offset must be aligned to the message's data type. Dwords have [1:0] = 0, Qwords have [2:0] = 0, and Hwords have [4:0] = 0.
2..7	191:0	**Reserved** Access: RO Format: MBZ

A64 Oword Data Blocks Message Descriptor Control Field

MDC_A64_DB_OW - A64 Oword Data Blocks Message Descriptor Control Field

Size (in bits): 3

Default Value: 0x00000000

DWord	Bit	Description
0	2:0	**Data Blocks** Specifies the number of Oword blocks to be read or written

Value	Name	Description
00h	OW1L	1 Oword, read into or written from the low 128 bits of the destination register
01h	Reserved	Reserved
02h	OW2	2 Owords
03h	OW4	4 Owords
04h	OW8	8 Owords
Others	Reserved	Ignored

A64 Scaled Header Present Message Descriptor Control Field

MDC_A64_MHP - A64 Scaled Header Present Message Descriptor Control Field

Size (in bits): 1

Default Value: 0x00000000

DWord	Bit	Description
0	0	**Message Header Present** Specifies if the message uses the optional message header to modify the A64 address calculation, in combination with MDC_A64_SSO field. **Value / Name / Description** 0h / No / Message header is not present 1h / Yes / Message header is present **Programming Notes** The access is Out-of-Bounds if the SideBand Offset is enabled when the Message Header is not present.

AddrSubRegNum

AddrSubRegNum

Source: EuIsa

Size (in bits): 4

Default Value: 0x00000000

Address Subregister Number This field provides the subregister number for the address register. The address register contains 8 sub-registers. The size of each subregister is one word. The address register contains the register address of the operand, when the operand is in register-indirect addressing mode. This field applies to the destination operand and the source operands. It is ignored (or not present in the instruction word) for an immediate source operand. This field is present if the operand is in register-indirect addressing mode; it is not present if the operand is directly addressed. An address subregister used for indirect addressing is often called an index register.

DWord	Bit	Description
0	3:0	**Address Subregister Number** Value: 0-15 — Name: Address Subregister Number

Any Binding Table Index Message Descriptor Control Field

MDC_BTS_SLM_A32 - Any Binding Table Index Message Descriptor Control Field

Size (in bits): 8

Default Value: 0x00000000

DWord	Bit	Description
0	7:0	**Binding Table Index** Specifies the surface for the message, which can be Surface State Model, SLM or Stateless.

Value	Name	Description
00h-0EFh	BTS	Index of Binding Table State Surfaces
F0h-0FBh	Reserved	Reserved for future use
0FCh	SSO	Specifies a Surface State Offset supplied by the extended message descriptor
0FEh	SLM	Specifies an SLM access
0FFh	A32_A64	Specifies a A32 or A64 Stateless access that is locally coherent (coherent within a thread group)
0FDh	A32_A64_NC	Specifies a A32 or A64 Stateless access that is non-coherent (coherent within a thread).

Restriction
When using A32_A64_NC, SW must ensure that 2 threads do not both access the same cache line (64B)

Atomic Float Binary Operation Message Descriptor Control Field

MDC_FOP2 - Atomic Float Binary Operation Message Descriptor Control Field

Size (in bits): 3

Default Value: 0x00000001

DWord	Bit	Description
0	2:0	**Atomic Float Operation Type** Specifies the atomic float binary operation to be performed

Value	Name	Description	Programming Notes
01h	AOP_FMAX **[Default]**	new_dst = fmax(old_dst, src0)	The fmax operation implements the IEEE specification, which differs slightly from the DX and OCL specifications when a source operand is a sNaN. fmax(x,qNaN) = fmax(qNaN,x) = x fmax(x,sNaN) = fmax(sNaN,x) = quietize(sNaN) fmax(sNaN,sNaN) = fmax(sNaN,qNaN) = fmax(qNan,sNaN) = quietize(sNaN) fmax(qNaN,qNan) = qNaN fmax(-0, +0) = fmax(+0, -0) = +0 Fmax with sNaN operand returns sNaN instead of quietize(sNaN)
02h	AOP_FMIN	new_dst = fmin(old_dst, src0)	The fmin operation implements the IEEE specification, which differs slightly from the DX and OCL specifications when a source operand is a sNaN. fmin(x,qNaN) = fmin(qNaN,x) = x fmin(x,sNaN) = fmin(sNaN,x) = quietize(sNaN) fmin(sNaN,sNaN) = fmin(sNaN,qNaN) = fmin(qNan,sNaN) = quietize(sNaN) fmin(qNaN,qNan) = qNaN fmin(+0, -0) = fmin(-0, +0) = -0 Fmin with sNaN operand returns sNaN instead of quietize(sNaN) [] Fmin(+0,-0) returns +0. Should be -0, to match EU Fmin instruction.

Atomic Float Operation Message Descriptor Control Field

MDC_FOP - Atomic Float Operation Message Descriptor Control Field

Size (in bits): 3

Default Value: 0x00000000

DWord	Bit	Description
0	2:0	**Atomic Float Operation Type** Specifies the atomic float operation to be performed.

Value	Name	Description	Programming Notes
01h	AOP2_FMAX	new_dst = fmax(old_dst, src0) (default binary opcode)	The fmax operation implements the IEEE specification, which differs slightly from the DX and OCL specifications when a source operand is a sNaN. fmax(x,qNaN) = fmax(qNaN,x) = x fmax(x,sNaN) = fmax(sNaN,x) = quietize(sNaN) fmax(sNaN,sNaN) = fmax(sNaN,qNaN) = fmax(qNan,sNaN) = quietize(sNaN) fmax(qNaN,qNan) = qNaN fmax(-0, +0) = fmax(+0, -0) = +0 Fmax with sNaN operand returns sNaN instead of quietize(sNaN)
02h	AOP2_FMIN	new_dst = fmin(old_dst, src0)	The fmin operation implements the IEEE specification, which differs slightly from the DX and OCL specifications when a source operand is a sNaN. fmin(x,qNaN) = fmin(qNaN,x) = x fmin(x,sNaN) = fmin(sNaN,x) = quietize(sNaN) fmin(sNaN,sNaN) = fmin(sNaN,qNaN) = fmin(qNan,sNaN) = quietize(sNaN) fmin(qNaN,qNan) = qNaN fmin(+0, -0) = fmin(-0, +0) = -0 Fmin with sNaN operand returns sNaN instead of quietize(sNaN) [] Fmin(+0,-0) returns +0. Should be -0, to match EU Fmin instruction.
03h	AOP3_FCMPWR	new_dst = (src0 == old_dst) ? src1 : old_dst (default ternary opcode)	The fcmpwr operation performs the comparison using IEEE specification rules, and performs the store as a raw move (so SNaN is not quietized).fcmpwr(NaN,x,y) = NaNfcmpwr(x, NaN,y) = xfcmpwr(x,x, NaN) = NaN
Others	Reserved	Ignored	

Programming Notes
Binary opcodes AOP2_FMAX, AOP2_FMIN, AOP2_FADD, AOP2_FSUB, AOP2_FADD_64b, and AOP2_FSUB_64b have one source data payload. Ternary opcode AOP3_FCMPWR has two source data payloads.

Atomic Float Ternary Operation Message Descriptor Control Field

MDC_FOP3 - Atomic Float Ternary Operation Message Descriptor Control Field

Size (in bits): 3

Default Value: 0x00000003

DWord	Bit	Description
0	2:0	**Atomic Float Operation Type** Specifies the atomic float ternary operation to be performed

Value	Name	Description	Programming Notes
03h	AOP_FCMPWR **[Default]**	new_dst = (src0 == old_dst) ? src1 : old_dst	The fcmpwr operation performs the comparison using IEEE specification rules, and performs the store as a raw move (so SNaN is not quietized).fcmpwr(NaN,x,y) = NaNfcmpwr(x, NaN,y) = xfcmpwr(x,x, NaN) = NaN
Others	Reserved	Ignored	

Programming Notes
When Return Data Control is set, old_dst is returned.

Atomic Half Float Binary Operation Message Descriptor Control Field

MDC_HFOP2 - Atomic Half Float Binary Operation Message Descriptor Control Field

Source: BSpec
Size (in bits): 3
Default Value: 0x00000001

DWord	Bit	Description
0	2:0	**Atomic Float Operation Type** Specifies the atomic float binary operation to be performed

Value	Name	Description	Programming Notes
01h	AOP_FMAX **[Default]**	new_dst = fmax(old_dst, src0)	The fmax operation implements the IEEE specification, which differs slightly from the DX and OCL specifications when a source operand is a sNaN. fmax(x,qNaN) = fmax(qNaN,x) = x fmax(x,sNaN) = fmax(sNaN,x) = quietize(sNaN) fmax(sNaN,sNaN) = fmax(sNaN,qNaN) = fmax(qNan,sNaN) = quietize(sNaN) fmax(qNaN,qNan) = qNaN fmax(-0, +0) = fmax(+0, -0) = +0 Fmax with sNaN operand returns sNaN instead of quietize(sNaN)
02h	AOP_FMIN	new_dst = fmin(old_dst, src0)	The fmin operation implements the IEEE specification, which differs slightly from the DX and OCL specifications when a source operand is a sNaN. fmin(x,qNaN) = fmin(qNaN,x) = x fmin(x,sNaN) = fmin(sNaN,x) = quietize(sNaN) fmin(sNaN,sNaN) = fmin(sNaN,qNaN) = fmin(qNan,sNaN) = quietize(sNaN) fmin(qNaN,qNan) = qNaN fmin(+0, -0) = fmin(-0, +0) = -0 Fmin with sNaN operand returns sNaN instead of quietize(sNaN) [] Fmin(+0,-0) returns +0. Should be -0, to match EU Fmin instruction.

Atomic Integer Binary Operation Message Descriptor Control Field

MDC_AOP2 - Atomic Integer Binary Operation Message Descriptor Control Field

Size (in bits): 4

Default Value: 0x00000001

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>3:0</td><td>Atomic Integer Operation Type
Specifies the atomic integer binary operation to be performed
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>01h</td><td>AOP_AND [Default]</td><td>new_dst = old_dst AND src0</td></tr>
<tr><td>02h</td><td>AOP_OR</td><td>new_dst = old_dst | src0</td></tr>
<tr><td>03h</td><td>AOP_XOR</td><td>new_dst = old_dst ^ src0</td></tr>
<tr><td>04h</td><td>AOP_MOV</td><td>new_dst = src0</td></tr>
<tr><td>07h</td><td>AOP_ADD</td><td>new_dst = old_dst + src0</td></tr>
<tr><td>08h</td><td>AOP_SUB</td><td>new_dst = old_dst - src0</td></tr>
<tr><td>09h</td><td>AOP_REVSUB</td><td>new_dst = src0 - old_dst</td></tr>
<tr><td>0Ah</td><td>AOP_IMAX</td><td>new_dst = imax(old_dst, src0)</td></tr>
<tr><td>0Bh</td><td>AOP_IMIN</td><td>new_dst = imin(old_dst, src0)</td></tr>
<tr><td>0Ch</td><td>AOP_UMAX</td><td>new_dst = umax(old_dst, src0)</td></tr>
<tr><td>0Dh</td><td>AOP_UMIN</td><td>new_dst = umin(old_dst, src0)</td></tr>
<tr><td>Others</td><td>Reserved</td><td>Ignored</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>When Return Data Control is set, old_dst is returned.</td></tr>
</table></td></tr>
</table>

Atomic Integer Operation Message Descriptor Control Field

MDC_AOP - Atomic Integer Operation Message Descriptor Control Field

Size (in bits): 4

Default Value: 0x00000000

DWord	Bit	Description
0	3:0	**Atomic Integer Operation Type** Specifies the atomic integer operation to be performed.

Value	Name	Description
05h	AOP1_INC	new_dst = old_dst + 1 (default unary opcode)
06h	AOP1_DEC	new_dst = old_dst - 1
0Fh	AOP1_PREDEC	new_dst = old_dst - 1
01h	AOP2_AND	new_dst = old_dst AND src0 (default binary opcode)
02h	AOP2_OR	new_dst = old_dst | src0
03h	AOP2_XOR	new_dst = old_dst ^ src0
04h	AOP2_MOV	new_dst = src0
07h	AOP2_ADD	new_dst = old_dst + src0
08h	AOP2_SUB	new_dst = old_dst - src0
09h	AOP2_REVSUB	new_dst = src0 - old_dst
0Ah	AOP2_IMAX	new_dst = imax(old_dst, src0)
0Bh	AOP2_IMIN	new_dst = imin(old_dst, src0)
0Ch	AOP2_UMAX	new_dst = umax(old_dst, src0)
0Dh	AOP2_UMIN	new_dst = umin(old_dst, src0)
00h	AOP3_CMPWR_2W	new_dst = (src0_2W == old_dst_2W) ? src1_2W : old_dst_2W
0Eh	AOP3_CMPWR	new_dst = (src0 == old_dst) ? src1 : old_dst (default ternary opcode)
Others	Reserved	Ignored

Programming Notes
For Unary Opcodes, when Return Data Control is set, new_dst is returned by AOP_PREDEC and otherwise old_dst is returned. For Binary and Ternary Opcodes, when Return Data Control is set, old_dst is returned.
Unary Opcodes AOP1_* do not have a source data payload. Binary Opcodes AOP2_* have one source data payload. Ternary Opcodes AOP3_* have two source data payloads.

Atomic Integer Ternary Operation Message Descriptor Control Field

MDC_AOP3 - Atomic Integer Ternary Operation Message Descriptor Control Field

Size (in bits): 4

Default Value: 0x0000000E

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>3:0</td><td>Atomic Integer Operation Type
Specifies the atomic integer ternary operation to be performed
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>00h</td><td>AOP_CMPWR_2W</td><td>new_dst = (src0_2W == old_dst_2W) ? src1_2W : old_dst_2W</td></tr>
<tr><td>0Eh</td><td>AOP_CMPWR [Default]</td><td>new_dst = (src0 == old_dst) ? src1 : old_dst</td></tr>
<tr><td>Others</td><td>Reserved</td><td>Ignored</td></tr>
</table>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>When Return Data Control is set, old_dst is returned.</td></tr>
</table></td></tr>
</table>

Atomic Integer Unary Operation Message Descriptor Control Field

MDC_AOP1 - Atomic Integer Unary Operation Message Descriptor Control Field

Size (in bits): 4

Default Value: 0x00000005

DWord	Bit	Description
0	3:0	**Atomic Integer Operation Type** Specifies the atomic integer unary operation to be performed

Value	Name	Description
05h	AOP_INC **[Default]**	new_dst = old_dst + 1
06h	AOP_DEC	new_dst = old_dst - 1
0Fh	AOP_PREDEC	new_dst = old_dst - 1
Others	Reserved	Ignored

Programming Notes
When Return Data Control is set, new_dst is returned by AOP_PREDEC and otherwise old_dst is returned.

Audio Power State Format

Audio Power State Format		
Size (in bits):	2	
Default Value:	0x00000003	
DWord	**Bit**	**Description**
0	1:0	**Power State** **Value** / **Name** / **Description** 00b / D0 / D0 01b,10b / Unsupported / Unsupported 11b / D3 **[Default]** / D3

AVC CABAC

AVC CABAC		
Source:	VideoCS	
Size (in bits):	16	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	15	**Reserved** Access: RO Format: MBZ
	14	**Coefficient level out-of-bound Error** This flag indicates the coded coefficient level SEs in the bit-stream is out-of-bound.
	13:12	**Reserved** Access: RO Format: MBZ
	11	**Temporal Direction Motion Vector Out-of-Bound Error** This flag indicates motion vectors calculated from Temporal Direct Motion Vector is larger than the allowed range specified by the AVC spec.
	10	**Reserved** Access: RO Format: MBZ
	9	**Motion Vector Delta SE Out-of-Bound Error** This flag indicates inconsistent Motion Vector Delta SEs coded in the bit-stream.
	8	**Reference Index SE Out-of-Bound Error** This flag indicates inconsistent Reference Index SEs coded in the bit-stream.
	7	**MacroBlock QpDelta Error** This flag indicates out-of-bound MB QP delta SEs coded in the bit-stream.
	6	**Motion Vector Delta SE Error** This flag indicates out-of-bound motion vector delta SEs coded in the bit-stream.
	5	**Reference Index SE Error** This flag indicates out-of-bound Refidx SEs coded in the bit-stream.
	4	**Residual Error** This flag indicates out-of-bound absolute coefficient level SEs coded in the bit-stream.
	3	**Slice end Error** This flag indicates a pre-matured slice_end SE or inconsistent slice end on the last MB of a slice.
	2	**Chroma Intra prediction Mode Error** This flag indicates inconsistent Chroma Intra prediction mode SEs coded in the bit-stream.
	1	**Luma Intra prediction Mode Error** This flag indicates inconsistent luma Intra prediction mode SE coded in the bit-stream.

AVC CABAC		
	0	**MB Concealment Flag** Each pulse from this flag indicates one MB is concealed by hardware.

AVC CAVLC

AVC CAVLC		
Source:	VideoCS	
Size (in bits):	16	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	15	**Total Zero out-of-bound Error** This flag indicates the Total zero SE count exceed the max number of coeffs allowed in an intra16x16 AC block.
	14	**Coefficient level out-of-bound Error** This flag indicates the coded coefficient level SEs in the bit-stream is out-of-bound.
	13	**RunBefore out-of-bound Error** This flag indicates the coded RunBefore SE value is larger than the remaining zero block count.
	12	**Total coefficient Out-of-bound Error** This flag indicates the coded total coeff SE count exceed the max number of coeffs allowed in an intra16x16 AC block.
	11	**Temporal Direction Motion Vector Out-of-Bound Error** This flag indicates motion vectors calculated from Temporal Direct Motion Vector is larger than the allowed range specified by the AVC spec.
	10	**Reserved** Access: RO Format: MBZ
	9	**Motion Vector Delta SE Out-of-Bound Error** This flag indicates inconsistent Motion Vector Delta SEs coded in the bit-stream.
	8	**Reference Index SE Out-of-Bound Error** This flag indicates inconsistent Reference Index SEs coded in the bit-stream.
	7	**RunBefore/TotalZero Error** This flag indicates one or more inconsistent RunBefore or TotalZero SEs coded in the bit-stream.
	6	**Exponential Golomb Error** This flag indicates hardware detects more than 18 leadzero for skip and more than 19 for other SEs from the Exponential Golomb Logic
	5	**Total Coeff SE Error** This flag indicates one or more inconsistent total coeff SEs coded in the bit-stream.
	4	**Macroblock Coded Block Pattern Error** This flag indicates inconsistent CBP SEs coded in the bit-stream.
	3	**Mbytpe/submbtype Error** This flag indicates inconsistent MBtype/SubMBtype SEs coded in the bit-stream.
	2	**Chroma Intra prediction Mode Error** This flag indicates inconsistent Chroma Intra prediction mode SEs coded in the bit-stream.

AVC CAVLC		
	1	**Luma Intra prediction Mode Error** This flag indicates inconsistent luma Intra prediction mode SE coded in the bit-stream.
	0	**MB Concealment Flag** Each pulse from this flag indicates one MB is concealed by hardware.

AVP_PAK_INSERT_OBJECT_INDIRECT_PAYLOAD

AVP_PAK_INSERT_OBJECT_INDIRECT_PAYLOAD		
Source:	VideoCS	
Size (in bits):	128	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0	31:0	**Indirect Payload Data Size in bits** Format: U32 Number of bits to be inserted. Not including those skipped bytes in the beginning.
1..2	63:0	**Indirect Payload Base Address** Format: SplitBaseAddress64ByteAligned 48-bit address of the indirect payload data in memory buffer. **Programming Notes** Payload must begin in a byte position, but the payload can be ended in a bit position.
3	31:0	**Indirect Payload Base Address** Format: MemoryAddressAttributes

AVP_REF_LIST_ENTRY_OLD

AVP_REF_LIST_ENTRY_OLD		
Source:	VideoCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:15	**Reserved** Access: RO Format: MBZ
	14:0	**Reference Picture Frame ID[i]** Format: U15 This array corresponding to the array **Reference Picture Address (RefAddr[0-7])**defined in the AVP_PIPE_BUF_ADDR_STATE command. Frame ID is in decoding order (not display order). Frame ID is a unique number identifying a reference frame. It is 15-bit quantity and wraps around after 2^15-1

Barrier Data Payload

MDP_Barrier - Barrier Data Payload

Source:	EuSubFunctionGateway
Size (in bits):	256
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0..1	63:0	**Reserved** Access: RO Format: MBZ
2	31	**Reserved** Access: RO Format: MBZ
	30:24	**Barrier ID** Format: U7 This field indicates which barrier state is updated. Range = [0,63]
	23	**Predicate Mask Enable** Format: Enable This bit indicates that the barrier is a predicated barrier and the SIMD channels passing the predicate should be summed. All threads sending this message to the same barrier should have an identical value for this field, and must specify a response length of 1 for the predicate sum response. Note that Global Barriers must not have the Predicate Mask Enable bit set. **Programming Notes** This control is intended only for GPGPU or Media threads. This control must not be set if the barrier is for a Hull Shader thread.
	22:16	**Reserved** Access: RO Format: MBZ
	15	**Barrier Count Enable** Format: Enable Allows the message to reprogram the terminating barrier count. If set, the stored value of the terminating barrier count is set to the value of Barrier Count field (below), and used for this barrier operation. If clear, the stored value of the terminating barrier count is not modified and the stored value is used for this barrier operation. **Programming Notes** This control is intended only for Hull Shader threads. This control must not be set if the barrier is allocated by a GPGPU or Media thread.

MDP_Barrier - Barrier Data Payload

	14:8	**Barrier Count** Format: U7 If Barrier Count Enable is set, this field specifies the terminating barrier count. Otherwise this field is ignored. All threads that belong to a single barrier must deliver the same value for this field for a particular barrier iteration.
	7:0	**Reserved** Access: RO Format: MBZ
3	31:0	**Predicate Mask** Format: U32 This field has a bit set per SIMD channel that passes the predicate. For SIMD8 and SIMD16 the rest of the bits must be 0. This field is ignored for non-predicated barriers.
4..7	127:0	**Reserved** Access: RO Format: MBZ

BaseAddress4KByteAligned

BaseAddress4KByteAligned		
Size (in bits):	64	
Default Value:	0x00000000, 0x00000000	
Specifies a 64-bit (48-bit canonical) 4K-byte aligned memory base address.		
DWord	**Bit**	**Description**
0..1	63:12	**Base Address** Format: VIRTUAL_ADDR[63:12]
	11:0	**Reserved** Access: RO Format: MBZ

Batch Buffer Stack Structure

BATCH_BUFFER_STACK_STRUCTURE - Batch Buffer Stack Structure		
Size (in bits):	192	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0..5	191:0	**Batch Stack Entries** Format: **BATCH_STACK_ENTRY[3]** Stack containing details of the batch buffers currently in execution. The top of stack is determined by the Batch Buffer Stack Pointer.

Batch Stack Entry

BATCH_STACK_ENTRY - Batch Stack Entry

Size (in bits): 64

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0..1	63:62	**Reserved** Access: RO Format: MBZ
	61:60	**Reserved** Access: RO Format: MBZ
	59	**POSH Start** Exists If: //POCS
	58	**POSH Enable** Exists If: //RCS, POCS This bit reflects the POSH Enable value programmed by the active MI_BATCH_BUFFER_START command.
	57	**Address Space Indicator** Format: U1 This field reflects the effective address space indicator security level and may not be the same as the Address Space Indicator written using MI_BATCH_BUFFER_START. Value / Name / Description 0h / GGTT **[Default]** / This Batch buffer is located in GGTT memory and is privileged. 1h / PPGTT / This Batch buffer is located in PPGTT memory and is non-privileged.
	56	**Reserved**
	55:48	**Reserved** Access: RO Format: MBZ
	47:2	**Batch Buffer Head Pointer** Format: GraphicsAddress[47:2]
	1:0	**Reserved** Access: RO Format: MBZ

BCS Hardware-Detected Error Bit Definitions

BCS Hardware-Detected Error Bit Definitions

Source: BlitterCS
Size (in bits): 16
Default Value: 0x00000000

DWord	Bit	Description
0	15:12	**Reserved** Access: RO Format: MBZ
	11	**Reserved** Access: RO Format: MBZ
	10:3	**Reserved** Access: RO Format: MBZ
	2	**Command Privilege Violation Error** This bit is set if a command classified as privileged is parsed in a non-privileged batch buffer. The command will be converted to a NOOP and parsing will continue.
	1	**Reserved** Access: RO Format: MBZ
	0	**Instruction Error** This bit is set when the Renderer Instruction Parser detects an error while parsing an instruction. Instruction errors include: • Client ID value (Bits 31:29 of the Header) is not supported (only MI, 2D and 3D are supported). • Defeatured MI Instruction Opcodes: Value: 1 — Name: — Description: Instruction Error detected **Programming Notes** This error indications cannot be cleared except by reset (i.e., it is a fatal error).

BINDING_TABLE_EDIT_ENTRY

BINDING_TABLE_EDIT_ENTRY		
Source:	RenderCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**Binding Table Index** Format: U8 This field specifies the index of binding table entry that will be updated.
	15:0	**Surface State Pointer** Format: SurfaceStateOffset[21:6]RENDER_SURFACE_STATE Surface State Pointer. This address points to a surface state block. This pointer is relative to the Surface State Base Address.

Bit Definition for Interrupt Control Registers - Media

Bit Definition for Interrupt Control Registers - Media

Source: VideoCS
Size (in bits): 32
Default Value: 0x00000000

DWord	Bit	Description
0	31:12	**Reserved** Access: RO Format: MBZ
	11	**Wait on Semaphore** Exec-List Scheduling: Set when MI_SEMAPHORE_WAIT command is un-successful and when "Inhibit Synchronous Context Switch" is set. Scheduler can use this interrupt to preempt the context waiting on semaphore wait. Ring Buffer Scheduling: Set when MI_SEMAPHORE_WAIT command is un-successful.
	10	**Reserved** Access: RO Format: MBZ
	9	**Reserved**
	8	**Context Switch Interrupt** Set when a context switch has just occurred. **Execlist Enable bit** needs to be set for this interrupt to occur.
	7	**Reserved** Access: RO Format: MBZ
	6	**Timeout Counter Expired** Set when the VCS timeout counter has reached the timeout thresh-hold value.
	5	**Reserved**
	4	**MI_FLUSH_DW Notify Interrupt** The Pipe Control packet (Fences) specified in 3D pipeline document may optionally generate an Interrupt. The Store QW associated with a fence is completed ahead of the interrupt.
	3	**Video Command Parser Error** When this status bit is set, it indicates that the hardware has detected an error. It is set by the device upon an error condition and cleared by a CPU write of a one to the appropriate bit contained in the Error ID register followed by a write of a one to this bit in the IIR. Further information on the source of the error comes from the "Error Status Register" which along with the "Error Mask Register" determine which error conditions will cause the error status bit to be set and the interrupt to occur. **Page Table Error:** Indicates a page table error. **Instruction Parser Error:** The Blitter Instruction Parser encounters an error while parsing an

Bit Definition for Interrupt Control Registers - Media		
		instruction.
	2:1	**Reserved** Access: RO Format: MBZ
	0	**Video Command Parser User Interrupt** This status bit is set when an MI_USER_INTERRUPT instruction is executed on the Video Command Parser. Note that instruction execution is not halted and proceeds normally. A mechanism such as an MI_STORE_DATA instruction is required to associate a particular meaning to a user interrupt.

BLEND_STATE_ENTRY

BLEND_STATE_ENTRY

Size (in bits): 64

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0..1	63	**Logic Op Enable** Format: Enable Enables the LogicOp function of the Pixel Processing pipeline. **Programming Notes** Enabling LogicOp and Color Buffer Blending at the same time is UNDEFINED
	62:59	**Logic Op Function** Format: 3D_Logic_Op_Function This field specifies the function to be performed (when enabled) in the Logic Op stage of the Pixel Processing pipeline. Note that the encoding of this field is one less than the corresponding "R2_" ROP code defined in WINGDI.H, and is a rather contorted mapping of the OpenGL LogicOp encodings. However, this field was defined such that, when the 4 bits are replicated to 8 bits, they coincide with the ROP codes used in the Blter. Note: if the Logic Op Function does not depend on "D", the dest buffer is not read.
	58:37	**Reserved** Access: RO Format: MBZ
	36	**Pre-Blend Source Only Clamp Enable** Format: Enable This field specifies whether the source(s) are clamped prior to blending, regardless of whether blending is enabled. If DISABLED, no clamping is performed prior to blending. If ENABLED, only source0 and source 1, if dual source is enabled, are clamped prior to the blend to the range specified by Color Clamp Range.

Value	Name	Description
0	Disabled	No clamping is performed prior to blending.
1	Enabled	Only Source(s) are clamped prior to blend function. Other inputs to blend must be clamped according to the behavior specified for "pre-blend color clamp disable" in the pre-blend color clamping table .

Programming Notes

This field is ignored (treated as DISABLED) for UINT and SINT RT surface formats. Blending is not supported for those RT surface formats. **When this bit is enabled Pre-Blend Color Clamp Enable must be disabled.**

BLEND_STATE_ENTRY

35:34 **Color Clamp Range**

Specifies the clamped range used in Pre-Blend and Post-Blend Color Clamp functions if one or both of those functions are enabled. Note that this range selection is shared between those functions.

Value	Name	Description
0	COLORCLAMP_UNORM	Clamp Range [0,1]
1	COLORCLAMP_SNORM	Clamp Range [-1,1]
2	COLORCLAMP_RTFORMAT	Clamp to the range of the RT surface format (Note: The Alpha component is clamped to FLOAT16 for R11G11B10_FLOAT format). Unsigned Floating Point components are clamped to positive zero.
3	Reserved	Reserved

Programming Notes
See table in Pre-Blending Color Clamp subsection for programming restrictions as a function of Blending and Pre-Blend Color Clamp.

33 **Pre-Blend Color Clamp Enable**

Format:	Enable

This field specifies whether the source, destination and constant color channels are clamped prior to blending, regardless of whether blending is enabled.If DISABLED, no clamping is performed prior to blending.If ENABLED, all inputs to the blend function are clamped prior to the blend to the range specified by Color Clamp Range.

Value	Name	Description
0	Disabled	No clamping is performed prior to blending.
1	Enabled	All inputs to the blend function are clamped prior to the blend to the range specified by Color Clamp Range.

Programming Notes
See table in Pre-Blending Color Clamp subsection for programming restrictions as a function of RT format. This field is ignored (treated as DISABLED) for UINT and SINT RT surface formats. Blending is not supported for those RT surface formats. The device will automatically clamp source color channels to the respective RT surface range.

32 **Post-Blend Color Clamp Enable**

Format:	Enable

Regardless of whether this clamping is enabled, the blending output channels will be clamped to the RT surface format just prior to being written.

Programming Notes
This field is ignored (treated as DISABLED) for UINT and SINT RT surface formats. Blending is not supported for those RT surface formats. Post Blend Clamp Enable must be programmed identical to Pre Blend Clamp Enable. The device will automatically clamp source color channels to the respective RT surface range. **When this bit is enabled Pre-Blend Source Only Clamp**

BLEND_STATE_ENTRY

	Enable must be disabled.
31	**Color Buffer Blend Enable** Format: Enable Enables the ColorBufferBlending (nee "alpha blending") function of the Pixel Processing Pipeline for this render target. **Programming Notes** Enabling LogicOp and ColorBufferBlending at the same time is UNDEFINED
30:26	**Source Blend Factor** Format: 3D_Color_Buffer_Blend_Factor Controls the "source factor" in the ColorBufferBlending function.Refer to Source Alpha Blend Factor for encodings.
25:21	**Destination Blend Factor** Format: 3D_Color_Buffer_Blend_Factor Controls the "destination factor" in the ColorBufferBlending function. Refer to Source Alpha Blend Factor for encodings.
20:18	**Color Blend Function** Format: 3D_Color_Buffer_Blend_Function This field specifies the function used to combine the color components in the ColorBufferBlending function of the Pixel Processing Pipeline. If Independent Alpha Blend Enable is disabled, this field will also control the blending of the alpha components in the ColorBufferBlending function.
17:13	**Source Alpha Blend Factor** Format: 3D_Color_Buffer_Blend_Factor Controls the "source factor" in alpha Color Buffer Blending stage.Note: For the source/destination alpha blend factors, the encodings indicating "COLOR" are the same as the encodings indicating "ALPHA", as the alpha component of the color is selected.
12:8	**Destination Alpha Blend Factor** Format: 3D_Color_Buffer_Blend_Factor Controls the "destination factor" in alpha Color Buffer Blending stage. Refer to Source Alpha Blend Factor for encodings.
7:5	**Alpha Blend Function** Format: 3D_Color_Buffer_Blend_Function This field specifies the function used to combine the alpha components in the Color Buffer blend stage of the Pixel Pipeline when the IndependentAlphaBlend state is enabled.
4	**Reserved** Access: RO Format: MBZ
3	**Write Disable Alpha** Format: Disable This field controls the writing of the alpha component into the Render Target.

BLEND_STATE_ENTRY

Value	Name	Description
0b	Enabled	Alpha component can be overwritten
1b	Disabled	Writes to the color buffer will not modify Alpha.

Programming Notes
For YUV surfaces, this field must be set to 0B (enabled).

2 **Write Disable Red**

Format:	Disable

This field controls the writing of the red component into the Render Target.

Value	Name	Description
0b	Enabled	Red component can be overwritten
1b	Disabled	Writes to the color buffer will not modify Red.

Programming Notes
For YUV surfaces, this field must be set to 0B (enabled).

1 **Write Disable Green**

Format:	Disable

This field controls the writing of the green component into the Render Target.

Value	Name	Description
0b	Enabled	Green component can be overwritten
1b	Disabled	Writes to the color buffer will not modify Green.

Programming Notes
For YUV surfaces, this field must be set to 0B (enabled).

0 **Write Disable Blue**

Format:	Disable

This field controls the writing of the Blue component into the Render Target.

Value	Name	Description
0b	Enabled	Blue component can be overwritten
1b	Disabled	Writes to the color buffer will not modify Blue.

Programming Notes
For YUV surfaces, this field must be set to 0B (enabled).

BLEND_STATE

BLEND_STATE

Size (in bits): 544

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

The blend state is stored as a structure containing a common DWORD that applies to all RTs and an array of up to 8 elements, each of which contains the two DWords for each. The start of each element is spaced 2 DWords apart. The blend state is aligned to a 64-byte boundary, which is pointed to by a field in 3DSTATE_BLEND_STATE_POINTERS. The 3-bit Render Target Index field in the Render Target Write data port message header is used to select which of the 8 elements from BLEND_STATE that is used on the current message.

DWord	Bit	Description
0	31	**Alpha To Coverage Enable** Format: Enable If set, Source0 Alpha is converted to a temporary 1/2/4-bit coverage mask and the mask bit corresponding to the sample# ANDed with the sample mask bit. If set, sample coverage is computed based on src0 alpha value. Value of 0 disables all samples and value of 1 enables all samples for that pixel. The same coverage needs to apply to all the RTs in MRT case. Further, any value of src0 alpha between 0 and 1 monotonically increases the number of enabled pixels.The field is applied to all the RTs in MRT case.
	30	**Independent Alpha Blend Enable** Format: Enable When enabled, the other fields in this instruction control the combination of the alpha components in the Color Buffer Blend stage. When disabled, the alpha components are combined in the same fashion as the color components.The field is applied to all the RTs in MRT case.
	29	**Alpha To One Enable** Format: Enable If set, Source0 Alpha is set to 1.0f after (possibly) being used to generate the AlphaToCoverage coverage mask.If Dual Source Blending is enabled, this bit must be disabled.The field is applied to all the RTs in MRT case.
	28	**Alpha To Coverage Dither Enable** Format: Enable If set, sample coverage is computed based on src0 alpha value and it modulates the sample coverage based on screen coordinates. Value of 0 disables all samples and value of 1 enables all samples for that pixel. The same coverage needs to apply to all the RTs in MRT case. Further, any value of src0 alpha between 0 and 1 monotonically increases the number of enabled pixels. If AlphaToCoverage is disabled, AlphaToCoverage Dither does not have any impact.The field is applied to all the RTs in MRT case.

BLEND_STATE

	27	**Alpha Test Enable** Format: Enable Enables the AlphaTest function of the Pixel Processing pipeline. The field is applied to all the RTs in MRT case. **Programming Notes** Alpha Test can only be enabled if Pixel Shader outputs a float alpha value. Alpha Test is applied independently on each render target by comparing that render target's alpha value against the alpha reference value. If the alpha test fails, the corresponding pixel write will be supressed only for that render target. The depth/stencil update will occur if alpha test passes for any render target.
	26:24	**Alpha Test Function** Format: 3D_Compare_Function This field specifies the comparison function used in the AlphaTest function The field is applied to all the RTs in MRT case.
	23	**Color Dither Enable** Format: Enable Enables dithering of colors (including any alpha component) before they are written to the Color Buffer. The field is applied to all the RTs in MRT case. **Programming Notes** For YUV render target formats, this field must be programmed to 0.
	22:21	**X Dither Offset** Format: U2 Specifies offset to apply to pixel X coordinate LSBs when accessing dither table. The field is applied to all the RTs in MRT case.
	20:19	**Y Dither Offset** Format: U2 Specifies offset to apply to pixel Y coordinate LSBs when accessing dither table. The field is applied to all the RTs in MRT case.
	18:0	**Reserved** Access: RO Format: MBZ
1..16	511:0	**Entry** Format: BLEND_STATE_ENTRY[8]

Blitter Interrupt Vector

BLITTER_INTR_VEC - Blitter Interrupt Vector		
Size (in bits):	16	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	15	**Catastrophic Error** This interrupt signals that a unrecoverable error (for e.g encountered fault when accessing a page mapped in Global GTT) during the engine processing. When Memory interface signals this error, the Command Streamer will stop parsing any more instructions. Scheduler is expected to reset the engine to evict the context
	14:12	**Reserved** Access: RO Format: MBZ
	11	**BCS Wait On Semaphore**
	10	**Reserved** Access: RO Format: MBZ
	9	**CS TR Invalid Tile Detection**
	8	**BCS Context Switch Interrupt**
	7	**Legacy Context Per Process Page Fault Interrupt** Fault interrupt is generated by GA fabric, not by the CS This interrupt is for handling Legacy context PPGTT Page Fault.
	6	**BCS Watchdog Counter Expired**
	5	**Reserved** Access: RO Format: MBZ
	4	**BCS MI Flush DW Notify**
	3	**BCS Error Interrupt**
	2:1	**Reserved** Access: RO Format: MBZ
	0	**BCS MI User Interrupt**

Block Dimensions Message Header Control

MHC_BDIM - Block Dimensions Message Header Control

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:22	**Reserved** Access: RO Format: MBZ
	21:20	**Block Height** Height in rows of block being accessed. Range = [0,3] representing 1 to 8 rows. 0h – H1 – Block height = 1 row 1h – H2 – Block height = 2 rows 2h – H4 – Block height = 4 rows 03h – H8 – Block height = 8 rows
	19:2	**Reserved** Access: RO Format: MBZ
	1:0	**Block Width** Width in Dwords of block being accessed. Range = [0,3] representing 1 to 8 Dwords. 0h – W1 – Block width = 1 Dword 1h – W2 – Block width = 2 Dwords 2h – W4 – Block width = 4 Dwords 03h – W8 – Block width = 8 Dwords

Block Message Header

MH_BTS_GO - Block Message Header

Source: EuSubFunctionDataPort0

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0..1	63:0	**Reserved** Access: RO Format: MBZ
2	31:0	**Global Offset** Format: U32 Specifies the global element index into the buffer, in units of Hwords, Owords, Dwords, or Bytes (depending on the message). **Programming Notes** The Global Offset for the Aligned Block operations is specified as a Dword-aligned byte offset (offset bits [1:0] = 0), Oword-aligned byte offset (offset bits [3:0]=0), or Hword-aligned byte offset (offset bits [4:0]=0). If the address offset calculated with the Global Offset is greater than the Surface Size, then the access is Out-of-Bounds.
3..7	159:0	**Reserved** Access: RO Format: MBZ

BR00 - BLT Opcode and Control

BR00 - BLT Opcode and Control

Source: BlitterCS

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31	**BLT Engine Busy** This bit indicates whether the BLT Engine is busy (1) or idle (0). This bit is replicated in the SETUP BLT Opcode and Control register. Value: 0 — Name: Idle **[Default]** Value: 1 — Name: Busy
	30	**Setup Instruction Instruction** Default Value: 0 The current instruction performs clipping (1).
	29	**Setup Monochrome Pattern** This bit is decoded from the Setup instruction opcode to identify whether a color (0) or monochrome (1) pattern is used with the SCANLINE_BLT instruction. Value: 0 — Name: Color **[Default]** Value: 1 — Name: Monochrome
	28:22	**Instruction Target (Opcode)** Default Value: 0000000b This is the contents of the Instruction Target field from the last BLT instruction. This field is used by the BLT Engine state machine to identify the BLT instruction it is to perform. The opcode specifies whether the source and pattern operands are color or monochrome.
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. Value: 00b — Name: **[Default]** Value: 1xb — Name: Write Alpha Channel Value: x1b — Name: Write RGB Channel
	19:17	**Monochrome Source Start** Default Value: 000b This field indicates the starting monochrome pixel bit position within a byte per scan line of the source operand. The monochrome source is word aligned which means that at the end of the scan line all bits should be discarded until the next word boundary.

BR00 - BLT Opcode and Control

16 Bit/Byte Packed

Byte packed is for the NT driver.

Value	Name
0b	Bit **[Default]**
1b	Byte

15 Src Tiling Enable

Value	Name
0b	Tiling Disabled (Linear) **[Default]**
1b	Tiling enabled: Tile-X or Tile-Y

14:12 Horizontal Pattern Seed

Default Value:	0b

This field indicates the pattern pixel position which corresponds to X = 0.

11 Dest Tiling Enable

When set to '1', this means that Blitter is executing in Tiled mode. If '0' it means that Blitter is in Linear mode. Pre-Dev Blitter never executes in Tiled-Y mode, DevGT+ Blitter supports both Tile-X and Tile-Y modes. On reset, this bit will be '0'. This definition applies to only X, Y Blits.

Value	Name
0b	Tiling Disabled (Linear blit) **[Default]**
1b	Tiling enabled: Tile-X or Tile-Y

10:8 Transparency Range Mode

These bits control whether or not the byte(s) at the destination corresponding to a given pixel will be conditionally written, and what those conditions are. This feature can make it possible to perform various masking functions in order to selectively write or preserve graphics data already at the destination.

Value	Name	Description
xx0b	**[Default]**	No color transparency mode enabled. This causes normal operation with regard to writing data to the destination.
001b		[Source color transparency] The Transparency Color Low: (Pixel Greater or Equal) (source background register) and the Transparency Color High: (Pixel Less or Equal) (source foreground register) are compared to the source pixels. The range comparisons are done on each component (R, G, B) and then logically ANDed. If the source pixel components are not within the range defined by the Transparency Color registers, then the byte(s) at the destination corresponding to the current pixel are written with the result of the bit-wise operation.
011b		[Source and Alpha color transparency] The Transparency Color Low: (Pixel Greater or Equal) (source background register) and the Transparency Color High: (Pixel Less or Equal) (source foreground register) are compared to the source pixels. The range comparisons are done on each component (A, R, G, B) and then logically ANDed. If the source pixel components are not within

				the range defined by the Transparency Color registers, then the byte(s) at the destination corresponding to the current pixel are written with the result of the bit-wise operation."
		101b		[Destination and Alpha color transparency] The Transparency Color Low: (Pixel Greater or Equal) (source background register) and the Transparency Color High: (Pixel Less or Equal) (source foreground register) are compared to the destination pixels. The range comparisons are done on each component (A, R, G, B) and then logically ANDed. If the destination pixels are within the range, then the byte(s) at the destination corresponding to the current pixel are written with the result of the bit-wise operation.
		111b		[Destination color transparency] The Transparency Color Low: (Pixel Greater or Equal) (source background register) and the Transparency Color High: (Pixel Less or Equal) (source foreground register) are compared to the destination pixels. The range comparisons are done on each component (R, G, B) and then logically ANDed. If the destination pixels are within the range, then the byte(s) at the destination corresponding to the current pixel are written with the result of the bit-wise operation.

7:5	**Pattern Vertical Seed** Default Value: 000b This field specifies the pattern scan line which corresponds to Y=0.
4	**Destination Read Modify Write** Default Value: 0b This bit is decoded from the last instruction's opcode field and Destination Transparency Mode to identify whether a Destination read is needed.
3	**Color Source** Default Value: 0b This bit is decoded from the last instructions opcode field to identify whether a color (1) source is used.
2	**Monochrome Source** Default Value: 0b This bit is decoded from the last instructions opcode field to identify whether a monochrome (1) source is used.
1	**Color Pattern** Default Value: 0b This bit is decoded from the last instructions opcode field to identify whether a color (1) pattern is used.

BR00 - BLT Opcode and Control		
	0	**Monochrome Pattern** Default Value: 0b This bit is decoded from the last instructions opcode field to identify whether a monochrome (1) pattern is used.

BR01 - Setup BLT Raster OP, Control, and Destination Offset

BR01 - Setup BLT Raster OP, Control, and Destination Offset

Source: BlitterCS
Size (in bits): 32
Default Value: 0x00000000

DWord	Bit	Description
0	31	**Solid Pattern Select** This bit applies only when the pattern data is monochrome. This bit determines whether or not the BLT Engine actually performs read operations from the frame buffer in order to load the pattern data. Use of this feature to prevent these read operations can increase BLT Engine performance, if use of the pattern data is indeed not necessary. The BLT Engine is configured to accept either monochrome or color pattern data via the opcode field.

Value	Name	Description
0b	**[Default]**	This causes normal operation with regard to the use of the pattern data. The BLT Engine proceeds with the process of reading the pattern data, and the pattern data is used as the pattern operand for all bit-wise operations.
1b		The BLT Engine forgoes the process of reading the pattern data, the presumption is made that all of the bits of the pattern data are set to 0, and the pattern operand for all bit-wise operations is forced to the background color specified in the Color Expansion Background Color Register.

DWord	Bit	Description
	30	**Clipping Enabled**

Value	Name
0b	**[Default]**
1b	

DWord	Bit	Description
	29	**Monochrome Source Transparency Mode** This bit applies only when the source data is in monochrome. This bit determines whether or not the byte(s) at the destination corresponding to the pixel to which a given bit of the source data also corresponds will actually be written if that source data bit has the value of 0. This feature can make it possible to use the source as a transparency mask. The BLT Engine is configured to accepted either monochrome or color source data via the opcode field.

Value	Name	Description
0b	**[Default]**	This causes normal operation with regard to the use of the source data. Wherever a bit in the source data has the value of 0, the color specified in the background color register is used as the source operand in the bit-wise operation for the pixel corresponding to the source data bit, and the bytes at the destination corresponding to that pixel are written with the result.
1b		Wherever a bit in the source data has the value of 0, the byte(s) at the destination corresponding to the pixel to which the source data bit also corresponds are simply not written, and the data at those byte(s) at the destination are allowed to remain unchanged.

BR01 - Setup BLT Raster OP, Control, and Destination Offset

28	**Monochrome Pattern Transparency Mode** This bit applies only when the pattern data is monochrome. This bit determines whether or not the byte(s) at the destination corresponding to the pixel to which a given bit of the pattern data also corresponds will actually be written if that pattern data bit has the value of 1. This feature can make it possible to use the pattern as a transparency mask. The BLT Engine is configured to accepted either monochrome or color pattern data via the opcode field.

Value	Name	Description
0b	**[Default]**	This causes normal operation with regard to the use of the pattern data. Wherever a bit in the pattern data has the value of 0, the color specified in the background color register is used as the pattern operand in the bit-wise operation for the pixel corresponding to the pattern data bit, and the bytes at the destination corresponding to that pixel are written with the result.
1b		Wherever a bit in the pattern data has the value of 0, the byte(s) at the destination corresponding to the pixel to which the pattern data bit also corresponds are simply not written, and the data at those byte(s) at the destination are allowed to remain unchanged.

27:26	**32bpp Byte Mask** This bit applies only when the pattern data is monochrome. This bit determines whether or not the byte(s) at the destination corresponding to the pixel to which a given bit of the pattern data also corresponds will actually be written if that pattern data bit has the value of 1. This feature can make it possible to use the pattern as a transparency mask. The BLT Engine is configured to accepted either monochrome or color pattern data via the opcode field.

Value	Name
00b	**[Default]**
1xb	Write Alpha Channel
x1b	Write RGB Channel

25:24	**Color Depth**

Value	Name
00b	8 Bit Color Depth **[Default]**
01b	16 Bit Color Depth
10b	Alternate 16 Bit Color Depth
11b	32 Bit Color Depth

23:16	**Raster Operation Select** These 8 bits are used to select which one of 256 possible raster operations is to be performed by the BLT Engine.
15:0	**Destination Pitch (Offset)** For non-XY Blits, the signed 16bit field allows for specifying up to + 32Kbytes signed pitches in bytes (same as before). For X, Y Blits with tiled-X surfaces, the pitch for Destination will be 512Byte aligned and should be programmable up to + 128Kbytes. For X, Y Blits with tiled-Y surfaces, the pitch for Destination will be 128Byte aligned and should be programmable upto + 128Kbytes. In this case, this 16bit signed pitch field is used to specify up to + 32KDWords. For X, Y blits with nontiled surfaces (linear surfaces), this 16bit field can be programmed to byte

BR01 - Setup BLT Raster OP, Control, and Destination Offset		
		specification of up to + 32Kbytes (same as before). These 16 bits store the signed memory address offset value by which the destination address originally specified in the Destination Address Register is incremented or decremented as each scan line's worth of destination data is written into the frame buffer by the BLT Engine, so that the destination address will point to the next memory address to which the next scan line's worth of destination data is to be written. If the intended destination of a BLT operation is within on-screen frame buffer memory, this offset is normally set so that each subsequent scan line's worth of destination data lines up vertically with the destination data in the scan line, above. However, if the intended destination of a BLT operation is within off-screen memory, this offset can be set so that each subsequent scan line's worth of destination data is stored at a location immediately after the location where the destination data for the last scan line ended, in order to create a single contiguous block of bytes of destination data at the destination.

BR05 - Setup Expansion Background Color

BR05 - Setup Expansion Background Color		
Source:	BlitterCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:0	**Setup Expansion Background Color Bits** These bits provide the one, two, or four bytes worth of color data that select the background color to be used in the color expansion of monochrome pattern or source data for either the SCANLINE_BLT or TEXT_BLT instructions. BR05 is also used as the solid pattern for the PIXEL_BLT instruction. Whether one, two, or three bytes worth of color data is needed depends upon the color depth to which the BLT Engine has been set. For a color depth of 32bpp, 16bpp and 8bpp, bits [31:0], [15:0] and [7:0], respectively, are used.

BR06 - Setup Expansion Foreground Color

BR06 - Setup Expansion Foreground Color		
Source:	BlitterCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:0	**Setup Expansion Foreground Color Bits** These bits provide the one, two, or four bytes worth of color data that select the foreground color to be used in the color expansion of monochrome pattern or source data for either the SCANLINE_BLT or TEXT_BLT instructions. Whether one, two, or three bytes worth of color data is needed depends upon the color depth to which the BLT Engine has been set. For a color depth of 32bpp, 16bpp and 8bpp, bits [31:0], [15:0] and [7:0], respectively, are used.

BR07 - Setup Blit Color Pattern Address Lower Order Address bits

BR07 - Setup Blit Color Pattern Address Lower Order Address bits

Source:	BlitterCS
Size (in bits):	32
Default Value:	0x00000000

DWord	Bit	Description
0	31:6	**Setup Blit Color Pattern Address** Format: GraphicsAddress[31:6] Lower 32bits of the 48bit addressing. These 26 bits specify the starting address of the (8X8) pixel color **pattern from the SETUP_BLT instruction.** This register works identically to the Pattern Address register (BR15), but this version is **only used with the SCANLINE_BLT instruction execution** (the actual programming for this, is done in XY_SETUP_BLT command). The pattern data must be located in linear memory. The pattern data must be located on a pattern-size boundary. The pattern is always of 8x8 pixels, and therefore, its size is dependent upon its pixel depth. The pixel depth may be 8, 16, or 32 bits per pixel if the pattern is in color (the pixel depth of a color pattern must match the pixel depth to which the graphics system has been set). Monochrome patterns require 8 bytes and is supplied through the instruction. Color patterns of 8, 16, and 32 bits per pixel color depth must start on 64-byte, 128-byte and 256-byte boundaries, respectively. The Pattern Base Address programmed, must always be Cache Line (64byte) aligned.
	5:0	**Reserved** Access: RO Format: MBZ

BR09 - Destination Address Lower Order Address Bits

BR09 - Destination Address Lower Order Address Bits		
Source:	BlitterCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:0	**Destination Address Bits** Format: GraphicsAddress[31:0] When tiling is enabled for XY-blits, this base address should be limited to 4KB. when tiling is disabled for XY-blits, this base address should be CL (64byte) aligned. These lower 32bits of the 48bit address, which specify the starting pixel address of the destination data. This register is also the working destination address register for the lower 32bits of the address, and changes as the BLT Engine performs the accesses. Used as the scan line address (Destination Y Address and Destination Y1 Address) for BLT instructions: PIXEL_BLT, SCANLINE_BLT, and TEXT_BLT. In this case the address points to the first pixel in a scan line and is compared with the ClipRect Y1 and Y2 address registers to determine whether the scan line should be written or not. The Destination Y1 address is the top scan line to be written for text. Note that for non-XY blits (COLOR_BLT, SRC_COPY_BLT), this address points to the first byte to be written. Note: Some instructions affect only one scan line (requiring only one coordinate); other instructions affect multiple scan lines and need both coordinates.

BR11 - BLT Source Pitch (Offset)

BR11 - BLT Source Pitch (Offset)		
Source:	BlitterCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:16	**Reserved**
	15:0	**Source Pitch (Offset)** For non-XY Blits with color source operand (SRC_COPY_BLT), the signed 16bit field allows for specifying up to + 32Kbytes signed pitch in bytes (same as before). For X, Y Blits with tiled-X surfaces, the pitch for Color Source will be 512Byte aligned and should be programmable up to + 128Kbytes. For X, Y Blits with tiled-Y surfaces, the pitch for Color Source will be 128Byte aligned and should be programmable upto + 128Kbytes. In this case, this 16bit signed pitch field is used to specify up to + 32KDWords. For X, Y blits with nontiled color source surfaces (linear surfaces), this 16bit field can be programmed to byte specification of up to + 32Kbytes (same as before). When the color source data is located within the frame buffer or AGP aperture, these signed 16 bits store the memory address offset (pitch) value by which the source address originally specified in the Source Address Register is incremented or decremented as each scan line's worth of source data is read from the frame buffer by the BLT Engine, so that the source address will point to the next memory address from which the next scan line's worth of source data is to be read. Note that if the intended source of a BLT operation is within on-screen frame buffer memory, this offset is normally set to accommodate the fact that each subsequent scan line's worth of source data lines up vertically with the source data in the scan line, above. However, if the intended source of a BLT operation is within off-screen memory, this offset can be set to accommodate a situation in which the source data exists as a single contiguous block of bytes where in each subsequent scan line's worth of source data is stored at a location immediately after the location where the source data for the last scan line ended.

BR12 - Source Address Lower order Address bits

BR12 - Source Address Lower order Address bits

Source:	BlitterCS
Size (in bits):	32
Default Value:	0x00000000

DWord	Bit	Description
0	31:0	**Source Address Bits** Format: GraphicsAddress[31:0] Lower 32bits of the 48bit addressing. When tiling is enabled for XY-blits with Color source surfaces, this base address should be aligned to 4KB. When tiling is disabled for XY-blits, this base address should be CL (64byte) aligned. Note that for non-XY blit with Color Source (SRC_COPY_BLT), this address points to the first byte to be read. These lower 32bits of the 48bit address, specify the starting pixel address of the color source data. The lower 3 bits are used to indicate the position of the first valid byte within the first Quadword of the source data. If this Source happens to be a Monosource surface, then this Monosource Base Address programmed, must always be Cache Line (64byte) aligned.

BR13 - BLT Raster OP, Control, and Destination Pitch

BR13 - BLT Raster OP, Control, and Destination Pitch

Source: BlitterCS

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31	**Solid Pattern Select** This bit applies only when the pattern data is monochrome. This bit determines whether or not the BLT Engine actually performs read operations from the frame buffer in order to load the pattern data. Use of this feature to prevent these read operations can increase BLT Engine performance, if use of the pattern data is indeed not necessary. The BLT Engine is configured to accept either monochrome or color pattern data via the opcode field.

Value	Name	Description
0	**[Default]**	This causes normal operation with regard to the use of the pattern data. The BLT Engine proceeds with the process of reading the pattern data, and the pattern data is used as the pattern operand for all bit-wise operations.
1		The BLT Engine forgoes the process of reading the pattern data, the presumption is made that all of the bits of the pattern data are set to 0, and the pattern operand for all bit-wise operations is forced to the background color specified in the Color Expansion Background Color Register.

	30	**Clipping Enabled**

Default Value:	0

	29	**Monochrome Source Transparency Mode** This bit applies only when the source data is in monochrome. This bit determines whether or not the byte(s) at the destination corresponding to the pixel to which a given bit of the source data also corresponds will actually be written if that source data bit has the value of 0. This feature can make it possible to use the source as a transparency mask. The BLT Engine is configured to accepted either monochrome or color source data via the opcode field.

Value	Name	Description
0	**[Default]**	This causes normal operation with regard to the use of the source data. Wherever a bit in the source data has the value of 0, the color specified in the background color register is used as the source operand in the bit-wise operation for the pixel corresponding to the source data bit, and the bytes at the destination corresponding to that pixel are written with the result.
1		Where a bit in the source data has the value of 0, the byte(s) at the destination corresponding to the pixel to which the source data bit also corresponds are simply not written, and the data at those byte(s) at the destination are allowed to remain unchanged.

	28	**Monochrome Pattern Transparency Mode** This bit applies only when the pattern data is monochrome. This bit determines whether or not the byte(s) at the destination corresponding to the pixel to which a given bit of the pattern data also corresponds will actually be written if that pattern data bit has the value of 1. This feature

BR13 - BLT Raster OP, Control, and Destination Pitch

can make it possible to use the pattern as a transparency mask. The BLT Engine is configured to accepted either monochrome or color pattern data via the opcode in the Opcode and Control register.

Value	Name	Description
0	**[Default]**	This causes normal operation with regard to the use of the pattern data. Where a bit in the pattern data has the value of 0, the color specified in the background color register is used as the pattern operand in the bit-wise operation for the pixel corresponding to the pattern data bit, and the bytes at the destination corresponding to that pixel are written with the result.
1		Wherever a bit in the pattern data has the value of 0, the byte(s) at the destination corresponding to the pixel to which the pattern data bit also corresponds are simply not written, and the data at those byte(s) at the destination are allowed to remain unchanged.

27:26 **32bpp Byte Mask**

This field is only used for 32bpp.

Value	Name
00b	**[Default]**
1xb	Write Alpha Channel
x1b	Write RGB Channel

25:24 **Color Depth**

Value	Name
00b	8 Bit Color Depth **[Default]**
01b	16 Bit Color Depth
10b	24 Bit Color Depth
11b	Reserved

23:16 **Raster Operation Select**

Default Value:	00000000b

These 8 bits are used to select which one of 256 possible raster operations is to be performed by the BLT Engine.

15:0 **Destination Pitch(Offset)**

These 16 bits store the signed memory address offset value by which the destination address originally specified in the Destination Address Register is incremented or decremented as each scan line's worth of destination data is written into the frame buffer by the BLT Engine, so that the destination address will point to the next memory address to which the next scan line's worth of destination data is to be written. If the intended destination of a BLT operation is within on-screen frame buffer memory, this offset is normally set so that each subsequent scan line's worth of destination data lines up vertically with the destination data in the scan line, above. However, if the intended destination of a BLT operation is within off-screen memory, this offset can be set so that each subsequent scan line's worth of destination data is stored at a location immediately after the location where the destination data for the last scan line ended, in order to create a single contiguous block of bytes of destination data at the destination.

BR14 - Destination Width and Height

BR14 - Destination Width and Height

Source:	BlitterCS
Size (in bits):	32
Default Value:	0x00000000

BR14 contains the values for the height and width of the data to be BLT. If these values are not correct, such that the BLT Engine is either expecting data it does not receive or receives data it did not expect, the system can hang.

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**Destination Height** These 13 bits specify the height of the destination data in terms of the number of scan lines. This is a working register.
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**Destination Byte Width** These 13 bits specify the width of the destination data in terms of the number of bytes per scan line. The number of pixels per scan line into which this value translates depends upon the color depth to which the graphics system has been set.

BR15 - Color Pattern Address Lower order Address bits

BR15 - Color Pattern Address Lower order Address bits

Source:	BlitterCS
Size (in bits):	32
Default Value:	0x00000000

DWord	Bit	Description
0	31:6	**Color Pattern Address** Format: GraphicsAddress[31:6] Lower 32bits of the 48bit addressing. There is no change to the Color Pattern address specification due to Non-Power-of-2 change. It remains the same as before. The pattern data must be located in linear memory. These 26 bits specify the starting address of the (8X8) pixel color pattern. The pattern data must be located on a pattern-size boundary. The pattern is always of 8x8 pixels, and therefore, its size is dependent upon its pixel depth. The pixel depth may be 8, 16, or 32 bits per pixel if the pattern is in color (the pixel depth of a color pattern must match the pixel depth to which the graphics system has been set). Monochrome patterns require 8 bytes and are applied through the instruction. Color patterns of 8, 16, and 32 bits per pixel color depth must start on 64-byte, 128-byte and 256-byte boundaries, respectively. The Pattern Base Address programmed, must always be Cache Line (64byte) aligned.
	5:0	**Reserved** Access: RO Format: MBZ

BR16 - Pattern Expansion Background and Solid Pattern Color

BR16 - Pattern Expansion Background and Solid Pattern Color		
Source:	BlitterCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:0	**Pattern Expansion Background Color Bits** These bits provide the one, two, or four bytes worth of color data that select the background color to be used in the color expansion of monochrome pattern data during BLT operations. Whether one, two, or four bytes worth of color data is needed depends upon the color depth to which the BLT Engine has been set. For a color depth of 32bpp, 16bpp and 8bpp, bits [31:0], [15:0] and [7:0], respectively, are used.

BR17 - Pattern Expansion Foreground Color

BR17 - Pattern Expansion Foreground Color		
Source:	BlitterCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:0	**Pattern Expansion Background Color Bits** These bits provide the one, two, or four bytes worth of color data that select the foreground color to be used in the color expansion of monochrome pattern data during BLT operations. Whether one, two, or four bytes worth of color data is needed depends upon the color depth to which the BLT Engine has been set. For a color depth of 32bpp, 16bpp and 8bpp, bits [31:0], [15:0] and [7:0], respectively, are used.

BR18 - Source Expansion Background and Destination Color

BR18 - Source Expansion Background and Destination Color		
Source:	BlitterCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:0	**Source Expansion Background Color Bits** These bits provide the one, two, or four bytes worth of color data that select the background color to be used in the color expansion of monochrome source data during BLT operations. This register is also used to support destination transparency mode and Solid color fill. Whether one, two, three, or four bytes worth of color data is needed depends upon the color depth to which the BLT Engine has been set. For a color depth of 32bpp, 16bpp and 8bpp, bits [31:0], [15:0] and [7:0], respectively, are used.

BR19 - Source Expansion Foreground Color

BR19 - Source Expansion Foreground Color		
Source:	BlitterCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:0	**Pattern/Source Expansion Foreground Color Bits** These bits provide the one, two, or four bytes worth of color data that select the foreground color to be used in the color expansion of monochrome source data during BLT operations. Whether one, two, or four bytes worth of color data is needed depends upon the color depth to which the BLT Engine has been set. For a color depth of 32bpp, 16bpp and 8bpp, bits [31:0], [15:0] and [7:0], respectively, are used.

BR27 - Destination Higher Order Address

BR27 - Destination Higher Order Address

Source:	BlitterCS
Size (in bits):	32
Default Value:	0x00000000

Upper 32 bits of the starting pixel address for the destination data. This structure is also the working location for the upper bits of the destination address, and changes as the BLT Engine performs the accesses. See BR09 for the lower 32 bits. When tiling is enabled for XY-blits, this base address should be limited to 4KB. Otherwise for XY blits, there is no restriction and it is same as before.
Used as the scan line address (Destination Y Address and Destination Y1 Address) for BLT instructions: PIXEL_BLT, SCANLINE_BLT, and TEXT_BLT. In this case the address points to the first pixel in a scan line and is compared with the ClipRect Y1 and Y2 address registers to determine whether the scan line should be written or not. The Destination Y1 address is the top scan line to be written for text.
Note that for non-XY blits (COLOR_BLT, SRC_COPY_BLT), the destination address points to the first byte to be written. This structure is always the last location written for a BLT drawing instruction. Writing to BR27 starts the BLT engine execution. Note: Some instructions affect only one scan line (requiring only one coordinate); other instructions affect multiple scan lines and need both coordinates.
GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The uppermost reserved bits are ignored and MBZ.

DWord	Bit	Description
0	31:25	**Reserved** Access: RO Format: MBZ
	24:16	**Reserved** Access: RO Format: MBZ
	15:0	**Destination Address Upper DWORD** Format: GraphicsAddress[47:32]

BR28 - Source Higher Order Address

BR28 - Source Higher Order Address		
Source:	BlitterCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0 Upper 32 bits of the Source address, specifying the starting pixel address of the color or mono source data. When tiling is enabled for XY-blits with Color source surfaces, this base address should be limited to 4KB. Otherwise for XY blits, there is no restriction and it is same as before, including for monosource and text blits. Note that for non-XY blit with Color Source (SRC_COPY_BLT), this address points to the first byte to be read. GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The uppermost reserved bits are ignored and MBZ.	31:25	**Reserved** Access: RO Format: MBZ
	24:16	**Reserved** Access: RO Format: MBZ
	15:0	**Source Address Upper DWORD** Format: GraphicsAddress[47:32]

BR29 - Color Pattern Higher Order Address

BR29 - Color Pattern Higher Order Address

Source:	BlitterCS
Size (in bits):	32
Default Value:	0x00000000

DWord	Bit	Description
0 Upper 32 bits of the Color Pattern address, specifying the starting location of the (8X8) pixel pattern. GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The uppermost reserved bits are ignored and MBZ.	31:25	**Reserved** Access: RO Format: MBZ
	24:16	**Reserved** Access: RO Format: MBZ
	15:0	**Color Pattern Address Upper DWORD** Format: GraphicsAddress[47:32]

BR30 - Setup Blit Color Pattern Higher Order Address

BR30 - Setup Blit Color Pattern Higher Order Address			
Source:	BlitterCS		
Size (in bits):	32		
Default Value:	0x00000000		
DWord	**Bit**	**Description**	
0 Upper 32 bits of the Color Pattern address, specifying the starting location of the (8X8) pixel pattern. GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The uppermost reserved bits are ignored and MBZ.	31:16	**Reserved**	
		Access:	RO
		Format:	MBZ
	15:0	**Setup Blit Color Pattern Upper DWORD**	
		Format:	GraphicsAddress[47:32]

Byte Masked Media Block Message Header

MH_MBBM - Byte Masked Media Block Message Header

Source: EuSubFunctionDataPort1

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:0	**X Offset** Format: S31 X offset (in bytes) of the upper left corner of the block into the surface. **Programming Notes** Must be DWord aligned (Bits 1:0 MBZ) for the write form of the message.
1	31:0	**Y Offset** Format: S31 Y offset (in rows) of the upper left corner of the block into the surface.
2	31:0	**Media Block Message Control** Format: MHC_MBBM_CONTROL Specifies the Byte Masked message subtype and its additional input parameters.
3	31:0	**Byte Mask** Format: U32 Specifies the Byte Mask for writes when Message Mode field is BYTE_MASK. **Programming Notes** The Byte mask applies horizontally to each row of output: bit 0 for byte 0, through bit 31 for byte 31.
4	31:0	**FFTID** Format: MHC_FFTID Fixed Function Thread ID
5..7	95:0	**Reserved** Access: RO Format: MBZ

Byte Masked Media Block Message Header Control

MHC_MBBM_CONTROL - Byte Masked Media Block Message Header Control

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:30	**Message Mode** Specifies the Media Block Write Message subtype is Byte Masked. Value / Name / Description: 02h / BYTE_MASK / The Block Height and Block Width fields are specified in this Dword. The Byte Mask qualifies which bytes are written. Others / Reserved / Reserved.
	29	**Reserved** Access: RO Format: MBZ
	28:24	**Sub-Register Offset** Format: U5 This field is ignored (reserved) for Media Block Write message.
	23:22	**Reserved** Access: RO Format: MBZ
	21:16	**Block Height** Format: U6 Height in rows of block being accessed. Range = [0,63] representing 1 to 64 rows Restriction: If Block Width (bytes), then Maximum Block Height (rows) is constrained by (# Dwords width) * (# rows) <= 64 Dwords.
	15:10	**Reserved** Access: RO Format: MBZ
	9:8	**Register Pitch Control** Format: U2 This field is ignored (reserved) for a Media Block Write message.

MHC_MBBM_CONTROL - Byte Masked Media Block Message Header Control

	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**Block Width** Format: U6 Width in bytes of the block being accessed. Range = [0,31] representing 1 to 32 Bytes. **Programming Notes** Must be DWord aligned for Media Block Write message.

CC_VIEWPORT

CC_VIEWPORT

Size (in bits): 64

Default Value: 0x00000000, 0x00000000

The viewport state is stored as an array of up to 16 elements, each of which contains the DWords described here. The start of each element is spaced 2 DWords apart. The first element of the viewport state array is aligned to a 32-byte boundary. The Minimum and Maximum Depth legal value ranges are dependent on the depth buffer format.

DWord	Bit	Description
0	31:0	**Minimum Depth** Format: IEEE_FLOAT Indicates the minimum depth. The interpolated or computed depth is clamped to this value prior to the depth test. **Programming Notes** The Minimum depth value must be less-than-or-equal to the Maximum depth value. The Minimum depth value cannot be NAN (Not-A-Number). For All depth formats:Minimum depth value must not be less than 0.0, also it may not be -0.0 (negative zero)
1	31:0	**Maximum Depth** Format: IEEE_FLOAT Indicates the maximum depth. The interpolated or computed depth is clamped to this value prior to the depth test. **Programming Notes** The Maximum depth value cannot be smaller than Minimum depth value. The Maximum depth value cannot be NAN (Not-A-Number). For all depth formats: The Maximum depth value must be between +0.0to +1.0.

Channel Mask Message Descriptor Control Field

MDC_CMASK - Channel Mask Message Descriptor Control Field

Size (in bits): 4

Default Value: 0x00000000

DWord	Bit	Description
0	3:0	**Mask** For the read message, indicates that which channels are read from the surface and included in the writeback message. For the write message, indicates which channels are included in the message payload and written to the surface.

Value	Name	Description
00h	RGBA **[Default]**	Red, Green, Blue, and Alpha are included
01h	GBA	Green, Blue, and Alpha are included
02h	RBA	Red, Blue, and Alpha are included
03h	BA	Blue and Alpha are included
04h	RGA	Red, Green, and Alpha are included
05h	GA	Green and Alpha are included
06h	RA	Red and Alpha are included
07h	A	Alpha is included
08h	RGB	Red, Green, and Blue are included
09h	GB	Green and Blue are included
0Ah	RB	Red and Blue are included
0Bh	B	Blue is included
0Ch	RG	Red and Green are included
0Dh	G	Green is included
0Eh	R	Red is included
0Fh	Reserved	Ignored

Channel Mode Message Descriptor Control Field

MDC_CMODE - Channel Mode Message Descriptor Control Field

Size (in bits): 1

Default Value: 0x00000000

DWord	Bit	Description
0	0	**Channel Mode** Format: Boolean Two modes of channel-enable are provided: a SIMD8 or SIMD16 Dword channel serial view of a register, and a SIMD4x2 view of a register. Value / Name / Description: 0 / Oword / All 4 Dwords are read or written if one or more of these channels are enabled 1 / Dword / Each Dword is read or written only if its corresponding channel is enabled.

Clear Color

CLEAR_COLOR - Clear Color		
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0 **Programming Notes:** Software shall write the Raw Clear Color channels such that the channel order matches the SURFACE_STATE.Shader Channel Select programming. Software shall write the converted Depth Clear to this dword	31:0	**Raw Clear Color : Red** Format: IEEE_FLOAT Format: U32 Format: S31 Format: **U24_X8** []
1 **Programming Notes:** Software shall write the Raw Clear Color channels such that the channel order matches the SURFACE_STATE.Shader Channel Select programming.	31:0	**Raw Clear Color: Blue** Format: IEEE_FLOAT Format: U32 Format: S31
2 **Programming Notes:** Software shall write the Raw Clear Color channels such that the channel order matches the SURFACE_STATE.Shader Channel Select programming.	31:0	**Raw Clear Color : Green** Format: IEEE_FLOAT Format: U32 Format: S31
3 **Programming Notes:** Software shall write the Raw Clear Color channels such that the channel order matches the SURFACE_STATE.Shader Channel Select programming.	31:0	**Raw Clear Color : Alpha** Format: IEEE_FLOAT Format: U32 Format: S31
4	31:0	**Converted Clear Color and Clear Depth** This DWORD stores the format converted clear color. If bits per pixel are 32, entire pixel's clear value is stored in this DWORD. If bits per pixel are 64, lower DOWRD is stored in this field. If bits per pixel are 128, this field is not used to store clear value. This field is packed according to the RT format

CLEAR_COLOR - Clear Color

5	31:0	**Converted Clear Color** This DWORD stores the format converted clear color. If bits per pixel are 64, upper DOWRD is stored in this field If bits per pixel are 32 or 128, this field is not used to store clear value. The field is packed according to the RT format
6	31:1	**Reserved** Access: RO Format: MBZ
	0	**Color Discard Enable** **Description** When this bit is set for a Render Target Surface, SW indicates HW that this surface's cachelines from on-chip caches do not need to be written back to memory after a complete Render Pass (aka Tile Pass). This bit applies to both the main and the associated AUX surfaces. **Programming Notes** This bit must be programmed before binding a surface to a Render Pass (Tile Pass). This bit must not be changed during the Tile Pass.
7	31:1	**Reserved** Access: RO Format: MBZ
	0	**Reserved** Access: RO Format: MBZ

Clock Gating Disable Format

Clock Gating Disable Format		
Size (in bits):	1	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	0	**Clock_Gate_Disable**

Value	Name	Description
0b	Enable	Clock gating controlled by unit logic
1b	Disable	Disable clock gating function

COLOR_CALC_STATE

COLOR_CALC_STATE

Size (in bits): 192

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

This definition applies to devices. It is pointed to by a field in 3DSTATE_CC_STATE_POINTERS, and stored at a 64-byte aligned boundary.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15	**Round Disable Function Disable** Format: Disable Disables the round-disable function of the color calculator. Value 0 – Name: Cancelled – Description: Dithering is cancelled based on the data used by blend to avoid drift. Value 1 – Name: Not Cancelled – Description: Dithering is NOT cancelled.
	14:1	**Reserved** Access: RO Format: MBZ
	0	**Alpha Test Format** This field selects the format for Alpha Reference Value and the format in which Alpha Test is performed. Value 0h – Name: ALPHATEST_UNORM8 – Description: UNorm8 Value 1h – Name: ALPHATEST_FLOAT32 – Description: Float32 **Programming Notes** Alpha-test format is independent of RT format. When PS outputs UNIT/SINT alpha-value, it will be treated as IEEE 32bit float number for the purpose of alpha-test.
1	31:0	**Alpha Reference Value As FLOAT32** Exists If: [Alpha Test Format] == 'ALPHATEST_FLOAT32' Format: IEEE_FLOAT This field specifies the alpha reference value to compare against in the Alpha Test function. **Programming Notes** This field should not be programmed to NaN.

COLOR_CALC_STATE

	7:0	**Alpha Reference Value As UNORM8** Exists If: [Alpha Test Format] == 'ALPHATEST_UNORM8' Format: UNORM8 This field specifies the alpha reference value to compare against in the Alpha Test function.
2	31:0	**Blend Constant Color Red** Format: IEEE_FLOAT This field specifies the Red channel of the Constant Color used in Color Buffer Blending.
3	31:0	**Blend Constant Color Green** Format: IEEE_FLOAT This field specifies the Green channel of the Constant Color used in Color Buffer Blending.
4	31:0	**Blend Constant Color Blue** Format: IEEE_FLOAT This field specifies the Blue channel of the Constant Color used in Color Buffer Blending.
5	31:0	**Blend Constant Color Alpha** Format: IEEE_FLOAT This field specifies the Alpha channel of the Constant Color used in Color Buffer Blending.

COLOR_PROCESSING_STATE - ACE State

COLOR_PROCESSING_STATE - ACE State

Size (in bits): 416

Default Value: 0x00000068, 0x4C382410, 0x9C887460, 0xEBD8C4B0, 0x604C3824, 0xB09C8874, 0x0000D8C4, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

This state structure contains the ACE state used by the color processing function. It corresponds to DW29..DW41 of the Color Processing State.

DWord	Bit	Description
0	31:7	**Reserved** Access: RO Format: MBZ
	6:2	**Skin Threshold** Format: U5 Used for Y analysis (min/max) for pixels which are higher than skin threshold. Value / Name 1-31 26 **[Default]**
	1	**Full Image Histogram** Default Value: 0 Format: Enable Used to ignore the area of interest for full image histogram.
	0	**ACE Enable** Format: Enable
1	31:24	**Y3** Default Value: 76 Format: U8 The value of the y_pixel for point 3 in PWL.
	23:16	**Y2** Default Value: 56 Format: U8 The value of the y_pixel for point 2 in PWL.

<table>
<tr><th colspan="3">COLOR_PROCESSING_STATE - ACE State</th></tr>
<tr><td></td><td>15:8</td><td>Y1<table><tr><td>Default Value:</td><td>36</td></tr><tr><td>Format:</td><td>U8</td></tr></table>The value of the y_pixel for point 1 in PWL.</td></tr>
<tr><td></td><td>7:0</td><td>Ymin<table><tr><td>Default Value:</td><td>16</td></tr><tr><td>Format:</td><td>U8</td></tr></table>The value of the y_pixel for point 0 in PWL.</td></tr>
<tr><td>2</td><td>31:24</td><td>Y7<table><tr><td>Default Value:</td><td>156</td></tr><tr><td>Format:</td><td>U8</td></tr></table>The value of the y_pixel for point 7 in PWL.</td></tr>
<tr><td></td><td>23:16</td><td>Y6<table><tr><td>Default Value:</td><td>136</td></tr><tr><td>Format:</td><td>U8</td></tr></table>The value of the y_pixel for point 6 in PWL.</td></tr>
<tr><td></td><td>15:8</td><td>Y5<table><tr><td>Default Value:</td><td>116</td></tr><tr><td>Format:</td><td>U8</td></tr></table>The value of the y_pixel for point 5 in PWL.</td></tr>
<tr><td></td><td>7:0</td><td>Y4<table><tr><td>Default Value:</td><td>96</td></tr><tr><td>Format:</td><td>U8</td></tr></table>The value of the y_pixel for point 4 in PWL.</td></tr>
<tr><td>3</td><td>31:24</td><td>Ymax<table><tr><td>Default Value:</td><td>235</td></tr><tr><td>Format:</td><td>U8</td></tr></table>The value of the y_pixel for point 11 in PWL.</td></tr>
<tr><td></td><td>23:16</td><td>Y10<table><tr><td>Default Value:</td><td>216</td></tr><tr><td>Format:</td><td>U8</td></tr></table>The value of the y_pixel for point 10 in PWL.</td></tr>
</table>

COLOR_PROCESSING_STATE - ACE State

	15:8	**Y9** Default Value: 196 Format: U8 The value of the y_pixel for point 9 in PWL.
	7:0	**Y8** Default Value: 176 Format: U8 The value of the y_pixel for point 8 in PWL.
4	31:24	**B4** Default Value: 96 Format: U8 The value of the bias for point 4 in PWL.
	23:16	**B3** Default Value: 76 Format: U8 The value of the bias for point 3 in PWL.
	15:8	**B2** Default Value: 56 Format: U8 The value of the bias for point 2 in PWL.
	7:0	**B1** Default Value: 36 Format: U8 The value of the bias for point 1 in PWL.
5	31:24	**B8** Default Value: 176 Format: U8 The value of the bias for point 8 in PWL.
	23:16	**B7** Default Value: 156 Format: U8 The value of the bias for point 7 in PWL.

COLOR_PROCESSING_STATE - ACE State

	15:8	**B6** Default Value: 136 Format: U8 The value of the bias for point 6 in PWL.
	7:0	**B5** Default Value: 116 Format: U8 The value of the bias for point 5 in PWL.
6	31:16	**Reserved** Access: RO Format: MBZ
	15:8	**B10** Default Value: 216 Format: U8 The value of the bias for point 10 in PWL.
	7:0	**B9** Default Value: 196 Format: U8 The value of the bias for point 9 in PWL.
7	31:27	**Reserved** Access: RO Format: MBZ
	26:16	**S1** Format: U1.10 The value of the slope for point 1 in PWL. The default is 1024/1024.
	15:11	**Reserved** Access: RO Format: MBZ
	10:0	**S0** Format: U1.10 The value of the slope for point 0 in PWL. The default is 1024/1024.

COLOR_PROCESSING_STATE - ACE State

8	31:27	**Reserved** Access: RO Format: MBZ
	26:16	**S3** Format: U1.10 The value of the slope for point 3 in PWL. The default is 1024/1024.
	15:11	**Reserved** Access: RO Format: MBZ
	10:0	**S2** Format: U1.10 The value of the slope for point 2 in PWL. The default is 1024/1024.
9	31:27	**Reserved** Access: RO Format: MBZ
	26:16	**S5** Format: U1.10 The value of the slope for point 5 in PWL. The default is 1024/1024.
	15:11	**Reserved** Access: RO Format: MBZ
	10:0	**S4** Format: U1.10 The value of the slope for point 4 in PWL. The default is 1024/1024.
10	31:27	**Reserved** Access: RO Format: MBZ
	26:16	**S7** Format: U1.10 The value of the slope for point 7 in PWL. The default is 1024/1024.
	15:11	**Reserved** Access: RO Format: MBZ

COLOR_PROCESSING_STATE - ACE State

	10:0	**S6** Format: U1.10 The value of the slope for point 6 in PWL. The default is 1024/1024.
11	31:27	**Reserved** Access: RO Format: MBZ
	26:16	**S9** Format: U1.10 The value of the slope for point 9 in PWL. The default is 1024/1024.
	15:11	**Reserved** Access: RO Format: MBZ
	10:0	**S8** Format: U1.10 The value of the slope for point 8 in PWL. The default is 1024/1024.
12	31:11	**Reserved** Access: RO Format: MBZ
	10:0	**S10** Format: U1.10 The value of the slope for point 10 in PWL. The default is 1024/1024.

COLOR_PROCESSING_STATE - PROCAMP State

COLOR_PROCESSING_STATE - PROCAMP State

Size (in bits): 64

Default Value: 0x00020001, 0x01000000

This state structure contains the PROCAMP state used by the color processing function. It corresponds to DW53..DW54 of the Color Processing State.

DWord	Bit	Description
0	31:28	**Reserved** Access: RO Format: MBZ
	27:17	**Contrast** Default Value: 1 Format: U4.7 Contrast magnitude.
	16:13	**Reserved** Access: RO Format: MBZ
	12:1	**Brightness** Default Value: 0 Format: S7.4 Brightness magnitude.
	0	**PROCAMP Enable** Default Value: 1 Format: Enable
1	31:16	**Cos_c_s** Default Value: 256 Format: S7.8 UV multiplication cosine factor.
	15:0	**Sin_c_s** Default Value: 0 Format: S7.8 UV multiplication sine factor.

COLOR_PROCESSING_STATE - STD/STE State

COLOR_PROCESSING_STATE - STD/STE State

Size (in bits): 928

Default Value: 0x9A6E39F0, 0x400C0000, 0x00001180, 0xFE2F2E00, 0x000000FF, 0x00140000, 0xD82E0000, 0x8285ECEC, 0x00008282, 0x00000000, 0x02117000, 0xA38FEC96, 0x00008CC8, 0x00000000, 0x01478000, 0x0007C300, 0x00000000, 0x00000000, 0x1C180000, 0x00000000, 0x00000000, 0x00000000, 0x0007CF80, 0x00000000, 0x00000000, 0x1C080000, 0x00000000, 0x00000000, 0x00000000

This state structure contains the STD/STE state used by the color processing function.

DWord	Bit	Description
0	31:24	**V_Mid** Default Value: 154 Format: U8 Rectangle middle-point V coordinate
	23:16	**U_Mid** Default Value: 110 Format: U8 Rectangle middle-point U coordinate
	15:10	**Hue Max** Default Value: 14 Format: U6 Rectangle half width
	9:4	**Sat Max** Default Value: 31 Format: U6 Rectangle half length.
	3	**Reserved** Access: RO Format: MBZ
	2	**Output Control** 0 – Output Pixels **[Default]** 1 – Output STD Decisions

COLOR_PROCESSING_STATE - STD/STE State

	1	**STE Enable** Format: Enable
	0	**STD Enable** Format: Enable
1	31	**Reserved** Access: RO Format: MBZ
	30:28	**Diamond Margin** Default Value: 4 Format: U3
	27:21	**Diamond du** Default Value: 0 Format: S6 Rhombus center shift in the sat-direction, relative to the rectangle center.
	20:18	**HS Margin** Default Value: 3 Format: U3
	17:10	**Cos(Î±)** Format: S0.7 The default is 79/128
	9:8	**Reserved** Access: RO Format: MBZ
	7:0	**Sin(Î±)** Format: S0.7 The default is 101/128
2	31:21	**Reserved** Access: RO Format: MBZ
	20:13	**Diamond Alpha** Format: U2.6 1 / tan() The default is 100/64

COLOR_PROCESSING_STATE - STD/STE State

	12:7	**Diamond Th** Default Value: 35 Format: U6 Half length of the rhombus axis in the sat-direction.
	6:0	**Diamond dv** Default Value: 0 Format: S6
3	31:24	**Y_point_3** Default Value: 254 Format: U8 Third point of the Y piecewise linear membership function.
	23:16	**Y_point_2** Default Value: 47 Format: U8 Second point of the Y piecewise linear membership function.
	15:8	**Y_point_1** Default Value: 46 Format: U8 First point of the Y piecewise linear membership function.
	7	**VY_STD_Enable** Format: Enable Enables STD in the VY subspace.
	6:0	**Reserved** Access: RO Format: MBZ
4	31:18	**Reserved** Access: RO Format: MBZ
	17:13	**Y_Slope_2** Format: U2.3 Slope between points Y3 and Y4. The default is 31/8.

COLOR_PROCESSING_STATE - STD/STE State

	12:8	**Y_Slope_1** Format: U2.3 Slope between points Y1 and Y2. The default is 31/8.
	7:0	**Y_point_4** Default Value: 255 Format: U8 Fourth point of the Y piecewise linear membership function
5	31:16	**INV_skin_types_margin** Format: U0.16 1/(2* Skin_types_margin) Value / Name / Description: 20 / **[Default]** / Skin_Type_margin
	15:0	**Inverse Margin VYL** Format: U0.16 1 / Margin_VYL The default is 3300/65536
6	31:24	**P1L** Default Value: 216 Format: U8 Y Point 1 of the lower part of the detection PWLF.
	23:16	**P0L** Default Value: 46 Format: U8 Y Point 0 of the lower part of the detection PWLF.
	15:0	**Inverse Margin VYU** Format: U0.16 1 / Margin_VYU The default is 1600/65536.
7	31:24	**B1L** Default Value: 130 Format: U8 V Bias 1 of the lower part of the detection PWLF.

COLOR_PROCESSING_STATE - STD/STE State		
	23:16	**B0L** Default Value: 133 Format: U8 V Bias 0 of the lower part of the detection PWLF.
	15:8	**P3L** Default Value: 236 Format: U8 Y Point 3 of the lower part of the detection PWLF.
	7:0	**P2L** Default Value: 236 Format: U8 Y point 2 of the lower part of the detection PWLF.
8	31:27	**Reserved** Access: RO Format: MBZ
	26:16	**S0L** Format: S2.8 Slope 0 of the lower part of the detection PWLF. The default is -5/256.
	15:8	**B3L** Default Value: 130 Format: U8 V Bias 3 of the lower part of the detection PWLF.
	7:0	**B2L** Default Value: 130 Format: U8 V Bias 2 of the lower part of the detection PWLF.
9	31:22	**Reserved** Access: RO Format: MBZ
	21:11	**S2L** Format: S2.8 Slope 2 of the lower part of the detection PWLF. The default is 0/256.

COLOR_PROCESSING_STATE - STD/STE State

	10:0	**S1L** Format: S2.8 Slope 1 of the lower part of the detection PWLF. The default is 0/256.
10	31:27	**Reserved** Access: RO Format: MBZ
	26:19	**P1U** Default Value: 66 Format: U8 Y Point 1 of the upper part of the detection PWLF.
	18:11	**P0U** Default Value: 46 Format: U8 Y Point 0 of the upper part of the detection PWLF.
	10:0	**S3L** Format: S2.8 Slope 3 of the lower part of the detection PWLF. The default is 0/256.
11	31:24	**B1U** Default Value: 163 Format: U8 V Bias 1 of the upper part of the detection PWLF.
	23:16	**B0U** Default Value: 143 Format: U8 V Bias 0 of the upper part of the detection PWLF.
	15:8	**P3U** Default Value: 236 Format: U8 Y Point 3 of the upper part of the detection PWLF.

COLOR_PROCESSING_STATE - STD/STE State		
	7:0	**P2U** Default Value: 150 Format: U8 Y Point 2 of the upper part of the detection PWLF.
12	31:27	**Reserved** Access: RO Format: MBZ
	26:16	**S0U** Format: S2.8 Slope 0 of the upper part of the detection PWLF. The default is 256/256.
	15:8	**B3U** Default Value: 140 Format: U8 V Bias 3 of the upper part of the detection PWLF.
	7:0	**B2U** Default Value: 200 Format: U8 V Bias 2 of the upper part of the detection PWLF.
13	31:22	**Reserved** Access: RO Format: MBZ
	21:11	**S2U** Format: S2.8 Slope 2 of the upper part of the detection PWLF. The default is -179/256.
	10:0	**S1U** Format: S2.8 Slope 1 of the upper part of the detection PWLF. The default is -113/256.
14	31:28	**Reserved** Access: RO Format: MBZ

COLOR_PROCESSING_STATE - STD/STE State

	27:20	**Skin Types Margin** Default Value: 20 Format: U8 Skin types Y margin.
	19:12	**Skin Types Thresh** Default Value: 120 Format: U8 Skin types Y threshold.
	11	**Skin Type Enable** Format: Enable Treat differently bright and dark skin types. Value / Name / Description 0 / **[Default]** / Disable
	10:0	**S3U** Format: S2.8 Slope 3 of the upper part of the detection PWLF. The default is 0/256.
15	31	**Reserved** Access: RO Format: MBZ
	30:21	**SATB1** Format: S7.2 First bias for the saturation PWLF (bright skin). The default is -8/4.
	20:14	**SATP3** Default Value: 31 Format: S6 Third point for the saturation PWLF (bright skin).
	13:7	**SATP2** Default Value: 6 Format: S6 Second point for the saturation PWLF (bright skin).
	6:0	**SATP1** Format: S6 First point for the saturation PWLF (bright skin). The default is -6.

COLOR_PROCESSING_STATE - STD/STE State

16	31	**Reserved** Access: RO Format: MBZ
	30:20	**SATS0** Format: U3.8 Zeroth slope for the saturation PWLF (bright skin). The default is 297/256.
	19:10	**SATB3** Format: S7.2 Third bias for the saturation PWLF (bright skin). The default is 124/4.
	9:0	**SATB2** Format: S7.2 Second bias for the saturation PWLF (bright skin). The default is 8/4.
17	31:22	**Reserved** Access: RO Format: MBZ
	21:11	**SATS2** Format: U3.8 Second slope for the saturation PWLF (bright skin). The default is 297/256.
	10:0	**SATS1** Format: U3.8 First slope for the saturation PWLF (bright skin). The default is 85/256.
18	31:25	**HUEP3** Default Value: 14 Format: S6 Third point for the hue PWLF (bright skin)
	24:18	**HUEP2** Default Value: 6 Format: S6 Second point for the hue PWLF (bright skin)
	17:11	**HUEP1** Format: S6 First point for the hue PWLF (bright skin). The default is -6.

COLOR_PROCESSING_STATE - STD/STE State		
	10:0	**SATS3** Format: U3.8 Thrid slope for the saturation PWLF (bright skin). The default is 256/256.
19	31:30	**Reserved** Access: RO Format: MBZ
	29:20	**HUEB3** Format: S7.2 Third bias for the hue PWLF (bright skin). The default is 56/4.
	19:10	**HUEB2** Format: S7.2 Second bias for the hue PWLF (bright skin). The default is 8/4.
	9:0	**HUEB1** Format: S7.2 First bias for the hue PWLF (bright skin). The default is -8/4.
20	31:22	**Reserved** Access: RO Format: MBZ
	21:11	**HUES1** Format: U3.8 First slope for the hue PWLF (bright skin) The default is 85/256.
	10:0	**HUES0** Format: U3.8 Zeroth slope for the hue PWLF (bright skin) The default is 384/256.
21	31:22	**Reserved** Access: RO Format: MBZ
	21:11	**HUES3** Format: U3.8 Third slope for the hue PWLF (bright skin) The default is 256/256.

COLOR_PROCESSING_STATE - STD/STE State		
	10:0	**HUES2** Format: U3.8 Second slope for the hue PWLF (bright skin) The default is 384/256.
22	31	**Reserved** Access: RO Format: MBZ
	30:21	**SATB1_DARK** Format: S7.2 First bias for the saturation PWLF (dark skin) The default is 0/4.
	20:14	**SATP3_DARK** Default Value: 31 Format: S6 Third point for the saturation PWLF (dark skin)
	13:7	**SATP2_DARK** Default Value: 31 Format: S6 Second point for the saturation PWLF (dark skin)
	6:0	**SATP1_DARK** Format: S6 First point for the saturation PWLF (dark skin). The default is -11.
23	31	**Reserved** Access: RO Format: MBZ
	30:20	**SATS0_DARK** Format: U3.8 Zeroth slope for the saturation PWLF (dark skin). The default is 397/256.
	19:10	**SATB3_DARK** Format: S7.2 Third bias for the saturation PWLF (dark skin). The default is 124/4.
	9:0	**SATB2_DARK** Format: S7.2 Second bias for the saturation PWLF (dark skin). The default is 124/4.

COLOR_PROCESSING_STATE - STD/STE State

24	31:22	**Reserved** Access: RO Format: MBZ
	21:11	**SATS2_DARK** Format: U3.8 Second slope for the saturation PWLF (dark skin). The default is 256/256.
	10:0	**SATS1_DARK** Format: U3.8 First slope for the saturation PWLF (dark skin). The default is 189/256.
25	31:25	**HUEP3_DARK** Default Value: 14 Format: S6 Third point for the hue PWLF (dark skin).
	24:18	**HUEP2_DARK** Default Value: 2 Format: S6 Third point for the hue PWLF (dark skin).
	17:11	**HUEP1_DARK** Default Value: 0 Format: S6 Third point for the hue PWLF (dark skin).
	10:0	**SATS3_DARK** Format: U3.8 Third slope for the saturation PWLF (dark skin). The default is 256/256.
26	31:30	**Reserved** Access: RO Format: MBZ
	29:20	**HUEB3_DARK** Format: S7.2 Third bias for the hue PWLF (dark skin). The default is 56/4.

COLOR_PROCESSING_STATE - STD/STE State		
	19:10	**HUEB2_DARK** Format: S7.2 Second bias for the hue PWLF (dark skin). The default is 0/4.
	9:0	**HUEB1_DARK** Format: S7.2 First bias for the hue PWLF (dark skin). The default is 0/4.
27	31:22	**Reserved** Access: RO Format: MBZ
	21:11	**HUES1_DARK** Format: U3.8 First slope for the hue PWLF (dark skin). The default is 0/256.
	10:0	**HUES0_DARK** Format: U3.8 Zeroth slope for the hue PWLF (dark skin). The default is 256/256.
28	31:22	**Reserved** Access: RO Format: MBZ
	21:11	**HUES3_DARK** Format: U3.8 Third slope for the hue PWLF (dark skin). The default is 256/256.
	10:0	**HUES2_DARK** Format: U3.8 Second slope for the hue PWLF (dark skin). The default is 299/256.

COLOR_PROCESSING_STATE - TCC State

COLOR_PROCESSING_STATE - TCC State

Size (in bits): 352

Default Value: 0xDCDCDC00, 0xDCDCDC00, 0x1E34CC91, 0x3E3CCE91, 0x02E80195, 0x0197046B, 0x01790174, 0x00096000, 0x00000000, 0x03030000, 0x009201C0

This state structure contains the TCC state used by the color processing function. It corresponds to DW42..DW52 of the Color Processing State.

DWord	Bit	Description
0	31:24	**SatFactor3** Default Value: 220 Format: U1.7 The saturation factor for yellow.
	23:16	**SatFactor2** Default Value: 220 Format: U1.7 The saturation factor for red.
	15:8	**SatFactor1** Default Value: 220 Format: U1.7 The saturation factor for magenta.
	7	**TCC Enable** Format: Enable
	6:0	**Reserved** Access: RO Format: MBZ
1	31:24	**SatFactor6** Default Value: 220 Format: U1.7 The saturation factor for blue.
	23:16	**SatFactor5** Default Value: 220 Format: U1.7 The saturation factor for cyan.

COLOR_PROCESSING_STATE - TCC State

	15:8	**SatFactor4** Default Value: 220 Format: U1.7 The saturation factor for green.
	7:0	**Reserved** Access: RO Format: MBZ
2	31:30	**Reserved** Access: RO Format: MBZ
	29:20	**Base Color 3** Default Value: 483 Format: U10
	19:10	**Base Color 2** Default Value: 307 Format: U10
	9:0	**Base Color 1** Default Value: 145 Format: U10
3	31:30	**Reserved** Access: RO Format: MBZ
	29:20	**Base Color 6** Default Value: 995 Format: U10
	19:10	**Base Color 5** Default Value: 819 Format: U10
	9:0	**Base Color 4** Default Value: 657 Format: U10
4	31:16	**Color Transit Slope 23** Default Value: 744 Format: U0.16 The calculation result of 1 / (BC3 - BC2) [1/62]

COLOR_PROCESSING_STATE - TCC State

	15:0	**Color Transit Slope 12** Default Value: 405 Format: U0.16 The calculation result of 1 / (BC2 - BC1) [1/57]
5	31:16	**Color Transit Slope 45** Default Value: 407 Format: U0.16 The calculation result of 1 / (BC5 - BC4) [1/57]
	15:0	**Color Transit Slope 34** Default Value: 1131 Format: U0.16 The calculation result of 1 / (BC4 - BC3) [1/61]
6	31:16	**Color Transit Slope 61** Default Value: 377 Format: U0.16 The calculation result of 1 / (BC1 - BC6) [1/62]
	15:0	**Color Transit Slope 56** Default Value: 372 Format: U0.16 The calculation result of 1 / (BC6 - BC5) [1/62]
7	31:22	**Color Bias 3** Default Value: 0 Format: U2.8 Color bias for BaseColor3.
	21:12	**Color Bias 2** Default Value: 150 Format: U2.8 Color bias for BaseColor2.
	11:2	**Color Bias 1** Default Value: 0 Format: U2.8 Color bias for BaseColor1.

COLOR_PROCESSING_STATE - TCC State

	1:0	**Reserved** Access: RO Format: MBZ
8	31:22	**Color Bias 6** Default Value: 0 Format: U2.8 Color bias for BaseColor6.
	21:12	**Color Bias 5** Default Value: 0 Format: U2.8 Color bias for BaseColor5.
	11:2	**ColorBias4** Default Value: 0 Format: U2.8 Color bias for BaseColor4.
	1:0	**Reserved** Access: RO Format: MBZ
9	31	**Reserved** Access: RO Format: MBZ
	30:24	**UV Threshold** Default Value: 3 Format: U7 Low UV threshold.
	23:19	**Reserved** Access: RO Format: MBZ
	18:16	**UV Threshold Bits** Default Value: 3 Format: U3 Low UV transition width bits.

COLOR_PROCESSING_STATE - TCC State

	15:13	**Reserved** Access: RO Format: MBZ
	12:8	**STE Threshold** Default Value: 0 Format: U5 Skin tone pixels enhancement threshold.
	7:3	**Reserved** Access: RO Format: MBZ
	2:0	**STE Slope Bits** Default Value: 0 Format: U3 Skin tone pixels enhancement slope bits.
10	31:16	**Inverse UVMax Color** Default Value: 146 Format: U0.16 1 / UVMaxColor. Used for the SFs2 calculation.
	15:9	**Reserved** Access: RO Format: MBZ
	8:0	**UVMax Color** Default Value: 448 Format: U9 The maximum absolute value of the legal UV pixels. Used for the SFs2 calculation.

Color Calculator State Pointer Message Header Control

MHC_RT_CCSP - Color Calculator State Pointer Message Header Control

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:6	**Color Calculator State Pointer** Format: GeneralStateOffset[31:6] Specifies the 64-byte aligned point to the color calculator state. This pointer is relative to the General State Base Address.
	5:0	**Reserved** Access: RO Format: MBZ

Color Code Message Header Control

MHC_RT_CC - Color Code Message Header Control		
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:10	**Reserved** Access: RO Format: MBZ
	9:8	**Color Code** Format: U2 This ID is assigned by the Windower unit and is used to track synchronizing events. Reserved for HW implementation use
	7:0	**FFTID** Format: U8 This ID is assigned by the fixed function unit and is a unique identifier for the thread. It is used to free up resources used by the thread upon thread completion.

COMPRESSION_PAIR_BIT

COMPRESSION_PAIR_BIT		
Size (in bits):	2	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	1:0	**COMPRESSION PAIRING BIT** This field defines which 2 cachelines are combined in a 128B memory compression block.

Value	Name	Description
0h	Bit_6 **[Default]**	Pairing bit is Addr[6]: Two consecutive cachelines form a 128B compression block.
1h	Bit_7	Pairing bit is Addr[7]: Two cachelines with a 128B stride form a 128B compression block.
2h	Bit_8	Pairing bit is Addr[8]: Two cachelines with a 256B stride form a 128B compression block.

ComputeCS Hardware-Detected Error Bit Definitions

ComputeCS Hardware-Detected Error Bit Definitions

Source: ComputeCS
Size (in bits): 32
Default Value: 0x00000000

DWord	Bit	Description
0	31:12	**Reserved** Access: RO Format: MBZ
	11	**Reserved** Access: RO Format: MBZ
	10:3	**Reserved** Access: RO Format: MBZ
	2	**Command Privilege Violation Error** This bit is set if a command classified as privileged is parsed in a non-privileged batch buffer. The command will be converted to a NOOP and parsing will continue.
	1	**Reserved** Access: RO Format: MBZ
	0	**Instruction Error** This bit is set when the Renderer Instruction Parser detects an error while parsing an instruction. Instruction errors include: • Client ID value (Bits 31:29 of the Header) is not supported (only MI, 2D and 3D are supported). • Defeatured MI Instruction Opcodes: Value: 1 — Name: — Description: Instruction Error detected **Programming Notes** This error indications cannot be cleared except by reset (i.e., it is a fatal error).

Compute Engine Interrupt Vector

COMPUTE_INTR_VEC - Compute Engine Interrupt Vector		
Size (in bits):	16	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	15	**Catastrophic Error** This interrupt signals that a unrecoverable error (for e.g encountered fault when accessing a page mapped in Global GTT) during the engine processing. When Memory interface signals this error, the Command Streamer will stop parsing any more instructions. Scheduler is expected to reset the engine to evict the context
	14	**EU Restart Interrupt** EU Restart Interrupt is generated by the GA fabric, and not by Render Command Streamer. GA routes this interrupt to GuC independently of Command Stream.
	13	**Context Stall** Command streamer will generate a Context Stall interrupt when a high priority context gets stalled due to the other command streamer executing a normal priority or low priority context is "Run Alone" mode OR Command streamer will generate a Context Stall interrupt when a high priority context gets stalled while procuring run alone mode.
	12	**Reserved** Access: RO Format: MBZ
	11	**CS Wait On Semaphore**
	10	**Spare 10**
	9	**CS TR Invalid Tile Detection**
	8	**CS Context Switch Interrupt**
	7	**Page Fault Interrupt** This interrupt is for handling Legacy Page Fault interface for all Command Streamers [BCS, RCS, VCS, VECS]. When Fault Repair Mode is enabled, Interrupt mask register value is not looked at to generate interrupt due to page fault. Please refer to vol1c "page fault support" section for more details. In Advanced (PRQ) Fault Interface is done through GUC interface.
	6	**CS Watchdog Counter Expired**
	5	**Spare 5**
	4	**CS PIPE_CONTROL Notify**
	3	**CS Error Interrupt**
	2:1	**Spare 2**

COMPUTE_INTR_VEC - Compute Engine Interrupt Vector		
	0	**CS MI User Interrupt**

Context Descriptor Format

CONTEXT_DESCRIPTOR - Context Descriptor Format

Size (in bits):	64
Default Value:	0x00000000, 0x00000000

Description

This is the format of context descriptors which make up submitted execlists.

Context ID is a unique field assigned by GFX driver when a new context is created by which it is identified across all hierarchies of SW and HW.

- Context ID is used for semaphore signaling by hardware and software.
- Context ID matching is used by hardware to detect Lite Restore.
- Context ID is used by hardware for page fault reporting and response with IOMMU.
- Context switch reason and the associated Context ID are reported to Context Switch Status Buffer by hardware on a context switch.

Context ID which is a 32 bit field is further divided in to following segments described below:

- **Bits[63:61] (Bits 31:29 of Context ID)** represents **Engine class**.
- **Bits[60:55] (Bits 28:23 of Context ID)** represents **SW Counter**.
- **Bit[54] (Bit 22 of Context ID)** MBZ for SW programming; this bit is used by hardware to distinguish between F&H vs F&S page requests and response messages to and from IOMMU. This bit is used by hardware on receiving page response to properly manage the page fault counters.
- **Bits[53:48] (Bits 21:16 of Context ID)** represents **Engine Instance** (within a Engine class).
- **Bits[47:37] (Bits 15:5 of Context ID)**represents **SW Context ID** which is a software assigned unique context ID. (supports 2048 contexts per virtual function)
- **Bits[36:32] (Bits 4:0 of Context ID)** represents **Virtual Function Number** (when virtualization is enabled). Set to zero when virtualization is not enabled. This field contains the bits [4:0] of the Virtual Function Number.

Programming Note: "**Virtual Function Number**" must be always programmed to value 0x0.

Hardware compares the following fields of the outgoing context to that of the incoming context to detect a lite restore. Lite restore is detected when the following fields are equal and the incoming context does not have the "Force Restore" bit set. On a lite restore hardware will only sample the tail pointer from memory (LRCA) and keep executing the ongoing context with out initiating any context switch flows (Flush, Context Save, Context Restore). Lite restore is HW detected context switch optimization transparent to SW, Context Switch Status report and Context Switch Interrupt generation happens on a lite restore, Hardware Front End may temporarily get stalled from parsing new commands.

- **DW1.SW Context ID**
- **DW1.Virtual Function Number**
- **DW0.Logical Ring Context Address (LRCA)**
- **DW0. Reserved Bits[11:9]**

Context ID issued for comparing during lite restore and context specific OA enabling.

CONTEXT_DESCRIPTOR - Context Descriptor Format

Context ID is reported by hardware to OABUFFER along with the performance statistics counters, Context ID is used for filtering the statistics on per context basis.

DWord	Bit	Description
0..1	63:61	**Context ID - Engine Class** Bits 31:29 of Context ID represents Engine class.
	60:55	**Context ID - SW Counter** Bits 28:23 of Context ID represents SW Counter.
	54	**Reserved** Bits 22 of Context ID. This must be zero for SW programming; this bit is used by hardware to distinguish between F&H vs F&S page requests and response messages to and from IOMMU. This bit is used by hardware on receiving page response to properly manage the page fault counters.
	53:48	**Context ID - Engine Instance** Bits 21:16 of Context ID represents Engine Instance.
	47:37	**Context ID - SW Context ID** Bits 15:5 of Context ID represents SW Context ID, which is a software assigned unique context ID. (supports 2048 contexts per virtual function).
	36:32	**Context ID - Virtual Function Number** Bits 4:0 of Context ID represents the Virtual Function Number (when virtualization is enabled). Set to zero when virtualization is not enabled. This field contains the bits [4:0] of the Virtual Function Number.
	31:12	**Logical Ring Context Address (LRCA)** Format: GraphicsAddress[31:12] This field contains the 4 KB-aligned address of the Logical Ring Context associated with his execlist element. LRCA must be always programmed in GGTT memory.
	11	**Reserved** Access: RO Format: MBZ
	10:9	**Context Priority** This field indicates the prioritization of the thread dispatch associated with the corresponding context. Note that Render Engine and Compute Engine are executing contexts of their own with the corresponding priority programmed. For e.g: When Compute Engine is executing lower priority context when compared to the context executed by render engine, then threads dispatched from render engine (3D - VS, HS, DS, GS & PSD and GPGPU -TSG threads corresponding to render engine) are given priority over the TSG threads dispatched for compute engine. Value / Name: 0 – Low Priority 1 – Normal Priority 2 – High Priority

CONTEXT_DESCRIPTOR - Context Descriptor Format

Programming Notes
This field is only functional for RenderCS and ComputeCS and must be only programmed for context descriptor submitted to RenderCS and ComputeCS.

8 **Privilege Access**

Description
This field when set indicates PPGTT enabled.

Programming Notes
This field must be always set.

7:6 **Fault Handling**

Source:	CommandStreamer

Value	Name	Description
0h	Fault and Hang	Fault model is not supported and fault occurrence is treated as catastrophic. GAM indicates Fault Error to Command streamer. Fault Error interrupt is reported to scheduler. CommandStreamer will not initiate context switch on occurrence of Fault Error.
Others	Reserved	Reserved

Programming Notes
When execlist mode is set to "Legacy Context mode" Fault Handling mode must be set to "Fault and Hang." For proper programming for Page Fault modes, refer to memory interface section for the corresponding generation.

5 **Reserved**

Access:	RO
Format:	MBZ

4:3 **Addressing Mode & Legacy Context**

Format:	U2

Legacy context set indicates GPU is operating in legacy context mode of operation and doesn't support any SVM features. Legacy context reset indicates GPU is operating in advanced context mode of operation and support SVM features. Based on the Context mode set Addressing mode is interpreted appropriately. The table below summarizes the combinations supported.
GFX engine always uses 32b virtual addressing mode when translated using GGTT irrespective of below options.

Value	Name	Description
01b	Legacy Context with no 64 bit VA support	GPU is enabled for legacy context mode of operation and DOESN'T support any SVM features. GPU supports 32b PPGTT graphics virtual addressing. PDP*_DESCRIPTOR contains the base address to 4GB of memory space supported.

CONTEXT_DESCRIPTOR - Context Descriptor Format

11b	Legacy Context with 64 bit VA support	GPU is enabled for legacy context mode of operation and DOESN'T support any SVM features. GPU supports 64b (48bit canonical) PPGTT graphics virtual addressing and PDP0_DESCRIPTORcontains the base address to PML4 and other PDP Descriptors are ignored.
Others	Reserved	Other values are not supported.

2	**Force Restore** Setting this bit will force a context restore operation when switching to this context even if the LRCA in the CCID register (normally the LRCA of the last context from the prior execlist)matches this one. Note that it is legal (and likely desirable) for the **Render Context Restore Inhibit** bit(part of the CTXT_SR_CTL register) in the context image being restored to also be set. The "ring"context is being forced to be restored from a newly initialized context despite a possible LRCA match. However, the render context for such a newly initialized context will likely be uninitialized and so should not be restored.
1	**Reserved** Access: RO Format: MBZ
0	**Valid** Set if this register holds a valid context descriptor. SW should set this bit in the Element registers that it has set up to contain valid context descriptors. Any execlist elements that are not used in a submitted execlist must have this bit clear.

Context Status

Context Status

Size (in bits): 64

Default Value: 0x00000000, 0x00000000

The context status is an update sent by a Command Streamer to the scheduler.
The Engine Class and Instance ID specifies the Command streamer the event came from.
See the **Engine ID Definition** structure.
Context ID Away: Context ID of the context that the command streamer is switching away from.
Context ID To: Context ID of the context that the command streamer is switching execution to

DWord	Bit	Description
0	31:26	**Context ID To SW Counter** Format: U6
	25:15	**Context ID To SW Context ID** Format: U11 0x7FF: Is reserved to indicate HW idle state. "Ctxt-ID To SW Context-ID"set to 0x7FF in the report indicates HW went to Idle following this context switch. Indicate Active to Idle switch
	14:12	**Context ID To Virtual Function Number** Format: U3 This field contains the bits [2:0] of the Virtual Function Number.
	11:6	**Engine Instance** Format: U6
	5:3	**Engine Class** Format: U3
	2	**Semaphore Wait Mode** This field indicates the Semaphore Wait Mode (Poll or Signal) when the context switch is due to "Wait on Semaphore". This field is only valid when "Switch Detail" indicates "Wait on Semaphore". Value: 0 — Name: Signal Mode Value: 1 — Name: Poll Mode
	1	**Reserved** Access: RO Format: MBZ
	0	**Switched to New Queue** Value: 0 — Name: False Value: 1 — Name: True

Context Status

1 — 31:26 — Context ID Away SW Counter

Format:	U6

25:15 — Context ID Away SW Context ID

Format:	U11

0x7FF: Is reserved to indicate HW idle state. Ctxt-ID Away SW Context-ID set to 0x7FF in the CSB report indicates HW was Idle with no valid context at the time of context switch. Indicates Idle to Active switch.

14:12 — Context ID Away Virtual Function Number

Format:	U3

This field contains the bits [2:0] of the Virtual Function Number.

11:5 — Wait Detail

Format:	U7

This field is only valid when Switch Detail" indicates "Wait on Sync Flip" or "Wait on Scanline" or "Wait on VBlank" or "Wait on Semaphore".

- This field indicates the Display Plane ID when the "Switch Detail" indicates "Wait on Sync Flip".
- This field indicates the Display PipeID when the "Switch Detail" indicates "Wait on Scanline" or "Wait on VBlank".
- This field indicates the Wait Token Number when the "Switch Detail" indicates "Wait on Semaphore".

Value	Name	Description
[0,27]	Wait on Display	The value entered here is the Display Plane ID.
[0,26]	Wait on Semaphore	The value entered here is the Wait Token Number.

4 — Reserved

Access:	RO
Format:	MBZ

3:0 — Switch Detail

Format:	U4

Any values not listed below are reserved.

Value	Name	Description
0	Context Complete	Indicates context is complete with Head Pointer equal to Tail Pointer. In case of Render Engine it implies both RenderCS and PositionCS are complete with head pointer equal to tail pointer.
1	Wait on Sync Flip	
2	Wait on VBlank	

Context Status				
		3	Wait on Scanline	
		4	Wait on Semaphore	
		5	Context Pre-empted	This field is set when the context is preempted on a preemptable command and the command is not MI_SEMAPHORE_WAIT or MI_WAIT_FOR_EVENT.
		[6h-Fh]	Reserved	

CPS_STATE

CPS_STATE

Source: RenderCS

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:27	**Reserved**
	26:16	**MinCPSizeY** Format: S3.7 This bit-field defines the minimum shading ratio in Y dimension in screen space. This value is used only when Coarse Pixel Shading is enabled. It also defines the floor of the non-quantized CPSizeY for CPS_MODE_RADIAL. HW quantizes this value to determine Decoupled Rate. This value is used to clamp the CPSizeY for the lowest bound.
	15	**RESERVED**
	14	**ScaleAxis** Format: U1 This bit defines which dimension (along X- or Y- axis) should be scaled when computing Coarse Pixel Size values along ellipse in CPS_MODE_RADIAL. **Value** / **Name** / **Description** 0h / X axis / Use aspect to scale X-dimension 1h / Y axis / Use aspect to scale Y-dimension
	13:12	**Coarse Pixel Shading Mode** Format: U2 This bit-field defines Coarse Pixel Shading Mode. **Value** / **Name** / **Description** 0h / CPS_MODE_NONE / Coarse Pixel Shading is disabled. HW may be required to drive default values to shader inputs e.g. ScaleX = ScaleY = 1 and LODCompX = LODCompY = 1. 1h / CPS_MODE_CONSTANT / Coarse Pixel Shading Ratios are defined per DRAW based on MinCPSizeX and MinCPSizeY fields in this state (constant across render target). 2h / CPS_MODE_RADIAL / Coarse Pixel Shading Ratio varies radially from a focal point defined by (X_Focal, Y_Focal) relative to the viewport X/Y origin. This mode is typically used when there is Depth of Field or Ring of Confusion camera effects are desired. 3h / Reserved /

CPS_STATE

		Programming Notes It is a valid configuration to set the CPS mode other than CPS_MODE_NONE and request per-pixel dispatch in 3DSTATE_PS_EXTRA. In such case, 3DSTATE_PS_EXTRA configuration overrides 3DSTATE_CPS configuration, and effective CPS mode is set to CPS_MODE_NONE for this draw primitive. It is an INVALID configuration to set the CPS mode other than CPS_MODE_NONE and request per-sample dispatch in 3DSTATE_PS_EXTRA. Such configuration should be disallowed at the API level, and rendering results are undefined. It is a valid configuration to set the CPS mode to CPS_MODE_NONE and at the same time set Pixel Shader Is Per Coarse Pixel in 3DSTATE_PS_EXTRA. In such case, 3DSTATE_PS_EXTRA bit is ignored and shader is dispatched at pixel-rate; shader inputs specific to coarse-rate have undefined value (ActualCoarsePixelSize for example).
	11	**Statistics Enable** Format: Enable
	10:0	**MinCPSizeX** Format: S3.7 This bit-field defines the minimum shading ratio in X dimension in screen space. This value is used only when Coarse Pixel Shading is enabled. It also defines the floor of the non-quantized ScaleX for Mode 1. HW quantizes this value to determine Decoupled Rate.
1	31:27	**Reserved** Access: RO Format: MBZ
	26:16	**MaxCPSizeY** Format: S3.7 This bit-field defines the maximum shading ratio in Y dimension in screen space. This value is used only when Coarse Pixel Shading is enabled and Coarse Pixel Shading Mode is set to CPS_MODE_RADIAL. This value is used to clamp the CPSizeY for the highest bound. MaxCPSizeY must be greater than or equal to MinCPSizeY when this value is used.
	15:11	**Reserved** Access: RO Format: MBZ
	10:0	**MaxCPSizeX** Format: S3.7 This bit-field defines the maximum shading ratio in X dimension in screen space. This value is used only when Coarse Pixel Shading is enabled and Coarse Pixel Shading Mode is set to CPS_MODE_RADIAL. This value is used to clamp the CPSizeX for the highest bound. MaxCPSizeX must be greater than or equal to MinCPSizeX when this value is used.
2	31:16	**Reserved** Access: RO Format: MBZ

CPS_STATE

	15:0	**Y_Focal** Format: S15 This field defines the Y-coordinate for a focal point with respect to which shading ratio is computed in CPS_MODE_RADIAL. **Programming Notes** The valid data range is (-2^14 to 2^14-1)
3	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**X_Focal** Format: S15 This field defines the X-coordinate for a focal point with respect to which shading ratio is computed in CPS_MODE_RADIAL. **Programming Notes** The valid data range is (-2^14 to 2^14-1)
4	31:0	**My** Format: IEEE_FLOAT32 This field defines the slope of the transfer function for computing CPSizeY for CPS_MODE_RADIAL. **Programming Notes** SW needs to compute this from API supplied parameters: $(M_x, M_y) = \left(\frac{S_x^{max}-S_x^{min}}{R_{max}-R_{min}}, \frac{S_y^{max}-S_y^{min}}{R_{max}-R_{min}} \right)$ Mymust be greater than or equal to zero
5	31:0	**Mx** Format: IEEE_FLOAT32 This field defines the slope of the transfer function for computing CPSizeX for CPS_MODE_RADIAL. **Programming Notes** SW needs to compute this from API supplied parameters: $(M_x, M_y) = \left(\frac{S_x^{max}-S_x^{min}}{R_{max}-R_{min}}, \frac{S_y^{max}-S_y^{min}}{R_{max}-R_{min}} \right)$ Mx must be greater than or equal to zero
6	31:0	**Rmin** Format: IEEE_FLOAT32 This field defines (smaller) radius of the inner ellipse for CPS_MODE_RADIAL. All points on inner ellipse have coarse point size = (MinCPSizeX, MinCPSizeY).

CPS_STATE		
7	31:0	**Aspect** Format: IEEE_FLOAT32 This field defines aspect for both inner and outer ellipses in CPS_MODE_RADIAL. The aspect parameter must be within <0,1> range and Driver must program it as ratio of smallest ellipse radius to larger ellipse radius: Aspect = min(radiusX, radiusY) / max(radiusX, radiusY) where radiusX and radiusY define ellipse radius along x- and y- axes respectively. Note: Aspect must be same for both inner and outer ellipses.

CSC COEFFICIENT FORMAT

CSC COEFFICIENT FORMAT

Size (in bits): 16

Default Value: 0x00000000

Coefficients for the CSC are stored in sign-exponent-mantissa format. Two CSC coefficients are stored in each dword, the table below show the data packing in each dword.

DWord	Bit	Description
0	15	**Sign** Value: 0b — Name: Positive Value: 1b — Name: Negative
	14:12	**Exponent_bits** Represented as 2^(-n) Value: 110b — Name: 4 — Description: 4 or mantissa is bb.bbbbbbb Value: 111b — Name: 2 — Description: 2 or mantissa is b.bbbbbbbb Value: 000b — Name: 1 — Description: 1 or mantissa is 0.bbbbbbbbb Value: 001b — Name: 0.5 — Description: 0.5 or mantissa is 0.0bbbbbbbbb Value: 010b — Name: 0.25 — Description: 0.25 or mantissa is 0.00bbbbbbbbb Value: 011b — Name: 0.125 — Description: 0.125 or mantissa is 0.000bbbbbbbbb Value: Others — Name: Reserved — Description: Reserved
	11:3	**Mantissa**
	2:0	**Reserved** Access: RO Format: MBZ

Data Port 0 Message Types

MT_DP0 - Data Port 0 Message Types

Source: EuSubFunctionDataPort0

Size (in bits): 5

Default Value: 0x00000000

Lists all the Message Types in a Data Port 0 Message Descriptor [18:14].The Legacy messages are encoded in Data Port 0 with Bit 18 set to zero. The Message Header is optional for many (but not all) of these operations. The Scratch Block messages are encoded in Data Port 0 with Bit 18 set to one. A Message Header is required.

DWord	Bit	Description
0	4	**Legacy DAP-DC Message** Legacy Message

Value	Name	Description
0h	No **[Default]**	Legacy DAP-DC Message
1h	Reserved	Scratch Block Message, descriptor uses different Message Type encoding

DWord	Bit	Description
0	3:0	**Message Type** Specifies type of message

Value	Name	Description
00h	MT0R_B **[Default]**	Block Read message
01h	MT0R_AB	Aligned Block Read message
03h	MT0R_DWS	Dword Scattered Read message
04h	MT0R_BS	Byte Scattered Read message
07h	MT0_MEMFENCE	Memory Fence message
08h	MT0W_B	Block Write message
0Bh	MT0W_DWS	Dword Scattered Write message
0Ch	MT0W_BS	Byte Scattered Write message
Others	Reserved	Ignored

Data Port 1 Message Types

MT_DP1 - Data Port 1 Message Types

Source: EuSubFunctionDataPort1

Size (in bits): 5

Default Value: 0x00000000

Lists all the Message Types in a Data Port 1 Message Descriptor [18:14].Most surface and atomic operations, both typed and untyped, are encoded on Data Port 1. The Message Header is optional for many (but not all) of these operations. Most A64 Stateless operations are also encoded on Data Port 1. The Message Header is forbidden for all A64 messages on Data Port 1.

DWord	Bit	Description
0	4:0	**Message Type** Specifies type of message

Value	Name	Description
01h	MT1R_US	Untyped Surface Read message
02h	MT1A_UI	Untyped Atomic Integer Operation message
03h	MT1A_UHI	Untyped Atomic Half Integer Operation message
04h	MT1R_MB	Media Block Read message
05h	MT1R_TS	Typed Surface Read message
06h	MT1A_TA	Typed Atomic Integer Operation message
07h	MT1A_TAH	Typed Atomic Half Integer Operation message
08h	Reserved	Ignored
09h	MT1W_US	Untyped Surface Write message
0Ah	MT1W_MB	Media Block Write message
0Bh	MT1A_TC	Typed Atomic Counter Operation message
0Ch	MT1A_TCH	Typed Atomic Half Counter Operation message
0Dh	MT1W_TS	Typed Surface Write message
0Eh	Reserved	Ignored
10h	MT1R_A64_SB	A64 Scattered Read message
11h	MT1R_A64_US	A64 Untyped Surface Read message
12h	MT1A_A64_UI	A64 Untyped Atomic Integer Operation message
13h	MT1A_A64_UHI	A64 Untyped Atomic Half Integer Operation message
14h	MT1R_A64_B	A64 Block Read message
15h	MT1W_A64_B	A64 Block Write message
18h	Reserved	Ignored
19h	MT1W_A64_US	A64 Untyped Surface Write message
1Ah	MT1W_A64_SB	A64 Scattered Write message
1Bh	MT1A_UF	Untyped Atomic Float Operation message

MT_DP1 - Data Port 1 Message Types				
		1Ch	MT1A_UHF	Untyped Atomic Half Float Operation message
		1Dh	MT1A_A64_UF	A64 Untyped Atomic Float Operation message
		1Eh	MT1A_A64_UHF	A64 Untyped Atomic Half Float Operation message
		Others	Reserved	Ignored

Data Port Bindless Surface Extended Message Descriptor

DP_EXTDESC_BTI252 - Data Port Bindless Surface Extended Message Descriptor

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:12	**Bindless Surface Offset** Format: SurfaceStateOffset[25:6] Specifies the bindless surface offset if the Binding Table Index is set to 252. Ignored otherwise. The bindless surface offset is added to the Bindless Surface Base Address as bits 25:6 of the byte-based address. The resulting address is the location of SURFACE_STATE for this message.
	11	**Reserved** Access: RO Format: MBZ
	10:0	**Reserved** Access: RO Format: MBZ

Data Size Message Descriptor Control Field

MDC_DS - Data Size Message Descriptor Control Field

Size (in bits): 2

Default Value: 0x00000000

DWord	Bit	Description
0	1:0	**Data Size** Specifies the number of Bytes to be read or written

Value	Name	Description
00h	B	1 Byte
01h	W	2 Bytes
02h	DW	4 Bytes
03h	Reserved	Reserved

Depth Clear Value Format

Depth Clear Value Format		
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0 This field defines the clear value that will be applied to the depth buffer if the Depth Buffer Clear field is enabled. It is valid only if Depth Buffer Clear Value Valid is set. **Programming Notes:** The clear value must be between the min and max depth values (inclusive) defined in the CC_VIEWPORT. If the depth buffer format is D32_FLOAT, then values must be limited to the range of +0.0f and 1.0f inclusive; values outside this range are reserved	31:0	**Depth Clear Value** Format: IEEE_FLOAT

Deptrh Clear Value Format

STRUCTURE_TEMPLATE - Deptrh Clear Value Format		
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:0	**Address1** Format: IEEE_FLOAT32 Format: UNORM24 Format: UNORM16 When this field contains 24-bit UNORM, the upper 8-bits are reserved (0's) When this field contains 16-bit UNORM the upper 16-bits are reserved (0's)

DirectOperand

DirectOperand

Size (in bits): 14

Default Value: 0x00000000

DWord	Bit	Description
0	13:6	**RegNum** Format: U8 **Description** This field provide the register number for the operand. For a GRF register, is the part of a register address that aligns to a 256-bit (32-byte) boundary. For an ARF register, this field is encoded such that MSBs identify the architecture register type and LSBs provide the register number. An ARF register can only be destination or Source 0. Any Source 1 or Source 2 operands cannot be ARF registers. RegNum and SubRegNum together provide the byte-aligned address for the origin of a register region. RegNum provides bits 12:5 of that address. This field applies to both source and destination operands.
	5:1	**SubRegNum** This field provide the subregister number for the operand. For a GRF register, is the byte address within a 256-bit (32-byte) register. For an ARF register, determines the sub-register number according to the specified encoding for the given architecture register. RegNum and SubRegNum together provide the byte-aligned address for the origin of a GRF register region. RegNum provides bits 12:5 of that address. For one-source and two-source instructions, SubregNum provides bits 4:0. For three-source instructions, the address must be Word-aligned; SubRegNum provides bits 4:1of the address and bits 0 are zero.
	0	**RegFile** This field indicate whether Architecture register file or General register file are selected. Value / Name / Description 0 / ARF / Architecture Register File. 1 / GRF / General Register File. Allowed for any Source or Destination.

Display Engine Render Response Message Definition

DE_RRMD - Display Engine Render Response Message Definition

Size (in bits): 96

Default Value: 0x00000000, 0x00000000, 0x00000000

The Display Engine Render Response Registers use bit definitions from this table.

Programming Notes

Some events can be sent to CS (Render Command Streamer) or BCS (Blitter Command Streamer). For render response messages sending flip done or scanline events, the destination, CS or BCS, is selected depending on the initiator of the flip or the load scanline command. For render response messages sending vertical blank events, the destinations, CS or BCS, or both CS and BCS, is selected depending on the DE_RR_DEST setting. Command Streamer Plane number to the Display Plane name mapping is available in the Display Plane Capability and Interoperability section.

The STEREO3D_EVENT_MASK selects between left eye and right eye reporting of vertical blank and scanline events in stereo 3D modes.

DWord	Bit	Description
0	31	**Spare 31**
	30	**Reserved** Access: RO Format: MBZ
	29	**Reserved**
	28	**Spare 28**
	27	**Spare 27**
	26	**VDEnc Session3 Start** This event is reported when the Display pipe for this session reaches Tail Initial Update Delay. This event is not masked by DE_GUCRMR since it is sent directly by transcoder WD.
	25	**VDEnc Session2 Start** This event is reported when the Display pipe for this session reaches Tail Initial Update Delay. This event is not masked by DE_GUCRMR since it is sent directly by transcoder WD.
	24	**VDEnc Session1 Start** This event is reported when the Display pipe for this session reaches Tail Initial Update Delay. This event is not masked by DE_GUCRMR since it is sent directly by transcoder WD.
	23	**VDEnc Session0 Start** This event is reported when the Display pipe for this session reaches Tail Initial Update Delay. This event is not masked by DE_GUCRMR since it is sent directly by transcoder WD.

DE_RRMD - Display Engine Render Response Message Definition		
	22	**Pipe_C_DPST_Histogram_Event** This event is reported on the start of the DPST Histogram event for pipe C. This event should only be unmasked for GuC.
	21	**Pipe_C_Start_of_Vertical_Blank_Event** This event is reported on the start of the vertical blank of the transcoder attached to Pipe C.
	20	**Plane_6_Flip_Done_Event** This event is reported on the completion of a flip for Plane 6.
	19	**Plane_12_Flip_Done_Event** This event is reported on the completion of a flip for Plane 12.
	18	**Plane_11_Flip_Done_Event** This event is reported on the completion of a flip for Plane 11.
	17	**Plane_10_Flip_Done_Event** This event is reported on the completion of a flip for Plane 10.
	16	**Plane_9_Flip_Done_Event** This event is reported on the completion of a flip for Plane 9.
	15	**Plane_3_Flip_Done_Event** This event is reported on the completion of a flip for Plane 3.
	14	**Pipe_C_Scanline_Event** This event is reported on the start of the selected scan line for the transcoder attached to Pipe C.
	13	**Reserved**
	12	**Spare 12** Unused
	11	**Pipe_B_Start_of_Vertical_Blank_Event** This event is reported on the start of the vertical blank of the transcoder attached to Pipe B.
	10	**Plane_5_Flip_Done_Event** This event is reported on the completion of a flip for Plane 5.
	9	**Plane_2_Flip_Done_Event** This event is reported on the completion of a flip for Plane 2.
	8	**Pipe_B_Scanline_Event** This event is reported on the start of the selected scan line for the transcoder attached to Pipe B.
	7	**Plane_8_Flip_Done_Event** This event is reported on the completion of a flip for Plane 8.
	6	**Plane_7_Flip_Done_Event** This event is reported on the completion of a flip for Plane 7.
	5	**Reserved**
	4	**Spare 4** Unused
	3	**Pipe_A_Start_of_Vertical_Blank_Event** This event is reported on the start of the vertical blank of the transcoder attached to Pipe A.
	2	**Plane_4_Flip_Done_Event** This event is reported on the completion of a flip for Plane 4.

DE_RRMD - Display Engine Render Response Message Definition		
	1	**Plane_1_Flip_Done_Event** This event is reported on the completion of a flip for Plane 1.
	0	**Pipe_A_Scanline_Event** This event is reported on the start of the selected scan line for the transcoder attached to Pipe A.
1	31	**Spare 31** Unused.
	30	**Spare 30** Unused.
	29	**Spare 29** Unused.
	28	**Spare 28** Unused
	27	**Spare 27** Unused
	26	**Spare 26** Unused
	25	**Spare 25** Unused
	24	**Spare 24** Unused
	23	**Spare 23** Unused
	22	**Spare 22** Unused
	21	**Spare 21** Unused
	20	**Spare 20** Unused
	19	**Spare 19** Unused
	18	**Spare 18** Unused
	17	**Spare 17** Unused
	16	**Spare 16** Unused
	15	**Spare 15** Unused
	14	**Spare 14** Unused

DE_RRMD - Display Engine Render Response Message Definition		
	13	**Spare 13** Unused
	12	**Spare 12** Unused
	11	**Spare 11** Unused
	10	**Spare 10** Unused
	9	**Spare 9** Unused
	8	**Spare 8** Unused
	7	**Spare 7** Unused
	6	**Spare 6** Unused
	5	**Spare 5** Unused
	4	**Spare 4** Unused
	3	**Spare 3** Unused
	2	**Reserved**
	1	**Pipe_D_Scanline_Event** This event is reported on the start of the selected scan line for the transcoder attached to Pipe D. Some SKUs may not have Pipe D.
	0	**Pipe_D_Start_of_Vertical_Blank_Event** This event is reported on the start of the vertical blank of the transcoder attached to Pipe D. Some SKUs may not have Pipe D.
2	31	**Spare 31** Unused.
	30	**Spare 30** Unused.
	29	**Spare 29** Unused.
	28	**Spare 28** Unused
	27	**Spare 27** Unused
	26	**Spare 26** Unused

DE_RRMD - Display Engine Render Response Message Definition		
	25	**Spare 25** Unused
	24	**Spare 24** Unused
	23	**Spare 23** Unused
	22	**Spare 22** Unused
	21	**Spare 21** Unused
	20	**Spare 20** Unused
	19	**Plane_32_Flip_Done_Event** This event is reported on the completion of a flip for Plane 32.
	18	**Plane_31_Flip_Done_Event** This event is reported on the completion of a flip for Plane 31.
	17	**Plane_30_Flip_Done_Event** This event is reported on the completion of a flip for Plane 30.
	16	**Plane_29_Flip_Done_Event** This event is reported on the completion of a flip for Plane 29.
	15	**Plane_28_Flip_Done_Event** This event is reported on the completion of a flip for Plane 28.
	14	**Plane_27_Flip_Done_Event** This event is reported on the completion of a flip for Plane 27.
	13	**Plane_26_Flip_Done_Event** This event is reported on the completion of a flip for Plane 26.
	12	**Plane_25_Flip_Done_Event** This event is reported on the completion of a flip for Plane 25.
	11	**Plane_24_Flip_Done_Event** This event is reported on the completion of a flip for Plane 24.
	10	**Plane_23_Flip_Done_Event** This event is reported on the completion of a flip for Plane 23.
	9	**Plane_22_Flip_Done_Event** This event is reported on the completion of a flip for Plane 22.
	8	**Plane_21_Flip_Done_Event** This event is reported on the completion of a flip for Plane 21.
	7	**Plane_20_Flip_Done_Event** This event is reported on the completion of a flip for Plane 20.
	6	**Plane_19_Flip_Done_Event** This event is reported on the completion of a flip for Plane 19.

DE_RRMD - Display Engine Render Response Message Definition		
	5	**Plane_18_Flip_Done_Event** This event is reported on the completion of a flip for Plane 18.
	4	**Plane_17_Flip_Done_Event** This event is reported on the completion of a flip for Plane 17.
	3	**Plane_16_Flip_Done_Event** This event is reported on the completion of a flip for Plane 16.
	2	**Plane_15_Flip_Done_Event** This event is reported on the completion of a flip for Plane 15.
	1	**Plane_14_Flip_Done_Event** This event is reported on the completion of a flip for Plane 14.
	0	**Plane_13_Flip_Done_Event** This event is reported on the completion of a flip for Plane 13.

DUALSUBSLICE_HASH_TABLE_8x8

DUALSUBSLICE_HASH_TABLE_8x8

Size (in bits): 64

Default Value: 0x00000000, 0x00000000

8x8 [Y][X] dualsubslice hashing table. Each entry is a single bit that indicates which dualSubSlice(DSS) the indicated xy location maps to. A value of 0 indicates the larger DSS, or DSS=0 if both DSS have are balanced(have same number of enabled lsubslices)

DWord	Bit	Description
0	31:24	**SubSlice Hashing Table Entries[3]x[7:0]** Format: U8 Indicates the dualsubslice_id for the pixel block that has y=3 and x=7..0
	23:16	**SubSlice Hashing Table Entries[2]x[7:0]** Format: U8 Indicates the dualsubslice_id for the pixel block that has y=2 and x=7..0
	15:8	**SubSlice Hashing Table Entries[1]x[7:0]** Format: U8 Indicates the dualsubslice_id for the pixel block that has y=1 and x=7..0
	7:0	**SubSlice Hashing Table Entries[0]x[7:0]** Format: U8 Indicates the dualsubslice_id for the pixel block that has y=0 and x=7..0
1	31:24	**SubSlice Hashing Table Entries[7]x[7:0]** Format: U8 Indicates the dualsubslice_id for the pixel block that has y=7 and x=7..0
	23:16	**SubSlice Hashing Table Entries[6]x[7:0]** Format: U8 Indicates the dualsubslice_id for the pixel block that has y=6 and x=7..0
	15:8	**SubSlice Hashing Table Entries[5]x[7:0]** Format: U8 Indicates the dualsubslice_id for the pixel block that has y=5 and x=7..0
	7:0	**SubSlice Hashing Table Entries[4]x[7:0]** Format: U8 Indicates the dualsubslice_id for the pixel block that has y=4 and x=7..0

DUALSUBSLICE_HASH_TABLE_16x8

DUALSUBSLICE_HASH_TABLE_16x8		
Size (in bits):	128	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000	
16x8 [Y][X] dualsubslice hashing table. Each entry is a single bit that indicates which dualSubSlice(DSS) the indicated xy location maps to. A value of 0 indicates the larger DSS, or DSS=0 if both DSS have are balanced(have same number of enabled lsubslices)		
DWord	**Bit**	**Description**
0	31:16	**SubSlice Hashing Table Entries y[1]x[15:0]** Format: U16 Indicates the dualsubslice_id for the pixel block that has y=1 and x=15..0
	15:0	**SubSlice Hashing Table Entries y[0]x[15:0]** Format: U16 Indicates the dualsubslice_id for the pixel block that has y=0 and x=15..0
1	31:16	**SubSlice Hashing Table Entries y[3]x[15:0]** Format: U16 Indicates the dualsubslice_id for the pixel block that has y=3 and x=15..0
	15:0	**SubSlice Hashing Table Entries y[2]x[15:0]** Format: U16 Indicates the dualsubslice_id for the pixel block that has y=2 and x=15..0
2	31:16	**SubSlice Hashing Table Entries y[5]x[15:0]** Format: U16 Indicates the dualsubslice_id for the pixel block that has y=5 and x=15..0
	15:0	**SubSlice Hashing Table Entries y[4]x[15:0]** Format: U16 Indicates the dualsubslice_id for the pixel block that has y=4 and x=15..0
3	31:16	**SubSlice Hashing Table Entries y[7]x[15:0]** Format: U16 Indicates the dualsubslice_id for the pixel block that has y=7 and x=15..0
	15:0	**SubSlice Hashing Table Entries y[6]x[15:0]** Format: U16 Indicates the dualsubslice_id for the pixel block that has y=6 and x=15..0

Dword Data Payload Register

MDCR_DW - Dword Data Payload Register		
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0	31:0	**Dword0** Format: U32 Specifies the slot 0 data in this payload register
0.1	31:0	**Dword1** Format: U32 Specifies the slot 1 data in this payload register
0.2	31:0	**Dword2** Format: U32 Specifies the slot 2 data in this payload register
0.3	31:0	**Dword3** Format: U32 Specifies the slot 3 data in this payload register
0.4	31:0	**Dword4** Format: U32 Specifies the slot 4 data in this payload register
0.5	31:0	**Dword5** Format: U32 Specifies the slot 5 data in this payload register
0.6	31:0	**Dword6** Format: U32 Specifies the slot 6 data in this payload register
0.7	31:0	**Dword7** Format: U32 Specifies the slot 7 data in this payload register

Dword SIMD8 Atomic Operation CMPWR Message Data Payload

MDP_AOP8_DW2 - Dword SIMD8 Atomic Operation CMPWR Message Data Payload		
Size (in bits):	512	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Src0** Format: MDCR_DW Specifies the Slot [7:0] Source 0 data
1.0-1.7	255:0	**Src1** Format: MDCR_DW Specifies the Slot [7:0] Source 1 data

Dword SIMD8 Data Payload

MDP_DW_SIMD8 - Dword SIMD8 Data Payload		
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Data[7:0]** Format: MDCR_DW Specifies the Slot [7:0] data

Dword SIMD16 Atomic Operation CMPWR Message Data Payload

MDP_AOP16_DW2 - Dword SIMD16 Atomic Operation CMPWR Message Data Payload

Size (in bits): 1024

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Src0[7:0]** Format: MDCR_DW Specifies the Source 0 data for Slot [7:0]
1.0-1.7	255:0	**Src0[15:8]** Format: MDCR_DW Specifies the Source 0 data for Slot [15:8]
2.0-2.7	255:0	**Src1[7:0]** Format: MDCR_DW Specifies the Source 1 data for Slot [7:0]
3.0-3.7	255:0	**Src1[15:8]** Format: MDCR_DW Specifies the Source 1 data for Slot [15:8]

Dword SIMD16 Data Payload

MDP_DW_SIMD16 - Dword SIMD16 Data Payload		
Size (in bits):	512	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Data[7:0]** Format: MDCR_DW Specifies the Slot [7:0] data
1.0-1.7	255:0	**Data[15:8]** Format: MDCR_DW Specifies the Slot [15:8] data

Encoder Statistics Format

Encoder Statistics Format

Source:	VideoEnhancementCS
Size (in bits):	128
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000

The per block data is intended for use by the video encoder and consists of 16 bytes of Denoise block data and FMD variances. Much of the data is encoded as an 8-bit mantissa with the leading 1 removed and a 4-bit shift. To recover the original 17-bit integer this code can be used: If (exp != 0) Number = ((0x100 | Mantissa) « exp) » 7; else Number = mantissa;

The values for STAD, SHCM and SVCM for each 4x4 are shited down by 2 bits to make 14-bit values before being summed for the 16x4 block to make a 16-bit value. The result is then converted into the mantissa/exp format.

DWord	Bit	Description
0	31:24	**Tearing_Count 1 (FMD Variance[8])** Format: U8 Number of pixels that have (diff_cTcB > diff_cTcT + diff_cBcB) Value 0: Name: Description: DI is Disabled
	23:16	**Tearing_Count 2** Format: U8 If the frame is Deinterlaced with Top First in the DN/DI state then this is (FMD Variance[9]) = Number of pixels that have (diff_cTpB > diff_cTcT + diff_pBpB) If the frame is bottom first then this is (FMD Variance[10]) = Number of pixels that have (diff_cBpT > diff_pTpT + diff_cBcB) Value 0: Name: Description: DI is Disabled
	15:8	**Motion_Count (FMD Variance[7])** Format: U8 Number of pixels that are moving (different above a threshold) Value 0: Name: Description: DI is Disabled
	7:0	**Reserved** Access: RO Format: MBZ

Encoder Statistics Format

1	31:28	**sSTAD** Format: U4 Shift for the Sum in time of absolute differences for 16x4. Value 0 — Name — Description: Temporal Denoise Filtering is Disabled.
	27:24	**sSHCM** Format: U4 Shift for the Sum horizontally of absolute differences. Value 0 — Name — Description: DN is Disabled
	23:20	**sSVCM** Format: U4 Shift for the Sum vertically of absolute differences.
	19:16	**sDiff_cTpT** Format: U4 Shift for the sum of differences in top fields of current and previous frame. Value 0 — Name — Description: DI is Disabled
	15:12	**sDiff_cBpB** Format: U4 Shift for the sum of differences in bottom field of current and previous frame. Value 0 — Name — Description: DI is Disabled
	11:8	**sDiff_cTcB** Format: U4 Shift for the sum of differences between top and bottom field in current frame. Value 0 — Name — Description: DI is Disabled
	7:4	**sDiff_cTpB** Format: U4 Shift for the sum of differences between current top and previous bottom. Value 0 — Name — Description: DI is Disabled

Encoder Statistics Format

	3:0	**sDiff_cBpT** Format: U4 Shift for the sum of differences between current bottom and previous top. Value: 0 — Name: — Description: DI is Disabled
2	31:24	**mDiff_cBpB (FMD Variance[1])** Format: U8 Mantissa of sum of differences in bottom field of current and previous frame. Value: 0 — Name: — Description: DI is Disabled
	23:16	**mDiff_cTcB (FMD Variance[2])** Format: U8 Mantissa of sum of differences between top and bottom field in current frame. Value: 0 — Name: — Description: DI is Disabled
	15:8	**mDiff_cTpB (FMD Variance[3])** Format: U8 Mantissa of sum of differences between current top and previous bottom. Value: 0 — Name: — Description: DI is Disabled
	7:0	**mDiff_cBpT (FMD Variance[4])** Format: U8 Mantissa of sum of differences between current bottom and previous top. Value: 0 — Name: — Description: DI is Disabled
3	31:24	**mSTAD** Format: U8 Mantissa of Sum in time of absolute differences for 16x4. Value: 0 — Name: — Description: Temporal Denoise Filtering is disabled.
	23:16	**mSHCM** Format: U8 Mantissa of Sum horizontally of absolute differences. Value: 0 — Name: — Description: DN is Disabled

Encoder Statistics Format

	15:8	**mSVCM** Format: U8 Mantissa of Sum vertically of absolute differences. **Value** / **Name** / **Description** 0 / / DN is Disabled
	7:0	**mDiff_cTpT (FMD Variance[0])** Format: U8 Mantissa of sum of differences in top fields of current and previous frame. **Value** / **Name** / **Description** 0 / / DI is Disabled

EU_INSTRUCTION_BASIC_ONE_SRC

EU_INSTRUCTION_BASIC_ONE_SRC

Source: EuIsa

Size (in bits): 128

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0..3	127:96	**Src0.ImmValue[31:0]** Exists If: ([Src0.IsImm]==true)
	95:92	**CondCtrl** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **FlagModifier**
	95:64	**Src0.ImmValue[63:32]** Exists If: ([Src0.IsImm]==true) AND (([Src0.DataType]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df))
	87:84	**Src0.VertStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **VertStride**
	83:81	**Src0.Width** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **Width**
	80	**Src0.AddrMode** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**

EU_INSTRUCTION_BASIC_ONE_SRC

	65:64	**Src0.HorzStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Reserved** Access: RO Format: MBZ
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
	45:44	**Src0.Mod** Format: **SrcMod**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: **RegDataType**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: **ImmDataType**
	39:36	**Dst.DataType** Format: **RegDataType**
	35	**Dst.AddrMode** Format: **AddrMode**
	34	**Saturate** Format: **Saturate**
	33	**AccWrCtrl** Format: **AccWrCtrl**

EU_INSTRUCTION_BASIC_ONE_SRC

	32	**AtomicCtrl** Format: **AtomicCtrl**
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". **Value** / **Name** / **Description** 0 / Normal **[Default]** / Normal. Per channel write enable used for final write enable generation. 1 / NoMask / NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	30	**Reserved**
	29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. **Value** / **Name** / **Description** 0 / NoCompaction **[Default]** / No compaction. 128-bit native instruction supporting all instruction options. 1 / Compacted / Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
	28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields **Value** / **Name** / **Description** 0 / Positive **[Default]** / Positive polarity of predication. Use the predication mask produced by PredCtrl. 1 / Negative / Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
	27:24	**PredCtrl** Format: **PredCtrl** This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

EU_INSTRUCTION_BASIC_ONE_SRC

	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

EU_INSTRUCTION_BASIC_THREE_SRC

EU_INSTRUCTION_BASIC_THREE_SRC		
Source:	EuIsa	
Size (in bits):	128	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0..3	127:114	**Src2.Operand** Exists If: ([Src2.IsImm]==false) AND ([Header][Opcode]!=madm) Format: **DirectOperand**
	127:114	**Src2.Operand** Exists If: ([Src2.IsImm]==false) AND ([Header][Opcode]==madm) Format: **MacroOperand**
	127:112	**Src2.ImmValue[15:0]** Exists If: ([Src2.IsImm]==true)
	113:112	**Src2.HorzStride** Exists If: ([Src2.IsImm]==false) Format: **HorzStride**
	111:98	**Src1.Operand** Exists If: ([Header][Opcode]!=madm) Format: **DirectOperand**
	111:98	**Src1.Operand** Exists If: ([Header][Opcode]==madm) Format: **MacroOperand**
	97:96	**Src1.HorzStride** Format: **HorzStride**
	95:92	**CondCtrl** Format: **FlagModifier**
	91	**Src1.VertStride[1]** Format: **TernaryVertStride[1:1]**
	90:88	**Src1.DataType** Format: **TernaryDataType**
	87:86	**Src1.Mod** Format: **SrcMod**
	85:84	**Src2.Mod** Format: **SrcMod**

EU_INSTRUCTION_BASIC_THREE_SRC

	83	**Src1.VertStride[0]** Format: **TernaryVertStride[0:0]**
	82:80	**Src2.DataType** Format: **TernaryDataType**
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) AND ([Header][Opcode]!=madm) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) AND ([Header][Opcode]==madm) Format: **MacroOperand**
	79:64	**Src0.ImmValue[15:0]** Exists If: ([Src0.IsImm]==true)
	65:64	**Src0.HorzStride** Exists If: ([Src0.IsImm]==false) Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Header][Opcode]!=madm) Format: **DirectOperand** **Programming Notes** The Dst.Operand must be 64 bit aligned. i.e. Dst.Operand.SubRegNum[2:0] must be zero,
	63:50	**Dst.Operand** Exists If: ([Header][Opcode]==madm) Format: **MacroOperand**
	49	**Reserved** Format: MBZ
	48	**Dst.HorzStride** This field provides the distance in unit of data elements between two adjacent data elements within a row (horizontal) in the register region for the operand. **Value** / **Name** 0 / 1 element 1 / 2 element
	47	**Src2.IsImm** This field indicate that Source 2 operand is carrying an immediate value. **Value** / **Name** 0 / false 1 / true

EU_INSTRUCTION_BASIC_THREE_SRC

	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. **Value** / **Name** 0 – false 1 – true
	45:44	**Src0.Mod** Format: **SrcMod**
	43	**Src0.VertStride[1]** Format: **TernaryVertStride[1:1]**
	42:40	**Src0.DataType** Format: **TernaryDataType**
	39	**ExecDataType** This field indicate the datatype mode of ternary instruction. Integer or Float. **Value** / **Name** 0 – Integer 1 – Float
	38:36	**Dst.DataType** Format: **TernaryDataType**
	35	**Src0.VertStride[0]** Format: **TernaryVertStride[0:0]**
	34	**Saturate** Format: **Saturate**
	33	**AccWrCtrl** Format: **AccWrCtrl**
	32	**AtomicCtrl** Format: **AtomicCtrl**
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the the per channel write enables are used to generate the final write enable. This field should be normally "0". **Value** / **Name** / **Description** 0 – Normal **[Default]** – Normal. Per channel write enable used for final write enable generation. 1 – NoMask – NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	30	**Reserved**
	29	**CmptCtrl** Format: MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

EU_INSTRUCTION_BASIC_THREE_SRC		
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

EU_INSTRUCTION_BASIC_TWO_SRC

EU_INSTRUCTION_BASIC_TWO_SRC

Source: EuIsa
Size (in bits): 128
Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0..3	127:126	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	127:96	**Src1.ImmValue[31:0]** Exists If: ([Src1.IsImm]==true)
	125:122	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	121:120	**Src1.Mod** Exists If: ([Src1.IsImm]==false) Format: **SrcMod**
	119:116	**Src1.VertStride** Exists If: ([Src1.IsImm]==false) Format: **VertStride**
	115:113	**Src1.Width** Exists If: ([Src1.IsImm]==false) Format: **Width**
	112	**Src1.AddrMode** Exists If: ([Src1.IsImm]==false) Format: **AddrMode**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect) Format: **IndirectOperand**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct) Format: **DirectOperand**
	97:96	**Src1.HorzStride** Exists If: ([Src1.IsImm]==false) Format: **HorzStride**

EU_INSTRUCTION_BASIC_TWO_SRC

	95:92	**CondCtrl** Format: **FlagModifier**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==true) Format: **ImmDataType**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==false) Format: **RegDataType**
	87:84	**Src0.VertStride** Format: **VertStride**
	83:81	**Src0.Width** Format: **Width**
	80	**Src0.AddrMode** Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value. Value: 0 — Name: false **[Default]** Value: 1 — Name: true

EU_INSTRUCTION_BASIC_TWO_SRC

	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
	45:44	**Src0.Mod** Format: SrcMod
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: RegDataType
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: ImmDataType
	39:36	**Dst.DataType** Format: RegDataType
	35	**Dst.AddrMode** Format: AddrMode
	34	**Saturate** Format: Saturate
	33	**AccWrCtrl** Format: AccWrCtrl
	32	**AtomicCtrl** Format: AtomicCtrl
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". Value: 0 — Name: Normal **[Default]** — Description: Normal. Per channel write enable used for final write enable generation. Value: 1 — Name: NoMask — Description: NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	30	**Reserved**
	29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations

supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

EU_INSTRUCTION_BASIC_TWO_SRC		
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

EU_INSTRUCTION_BRANCH_ONE_SRC

EU_INSTRUCTION_BRANCH_ONE_SRC

Source: EuIsa
Size (in bits): 128
Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0..3	127:96	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	127:96	**JIP** Exists If: ([Src0.IsImm]==true) Format: S31 The byte-aligned jump distance if a jump is taken for the channel
	95:80	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	95:64	**Reserved** Exists If: ([Src0.IsImm]==true) Format: MBZ
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) Format: **DirectOperand**
	65:64	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	63:50	**Dst.Operand** Format: **DirectOperand**
	49:47	**Reserved** Access: RO Format: MBZ
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — Name: false Value: 1 — Name: true

EU_INSTRUCTION_BRANCH_ONE_SRC

45:34	**Reserved** Access: RO Format: MBZ
33	**BranchCtrl** This field is used by *goto* ,*if*, and *else* instructions to control branching. See the **goto** instruction description for more information about BranchCtrl.
32	**AtomicCtrl** Format: **AtomicCtrl**
31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30	**Reserved**
29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

EU_INSTRUCTION_BRANCH_ONE_SRC

	27:24	**PredCtrl** Format: **PredCtrl** This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.
	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

EU_INSTRUCTION_BRANCH_TWO_SRC

EU_INSTRUCTION_BRANCH_TWO_SRC

Source: EuIsa

Size (in bits): 128

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0..3	127:96	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	127:96	**JIP** Exists If: ([Src0.IsImm]==true) Format: S31 The byte-aligned jump distance if a jump is taken for the channel.
	95:80	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	95:64	**Reserved** Exists If: ([Src0.IsImm]==true) AND ([Src1.IsImm]==false) Format: MBZ
	95:64	**UIP** Exists If: ([Src0.IsImm]==true) AND ([Src1.IsImm]==true) Format: S31 The byte aligned jump distance if a jump is taken for the instruction.
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) Format: **DirectOperand**
	65:64	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	63:50	**Dst.Operand** Format: **DirectOperand**
	49:48	**Reserved** Access: RO Format: MBZ

EU_INSTRUCTION_BRANCH_TWO_SRC

47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value

Value	Name
0	false
1	true

46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value

Value	Name
0	false
1	true

45:34	**Reserved**

Access:	RO
Format:	MBZ

33	**BranchCtrl** This field is used by *goto*, *if*, and *else* instructions to control branching. See the goto instruction description for more information about BranchCtrl.
32	**AtomicCtrl**

Format:	AtomicCtrl

31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30	**Reserved**
29	**CmptCtrl**

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

EU_INSTRUCTION_BRANCH_TWO_SRC

	28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

	27:24	**PredCtrl** Format: **PredCtrl** This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.
	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

EU_INSTRUCTION_ILLEGAL

EU_INSTRUCTION_ILLEGAL		
Source:	EuIsa	
Size (in bits):	128	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0..3	127:7	**Reserved** Access: RO Format: MBZ
	6:0	**Opcode** Format: EU_OPCODE

EU_INSTRUCTION_MATH

EU_INSTRUCTION_MATH		
Source:	EuIsa	
Size (in bits):	128	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0..3	127:126	**Reserved** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==false) Format: MBZ
	127:96	**Src0.ImmValue[31:0]** Exists If: ([Src0.IsImm]==true) AND ([Src1.IsImm]==false)
	127:96	**Src1.ImmValue[31:0]** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==true)
	125:122	**Reserved** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==false) Format: MBZ
	121:120	**Src1.Mod** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==false) Format: **SrcMod**
	119:116	**Src1.VertStride** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==false) Format: **VertStride**
	115:113	**Src1.Width** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==false) Format: **Width**
	112	**Reserved** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==false) Format: MBZ
	111:98	**Src1.Operand** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==false) AND ([FuncCtrl]==INVM) Format: **MacroOperand**
	111:98	**Src1.Operand** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==false) AND ([FuncCtrl]!=INVM) Format: **DirectOperand**

EU_INSTRUCTION_MATH

	97:96	**Src1.HorzStride** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==false) Format: **HorzStride**
	95:92	**FuncCtrl** Format: **MathFC**
	91:88	**Src1.DataType** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==true) Format: **ImmDataType**
	91:88	**Src1.DataType** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==false) Format: **RegDataType**
	91:64	**Reserved** Exists If: ([Src0.IsImm]==true) Format: MBZ
	87:84	**Src0.VertStride** Exists If: ([Src0.IsImm]==false) Format: **VertStride**
	83:81	**Src0.Width** Exists If: ([Src0.IsImm]==false) Format: **Width**
	80	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) AND (([FuncCtrl]==INVM) OR ([FuncCtrl]==RSQTM)) Format: **MacroOperand**
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) AND (([FuncCtrl]!=INVM) AND ([FuncCtrl]!=RSQTM)) Format: **DirectOperand**
	65:64	**Src0.HorzStride** Exists If: ([Src0.IsImm]==false) Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([FuncCtrl]!=INVM) AND ([FuncCtrl]!=RSQTM) Format: **DirectOperand**

EU_INSTRUCTION_MATH

	63:50	**Dst.Operand** Exists If: ([FuncCtrl]==INVM) OR ([FuncCtrl]==RSQTM) Format: **MacroOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value Value: 0 — Name: false Value: 1 — Name: true
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — Name: false Value: 1 — Name: true
	45:44	**Src0.Mod** Exists If: ([Src0.IsImm]==false) Format: **SrcMod**
	45:44	**Reserved** Exists If: ([Src0.IsImm]==true) Format: MBZ
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: **RegDataType**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: **ImmDataType**
	39:36	**Dst.DataType** Format: **RegDataType**
	35	**Reserved** Access: RO Format: MBZ
	34	**Saturate** Format: **Saturate**
	33	**AccWrCtrl** Format: **AccWrCtrl**

EU_INSTRUCTION_MATH

	32	**AtomicCtrl** Format: **AtomicCtrl**
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the the per channel write enables are used to generate the final write enable. This field should be normally "0". **Value** / **Name** / **Description** 0 / Normal **[Default]** / Normal. Per channel write enable used for final write enable generation. 1 / NoMask / NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	30	**Reserved**
	29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. **Value** / **Name** / **Description** 0 / NoCompaction **[Default]** / No compaction. 128-bit native instruction supporting all instruction options. 1 / Compacted / Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
	28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields **Value** / **Name** / **Description** 0 / Positive **[Default]** / Positive polarity of predication. Use the predication mask produced by PredCtrl. 1 / Negative / Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
	27:24	**PredCtrl** Format: **PredCtrl** This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

EU_INSTRUCTION_MATH

	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

EU_INSTRUCTION_NOP

EU_INSTRUCTION_NOP		
Source:	Eulsa	
Size (in bits):	128	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0..3	127:31	**Reserved** Access: RO Format: MBZ
	30	**Reserved**
	29:28	**Reserved** Access: RO Format: MBZ
	27:26	**Reserved** Format: MBZ
	25:18	**Reserved** Access: RO Format: MBZ
	17:16	**Reserved** Format: MBZ
	15:0	**Header** Format: **Header**

EU_INSTRUCTION_SEND

EU_INSTRUCTION_SEND		
Source:	EuIsa	
Size (in bits):	128	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0..3	127:124	**ExDesc[31:28]** Exists If: ([ExDesc.IsReg]==false) Format: **ExMsgDesc[31:28]**
	127:124	**Reserved** Exists If: ([ExDesc.IsReg]==true) Format: MBZ
	123:122	**Desc[31:30]** Exists If: ([Desc.IsReg]==false) Format: **MsgDesc[31:30]**
	123:113	**Reserved** Exists If: ([Desc.IsReg]==true) Format: MBZ
	121:113	**Desc[19:11]** Exists If: ([Desc.IsReg]==false) Format: **MsgDesc[19:11]**
	112	**Reserved** Access: RO Format: MBZ
	111:104	**Src1.RegNum** Format: **DirectOperand[13:6]**
	103:99	**Src1.Length** Exists If: ([ExDesc.IsReg]==false) Format: **ExMsgDesc[10:6]**
	103:99	**Reserved** Exists If: ([ExDesc.IsReg]==true) Format: MBZ
	98	**Src1.RegFile** Format: **DirectOperand[0]**

EU_INSTRUCTION_SEND

	97:96	**Reserved** Exists If: ([ExDesc.IsReg]==true) Format: MBZ
	97:96	**ExDesc[27:26]** Exists If: ([ExDesc.IsReg]==false) Format: **ExMsgDesc[27:26]**
	95:92	**SFID** Format: **SFID**
	91:81	**Reserved** Exists If: ([Desc.IsReg]==true) Format: MBZ
	91:81	**Desc[10:0]** Exists If: ([Desc.IsReg]==false) Format: **MsgDesc[10:0]**
	80	**Reserved** Access: RO Format: MBZ
	79:72	**Src0.RegNum** Format: **DirectOperand[13:6]**
	71	**MsgDesc[29]** Exists If: ([Desc.IsReg]==false) Format: **MsgDesc[29:29]**
	71:67	**Reserved** Exists If: ([Desc.IsReg]==true) Format: MBZ
	70:67	**Src0.Length** Exists If: ([Desc.IsReg]==false) Format: **MsgDesc[28:25]**
	66	**Src0.RegFile** Format: **DirectOperand[0]**
	65:64	**Reserved** Exists If: ([ExDesc.IsReg]==true) Format: MBZ
	65:64	**ExDesc[25:24]** Exists If: ([ExDesc.IsReg]==false) Format: **ExMsgDesc[25:24]**

EU_INSTRUCTION_SEND

	63:56	**Dst.RegNum** Format: **DirectOperand[13:6]**
	55:51	**Reserved** Exists If: ([Desc.IsReg]==true) Format: MBZ
	55:51	**Dst.Length** Exists If: ([Desc.IsReg]==false) Format: **MsgDesc[24:20]**
	50	**Dst.RegFile** Format: **DirectOperand[0]**
	49	**ExDesc.IsReg** This field indicates that the extended message descriptor is provided by the address register, selected by the AddrSubRegNum[3:1]. Value: 0 — Name: false Value: 1 — Name: true
	48	**Desc.IsReg** This field indicates that the message descriptor is provided by the address subregister a0.0. Value: 0 — Name: false Value: 1 — Name: true
	47:43	**Reserved** Exists If: ([ExDesc.IsReg]==true) Format: MBZ
	47:35	**ExDesc[23:11]** Exists If: ([ExDesc.IsReg]==false) Format: **ExMsgDesc[23:11]**
	42:40	**AddrSubRegNum[3:1]** Exists If: ([ExDesc.IsReg]==true) Format: **AddrSubRegNum[3:1]**
	38:36	**Reserved** Exists If: ([ExDesc.IsReg]==true) Format: MBZ
	35	**Reserved** Exists If: ([ExDesc.IsReg]==true) Format: MBZ

EU_INSTRUCTION_SEND

34 EOT

This field controls the termination of the thread. For a send instruction, if this field is set, EU will terminate the thread and also set the EOT bit in the message sideband. This field only applies to the send instruction. It is not present for other instructions.

Value	Name
0	Thread is not terminated
1	EOT

33 FusionCtrl

This field provides explicit control for EU fusion lock-step execution. When this bit is set to 1b, the instruction is executed serially starting from the first EU to the last EU in the fused set.

Value	Name
0	Normal lockstep execution
1	Serialized execution

32 AtomicCtrl

Format:	AtomicCtrl

31 MaskCtrl

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 Reserved

29 CmptCtrl

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16 **ExecSize**

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

15:0 **Header**

Format:	Header

EU_INSTRUCTION_SYNC

EU_INSTRUCTION_SYNC

Source: EuIsa
Size (in bits): 128
Default Value: 0x00000000, 0x00010000, 0x00000000, 0x00000000

DWord	Bit	Description
0..3	127:96	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	127:96	**Src0.ImmValue[31:0]** Exists If: ([Src0.IsImm]==true)
	95:92	**SyncCtrl** Format: SyncFC
	91:88	**Reserved** Format: MBZ
	87	**Reserved** Format: MBZ
	86:80	**Reserved** Format: MBZ
	79:66	**Reserved** Format: MBZ
	65:50	**Reserved** Format: MBZ
	49:48	**Dst.HorzStride** Value: 01b — Name: 1 elements **[Default]** Value: Others — Name: Reserved
	47	**Reserved** Format: MBZ
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — Name: false Value: 1 — Name: true
	45:44	**Reserved** Format: MBZ

EU_INSTRUCTION_SYNC

43:40 **Src0.DataType**

Exists If:	([Src0.IsImm]==true)
Format:	**ImmDataType**

43:40 **Reserved**

Exists If:	([Src0.IsImm]==false)
Format:	MBZ

39:33 **Reserved**

Format:	MBZ

32 **AtomicCtrl**

Format:	**AtomicCtrl**

31 **MaskCtrl**

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 **Reserved**

29 **CmptCtrl**

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive	Positive polarity of predication. Use the predication mask produced

	[Default]	by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 PredCtrl

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 FlagRegNum[0]

This field specifies bit[0] of the register number for a flag register operand.

22 FlagSubRegNum

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 ChanOff

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16 ExecSize

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

15:0 Header

Format:	Header

Event Data Payload

MDP_EVENT - Event Data Payload		
Source:	EuSubFunctionGateway	
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0	31:24	**Reserved** Access: RO Format: MBZ
	23:0	**Event ID** Format: U24 Indicates the ID of the event to be signalled.
1..7	223:0	**Reserved** Access: RO Format: MBZ

ExMsgDesc

ExMsgDesc		
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:12	**ExtendedFunctionControl[31:12]** Format: U20 This field is sent to the target function unit as extended function control.
	11	**CPS LOD Compensation**
	10:6	**Extended Message Length** Format: U5 This field specifies the number of 256-bit GRF registers starting from <Src1.RegNum> to be sent out on the request message payload **Programming Notes** When <Src1> is null this field must be 0.
	5:0	**Reserved** Format: MBZ

Extended Message Descriptor Render Target

Extended Message Descriptor Render Target

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:25	**Reserved** Access: RO Format: MBZ
	24:21	**Reserved** Access: RO Format: MBZ
	20	**Null Render Target** **Description** When this bit is set, RT write or read message is considered to be a dummy message and as if it is directed to the NULL render target. Setting this bit in the descriptor, allows SW to not use any entry from the Binding Table to convey NULL RT. **Programming Notes** SW must set this bit for Render Target Write just to clear the Pixel Scoreboard without allocating an entry in the Binding Table. For Texel Shaders, this bit must be set while sending a NULL Render Target Write Message to communicate End of Texel Shader Thread (EOT) to the AMFSunit.
	19:16	**Pixel shading phase for CPS+PS inner loop** Format: U4 The loop counter value of a PS phase within CPS+PS(+S) monolithic shader; this value is same as value delivered to Pixel Interpolator when requesting input data for a new PS loop phase. Data Port uses this index to match pixel XY positions delivered by bypass path from PI hardware when a new phase started. **Programming Notes** The SIMD width of a render target read/write message with PS phase counter must match SIMD width of the Pixel Interpolator Pull message which returns PS phase counter.
	15	**Src0 Alpha Present** **Description** Setting this bit indicates that Src0 Alpha is present in the Render Target Write Message. **Programming Notes** SW must not send a header to send Src0 Alpha present, but instead, it must set this bit and avoid sending the header for RT write messages.

Extended Message Descriptor Render Target

	14:12	**Render Target Array Index** Format: U3 **Description** This bit-field is used to set the Render Target Index for MRT messages. **Programming Notes** SW must not send a header to send Render Target Array Index, but instead, it must set this bit-field appropriately and avoid sending the header for RT messages.
	11:10	**Reserved** Access: RO Format: MBZ
	9:6	**Extended Message Length** Format: U4 This field specifies the number of 256-bit GRF registers starting from <src1> to be sent out on the request message payload. Valid value ranges from 0 to 15. Must be 0 when <src1> is null register.
	5	**End of Thread** This field, if set, indicates that this is the final message of the thread and the threads resources can be reclaimed.
	4	**Reserved** Access: RO Format: MBZ
	3:0	**Target Function ID** This field indicates the function unit for which the message is intended. *Refer to GPU Overview document for the mapping of Shared Function IDs*

Extended Message Descriptor - Sampling Engine

<table>
<tr><th colspan="3">Extended Message Descriptor - Sampling Engine</th></tr>
<tr><td colspan="3">Size (in bits): 32
Default Value: 0x00000000</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="3">0</td><td>31:12</td><td>Bindless Surface Offset
<table><tr><td>Format:</td><td>SurfaceStateOffset[25:6]</td></tr></table>Specifies the bindless surface offset if the Binding Table Index is set to 252. Ignored otherwise. The bindless surface offset is added to the Bindless Surface Base Address as bits 25:6 of the byte-based address. The resulting address is the location of SURFACE_STATE for this message.</td></tr>
<tr><td>11</td><td>CPS Message LOD Compensation Enable
<table><tr><td>Format:</td><td>Enable</td></tr></table>Specifies whether LOD Compenstation is enabled for this message. See CPS LOD Compensation Enable in SAMPLER_STATE for more details.<table><tr><th>Programming Notes</th></tr><tr><td>This field must be disabled if the response length of the message is zero.</td></tr><tr><td>This field must be disabled if the messages is from a 32-pixel dispatch thread.</td></tr><tr><td>This field must be disabled unless SIMD Mode is SIMD8* or SIMD16*.</td></tr></table></td></tr>
<tr><td>10:0</td><td>Execution Unit Extended Message Descriptor Definition
<table><tr><td>Format:</td><td>ExMsgDesc[10:0]</td></tr></table></td></tr>
</table>

FFTID Message Header Control

MHC_FFTID - FFTID Message Header Control

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7:0	**FFTID** Format: U8 Fixed function thread ID, used to free up resources by the thread on thread completion.

Filter_Coefficient

Filter_Coefficient		
Size (in bits):	8	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	7:0	**Filter Coefficient** Format: S1.6 Range : [-1 63/64, +1 63/64]

Filter_Coefficients

Filter_Coefficients		
Size (in bits):	64	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0	63:56	**Filter Coefficient Offset 7** Format: **Filter_Coefficient**
	55:48	**Filter Coefficient Offset 6** Format: **Filter_Coefficient**
	47:40	**Filter Coefficient Offset 5** Format: **Filter_Coefficient**
	39:32	**Filter Coefficient Offset 4** Format: **Filter_Coefficient**
	31:24	**Filter Coefficient Offset 3** Format: **Filter_Coefficient**
	23:16	**Filter Coefficient Offset 2** Format: **Filter_Coefficient**
	15:8	**Filter Coefficient Offset 1** Format: **Filter_Coefficient**
	7:0	**Filter Coefficient Offset 0** Format: **Filter_Coefficient**

FrameDeltaQp

FrameDeltaQp		
Size (in bits):	64	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0..1	63:56	**FrameDeltaQp[7]** Format: S7
	55:48	**FrameDeltaQp[6]** Format: S7
	47:40	**FrameDeltaQp[5]** Format: S7
	39:32	**FrameDeltaQp[4]** Format: S7
	31:24	**FrameDeltaQp[3]** Format: S7
	23:16	**FrameDeltaQp[2]** Format: S7
	15:8	**FrameDeltaQp[1]** Format: S7
	7:0	**FrameDeltaQp[0]** Format: S7

FrameDeltaQpRange

FrameDeltaQpRange		
Size (in bits):	64	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0..1	63:56	**FrameDeltaQpRange[7]** Format: U8
	55:48	**FrameDeltaQpRange[6]** Format: U8
	47:40	**FrameDeltaQpRange[5]** Format: U8
	39:32	**FrameDeltaQpRange[4]** Format: U8
	31:24	**FrameDeltaQpRange[3]** Format: U8
	23:16	**FrameDeltaQpRange[2]** Format: U8
	15:8	**FrameDeltaQpRange[1]** Format: U8
	7:0	**FrameDeltaQpRange[0]** Format: U8

Gamut_Expansion_Gamma_Correction

Gamut_Expansion_Gamma_Correction	
Source:	VideoEnhancementCS
Size (in bits):	32768
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x01000100, 0x01000100, 0x01000100, 0x01000100, 0x02000200, 0x02000200, 0x02000200, 0x02000200, 0x03000300, 0x03000300, 0x03000300, 0x03000300, 0x04000400, 0x04000400, 0x04000400, 0x04000400, 0x05000500, 0x05000500, 0x05000500, 0x05000500, 0x06000600, 0x06000600, 0x06000600, 0x06000600, 0x07000700, 0x07000700, 0x07000700, 0x07000700, 0x08000800, 0x08000800, 0x08000800, 0x08000800, 0x09000900, 0x09000900, 0x09000900, 0x09000900, 0x0A000A00, 0x0A000A00, 0x0A000A00, 0x0A000A00, 0x0B000B00, 0x0B000B00, 0x0B000B00, 0x0B000B00, 0x0C000C00, 0x0C000C00, 0x0C000C00, 0x0C000C00, 0x0D000D00, 0x0D000D00, 0x0D000D00, 0x0D000D00, 0x0E000E00, 0x0E000E00, 0x0E000E00, 0x0E000E00, 0x0F000F00, 0x0F000F00, 0x0F000F00, 0x0F000F00, 0x10001000, 0x10001000, 0x10001000, 0x10001000, 0x11001100, 0x11001100, 0x11001100, 0x11001100, 0x12001200, 0x12001200, 0x12001200, 0x12001200, 0x13001300, 0x13001300, 0x13001300, 0x13001300, 0x14001400, 0x14001400, 0x14001400, 0x14001400, 0x15001500, 0x15001500, 0x15001500, 0x15001500, 0x16001600, 0x16001600, 0x16001600, 0x16001600, 0x17001700, 0x17001700, 0x17001700, 0x17001700, 0x18001800, 0x18001800, 0x18001800, 0x18001800, 0x19001900, 0x19001900, 0x19001900, 0x19001900, 0x1A001A00, 0x1A001A00, 0x1A001A00, 0x1A001A00, 0x1B001B00, 0x1B001B00, 0x1B001B00, 0x1B001B00, 0x1C001C00, 0x1C001C00, 0x1C001C00, 0x1C001C00, 0x1D001D00, 0x1D001D00, 0x1D001D00, 0x1D001D00, 0x1E001E00, 0x1E001E00, 0x1E001E00, 0x1E001E00, 0x1F001F00, 0x1F001F00, 0x1F001F00, 0x1F001F00, 0x20002000, 0x20002000, 0x20002000, 0x20002000, 0x21002100, 0x21002100, 0x21002100, 0x21002100, 0x22002200, 0x22002200, 0x22002200, 0x22002200, 0x23002300, 0x23002300, 0x23002300, 0x23002300, 0x24002400, 0x24002400, 0x24002400, 0x24002400, 0x25002500, 0x25002500, 0x25002500, 0x25002500, 0x26002600, 0x26002600, 0x26002600, 0x26002600, 0x27002700, 0x27002700, 0x27002700, 0x27002700, 0x28002800, 0x28002800, 0x28002800, 0x28002800, 0x29002900, 0x29002900, 0x29002900, 0x29002900, 0x2A002A00, 0x2A002A00, 0x2A002A00, 0x2A002A00, 0x2B002B00, 0x2B002B00, 0x2B002B00, 0x2B002B00, 0x2C002C00, 0x2C002C00, 0x2C002C00, 0x2C002C00, 0x2D002D00, 0x2D002D00, 0x2D002D00, 0x2D002D00, 0x2E002E00, 0x2E002E00, 0x2E002E00, 0x2E002E00, 0x2F002F00, 0x2F002F00, 0x2F002F00, 0x2F002F00, 0x30003000, 0x30003000, 0x30003000, 0x30003000, 0x31003100, 0x31003100, 0x31003100, 0x31003100, 0x32003200, 0x32003200, 0x32003200, 0x32003200, 0x33003300, 0x33003300, 0x33003300, 0x33003300, 0x34003400, 0x34003400, 0x34003400, 0x34003400, 0x35003500, 0x35003500, 0x35003500, 0x35003500, 0x36003600, 0x36003600, 0x36003600, 0x36003600, 0x37003700, 0x37003700, 0x37003700, 0x37003700, 0x38003800, 0x38003800, 0x38003800, 0x38003800, 0x39003900, 0x39003900, 0x39003900, 0x39003900, 0x3A003A00, 0x3A003A00, 0x3A003A00, 0x3A003A00, 0x3B003B00, 0x3B003B00, 0x3B003B00, 0x3B003B00, 0x3C003C00, 0x3C003C00, 0x3C003C00, 0x3C003C00, 0x3D003D00, 0x3D003D00, 0x3D003D00,

Gamut_Expansion_Gamma_Correction

0x3D003D00, 0x3E003E00, 0x3E003E00, 0x3E003E00, 0x3E003E00, 0x3F003F00,
0x3F003F00, 0x3F003F00, 0x3F003F00, 0x40004000, 0x40004000, 0x40004000,
0x40004000, 0x41004100, 0x41004100, 0x41004100, 0x41004100, 0x42004200,
0x42004200, 0x42004200, 0x42004200, 0x43004300, 0x43004300, 0x43004300,
0x43004300, 0x44004400, 0x44004400, 0x44004400, 0x44004400, 0x45004500,
0x45004500, 0x45004500, 0x45004500, 0x46004600, 0x46004600, 0x46004600,
0x46004600, 0x47004700, 0x47004700, 0x47004700, 0x47004700, 0x48004800,
0x48004800, 0x48004800, 0x48004800, 0x49004900, 0x49004900, 0x49004900,
0x49004900, 0x4A004A00, 0x4A004A00, 0x4A004A00, 0x4A004A00, 0x4B004B00,
0x4B004B00, 0x4B004B00, 0x4B004B00, 0x4C004C00, 0x4C004C00, 0x4C004C00,
0x4C004C00, 0x4D004D00, 0x4D004D00, 0x4D004D00, 0x4D004D00, 0x4E004E00,
0x4E004E00, 0x4E004E00, 0x4E004E00, 0x4F004F00, 0x4F004F00, 0x4F004F00, 0x4F004F00,
0x50005000, 0x50005000, 0x50005000, 0x50005000, 0x51005100, 0x51005100,
0x51005100, 0x51005100, 0x52005200, 0x52005200, 0x52005200, 0x52005200,
0x53005300, 0x53005300, 0x53005300, 0x53005300, 0x54005400, 0x54005400,
0x54005400, 0x54005400, 0x55005500, 0x55005500, 0x55005500, 0x55005500,
0x56005600, 0x56005600, 0x56005600, 0x56005600, 0x57005700, 0x57005700,
0x57005700, 0x57005700, 0x58005800, 0x58005800, 0x58005800, 0x58005800,
0x59005900, 0x59005900, 0x59005900, 0x59005900, 0x5A005A00, 0x5A005A00,
0x5A005A00, 0x5A005A00, 0x5B005B00, 0x5B005B00, 0x5B005B00, 0x5B005B00,
0x5C005C00, 0x5C005C00, 0x5C005C00, 0x5C005C00, 0x5D005D00, 0x5D005D00,
0x5D005D00, 0x5D005D00, 0x5E005E00, 0x5E005E00, 0x5E005E00, 0x5E005E00,
0x5F005F00, 0x5F005F00, 0x5F005F00, 0x5F005F00, 0x60006000, 0x60006000,
0x60006000, 0x60006000, 0x61006100, 0x61006100, 0x61006100, 0x61006100,
0x62006200, 0x62006200, 0x62006200, 0x62006200, 0x63006300, 0x63006300,
0x63006300, 0x63006300, 0x64006400, 0x64006400, 0x64006400, 0x64006400,
0x65006500, 0x65006500, 0x65006500, 0x65006500, 0x66006600, 0x66006600,
0x66006600, 0x66006600, 0x67006700, 0x67006700, 0x67006700, 0x67006700,
0x68006800, 0x68006800, 0x68006800, 0x68006800, 0x69006900, 0x69006900,
0x69006900, 0x69006900, 0x6A006A00, 0x6A006A00, 0x6A006A00, 0x6A006A00,
0x6B006B00, 0x6B006B00, 0x6B006B00, 0x6B006B00, 0x6C006C00, 0x6C006C00,
0x6C006C00, 0x6C006C00, 0x6D006D00, 0x6D006D00, 0x6D006D00, 0x6D006D00,
0x6E006E00, 0x6E006E00, 0x6E006E00, 0x6E006E00, 0x6F006F00, 0x6F006F00, 0x6F006F00,
0x6F006F00, 0x70007000, 0x70007000, 0x70007000, 0x70007000, 0x71007100,
0x71007100, 0x71007100, 0x71007100, 0x72007200, 0x72007200, 0x72007200,
0x72007200, 0x73007300, 0x73007300, 0x73007300, 0x73007300, 0x74007400,
0x74007400, 0x74007400, 0x74007400, 0x75007500, 0x75007500, 0x75007500,
0x75007500, 0x76007600, 0x76007600, 0x76007600, 0x76007600, 0x77007700,
0x77007700, 0x77007700, 0x77007700, 0x78007800, 0x78007800, 0x78007800,
0x78007800, 0x79007900, 0x79007900, 0x79007900, 0x79007900, 0x7A007A00,
0x7A007A00, 0x7A007A00, 0x7A007A00, 0x7B007B00, 0x7B007B00, 0x7B007B00,
0x7B007B00, 0x7C007C00, 0x7C007C00, 0x7C007C00, 0x7C007C00, 0x7D007D00,
0x7D007D00, 0x7D007D00, 0x7D007D00, 0x7E007E00, 0x7E007E00, 0x7E007E00,
0x7E007E00, 0x7F007F00, 0x7F007F00, 0x7F007F00, 0x7F007F00, 0x80008000,
0x80008000, 0x80008000, 0x80008000, 0x81008100, 0x81008100, 0x81008100,
0x81008100, 0x82008200, 0x82008200, 0x82008200, 0x82008200, 0x83008300,

Gamut_Expansion_Gamma_Correction

0x83008300, 0x83008300, 0x83008300, 0x84008400, 0x84008400, 0x84008400,
0x84008400, 0x85008500, 0x85008500, 0x85008500, 0x85008500, 0x86008600,
0x86008600, 0x86008600, 0x86008600, 0x87008700, 0x87008700, 0x87008700,
0x87008700, 0x88008800, 0x88008800, 0x88008800, 0x88008800, 0x89008900,
0x89008900, 0x89008900, 0x89008900, 0x8A008A00, 0x8A008A00, 0x8A008A00,
0x8A008A00, 0x8B008B00, 0x8B008B00, 0x8B008B00, 0x8B008B00, 0x8C008C00,
0x8C008C00, 0x8C008C00, 0x8C008C00, 0x8D008D00, 0x8D008D00, 0x8D008D00,
0x8D008D00, 0x8E008E00, 0x8E008E00, 0x8E008E00, 0x8E008E00, 0x8F008F00,
0x8F008F00, 0x8F008F00, 0x8F008F00, 0x90009000, 0x90009000, 0x90009000,
0x90009000, 0x91009100, 0x91009100, 0x91009100, 0x91009100, 0x92009200,
0x92009200, 0x92009200, 0x92009200, 0x93009300, 0x93009300, 0x93009300,
0x93009300, 0x94009400, 0x94009400, 0x94009400, 0x94009400, 0x95009500,
0x95009500, 0x95009500, 0x95009500, 0x96009600, 0x96009600, 0x96009600,
0x96009600, 0x97009700, 0x97009700, 0x97009700, 0x97009700, 0x98009800,
0x98009800, 0x98009800, 0x98009800, 0x99009900, 0x99009900, 0x99009900,
0x99009900, 0x9A009A00, 0x9A009A00, 0x9A009A00, 0x9A009A00, 0x9B009B00,
0x9B009B00, 0x9B009B00, 0x9B009B00, 0x9C009C00, 0x9C009C00, 0x9C009C00,
0x9C009C00, 0x9D009D00, 0x9D009D00, 0x9D009D00, 0x9D009D00, 0x9E009E00,
0x9E009E00, 0x9E009E00, 0x9E009E00, 0x9F009F00, 0x9F009F00, 0x9F009F00, 0x9F009F00,
0xA000A000, 0xA000A000, 0xA000A000, 0xA000A000, 0xA100A100, 0xA100A100,
0xA100A100, 0xA100A100, 0xA200A200, 0xA200A200, 0xA200A200, 0xA200A200,
0xA300A300, 0xA300A300, 0xA300A300, 0xA300A300, 0xA400A400, 0xA400A400,
0xA400A400, 0xA400A400, 0xA500A500, 0xA500A500, 0xA500A500, 0xA500A500,
0xA600A600, 0xA600A600, 0xA600A600, 0xA600A600, 0xA700A700, 0xA700A700,
0xA700A700, 0xA700A700, 0xA800A800, 0xA800A800, 0xA800A800, 0xA800A800,
0xA900A900, 0xA900A900, 0xA900A900, 0xA900A900, 0xAA00AA00, 0xAA00AA00,
0xAA00AA00, 0xAA00AA00, 0xAB00AB00, 0xAB00AB00, 0xAB00AB00, 0xAB00AB00,
0xAC00AC00, 0xAC00AC00, 0xAC00AC00, 0xAC00AC00, 0xAD00AD00, 0xAD00AD00,
0xAD00AD00, 0xAD00AD00, 0xAE00AE00, 0xAE00AE00, 0xAE00AE00, 0xAE00AE00,
0xAF00AF00, 0xAF00AF00, 0xAF00AF00, 0xAF00AF00, 0xB000B000, 0xB000B000,
0xB000B000, 0xB000B000, 0xB100B100, 0xB100B100, 0xB100B100, 0xB100B100,
0xB200B200, 0xB200B200, 0xB200B200, 0xB200B200, 0xB300B300, 0xB300B300,
0xB300B300, 0xB300B300, 0xB400B400, 0xB400B400, 0xB400B400, 0xB400B400,
0xB500B500, 0xB500B500, 0xB500B500, 0xB500B500, 0xB600B600, 0xB600B600,
0xB600B600, 0xB600B600, 0xB700B700, 0xB700B700, 0xB700B700, 0xB700B700,
0xB800B800, 0xB800B800, 0xB800B800, 0xB800B800, 0xB900B900, 0xB900B900,
0xB900B900, 0xB900B900, 0xBA00BA00, 0xBA00BA00, 0xBA00BA00, 0xBA00BA00,
0xBB00BB00, 0xBB00BB00, 0xBB00BB00, 0xBB00BB00, 0xBC00BC00, 0xBC00BC00,
0xBC00BC00, 0xBC00BC00, 0xBD00BD00, 0xBD00BD00, 0xBD00BD00, 0xBD00BD00,
0xBE00BE00, 0xBE00BE00, 0xBE00BE00, 0xBE00BE00, 0xBF00BF00, 0xBF00BF00,
0xBF00BF00, 0xBF00BF00, 0xC000C000, 0xC000C000, 0xC000C000, 0xC000C000,
0xC100C100, 0xC100C100, 0xC100C100, 0xC100C100, 0xC200C200, 0xC200C200,
0xC200C200, 0xC200C200, 0xC300C300, 0xC300C300, 0xC300C300, 0xC300C300,
0xC400C400, 0xC400C400, 0xC400C400, 0xC400C400, 0xC500C500, 0xC500C500,
0xC500C500, 0xC500C500, 0xC600C600, 0xC600C600, 0xC600C600, 0xC600C600,
0xC700C700, 0xC700C700, 0xC700C700, 0xC700C700, 0xC800C800, 0xC800C800,

Gamut_Expansion_Gamma_Correction

0xC800C800, 0xC800C800, 0xC900C900, 0xC900C900, 0xC900C900, 0xC900C900,
0xCA00CA00, 0xCA00CA00, 0xCA00CA00, 0xCA00CA00, 0xCB00CB00, 0xCB00CB00,
0xCB00CB00, 0xCB00CB00, 0xCC00CC00, 0xCC00CC00, 0xCC00CC00, 0xCC00CC00,
0xCD00CD00, 0xCD00CD00, 0xCD00CD00, 0xCD00CD00, 0xCE00CE00, 0xCE00CE00,
0xCE00CE00, 0xCE00CE00, 0xCF00CF00, 0xCF00CF00, 0xCF00CF00, 0xCF00CF00,
0xD000D000, 0xD000D000, 0xD000D000, 0xD000D000, 0xD100D100, 0xD100D100,
0xD100D100, 0xD100D100, 0xD200D200, 0xD200D200, 0xD200D200, 0xD200D200,
0xD300D300, 0xD300D300, 0xD300D300, 0xD300D300, 0xD400D400, 0xD400D400,
0xD400D400, 0xD400D400, 0xD500D500, 0xD500D500, 0xD500D500, 0xD500D500,
0xD600D600, 0xD600D600, 0xD600D600, 0xD600D600, 0xD700D700, 0xD700D700,
0xD700D700, 0xD700D700, 0xD800D800, 0xD800D800, 0xD800D800, 0xD800D800,
0xD900D900, 0xD900D900, 0xD900D900, 0xD900D900, 0xDA00DA00, 0xDA00DA00,
0xDA00DA00, 0xDA00DA00, 0xDB00DB00, 0xDB00DB00, 0xDB00DB00, 0xDB00DB00,
0xDC00DC00, 0xDC00DC00, 0xDC00DC00, 0xDC00DC00, 0xDD00DD00, 0xDD00DD00,
0xDD00DD00, 0xDD00DD00, 0xDE00DE00, 0xDE00DE00, 0xDE00DE00, 0xDE00DE00,
0xDF00DF00, 0xDF00DF00, 0xDF00DF00, 0xDF00DF00, 0xE000E000, 0xE000E000,
0xE000E000, 0xE000E000, 0xE100E100, 0xE100E100, 0xE100E100, 0xE100E100,
0xE200E200, 0xE200E200, 0xE200E200, 0xE200E200, 0xE300E300, 0xE300E300,
0xE300E300, 0xE300E300, 0xE400E400, 0xE400E400, 0xE400E400, 0xE400E400,
0xE500E500, 0xE500E500, 0xE500E500, 0xE500E500, 0xE600E600, 0xE600E600,
0xE600E600, 0xE600E600, 0xE700E700, 0xE700E700, 0xE700E700, 0xE700E700,
0xE800E800, 0xE800E800, 0xE800E800, 0xE800E800, 0xE900E900, 0xE900E900,
0xE900E900, 0xE900E900, 0xEA00EA00, 0xEA00EA00, 0xEA00EA00, 0xEA00EA00,
0xEB00EB00, 0xEB00EB00, 0xEB00EB00, 0xEB00EB00, 0xEC00EC00, 0xEC00EC00,
0xEC00EC00, 0xEC00EC00, 0xED00ED00, 0xED00ED00, 0xED00ED00, 0xED00ED00,
0xEE00EE00, 0xEE00EE00, 0xEE00EE00, 0xEE00EE00, 0xEF00EF00, 0xEF00EF00, 0xEF00EF00,
0xEF00EF00, 0xF000F000, 0xF000F000, 0xF000F000, 0xF000F000, 0xF100F100, 0xF100F100,
0xF100F100, 0xF100F100, 0xF200F200, 0xF200F200, 0xF200F200, 0xF200F200, 0xF300F300,
0xF300F300, 0xF300F300, 0xF300F300, 0xF400F400, 0xF400F400, 0xF400F400, 0xF400F400,
0xF500F500, 0xF500F500, 0xF500F500, 0xF500F500, 0xF600F600, 0xF600F600, 0xF600F600,
0xF600F600, 0xF700F700, 0xF700F700, 0xF700F700, 0xF700F700, 0xF800F800, 0xF800F800,
0xF800F800, 0xF800F800, 0xF900F900, 0xF900F900, 0xF900F900, 0xF900F900,
0xFA00FA00, 0xFA00FA00, 0xFA00FA00, 0xFA00FA00, 0xFB00FB00, 0xFB00FB00,
0xFB00FB00, 0xFB00FB00, 0xFC00FC00, 0xFC00FC00, 0xFC00FC00, 0xFC00FC00,
0xFD00FD00, 0xFD00FD00, 0xFD00FD00, 0xFD00FD00, 0xFE00FE00, 0xFE00FE00,
0xFE00FE00, 0xFE00FE00, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF

Programming Notes

The default values follow the pattern suggested by incomplete table below.

DWords	DWord 0	DWord 1	DWord 2	DWord 3
0..3 : Point[0]	00000000h	00000000h	00000000h	00000000h
4..7 : Point[1]	01000100h	01000100h	01000100h	01000100h
8..11 : Point[2]	02000200h	02000200h	02000200h	02000200h
12..15 : Point[3]	03000300h	03000300h	03000300h	03000300h
...				

Gamut_Expansion_Gamma_Correction

1016..1019 : Point[254]	fe00fe00h	fe00fe00h	fe00fe00h	fe00fe00h
1020..1023 : Point[255]	ffffffffh	ffffffffh	ffffffffh	ffffffffh

DWord	Bit	Description
0..1	63:48	**Inverse R-ch Gamma Corrected Value 0** Default Value: 0000h Format: U16
	47:32	**Inverse Pixel Value 0** Default Value: 0000h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 0** Default Value: 0000h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 0** Default Value: 0000h Format: U16
2..3	63:48	**Forward R-ch Gamma Corrected Value 0** Default Value: 0000h Format: U16
	47:32	**Forward Pixel Value 0** Default Value: 0000h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 0** Default Value: 0000h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 0** Default Value: 0000h Format: U16
4..5	63:48	**Inverse R-ch Gamma Corrected Value 1** Default Value: 0100h Format: U16
	47:32	**Inverse Pixel Value 1** Default Value: 0100h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 1** Default Value: 0100h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 1** Default Value: 0100h Format: U16
6..7	63:48	**Forward R-ch Gamma Corrected Value 1** Default Value: 0100h Format: U16
	47:32	**Forward Pixel Value 1** Default Value: 0100h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 1** Default Value: 0100h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 1** Default Value: 0100h Format: U16
8..9	63:48	**Inverse R-ch Gamma Corrected Value 2** Default Value: 0200h Format: U16
	47:32	**Inverse Pixel Value 2** Default Value: 0200h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 2** Default Value: 0200h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 2** Default Value: 0200h Format: U16
10..11	63:48	**Forward R-ch Gamma Corrected Value 2** Default Value: 0200h Format: U16
	47:32	**Forward Pixel Value 2** Default Value: 0200h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 2** Default Value: 0200h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 2** Default Value: 0200h Format: U16
12..13	63:48	**Inverse R-ch Gamma Corrected Value 3** Default Value: 0300h Format: U16
	47:32	**Inverse Pixel Value 3** Default Value: 0300h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 3** Default Value: 0300h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 3** Default Value: 0300h Format: U16
14..15	63:48	**Forward R-ch Gamma Corrected Value 3** Default Value: 0300h Format: U16
	47:32	**Forward Pixel Value 3** Default Value: 0300h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 3** Default Value: 0300h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 3** Default Value: 0300h Format: U16
16..17	63:48	**Inverse R-ch Gamma Corrected Value 4** Default Value: 0400h Format: U16
	47:32	**Inverse Pixel Value 4** Default Value: 0400h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 4** Default Value: 0400h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 4** Default Value: 0400h Format: U16
18..19	63:48	**Forward R-ch Gamma Corrected Value 4** Default Value: 0400h Format: U16
	47:32	**Forward Pixel Value 4** Default Value: 0400h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 4** Default Value: 0400h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 4** Default Value: 0400h Format: U16
20..21	63:48	**Inverse R-ch Gamma Corrected Value 5** Default Value: 0500h Format: U16
	47:32	**Inverse Pixel Value 5** Default Value: 0500h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 5** Default Value: 0500h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 5** Default Value: 0500h Format: U16
22..23	63:48	**Forward R-ch Gamma Corrected Value 5** Default Value: 0500h Format: U16
	47:32	**Forward Pixel Value 5** Default Value: 0500h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 5** Default Value: 0500h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 5** Default Value: 0500h Format: U16
24..25	63:48	**Inverse R-ch Gamma Corrected Value 6** Default Value: 0600h Format: U16
	47:32	**Inverse Pixel Value 6** Default Value: 0600h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 6** Default Value: 0600h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 6** Default Value: 0600h Format: U16
26..27	63:48	**Forward R-ch Gamma Corrected Value 6** Default Value: 0600h Format: U16
	47:32	**Forward Pixel Value 6** Default Value: 0600h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 6** Default Value: 0600h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 6** Default Value: 0600h Format: U16
28..29	63:48	**Inverse R-ch Gamma Corrected Value 7** Default Value: 0700h Format: U16
	47:32	**Inverse Pixel Value 7** Default Value: 0700h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 7** Default Value: 0700h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 7** Default Value: 0700h Format: U16
30..31	63:48	**Forward R-ch Gamma Corrected Value 7** Default Value: 0700h Format: U16
	47:32	**Forward Pixel Value 7** Default Value: 0700h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 7** Default Value: 0700h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 7** Default Value: 0700h Format: U16
32..33	63:48	**Inverse R-ch Gamma Corrected Value 8** Default Value: 0800h Format: U16
	47:32	**Inverse Pixel Value 8** Default Value: 0800h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 8** Default Value: 0800h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 8** Default Value: 0800h Format: U16
34..35	63:48	**Forward R-ch Gamma Corrected Value 8** Default Value: 0800h Format: U16
	47:32	**Forward Pixel Value 8** Default Value: 0800h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 8** Default Value: 0800h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 8** Default Value: 0800h Format: U16
36..37	63:48	**Inverse R-ch Gamma Corrected Value 9** Default Value: 0900h Format: U16
	47:32	**Inverse Pixel Value 9** Default Value: 0900h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 9** Default Value: 0900h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 9** Default Value: 0900h Format: U16
38..39	63:48	**Forward R-ch Gamma Corrected Value 9** Default Value: 0900h Format: U16
	47:32	**Forward Pixel Value 9** Default Value: 0900h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 9** Default Value: 0900h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 9** Default Value: 0900h Format: U16
40..41	63:48	**Inverse R-ch Gamma Corrected Value 10** Default Value: 0a00h Format: U16
	47:32	**Inverse Pixel Value 10** Default Value: 0a00h Format: U16

Gamut_Expansion_Gamma_Correction

	31:16	**Inverse B-ch Gamma Corrected Value 10** Default Value: 0a00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 10** Default Value: 0a00h Format: U16
42..43	63:48	**Forward R-ch Gamma Corrected Value 10** Default Value: 0a00h Format: U16
	47:32	**Forward Pixel Value 10** Default Value: 0a00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 10** Default Value: 0a00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 10** Default Value: 0a00h Format: U16
44..45	63:48	**Inverse R-ch Gamma Corrected Value 11** Default Value: 0b00h Format: U16
	47:32	**Inverse Pixel Value 11** Default Value: 0b00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 11** Default Value: 0b00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 11** Default Value: 0b00h Format: U16
46..47	63:48	**Forward R-ch Gamma Corrected Value 11** Default Value: 0b00h Format: U16
	47:32	**Forward Pixel Value 11** Default Value: 0b00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 11** Default Value: 0b00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 11** Default Value: 0b00h Format: U16
48..49	63:48	**Inverse R-ch Gamma Corrected Value 12** Default Value: 0c00h Format: U16
	47:32	**Inverse Pixel Value 12** Default Value: 0c00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 12** Default Value: 0c00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 12** Default Value: 0c00h Format: U16
50..51	63:48	**Forward R-ch Gamma Corrected Value 12** Default Value: 0c00h Format: U16
	47:32	**Forward Pixel Value 12** Default Value: 0c00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 12** Default Value: 0c00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 12** Default Value: 0c00h Format: U16
52..53	63:48	**Inverse R-ch Gamma Corrected Value 13** Default Value: 0d00h Format: U16
	47:32	**Inverse Pixel Value 13** Default Value: 0d00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 13** Default Value: 0d00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 13** Default Value: 0d00h Format: U16
54..55	63:48	**Forward R-ch Gamma Corrected Value 13** Default Value: 0d00h Format: U16
	47:32	**Forward Pixel Value 13** Default Value: 0d00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 13** Default Value: 0d00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 13** Default Value: 0d00h Format: U16
56..57	63:48	**Inverse R-ch Gamma Corrected Value 14** Default Value: 0e00h Format: U16
	47:32	**Inverse Pixel Value 14** Default Value: 0e00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 14** Default Value: 0e00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 14** Default Value: 0e00h Format: U16
58..59	63:48	**Forward R-ch Gamma Corrected Value 14** Default Value: 0e00h Format: U16
	47:32	**Forward Pixel Value 14** Default Value: 0e00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 14** Default Value: 0e00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 14** Default Value: 0e00h Format: U16
60..61	63:48	**Inverse R-ch Gamma Corrected Value 15** Default Value: 0f00h Format: U16
	47:32	**Inverse Pixel Value 15** Default Value: 0f00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 15** Default Value: 0f00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 15** Default Value: 0f00h Format: U16
62..63	63:48	**Forward R-ch Gamma Corrected Value 15** Default Value: 0f00h Format: U16
	47:32	**Forward Pixel Value 15** Default Value: 0f00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 15** Default Value: 0f00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 15** Default Value: 0f00h Format: U16
64..65	63:48	**Inverse R-ch Gamma Corrected Value 16** Default Value: 1000h Format: U16
	47:32	**Inverse Pixel Value 16** Default Value: 1000h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 16** Default Value: 1000h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 16** Default Value: 1000h Format: U16
66..67	63:48	**Forward R-ch Gamma Corrected Value 16** Default Value: 1000h Format: U16
	47:32	**Forward Pixel Value 16** Default Value: 1000h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 16** Default Value: 1000h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 16** Default Value: 1000h Format: U16
68..69	63:48	**Inverse R-ch Gamma Corrected Value 17** Default Value: 1100h Format: U16
	47:32	**Inverse Pixel Value 17** Default Value: 1100h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 17** Default Value: 1100h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 17** Default Value: 1100h Format: U16
70..71	63:48	**Forward R-ch Gamma Corrected Value 17** Default Value: 1100h Format: U16
	47:32	**Forward Pixel Value 17** Default Value: 1100h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 17** Default Value: 1100h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 17** Default Value: 1100h Format: U16
72..73	63:48	**Inverse R-ch Gamma Corrected Value 18** Default Value: 1200h Format: U16
	47:32	**Inverse Pixel Value 18** Default Value: 1200h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 18** Default Value: 1200h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 18** Default Value: 1200h Format: U16
74..75	63:48	**Forward R-ch Gamma Corrected Value 18** Default Value: 1200h Format: U16
	47:32	**Forward Pixel Value 18** Default Value: 1200h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 18** Default Value: 1200h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 18** Default Value: 1200h Format: U16
76..77	63:48	**Inverse R-ch Gamma Corrected Value 19** Default Value: 1300h Format: U16
	47:32	**Inverse Pixel Value 19** Default Value: 1300h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 19** Default Value: 1300h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 19** Default Value: 1300h Format: U16
78..79	63:48	**Forward R-ch Gamma Corrected Value 19** Default Value: 1300h Format: U16
	47:32	**Forward Pixel Value 19** Default Value: 1300h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 19** Default Value: 1300h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 19** Default Value: 1300h Format: U16
80..81	63:48	**Inverse R-ch Gamma Corrected Value 20** Default Value: 1400h Format: U16
	47:32	**Inverse Pixel Value 20** Default Value: 1400h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 20** Default Value: 1400h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 20** Default Value: 1400h Format: U16
82..83	63:48	**Forward R-ch Gamma Corrected Value 20** Default Value: 1400h Format: U16
	47:32	**Forward Pixel Value 20** Default Value: 1400h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 20** Default Value: 1400h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 20** Default Value: 1400h Format: U16
84..85	63:48	**Inverse R-ch Gamma Corrected Value 21** Default Value: 1500h Format: U16
	47:32	**Inverse Pixel Value 21** Default Value: 1500h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 21** Default Value: 1500h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 21** Default Value: 1500h Format: U16
86..87	63:48	**Forward R-ch Gamma Corrected Value 21** Default Value: 1500h Format: U16
	47:32	**Forward Pixel Value 21** Default Value: 1500h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 21** Default Value: 1500h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 21** Default Value: 1500h Format: U16
88..89	63:48	**Inverse R-ch Gamma Corrected Value 22** Default Value: 1600h Format: U16
	47:32	**Inverse Pixel Value 22** Default Value: 1600h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 22** Default Value: 1600h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 22** Default Value: 1600h Format: U16
90..91	63:48	**Forward R-ch Gamma Corrected Value 22** Default Value: 1600h Format: U16
	47:32	**Forward Pixel Value 22** Default Value: 1600h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 22** Default Value: 1600h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 22** Default Value: 1600h Format: U16
92..93	63:48	**Inverse R-ch Gamma Corrected Value 23** Default Value: 1700h Format: U16
	47:32	**Inverse Pixel Value 23** Default Value: 1700h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 23** Default Value: 1700h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 23** Default Value: 1700h Format: U16
94..95	63:48	**Forward R-ch Gamma Corrected Value 23** Default Value: 1700h Format: U16
	47:32	**Forward Pixel Value 23** Default Value: 1700h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 23** Default Value: 1700h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 23** Default Value: 1700h Format: U16
96..97	63:48	**Inverse R-ch Gamma Corrected Value 24** Default Value: 1800h Format: U16
	47:32	**Inverse Pixel Value 24** Default Value: 1800h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 24** Default Value: 1800h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 24** Default Value: 1800h Format: U16
98..99	63:48	**Forward R-ch Gamma Corrected Value 24** Default Value: 1800h Format: U16
	47:32	**Forward Pixel Value 24** Default Value: 1800h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 24** Default Value: 1800h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 24** Default Value: 1800h Format: U16
100..101	63:48	**Inverse R-ch Gamma Corrected Value 25** Default Value: 1900h Format: U16
	47:32	**Inverse Pixel Value 25** Default Value: 1900h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 25** Default Value: 1900h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 25** Default Value: 1900h Format: U16
102..103	63:48	**Forward R-ch Gamma Corrected Value 25** Default Value: 1900h Format: U16
	47:32	**Forward Pixel Value 25** Default Value: 1900h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 25** Default Value: 1900h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 25** Default Value: 1900h Format: U16
104..105	63:48	**Inverse R-ch Gamma Corrected Value 26** Default Value: 1a00h Format: U16
	47:32	**Inverse Pixel Value 26** Default Value: 1a00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 26** Default Value: 1a00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 26** Default Value: 1a00h Format: U16
106..107	63:48	**Forward R-ch Gamma Corrected Value 26** Default Value: 1a00h Format: U16
	47:32	**Forward Pixel Value 26** Default Value: 1a00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 26** Default Value: 1a00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 26** Default Value: 1a00h Format: U16
108..109	63:48	**Inverse R-ch Gamma Corrected Value 27** Default Value: 1b00h Format: U16
	47:32	**Inverse Pixel Value 27** Default Value: 1b00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 27** Default Value: 1b00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 27** Default Value: 1b00h Format: U16
110..111	63:48	**Forward R-ch Gamma Corrected Value 27** Default Value: 1b00h Format: U16
	47:32	**Forward Pixel Value 27** Default Value: 1b00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 27** Default Value: 1b00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 27** Default Value: 1b00h Format: U16
112..113	63:48	**Inverse R-ch Gamma Corrected Value 28** Default Value: 1c00h Format: U16
	47:32	**Inverse Pixel Value 28** Default Value: 1c00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 28** Default Value: 1c00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 28** Default Value: 1c00h Format: U16
114..115	63:48	**Forward R-ch Gamma Corrected Value 28** Default Value: 1c00h Format: U16
	47:32	**Forward Pixel Value 28** Default Value: 1c00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 28** Default Value: 1c00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 28** Default Value: 1c00h Format: U16
116..117	63:48	**Inverse R-ch Gamma Corrected Value 29** Default Value: 1d00h Format: U16
	47:32	**Inverse Pixel Value 29** Default Value: 1d00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 29** Default Value: 1d00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 29** Default Value: 1d00h Format: U16
118..119	63:48	**Forward R-ch Gamma Corrected Value 29** Default Value: 1d00h Format: U16
	47:32	**Forward Pixel Value 29** Default Value: 1d00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 29** Default Value: 1d00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 29** Default Value: 1d00h Format: U16
120..121	63:48	**Inverse R-ch Gamma Corrected Value 30** Default Value: 1e00h Format: U16
	47:32	**Inverse Pixel Value 30** Default Value: 1e00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 30** Default Value: 1e00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 30** Default Value: 1e00h Format: U16
122..123	63:48	**Forward R-ch Gamma Corrected Value 30** Default Value: 1e00h Format: U16
	47:32	**Forward Pixel Value 30** Default Value: 1e00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 30** Default Value: 1e00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 30** Default Value: 1e00h Format: U16
124..125	63:48	**Inverse R-ch Gamma Corrected Value 31** Default Value: 1f00h Format: U16
	47:32	**Inverse Pixel Value 31** Default Value: 1f00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 31** Default Value: 1f00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 31** Default Value: 1f00h Format: U16
126..127	63:48	**Forward R-ch Gamma Corrected Value 31** Default Value: 1f00h Format: U16
	47:32	**Forward Pixel Value 31** Default Value: 1f00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 31** Default Value: 1f00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 31** Default Value: 1f00h Format: U16
128..129	63:48	**Inverse R-ch Gamma Corrected Value 32** Default Value: 2000h Format: U16
	47:32	**Inverse Pixel Value 32** Default Value: 2000h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 32** Default Value: 2000h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 32** Default Value: 2000h Format: U16
130..131	63:48	**Forward R-ch Gamma Corrected Value 32** Default Value: 2000h Format: U16
	47:32	**Forward Pixel Value 32** Default Value: 2000h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 32** Default Value: 2000h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 32** Default Value: 2000h Format: U16
132..133	63:48	**Inverse R-ch Gamma Corrected Value 33** Default Value: 2100h Format: U16
	47:32	**Inverse Pixel Value 33** Default Value: 2100h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 33** Default Value: 2100h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 33** Default Value: 2100h Format: U16
134..135	63:48	**Forward R-ch Gamma Corrected Value 33** Default Value: 2100h Format: U16
	47:32	**Forward Pixel Value 33** Default Value: 2100h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 33** Default Value: 2100h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 33** Default Value: 2100h Format: U16
136..137	63:48	**Inverse R-ch Gamma Corrected Value 34** Default Value: 2200h Format: U16
	47:32	**Inverse Pixel Value 34** Default Value: 2200h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 34** Default Value: 2200h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 34** Default Value: 2200h Format: U16
138..139	63:48	**Forward R-ch Gamma Corrected Value 34** Default Value: 2200h Format: U16
	47:32	**Forward Pixel Value 34** Default Value: 2200h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 34** Default Value: 2200h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 34** Default Value: 2200h Format: U16
140..141	63:48	**Inverse R-ch Gamma Corrected Value 35** Default Value: 2300h Format: U16
	47:32	**Inverse Pixel Value 35** Default Value: 2300h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 35** Default Value: 2300h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 35** Default Value: 2300h Format: U16
142..143	63:48	**Forward R-ch Gamma Corrected Value 35** Default Value: 2300h Format: U16
	47:32	**Forward Pixel Value 35** Default Value: 2300h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 35** Default Value: 2300h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 35** Default Value: 2300h Format: U16
144..145	63:48	**Inverse R-ch Gamma Corrected Value 36** Default Value: 2400h Format: U16
	47:32	**Inverse Pixel Value 36** Default Value: 2400h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 36** Default Value: 2400h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 36** Default Value: 2400h Format: U16
146..147	63:48	**Forward R-ch Gamma Corrected Value 36** Default Value: 2400h Format: U16
	47:32	**Forward Pixel Value 36** Default Value: 2400h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 36** Default Value: 2400h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 36** Default Value: 2400h Format: U16
148..149	63:48	**Inverse R-ch Gamma Corrected Value 37** Default Value: 2500h Format: U16
	47:32	**Inverse Pixel Value 37** Default Value: 2500h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 37** Default Value: 2500h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 37** Default Value: 2500h Format: U16
150..151	63:48	**Forward R-ch Gamma Corrected Value 37** Default Value: 2500h Format: U16
	47:32	**Forward Pixel Value 37** Default Value: 2500h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 37** Default Value: 2500h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 37** Default Value: 2500h Format: U16
152..153	63:48	**Inverse R-ch Gamma Corrected Value 38** Default Value: 2600h Format: U16
	47:32	**Inverse Pixel Value 38** Default Value: 2600h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 38** Default Value: 2600h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 38** Default Value: 2600h Format: U16
154..155	63:48	**Forward R-ch Gamma Corrected Value 38** Default Value: 2600h Format: U16
	47:32	**Forward Pixel Value 38** Default Value: 2600h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 38** Default Value: 2600h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 38** Default Value: 2600h Format: U16
156..157	63:48	**Inverse R-ch Gamma Corrected Value 39** Default Value: 2700h Format: U16
	47:32	**Inverse Pixel Value 39** Default Value: 2700h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 39** Default Value: 2700h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 39** Default Value: 2700h Format: U16
158..159	63:48	**Forward R-ch Gamma Corrected Value 39** Default Value: 2700h Format: U16
	47:32	**Forward Pixel Value 39** Default Value: 2700h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 39** Default Value: 2700h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 39** Default Value: 2700h Format: U16
160..161	63:48	**Inverse R-ch Gamma Corrected Value 40** Default Value: 2800h Format: U16
	47:32	**Inverse Pixel Value 40** Default Value: 2800h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 40** Default Value: 2800h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 40** Default Value: 2800h Format: U16
162..163	63:48	**Forward R-ch Gamma Corrected Value 40** Default Value: 2800h Format: U16
	47:32	**Forward Pixel Value 40** Default Value: 2800h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 40** Default Value: 2800h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 40** Default Value: 2800h Format: U16
164..165	63:48	**Inverse R-ch Gamma Corrected Value 41** Default Value: 2900h Format: U16
	47:32	**Inverse Pixel Value 41** Default Value: 2900h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 41** Default Value: 2900h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 41** Default Value: 2900h Format: U16
166..167	63:48	**Forward R-ch Gamma Corrected Value 41** Default Value: 2900h Format: U16
	47:32	**Forward Pixel Value 41** Default Value: 2900h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 41** Default Value: 2900h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 41** Default Value: 2900h Format: U16
168..169	63:48	**Inverse R-ch Gamma Corrected Value 42** Default Value: 2a00h Format: U16
	47:32	**Inverse Pixel Value 42** Default Value: 2a00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 42** Default Value: 2a00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 42** Default Value: 2a00h Format: U16
170..171	63:48	**Forward R-ch Gamma Corrected Value 42** Default Value: 2a00h Format: U16
	47:32	**Forward Pixel Value 42** Default Value: 2a00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 42** Default Value: 2a00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 42** Default Value: 2a00h Format: U16
172..173	63:48	**Inverse R-ch Gamma Corrected Value 43** Default Value: 2b00h Format: U16
	47:32	**Inverse Pixel Value 43** Default Value: 2b00h Format: U16

Gamut_Expansion_Gamma_Correction

	31:16	**Inverse B-ch Gamma Corrected Value 43** Default Value: 2b00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 43** Default Value: 2b00h Format: U16
174..175	63:48	**Forward R-ch Gamma Corrected Value 43** Default Value: 2b00h Format: U16
	47:32	**Forward Pixel Value 43** Default Value: 2b00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 43** Default Value: 2b00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 43** Default Value: 2b00h Format: U16
176..177	63:48	**Inverse R-ch Gamma Corrected Value 44** Default Value: 2c00h Format: U16
	47:32	**Inverse Pixel Value 44** Default Value: 2c00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 44** Default Value: 2c00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 44** Default Value: 2c00h Format: U16
178..179	63:48	**Forward R-ch Gamma Corrected Value 44** Default Value: 2c00h Format: U16
	47:32	**Forward Pixel Value 44** Default Value: 2c00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 44** Default Value: 2c00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 44** Default Value: 2c00h Format: U16
180..181	63:48	**Inverse R-ch Gamma Corrected Value 45** Default Value: 2d00h Format: U16
	47:32	**Inverse Pixel Value 45** Default Value: 2d00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 45** Default Value: 2d00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 45** Default Value: 2d00h Format: U16
182..183	63:48	**Forward R-ch Gamma Corrected Value 45** Default Value: 2d00h Format: U16
	47:32	**Forward Pixel Value 45** Default Value: 2d00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 45** Default Value: 2d00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 45** Default Value: 2d00h Format: U16
184..185	63:48	**Inverse R-ch Gamma Corrected Value 46** Default Value: 2e00h Format: U16
	47:32	**Inverse Pixel Value 46** Default Value: 2e00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 46** Default Value: 2e00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 46** Default Value: 2e00h Format: U16
186..187	63:48	**Forward R-ch Gamma Corrected Value 46** Default Value: 2e00h Format: U16
	47:32	**Forward Pixel Value 46** Default Value: 2e00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 46** Default Value: 2e00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 46** Default Value: 2e00h Format: U16
188..189	63:48	**Inverse R-ch Gamma Corrected Value 47** Default Value: 2f00h Format: U16
	47:32	**Inverse Pixel Value 47** Default Value: 2f00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 47** Default Value: 2f00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 47** Default Value: 2f00h Format: U16
190..191	63:48	**Forward R-ch Gamma Corrected Value 47** Default Value: 2f00h Format: U16
	47:32	**Forward Pixel Value 47** Default Value: 2f00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 47** Default Value: 2f00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 47** Default Value: 2f00h Format: U16
192..193	63:48	**Inverse R-ch Gamma Corrected Value 48** Default Value: 3000h Format: U16
	47:32	**Inverse Pixel Value 48** Default Value: 3000h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 48** Default Value: 3000h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 48** Default Value: 3000h Format: U16
194..195	63:48	**Forward R-ch Gamma Corrected Value 48** Default Value: 3000h Format: U16
	47:32	**Forward Pixel Value 48** Default Value: 3000h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 48** Default Value: 3000h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 48** Default Value: 3000h Format: U16
196..197	63:48	**Inverse R-ch Gamma Corrected Value 49** Default Value: 3100h Format: U16
	47:32	**Inverse Pixel Value 49** Default Value: 3100h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 49** Default Value: 3100h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 49** Default Value: 3100h Format: U16
198..199	63:48	**Forward R-ch Gamma Corrected Value 49** Default Value: 3100h Format: U16
	47:32	**Forward Pixel Value 49** Default Value: 3100h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 49** Default Value: 3100h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 49** Default Value: 3100h Format: U16
200..201	63:48	**Inverse R-ch Gamma Corrected Value 50** Default Value: 3200h Format: U16
	47:32	**Inverse Pixel Value 50** Default Value: 3200h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 50** Default Value: 3200h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 50** Default Value: 3200h Format: U16
202..203	63:48	**Forward R-ch Gamma Corrected Value 50** Default Value: 3200h Format: U16
	47:32	**Forward Pixel Value 50** Default Value: 3200h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 50** Default Value: 3200h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 50** Default Value: 3200h Format: U16
204..205	63:48	**Inverse R-ch Gamma Corrected Value 51** Default Value: 3300h Format: U16
	47:32	**Inverse Pixel Value 51** Default Value: 3300h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 51** Default Value: 3300h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 51** Default Value: 3300h Format: U16
206..207	63:48	**Forward R-ch Gamma Corrected Value 51** Default Value: 3300h Format: U16
	47:32	**Forward Pixel Value 51** Default Value: 3300h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 51** Default Value: 3300h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 51** Default Value: 3300h Format: U16
208..209	63:48	**Inverse R-ch Gamma Corrected Value 52** Default Value: 3400h Format: U16
	47:32	**Inverse Pixel Value 52** Default Value: 3400h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 52** Default Value: 3400h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 52** Default Value: 3400h Format: U16
210..211	63:48	**Forward R-ch Gamma Corrected Value 52** Default Value: 3400h Format: U16
	47:32	**Forward Pixel Value 52** Default Value: 3400h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 52** Default Value: 3400h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 52** Default Value: 3400h Format: U16
212..213	63:48	**Inverse R-ch Gamma Corrected Value 53** Default Value: 3500h Format: U16
	47:32	**Inverse Pixel Value 53** Default Value: 3500h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 53** Default Value: 3500h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 53** Default Value: 3500h Format: U16
214..215	63:48	**Forward R-ch Gamma Corrected Value 53** Default Value: 3500h Format: U16
	47:32	**Forward Pixel Value 53** Default Value: 3500h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 53** Default Value: 3500h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 53** Default Value: 3500h Format: U16
216..217	63:48	**Inverse R-ch Gamma Corrected Value 54** Default Value: 3600h Format: U16
	47:32	**Inverse Pixel Value 54** Default Value: 3600h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 54** Default Value: 3600h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 54** Default Value: 3600h Format: U16
218..219	63:48	**Forward R-ch Gamma Corrected Value 54** Default Value: 3600h Format: U16
	47:32	**Forward Pixel Value 54** Default Value: 3600h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 54** Default Value: 3600h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 54** Default Value: 3600h Format: U16
220..221	63:48	**Inverse R-ch Gamma Corrected Value 55** Default Value: 3700h Format: U16
	47:32	**Inverse Pixel Value 55** Default Value: 3700h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 55** Default Value: 3700h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 55** Default Value: 3700h Format: U16
222..223	63:48	**Forward R-ch Gamma Corrected Value 55** Default Value: 3700h Format: U16
	47:32	**Forward Pixel Value 55** Default Value: 3700h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 55** Default Value: 3700h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 55** Default Value: 3700h Format: U16
224..225	63:48	**Inverse R-ch Gamma Corrected Value 56** Default Value: 3800h Format: U16
	47:32	**Inverse Pixel Value 56** Default Value: 3800h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 56** Default Value: 3800h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 56** Default Value: 3800h Format: U16
226..227	63:48	**Forward R-ch Gamma Corrected Value 56** Default Value: 3800h Format: U16
	47:32	**Forward Pixel Value 56** Default Value: 3800h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 56** Default Value: 3800h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 56** Default Value: 3800h Format: U16
228..229	63:48	**Inverse R-ch Gamma Corrected Value 57** Default Value: 3900h Format: U16
	47:32	**Inverse Pixel Value 57** Default Value: 3900h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 57** Default Value: 3900h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 57** Default Value: 3900h Format: U16
230..231	63:48	**Forward R-ch Gamma Corrected Value 57** Default Value: 3900h Format: U16
	47:32	**Forward Pixel Value 57** Default Value: 3900h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 57** Default Value: 3900h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 57** Default Value: 3900h Format: U16
232..233	63:48	**Inverse R-ch Gamma Corrected Value 58** Default Value: 3a00h Format: U16
	47:32	**Inverse Pixel Value 58** Default Value: 3a00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 58** Default Value: 3a00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 58** Default Value: 3a00h Format: U16
234..235	63:48	**Forward R-ch Gamma Corrected Value 58** Default Value: 3a00h Format: U16
	47:32	**Forward Pixel Value 58** Default Value: 3a00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 58** Default Value: 3a00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 58** Default Value: 3a00h Format: U16
236..237	63:48	**Inverse R-ch Gamma Corrected Value 59** Default Value: 3b00h Format: U16
	47:32	**Inverse Pixel Value 59** Default Value: 3b00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 59** Default Value: 3b00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 59** Default Value: 3b00h Format: U16
238..239	63:48	**Forward R-ch Gamma Corrected Value 59** Default Value: 3b00h Format: U16
	47:32	**Forward Pixel Value 59** Default Value: 3b00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 59** Default Value: 3b00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 59** Default Value: 3b00h Format: U16
240..241	63:48	**Inverse R-ch Gamma Corrected Value 60** Default Value: 3c00h Format: U16
	47:32	**Inverse Pixel Value 60** Default Value: 3c00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 60** Default Value: 3c00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 60** Default Value: 3c00h Format: U16
242..243	63:48	**Forward R-ch Gamma Corrected Value 60** Default Value: 3c00h Format: U16
	47:32	**Forward Pixel Value 60** Default Value: 3c00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 60** Default Value: 3c00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 60** Default Value: 3c00h Format: U16
244..245	63:48	**Inverse R-ch Gamma Corrected Value 61** Default Value: 3d00h Format: U16
	47:32	**Inverse Pixel Value 61** Default Value: 3d00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 61** Default Value: 3d00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 61** Default Value: 3d00h Format: U16
246..247	63:48	**Forward R-ch Gamma Corrected Value 61** Default Value: 3d00h Format: U16
	47:32	**Forward Pixel Value 61** Default Value: 3d00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 61** Default Value: 3d00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 61** Default Value: 3d00h Format: U16
248..249	63:48	**Inverse R-ch Gamma Corrected Value 62** Default Value: 3e00h Format: U16
	47:32	**Inverse Pixel Value 62** Default Value: 3e00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 62** Default Value: 3e00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 62** Default Value: 3e00h Format: U16
250..251	63:48	**Forward R-ch Gamma Corrected Value 62** Default Value: 3e00h Format: U16
	47:32	**Forward Pixel Value 62** Default Value: 3e00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 62** Default Value: 3e00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 62** Default Value: 3e00h Format: U16
252..253	63:48	**Inverse R-ch Gamma Corrected Value 63** Default Value: 3f00h Format: U16
	47:32	**Inverse Pixel Value 63** Default Value: 3f00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 63** Default Value: 3f00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 63** Default Value: 3f00h Format: U16
254..255	63:48	**Forward R-ch Gamma Corrected Value 63** Default Value: 3f00h Format: U16
	47:32	**Forward Pixel Value 63** Default Value: 3f00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 63** Default Value: 3f00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 63** Default Value: 3f00h Format: U16
256..257	63:48	**Inverse R-ch Gamma Corrected Value 64** Default Value: 4000h Format: U16
	47:32	**Inverse Pixel Value 64** Default Value: 4000h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 64** Default Value: 4000h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 64** Default Value: 4000h Format: U16
258..259	63:48	**Forward R-ch Gamma Corrected Value 64** Default Value: 4000h Format: U16
	47:32	**Forward Pixel Value 64** Default Value: 4000h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 64** Default Value: 4000h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 64** Default Value: 4000h Format: U16
260..261	63:48	**Inverse R-ch Gamma Corrected Value 65** Default Value: 4100h Format: U16
	47:32	**Inverse Pixel Value 65** Default Value: 4100h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 65** Default Value: 4100h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 65** Default Value: 4100h Format: U16
262..263	63:48	**Forward R-ch Gamma Corrected Value 65** Default Value: 4100h Format: U16
	47:32	**Forward Pixel Value 65** Default Value: 4100h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 65** Default Value: 4100h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 65** Default Value: 4100h Format: U16
264..265	63:48	**Inverse R-ch Gamma Corrected Value 66** Default Value: 4200h Format: U16
	47:32	**Inverse Pixel Value 66** Default Value: 4200h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 66** Default Value: 4200h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 66** Default Value: 4200h Format: U16
266..267	63:48	**Forward R-ch Gamma Corrected Value 66** Default Value: 4200h Format: U16
	47:32	**Forward Pixel Value 66** Default Value: 4200h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 66** Default Value: 4200h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 66** Default Value: 4200h Format: U16
268..269	63:48	**Inverse R-ch Gamma Corrected Value 67** Default Value: 4300h Format: U16
	47:32	**Inverse Pixel Value 67** Default Value: 4300h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 67** Default Value: 4300h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 67** Default Value: 4300h Format: U16
270..271	63:48	**Forward R-ch Gamma Corrected Value 67** Default Value: 4300h Format: U16
	47:32	**Forward Pixel Value 67** Default Value: 4300h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 67** Default Value: 4300h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 67** Default Value: 4300h Format: U16
272..273	63:48	**Inverse R-ch Gamma Corrected Value 68** Default Value: 4400h Format: U16
	47:32	**Inverse Pixel Value 68** Default Value: 4400h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 68** Default Value: 4400h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 68** Default Value: 4400h Format: U16
274..275	63:48	**Forward R-ch Gamma Corrected Value 68** Default Value: 4400h Format: U16
	47:32	**Forward Pixel Value 68** Default Value: 4400h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 68** Default Value: 4400h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 68** Default Value: 4400h Format: U16
276..277	63:48	**Inverse R-ch Gamma Corrected Value 69** Default Value: 4500h Format: U16
	47:32	**Inverse Pixel Value 69** Default Value: 4500h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 69** Default Value: 4500h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 69** Default Value: 4500h Format: U16
278..279	63:48	**Forward R-ch Gamma Corrected Value 69** Default Value: 4500h Format: U16
	47:32	**Forward Pixel Value 69** Default Value: 4500h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 69** Default Value: 4500h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 69** Default Value: 4500h Format: U16
280..281	63:48	**Inverse R-ch Gamma Corrected Value 70** Default Value: 4600h Format: U16
	47:32	**Inverse Pixel Value 70** Default Value: 4600h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 70** Default Value: 4600h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 70** Default Value: 4600h Format: U16
282..283	63:48	**Forward R-ch Gamma Corrected Value 70** Default Value: 4600h Format: U16
	47:32	**Forward Pixel Value 70** Default Value: 4600h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 70** Default Value: 4600h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 70** Default Value: 4600h Format: U16
284..285	63:48	**Inverse R-ch Gamma Corrected Value 71** Default Value: 4700h Format: U16
	47:32	**Inverse Pixel Value 71** Default Value: 4700h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 71** Default Value: 4700h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 71** Default Value: 4700h Format: U16
286..287	63:48	**Forward R-ch Gamma Corrected Value 71** Default Value: 4700h Format: U16
	47:32	**Forward Pixel Value 71** Default Value: 4700h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 71** Default Value: 4700h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 71** Default Value: 4700h Format: U16
288..289	63:48	**Inverse R-ch Gamma Corrected Value 72** Default Value: 4800h Format: U16
	47:32	**Inverse Pixel Value 72** Default Value: 4800h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 72** Default Value: 4800h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 72** Default Value: 4800h Format: U16
290..291	63:48	**Forward R-ch Gamma Corrected Value 72** Default Value: 4800h Format: U16
	47:32	**Forward Pixel Value 72** Default Value: 4800h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 72** Default Value: 4800h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 72** Default Value: 4800h Format: U16
292..293	63:48	**Inverse R-ch Gamma Corrected Value 73** Default Value: 4900h Format: U16
	47:32	**Inverse Pixel Value 73** Default Value: 4900h Format: U16

Gamut_Expansion_Gamma_Correction

	31:16	**Inverse B-ch Gamma Corrected Value 73** Default Value: 4900h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 73** Default Value: 4900h Format: U16
294..295	63:48	**Forward R-ch Gamma Corrected Value 73** Default Value: 4900h Format: U16
	47:32	**Forward Pixel Value 73** Default Value: 4900h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 73** Default Value: 4900h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 73** Default Value: 4900h Format: U16
296..297	63:48	**Inverse R-ch Gamma Corrected Value 74** Default Value: 4a00h Format: U16
	47:32	**Inverse Pixel Value 74** Default Value: 4a00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 74** Default Value: 4a00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 74** Default Value: 4a00h Format: U16
298..299	63:48	**Forward R-ch Gamma Corrected Value 74** Default Value: 4a00h Format: U16
	47:32	**Forward Pixel Value 74** Default Value: 4a00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 74** Default Value: 4a00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 74** Default Value: 4a00h Format: U16
300..301	63:48	**Inverse R-ch Gamma Corrected Value 75** Default Value: 4b00h Format: U16
	47:32	**Inverse Pixel Value 75** Default Value: 4b00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 75** Default Value: 4b00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 75** Default Value: 4b00h Format: U16
302..303	63:48	**Forward R-ch Gamma Corrected Value 75** Default Value: 4b00h Format: U16
	47:32	**Forward Pixel Value 75** Default Value: 4b00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 75** Default Value: 4b00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 75** Default Value: 4b00h Format: U16
304..305	63:48	**Inverse R-ch Gamma Corrected Value 76** Default Value: 4c00h Format: U16
	47:32	**Inverse Pixel Value 76** Default Value: 4c00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 76** Default Value: 4c00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 76** Default Value: 4c00h Format: U16
306..307	63:48	**Forward R-ch Gamma Corrected Value 76** Default Value: 4c00h Format: U16
	47:32	**Forward Pixel Value 76** Default Value: 4c00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 76** Default Value: 4c00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 76** Default Value: 4c00h Format: U16
308..309	63:48	**Inverse R-ch Gamma Corrected Value 77** Default Value: 4d00h Format: U16
	47:32	**Inverse Pixel Value 77** Default Value: 4d00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 77** Default Value: 4d00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 77** Default Value: 4d00h Format: U16
310..311	63:48	**Forward R-ch Gamma Corrected Value 77** Default Value: 4d00h Format: U16
	47:32	**Forward Pixel Value 77** Default Value: 4d00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 77** Default Value: 4d00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 77** Default Value: 4d00h Format: U16
312..313	63:48	**Inverse R-ch Gamma Corrected Value 78** Default Value: 4e00h Format: U16
	47:32	**Inverse Pixel Value 78** Default Value: 4e00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 78** Default Value: 4e00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 78** Default Value: 4e00h Format: U16
314..315	63:48	**Forward R-ch Gamma Corrected Value 78** Default Value: 4e00h Format: U16
	47:32	**Forward Pixel Value 78** Default Value: 4e00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 78** Default Value: 4e00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 78** Default Value: 4e00h Format: U16
316..317	63:48	**Inverse R-ch Gamma Corrected Value 79** Default Value: 4f00h Format: U16
	47:32	**Inverse Pixel Value 79** Default Value: 4f00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 79** Default Value: 4f00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 79** Default Value: 4f00h Format: U16
318..319	63:48	**Forward R-ch Gamma Corrected Value 79** Default Value: 4f00h Format: U16
	47:32	**Forward Pixel Value 79** Default Value: 4f00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 79** Default Value: 4f00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 79** Default Value: 4f00h Format: U16
320..321	63:48	**Inverse R-ch Gamma Corrected Value 80** Default Value: 5000h Format: U16
	47:32	**Inverse Pixel Value 80** Default Value: 5000h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 80** Default Value: 5000h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 80** Default Value: 5000h Format: U16
322..323	63:48	**Forward R-ch Gamma Corrected Value 80** Default Value: 5000h Format: U16
	47:32	**Forward Pixel Value 80** Default Value: 5000h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 80** Default Value: 5000h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 80** Default Value: 5000h Format: U16
324..325	63:48	**Inverse R-ch Gamma Corrected Value 81** Default Value: 5100h Format: U16
	47:32	**Inverse Pixel Value 81** Default Value: 5100h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 81** Default Value: 5100h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 81** Default Value: 5100h Format: U16
326..327	63:48	**Forward R-ch Gamma Corrected Value 81** Default Value: 5100h Format: U16
	47:32	**Forward Pixel Value 81** Default Value: 5100h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 81** Default Value: 5100h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 81** Default Value: 5100h Format: U16
328..329	63:48	**Inverse R-ch Gamma Corrected Value 82** Default Value: 5200h Format: U16
	47:32	**Inverse Pixel Value 82** Default Value: 5200h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 82** Default Value: 5200h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 82** Default Value: 5200h Format: U16
330..331	63:48	**Forward R-ch Gamma Corrected Value 82** Default Value: 5200h Format: U16
	47:32	**Forward Pixel Value 82** Default Value: 5200h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 82** Default Value: 5200h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 82** Default Value: 5200h Format: U16
332..333	63:48	**Inverse R-ch Gamma Corrected Value 83** Default Value: 5300h Format: U16
	47:32	**Inverse Pixel Value 83** Default Value: 5300h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 83** Default Value: 5300h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 83** Default Value: 5300h Format: U16
334..335	63:48	**Forward R-ch Gamma Corrected Value 83** Default Value: 5300h Format: U16
	47:32	**Forward Pixel Value 83** Default Value: 5300h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 83** Default Value: 5300h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 83** Default Value: 5300h Format: U16
336..337	63:48	**Inverse R-ch Gamma Corrected Value 84** Default Value: 5400h Format: U16
	47:32	**Inverse Pixel Value 84** Default Value: 5400h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 84** Default Value: 5400h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 84** Default Value: 5400h Format: U16
338..339	63:48	**Forward R-ch Gamma Corrected Value 84** Default Value: 5400h Format: U16
	47:32	**Forward Pixel Value 84** Default Value: 5400h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 84** Default Value: 5400h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 84** Default Value: 5400h Format: U16
340..341	63:48	**Inverse R-ch Gamma Corrected Value 85** Default Value: 5500h Format: U16
	47:32	**Inverse Pixel Value 85** Default Value: 5500h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 85** Default Value: 5500h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 85** Default Value: 5500h Format: U16
342..343	63:48	**Forward R-ch Gamma Corrected Value 85** Default Value: 5500h Format: U16
	47:32	**Forward Pixel Value 85** Default Value: 5500h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 85** Default Value: 5500h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 85** Default Value: 5500h Format: U16
344..345	63:48	**Inverse R-ch Gamma Corrected Value 86** Default Value: 5600h Format: U16
	47:32	**Inverse Pixel Value 86** Default Value: 5600h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 86** Default Value: 5600h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 86** Default Value: 5600h Format: U16
346..347	63:48	**Forward R-ch Gamma Corrected Value 86** Default Value: 5600h Format: U16
	47:32	**Forward Pixel Value 86** Default Value: 5600h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 86** Default Value: 5600h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 86** Default Value: 5600h Format: U16
348..349	63:48	**Inverse R-ch Gamma Corrected Value 87** Default Value: 5700h Format: U16
	47:32	**Inverse Pixel Value 87** Default Value: 5700h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 87** Default Value: 5700h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 87** Default Value: 5700h Format: U16
350..351	63:48	**Forward R-ch Gamma Corrected Value 87** Default Value: 5700h Format: U16
	47:32	**Forward Pixel Value 87** Default Value: 5700h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 87** Default Value: 5700h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 87** Default Value: 5700h Format: U16
352..353	63:48	**Inverse R-ch Gamma Corrected Value 88** Default Value: 5800h Format: U16
	47:32	**Inverse Pixel Value 88** Default Value: 5800h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 88** Default Value: 5800h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 88** Default Value: 5800h Format: U16
354..355	63:48	**Forward R-ch Gamma Corrected Value 88** Default Value: 5800h Format: U16
	47:32	**Forward Pixel Value 88** Default Value: 5800h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 88** Default Value: 5800h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 88** Default Value: 5800h Format: U16
356..357	63:48	**Inverse R-ch Gamma Corrected Value 89** Default Value: 5900h Format: U16
	47:32	**Inverse Pixel Value 89** Default Value: 5900h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 89** Default Value: 5900h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 89** Default Value: 5900h Format: U16
358..359	63:48	**Forward R-ch Gamma Corrected Value 89** Default Value: 5900h Format: U16
	47:32	**Forward Pixel Value 89** Default Value: 5900h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 89** Default Value: 5900h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 89** Default Value: 5900h Format: U16
360..361	63:48	**Inverse R-ch Gamma Corrected Value 90** Default Value: 5a00h Format: U16
	47:32	**Inverse Pixel Value 90** Default Value: 5a00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 90** Default Value: 5a00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 90** Default Value: 5a00h Format: U16
362..363	63:48	**Forward R-ch Gamma Corrected Value 90** Default Value: 5a00h Format: U16
	47:32	**Forward Pixel Value 90** Default Value: 5a00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 90** Default Value: 5a00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 90** Default Value: 5a00h Format: U16
364..365	63:48	**Inverse R-ch Gamma Corrected Value 91** Default Value: 5b00h Format: U16
	47:32	**Inverse Pixel Value 91** Default Value: 5b00h Format: U16

Gamut_Expansion_Gamma_Correction

	31:16	**Inverse B-ch Gamma Corrected Value 91** Default Value: 5b00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 91** Default Value: 5b00h Format: U16
366..367	63:48	**Forward R-ch Gamma Corrected Value 91** Default Value: 5b00h Format: U16
	47:32	**Forward Pixel Value 91** Default Value: 5b00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 91** Default Value: 5b00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 91** Default Value: 5b00h Format: U16
368..369	63:48	**Inverse R-ch Gamma Corrected Value 92** Default Value: 5c00h Format: U16
	47:32	**Inverse Pixel Value 92** Default Value: 5c00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 92** Default Value: 5c00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 92** Default Value: 5c00h Format: U16
370..371	63:48	**Forward R-ch Gamma Corrected Value 92** Default Value: 5c00h Format: U16
	47:32	**Forward Pixel Value 92** Default Value: 5c00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 92** Default Value: 5c00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 92** Default Value: 5c00h Format: U16
372..373	63:48	**Inverse R-ch Gamma Corrected Value 93** Default Value: 5d00h Format: U16
	47:32	**Inverse Pixel Value 93** Default Value: 5d00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 93** Default Value: 5d00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 93** Default Value: 5d00h Format: U16
374..375	63:48	**Forward R-ch Gamma Corrected Value 93** Default Value: 5d00h Format: U16
	47:32	**Forward Pixel Value 93** Default Value: 5d00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 93** Default Value: 5d00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 93** Default Value: 5d00h Format: U16
376..377	63:48	**Inverse R-ch Gamma Corrected Value 94** Default Value: 5e00h Format: U16
	47:32	**Inverse Pixel Value 94** Default Value: 5e00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 94** Default Value: 5e00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 94** Default Value: 5e00h Format: U16
378..379	63:48	**Forward R-ch Gamma Corrected Value 94** Default Value: 5e00h Format: U16
	47:32	**Forward Pixel Value 94** Default Value: 5e00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 94** Default Value: 5e00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 94** Default Value: 5e00h Format: U16
380..381	63:48	**Inverse R-ch Gamma Corrected Value 95** Default Value: 5f00h Format: U16
	47:32	**Inverse Pixel Value 95** Default Value: 5f00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 95** Default Value: 5f00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 95** Default Value: 5f00h Format: U16
382..383	63:48	**Forward R-ch Gamma Corrected Value 95** Default Value: 5f00h Format: U16
	47:32	**Forward Pixel Value 95** Default Value: 5f00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 95** Default Value: 5f00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 95** Default Value: 5f00h Format: U16
384..385	63:48	**Inverse R-ch Gamma Corrected Value 96** Default Value: 6000h Format: U16
	47:32	**Inverse Pixel Value 96** Default Value: 6000h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 96** Default Value: 6000h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 96** Default Value: 6000h Format: U16
386..387	63:48	**Forward R-ch Gamma Corrected Value 96** Default Value: 6000h Format: U16
	47:32	**Forward Pixel Value 96** Default Value: 6000h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 96** Default Value: 6000h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 96** Default Value: 6000h Format: U16
388..389	63:48	**Inverse R-ch Gamma Corrected Value 97** Default Value: 6100h Format: U16
	47:32	**Inverse Pixel Value 97** Default Value: 6100h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 97** Default Value: 6100h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 97** Default Value: 6100h Format: U16
390..391	63:48	**Forward R-ch Gamma Corrected Value 97** Default Value: 6100h Format: U16
	47:32	**Forward Pixel Value 97** Default Value: 6100h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 97** Default Value: 6100h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 97** Default Value: 6100h Format: U16
392..393	63:48	**Inverse R-ch Gamma Corrected Value 98** Default Value: 6200h Format: U16
	47:32	**Inverse Pixel Value 98** Default Value: 6200h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 98** Default Value: 6200h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 98** Default Value: 6200h Format: U16
394..395	63:48	**Forward R-ch Gamma Corrected Value 98** Default Value: 6200h Format: U16
	47:32	**Forward Pixel Value 98** Default Value: 6200h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 98** Default Value: 6200h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 98** Default Value: 6200h Format: U16
396..397	63:48	**Inverse R-ch Gamma Corrected Value 99** Default Value: 6300h Format: U16
	47:32	**Inverse Pixel Value 99** Default Value: 6300h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 99** Default Value: 6300h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 99** Default Value: 6300h Format: U16
398..399	63:48	**Forward R-ch Gamma Corrected Value 99** Default Value: 6300h Format: U16
	47:32	**Forward Pixel Value 99** Default Value: 6300h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 99** Default Value: 6300h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 99** Default Value: 6300h Format: U16
400..401	63:48	**Inverse R-ch Gamma Corrected Value 100** Default Value: 6400h Format: U16
	47:32	**Inverse Pixel Value 100** Default Value: 6400h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 100** Default Value: 6400h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 100** Default Value: 6400h Format: U16
402..403	63:48	**Forward R-ch Gamma Corrected Value 100** Default Value: 6400h Format: U16
	47:32	**Forward Pixel Value 100** Default Value: 6400h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 100** Default Value: 6400h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 100** Default Value: 6400h Format: U16
404..405	63:48	**Inverse R-ch Gamma Corrected Value 101** Default Value: 6500h Format: U16
	47:32	**Inverse Pixel Value 101** Default Value: 6500h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 101** Default Value: 6500h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 101** Default Value: 6500h Format: U16
406..407	63:48	**Forward R-ch Gamma Corrected Value 101** Default Value: 6500h Format: U16
	47:32	**Forward Pixel Value 101** Default Value: 6500h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 101** Default Value: 6500h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 101** Default Value: 6500h Format: U16
408..409	63:48	**Inverse R-ch Gamma Corrected Value 102** Default Value: 6600h Format: U16
	47:32	**Inverse Pixel Value 102** Default Value: 6600h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 102** Default Value: 6600h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 102** Default Value: 6600h Format: U16
410..411	63:48	**Forward R-ch Gamma Corrected Value 102** Default Value: 6600h Format: U16
	47:32	**Forward Pixel Value 102** Default Value: 6600h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 102** Default Value: 6600h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 102** Default Value: 6600h Format: U16
412..413	63:48	**Inverse R-ch Gamma Corrected Value 103** Default Value: 6700h Format: U16
	47:32	**Inverse Pixel Value 103** Default Value: 6700h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 103** Default Value: 6700h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 103** Default Value: 6700h Format: U16
414..415	63:48	**Forward R-ch Gamma Corrected Value 103** Default Value: 6700h Format: U16
	47:32	**Forward Pixel Value 103** Default Value: 6700h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 103** Default Value: 6700h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 103** Default Value: 6700h Format: U16
416..417	63:48	**Inverse R-ch Gamma Corrected Value 104** Default Value: 6800h Format: U16
	47:32	**Inverse Pixel Value 104** Default Value: 6800h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 104** Default Value: 6800h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 104** Default Value: 6800h Format: U16
418..419	63:48	**Forward R-ch Gamma Corrected Value 104** Default Value: 6800h Format: U16
	47:32	**Forward Pixel Value 104** Default Value: 6800h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 104** Default Value: 6800h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 104** Default Value: 6800h Format: U16
420..421	63:48	**Inverse R-ch Gamma Corrected Value 105** Default Value: 6900h Format: U16
	47:32	**Inverse Pixel Value 105** Default Value: 6900h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 105** Default Value: 6900h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 105** Default Value: 6900h Format: U16
422..423	63:48	**Forward R-ch Gamma Corrected Value 105** Default Value: 6900h Format: U16
	47:32	**Forward Pixel Value 105** Default Value: 6900h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 105** Default Value: 6900h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 105** Default Value: 6900h Format: U16
424..425	63:48	**Inverse R-ch Gamma Corrected Value 106** Default Value: 6a00h Format: U16
	47:32	**Inverse Pixel Value 106** Default Value: 6a00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 106** Default Value: 6a00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 106** Default Value: 6a00h Format: U16
426..427	63:48	**Forward R-ch Gamma Corrected Value 106** Default Value: 6a00h Format: U16
	47:32	**Forward Pixel Value 106** Default Value: 6a00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 106** Default Value: 6a00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 106** Default Value: 6a00h Format: U16
428..429	63:48	**Inverse R-ch Gamma Corrected Value 107** Default Value: 6b00h Format: U16
	47:32	**Inverse Pixel Value 107** Default Value: 6b00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 107** Default Value: 6b00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 107** Default Value: 6b00h Format: U16
430..431	63:48	**Forward R-ch Gamma Corrected Value 107** Default Value: 6b00h Format: U16
	47:32	**Forward Pixel Value 107** Default Value: 6b00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 107** Default Value: 6b00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 107** Default Value: 6b00h Format: U16
432..433	63:48	**Inverse R-ch Gamma Corrected Value 108** Default Value: 6c00h Format: U16
	47:32	**Inverse Pixel Value 108** Default Value: 6c00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 108** Default Value: 6c00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 108** Default Value: 6c00h Format: U16
434..435	63:48	**Forward R-ch Gamma Corrected Value 108** Default Value: 6c00h Format: U16
	47:32	**Forward Pixel Value 108** Default Value: 6c00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 108** Default Value: 6c00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 108** Default Value: 6c00h Format: U16
436..437	63:48	**Inverse R-ch Gamma Corrected Value 109** Default Value: 6d00h Format: U16
	47:32	**Inverse Pixel Value 109** Default Value: 6d00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 109** Default Value: 6d00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 109** Default Value: 6d00h Format: U16
438..439	63:48	**Forward R-ch Gamma Corrected Value 109** Default Value: 6d00h Format: U16
	47:32	**Forward Pixel Value 109** Default Value: 6d00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 109** Default Value: 6d00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 109** Default Value: 6d00h Format: U16
440..441	63:48	**Inverse R-ch Gamma Corrected Value 110** Default Value: 6e00h Format: U16
	47:32	**Inverse Pixel Value 110** Default Value: 6e00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 110** Default Value: 6e00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 110** Default Value: 6e00h Format: U16
442..443	63:48	**Forward R-ch Gamma Corrected Value 110** Default Value: 6e00h Format: U16
	47:32	**Forward Pixel Value 110** Default Value: 6e00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 110** Default Value: 6e00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 110** Default Value: 6e00h Format: U16
444..445	63:48	**Inverse R-ch Gamma Corrected Value 111** Default Value: 6f00h Format: U16
	47:32	**Inverse Pixel Value 111** Default Value: 6f00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 111** Default Value: 6f00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 111** Default Value: 6f00h Format: U16
446..447	63:48	**Forward R-ch Gamma Corrected Value 111** Default Value: 6f00h Format: U16
	47:32	**Forward Pixel Value 111** Default Value: 6f00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 111** Default Value: 6f00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 111** Default Value: 6f00h Format: U16
448..449	63:48	**Inverse R-ch Gamma Corrected Value 112** Default Value: 7000h Format: U16
	47:32	**Inverse Pixel Value 112** Default Value: 7000h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 112** Default Value: 7000h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 112** Default Value: 7000h Format: U16
450..451	63:48	**Forward R-ch Gamma Corrected Value 112** Default Value: 7000h Format: U16
	47:32	**Forward Pixel Value 112** Default Value: 7000h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 112** Default Value: 7000h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 112** Default Value: 7000h Format: U16
452..453	63:48	**Inverse R-ch Gamma Corrected Value 113** Default Value: 7100h Format: U16
	47:32	**Inverse Pixel Value 113** Default Value: 7100h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 113** Default Value: 7100h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 113** Default Value: 7100h Format: U16
454..455	63:48	**Forward R-ch Gamma Corrected Value 113** Default Value: 7100h Format: U16
	47:32	**Forward Pixel Value 113** Default Value: 7100h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 113** Default Value: 7100h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 113** Default Value: 7100h Format: U16
456..457	63:48	**Inverse R-ch Gamma Corrected Value 114** Default Value: 7200h Format: U16
	47:32	**Inverse Pixel Value 114** Default Value: 7200h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 114** Default Value: 7200h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 114** Default Value: 7200h Format: U16
458..459	63:48	**Forward R-ch Gamma Corrected Value 114** Default Value: 7200h Format: U16
	47:32	**Forward Pixel Value 114** Default Value: 7200h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 114** Default Value: 7200h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 114** Default Value: 7200h Format: U16
460..461	63:48	**Inverse R-ch Gamma Corrected Value 115** Default Value: 7300h Format: U16
	47:32	**Inverse Pixel Value 115** Default Value: 7300h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 115** Default Value: 7300h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 115** Default Value: 7300h Format: U16
462..463	63:48	**Forward R-ch Gamma Corrected Value 115** Default Value: 7300h Format: U16
	47:32	**Forward Pixel Value 115** Default Value: 7300h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 115** Default Value: 7300h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 115** Default Value: 7300h Format: U16
464..465	63:48	**Inverse R-ch Gamma Corrected Value 116** Default Value: 7400h Format: U16
	47:32	**Inverse Pixel Value 116** Default Value: 7400h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 116** Default Value: 7400h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 116** Default Value: 7400h Format: U16
466..467	63:48	**Forward R-ch Gamma Corrected Value 116** Default Value: 7400h Format: U16
	47:32	**Forward Pixel Value 116** Default Value: 7400h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 116** Default Value: 7400h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 116** Default Value: 7400h Format: U16
468..469	63:48	**Inverse R-ch Gamma Corrected Value 117** Default Value: 7500h Format: U16
	47:32	**Inverse Pixel Value 117** Default Value: 7500h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 117** Default Value: 7500h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 117** Default Value: 7500h Format: U16
470..471	63:48	**Forward R-ch Gamma Corrected Value 117** Default Value: 7500h Format: U16
	47:32	**Forward Pixel Value 117** Default Value: 7500h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 117** Default Value: 7500h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 117** Default Value: 7500h Format: U16
472..473	63:48	**Inverse R-ch Gamma Corrected Value 118** Default Value: 7600h Format: U16
	47:32	**Inverse Pixel Value 118** Default Value: 7600h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 118** Default Value: 7600h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 118** Default Value: 7600h Format: U16
474..475	63:48	**Forward R-ch Gamma Corrected Value 118** Default Value: 7600h Format: U16
	47:32	**Forward Pixel Value 118** Default Value: 7600h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 118** Default Value: 7600h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 118** Default Value: 7600h Format: U16
476..477	63:48	**Inverse R-ch Gamma Corrected Value 119** Default Value: 7700h Format: U16
	47:32	**Inverse Pixel Value 119** Default Value: 7700h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 119** Default Value: 7700h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 119** Default Value: 7700h Format: U16
478..479	63:48	**Forward R-ch Gamma Corrected Value 119** Default Value: 7700h Format: U16
	47:32	**Forward Pixel Value 119** Default Value: 7700h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 119** Default Value: 7700h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 119** Default Value: 7700h Format: U16
480..481	63:48	**Inverse R-ch Gamma Corrected Value 120** Default Value: 7800h Format: U16
	47:32	**Inverse Pixel Value 120** Default Value: 7800h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 120** Default Value: 7800h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 120** Default Value: 7800h Format: U16
482..483	63:48	**Forward R-ch Gamma Corrected Value 120** Default Value: 7800h Format: U16
	47:32	**Forward Pixel Value 120** Default Value: 7800h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 120** Default Value: 7800h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 120** Default Value: 7800h Format: U16
484..485	63:48	**Inverse R-ch Gamma Corrected Value 121** Default Value: 7900h Format: U16
	47:32	**Inverse Pixel Value 121** Default Value: 7900h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 121** Default Value: 7900h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 121** Default Value: 7900h Format: U16
486..487	63:48	**Forward R-ch Gamma Corrected Value 121** Default Value: 7900h Format: U16
	47:32	**Forward Pixel Value 121** Default Value: 7900h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 121** Default Value: 7900h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 121** Default Value: 7900h Format: U16
488..489	63:48	**Inverse R-ch Gamma Corrected Value 122** Default Value: 7a00h Format: U16
	47:32	**Inverse Pixel Value 122** Default Value: 7a00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 122** Default Value: 7a00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 122** Default Value: 7a00h Format: U16
490..491	63:48	**Forward R-ch Gamma Corrected Value 122** Default Value: 7a00h Format: U16
	47:32	**Forward Pixel Value 122** Default Value: 7a00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 122** Default Value: 7a00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 122** Default Value: 7a00h Format: U16
492..493	63:48	**Inverse R-ch Gamma Corrected Value 123** Default Value: 7b00h Format: U16
	47:32	**Inverse Pixel Value 123** Default Value: 7b00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 123** Default Value: 7b00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 123** Default Value: 7b00h Format: U16
494..495	63:48	**Forward R-ch Gamma Corrected Value 123** Default Value: 7b00h Format: U16
	47:32	**Forward Pixel Value 123** Default Value: 7b00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 123** Default Value: 7b00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 123** Default Value: 7b00h Format: U16
496..497	63:48	**Inverse R-ch Gamma Corrected Value 124** Default Value: 7c00h Format: U16
	47:32	**Inverse Pixel Value 124** Default Value: 7c00h Format: U16

Gamut_Expansion_Gamma_Correction

	31:16	**Inverse B-ch Gamma Corrected Value 124** Default Value: 7c00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 124** Default Value: 7c00h Format: U16
498..499	63:48	**Forward R-ch Gamma Corrected Value 124** Default Value: 7c00h Format: U16
	47:32	**Forward Pixel Value 124** Default Value: 7c00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 124** Default Value: 7c00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 124** Default Value: 7c00h Format: U16
500..501	63:48	**Inverse R-ch Gamma Corrected Value 125** Default Value: 7d00h Format: U16
	47:32	**Inverse Pixel Value 125** Default Value: 7d00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 125** Default Value: 7d00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 125** Default Value: 7d00h Format: U16
502..503	63:48	**Forward R-ch Gamma Corrected Value 125** Default Value: 7d00h Format: U16
	47:32	**Forward Pixel Value 125** Default Value: 7d00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 125** Default Value: 7d00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 125** Default Value: 7d00h Format: U16
504..505	63:48	**Inverse R-ch Gamma Corrected Value 126** Default Value: 7e00h Format: U16
	47:32	**Inverse Pixel Value 126** Default Value: 7e00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 126** Default Value: 7e00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 126** Default Value: 7e00h Format: U16
506..507	63:48	**Forward R-ch Gamma Corrected Value 126** Default Value: 7e00h Format: U16
	47:32	**Forward Pixel Value 126** Default Value: 7e00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 126** Default Value: 7e00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 126** Default Value: 7e00h Format: U16
508..509	63:48	**Inverse R-ch Gamma Corrected Value 127** Default Value: 7f00h Format: U16
	47:32	**Inverse Pixel Value 127** Default Value: 7f00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 127** Default Value: 7f00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 127** Default Value: 7f00h Format: U16
510..511	63:48	**Forward R-ch Gamma Corrected Value 127** Default Value: 7f00h Format: U16
	47:32	**Forward Pixel Value 127** Default Value: 7f00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 127** Default Value: 7f00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 127** Default Value: 7f00h Format: U16
512..513	63:48	**Inverse R-ch Gamma Corrected Value 128** Default Value: 8000h Format: U16
	47:32	**Inverse Pixel Value 128** Default Value: 8000h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 128** Default Value: 8000h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 128** Default Value: 8000h Format: U16
514..515	63:48	**Forward R-ch Gamma Corrected Value 128** Default Value: 8000h Format: U16
	47:32	**Forward Pixel Value 128** Default Value: 8000h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 128** Default Value: 8000h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 128** Default Value: 8000h Format: U16
516..517	63:48	**Inverse R-ch Gamma Corrected Value 129** Default Value: 8100h Format: U16
	47:32	**Inverse Pixel Value 129** Default Value: 8100h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 129** Default Value: 8100h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 129** Default Value: 8100h Format: U16
518..519	63:48	**Forward R-ch Gamma Corrected Value 129** Default Value: 8100h Format: U16
	47:32	**Forward Pixel Value 129** Default Value: 8100h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 129** Default Value: 8100h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 129** Default Value: 8100h Format: U16
520..521	63:48	**Inverse R-ch Gamma Corrected Value 130** Default Value: 8200h Format: U16
	47:32	**Inverse Pixel Value 130** Default Value: 8200h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 130** Default Value: 8200h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 130** Default Value: 8200h Format: U16
522..523	63:48	**Forward R-ch Gamma Corrected Value 130** Default Value: 8200h Format: U16
	47:32	**Forward Pixel Value 130** Default Value: 8200h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 130** Default Value: 8200h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 130** Default Value: 8200h Format: U16
524..525	63:48	**Inverse R-ch Gamma Corrected Value 131** Default Value: 8300h Format: U16
	47:32	**Inverse Pixel Value 131** Default Value: 8300h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 131** Default Value: 8300h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 131** Default Value: 8300h Format: U16
526..527	63:48	**Forward R-ch Gamma Corrected Value 131** Default Value: 8300h Format: U16
	47:32	**Forward Pixel Value 131** Default Value: 8300h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 131** Default Value: 8300h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 131** Default Value: 8300h Format: U16
528..529	63:48	**Inverse R-ch Gamma Corrected Value 132** Default Value: 8400h Format: U16
	47:32	**Inverse Pixel Value 132** Default Value: 8400h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 132** Default Value: 8400h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 132** Default Value: 8400h Format: U16
530..531	63:48	**Forward R-ch Gamma Corrected Value 132** Default Value: 8400h Format: U16
	47:32	**Forward Pixel Value 132** Default Value: 8400h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 132** Default Value: 8400h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 132** Default Value: 8400h Format: U16
532..533	63:48	**Inverse R-ch Gamma Corrected Value 133** Default Value: 8500h Format: U16
	47:32	**Inverse Pixel Value 133** Default Value: 8500h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 133** Default Value: 8500h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 133** Default Value: 8500h Format: U16
534..535	63:48	**Forward R-ch Gamma Corrected Value 133** Default Value: 8500h Format: U16
	47:32	**Forward Pixel Value 133** Default Value: 8500h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 133** Default Value: 8500h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 133** Default Value: 8500h Format: U16
536..537	63:48	**Inverse R-ch Gamma Corrected Value 134** Default Value: 8600h Format: U16
	47:32	**Inverse Pixel Value 134** Default Value: 8600h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 134** Default Value: 8600h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 134** Default Value: 8600h Format: U16
538..539	63:48	**Forward R-ch Gamma Corrected Value 134** Default Value: 8600h Format: U16
	47:32	**Forward Pixel Value 134** Default Value: 8600h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 134** Default Value: 8600h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 134** Default Value: 8600h Format: U16
540..541	63:48	**Inverse R-ch Gamma Corrected Value 135** Default Value: 8700h Format: U16
	47:32	**Inverse Pixel Value 135** Default Value: 8700h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 135** Default Value: 8700h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 135** Default Value: 8700h Format: U16
542..543	63:48	**Forward R-ch Gamma Corrected Value 135** Default Value: 8700h Format: U16
	47:32	**Forward Pixel Value 135** Default Value: 8700h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 135** Default Value: 8700h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 135** Default Value: 8700h Format: U16
544..545	63:48	**Inverse R-ch Gamma Corrected Value 136** Default Value: 8800h Format: U16
	47:32	**Inverse Pixel Value 136** Default Value: 8800h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 136** Default Value: 8800h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 136** Default Value: 8800h Format: U16
546..547	63:48	**Forward R-ch Gamma Corrected Value 136** Default Value: 8800h Format: U16
	47:32	**Forward Pixel Value 136** Default Value: 8800h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 136** Default Value: 8800h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 136** Default Value: 8800h Format: U16
548..549	63:48	**Inverse R-ch Gamma Corrected Value 137** Default Value: 8900h Format: U16
	47:32	**Inverse Pixel Value 137** Default Value: 8900h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 137** Default Value: 8900h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 137** Default Value: 8900h Format: U16
550..551	63:48	**Forward R-ch Gamma Corrected Value 137** Default Value: 8900h Format: U16
	47:32	**Forward Pixel Value 137** Default Value: 8900h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 137** Default Value: 8900h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 137** Default Value: 8900h Format: U16
552..553	63:48	**Inverse R-ch Gamma Corrected Value 138** Default Value: 8a00h Format: U16
	47:32	**Inverse Pixel Value 138** Default Value: 8a00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 138** Default Value: 8a00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 138** Default Value: 8a00h Format: U16
554..555	63:48	**Forward R-ch Gamma Corrected Value 138** Default Value: 8a00h Format: U16
	47:32	**Forward Pixel Value 138** Default Value: 8a00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 138** Default Value: 8a00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 138** Default Value: 8a00h Format: U16
556..557	63:48	**Inverse R-ch Gamma Corrected Value 139** Default Value: 8b00h Format: U16
	47:32	**Inverse Pixel Value 139** Default Value: 8b00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 139** Default Value: 8b00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 139** Default Value: 8b00h Format: U16
558..559	63:48	**Forward R-ch Gamma Corrected Value 139** Default Value: 8b00h Format: U16
	47:32	**Forward Pixel Value 139** Default Value: 8b00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 139** Default Value: 8b00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 139** Default Value: 8b00h Format: U16
560..561	63:48	**Inverse R-ch Gamma Corrected Value 140** Default Value: 8c00h Format: U16
	47:32	**Inverse Pixel Value 140** Default Value: 8c00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 140** Default Value: 8c00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 140** Default Value: 8c00h Format: U16
562..563	63:48	**Forward R-ch Gamma Corrected Value 140** Default Value: 8c00h Format: U16
	47:32	**Forward Pixel Value 140** Default Value: 8c00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 140** Default Value: 8c00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 140** Default Value: 8c00h Format: U16
564..565	63:48	**Inverse R-ch Gamma Corrected Value 141** Default Value: 8d00h Format: U16
	47:32	**Inverse Pixel Value 141** Default Value: 8d00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 141** Default Value: 8d00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 141** Default Value: 8d00h Format: U16
566..567	63:48	**Forward R-ch Gamma Corrected Value 141** Default Value: 8d00h Format: U16
	47:32	**Forward Pixel Value 141** Default Value: 8d00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 141** Default Value: 8d00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 141** Default Value: 8d00h Format: U16
568..569	63:48	**Inverse R-ch Gamma Corrected Value 142** Default Value: 8e00h Format: U16
	47:32	**Inverse Pixel Value 142** Default Value: 8e00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 142** Default Value: 8e00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 142** Default Value: 8e00h Format: U16
570..571	63:48	**Forward R-ch Gamma Corrected Value 142** Default Value: 8e00h Format: U16
	47:32	**Forward Pixel Value 142** Default Value: 8e00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 142** Default Value: 8e00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 142** Default Value: 8e00h Format: U16
572..573	63:48	**Inverse R-ch Gamma Corrected Value 143** Default Value: 8f00h Format: U16
	47:32	**Inverse Pixel Value 143** Default Value: 8f00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 143** Default Value: 8f00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 143** Default Value: 8f00h Format: U16
574..575	63:48	**Forward R-ch Gamma Corrected Value 143** Default Value: 8f00h Format: U16
	47:32	**Forward Pixel Value 143** Default Value: 8f00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 143** Default Value: 8f00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 143** Default Value: 8f00h Format: U16
576..577	63:48	**Inverse R-ch Gamma Corrected Value 144** Default Value: 9000h Format: U16
	47:32	**Inverse Pixel Value 144** Default Value: 9000h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 144** Default Value: 9000h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 144** Default Value: 9000h Format: U16
578..579	63:48	**Forward R-ch Gamma Corrected Value 144** Default Value: 9000h Format: U16
	47:32	**Forward Pixel Value 144** Default Value: 9000h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 144** Default Value: 9000h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 144** Default Value: 9000h Format: U16
580..581	63:48	**Inverse R-ch Gamma Corrected Value 145** Default Value: 9100h Format: U16
	47:32	**Inverse Pixel Value 145** Default Value: 9100h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 145** Default Value: 9100h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 145** Default Value: 9100h Format: U16
582..583	63:48	**Forward R-ch Gamma Corrected Value 145** Default Value: 9100h Format: U16
	47:32	**Forward Pixel Value 145** Default Value: 9100h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 145** Default Value: 9100h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 145** Default Value: 9100h Format: U16
584..585	63:48	**Inverse R-ch Gamma Corrected Value 146** Default Value: 9200h Format: U16
	47:32	**Inverse Pixel Value 146** Default Value: 9200h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 146** Default Value: 9200h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 146** Default Value: 9200h Format: U16
586..587	63:48	**Forward R-ch Gamma Corrected Value 146** Default Value: 9200h Format: U16
	47:32	**Forward Pixel Value 146** Default Value: 9200h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 146** Default Value: 9200h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 146** Default Value: 9200h Format: U16
588..589	63:48	**Inverse R-ch Gamma Corrected Value 147** Default Value: 9300h Format: U16
	47:32	**Inverse Pixel Value 147** Default Value: 9300h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 147** Default Value: 9300h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 147** Default Value: 9300h Format: U16
590..591	63:48	**Forward R-ch Gamma Corrected Value 147** Default Value: 9300h Format: U16
	47:32	**Forward Pixel Value 147** Default Value: 9300h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 147** Default Value: 9300h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 147** Default Value: 9300h Format: U16
592..593	63:48	**Inverse R-ch Gamma Corrected Value 148** Default Value: 9400h Format: U16
	47:32	**Inverse Pixel Value 148** Default Value: 9400h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 148** Default Value: 9400h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 148** Default Value: 9400h Format: U16
594..595	63:48	**Forward R-ch Gamma Corrected Value 148** Default Value: 9400h Format: U16
	47:32	**Forward Pixel Value 148** Default Value: 9400h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 148** Default Value: 9400h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 148** Default Value: 9400h Format: U16
596..597	63:48	**Inverse R-ch Gamma Corrected Value 149** Default Value: 9500h Format: U16
	47:32	**Inverse Pixel Value 149** Default Value: 9500h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 149** Default Value: 9500h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 149** Default Value: 9500h Format: U16
598..599	63:48	**Forward R-ch Gamma Corrected Value 149** Default Value: 9500h Format: U16
	47:32	**Forward Pixel Value 149** Default Value: 9500h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 149** Default Value: 9500h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 149** Default Value: 9500h Format: U16
600..601	63:48	**Inverse R-ch Gamma Corrected Value 150** Default Value: 9600h Format: U16
	47:32	**Inverse Pixel Value 150** Default Value: 9600h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 150** Default Value: 9600h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 150** Default Value: 9600h Format: U16
602..603	63:48	**Forward R-ch Gamma Corrected Value 150** Default Value: 9600h Format: U16
	47:32	**Forward Pixel Value 150** Default Value: 9600h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 150** Default Value: 9600h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 150** Default Value: 9600h Format: U16
604..605	63:48	**Inverse R-ch Gamma Corrected Value 151** Default Value: 9700h Format: U16
	47:32	**Inverse Pixel Value 151** Default Value: 9700h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 151** Default Value: 9700h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 151** Default Value: 9700h Format: U16
606..607	63:48	**Forward R-ch Gamma Corrected Value 151** Default Value: 9700h Format: U16
	47:32	**Forward Pixel Value 151** Default Value: 9700h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 151** Default Value: 9700h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 151** Default Value: 9700h Format: U16
608..609	63:48	**Inverse R-ch Gamma Corrected Value 152** Default Value: 9800h Format: U16
	47:32	**Inverse Pixel Value 152** Default Value: 9800h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 152** Default Value: 9800h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 152** Default Value: 9800h Format: U16
610..611	63:48	**Forward R-ch Gamma Corrected Value 152** Default Value: 9800h Format: U16
	47:32	**Forward Pixel Value 152** Default Value: 9800h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 152** Default Value: 9800h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 152** Default Value: 9800h Format: U16
612..613	63:48	**Inverse R-ch Gamma Corrected Value 153** Default Value: 9900h Format: U16
	47:32	**Inverse Pixel Value 153** Default Value: 9900h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 153** Default Value: 9900h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 153** Default Value: 9900h Format: U16
614..615	63:48	**Forward R-ch Gamma Corrected Value 153** Default Value: 9900h Format: U16
	47:32	**Forward Pixel Value 153** Default Value: 9900h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 153** Default Value: 9900h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 153** Default Value: 9900h Format: U16
616..617	63:48	**Inverse R-ch Gamma Corrected Value 154** Default Value: 9a00h Format: U16
	47:32	**Inverse Pixel Value 154** Default Value: 9a00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 154** Default Value: 9a00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 154** Default Value: 9a00h Format: U16
618..619	63:48	**Forward R-ch Gamma Corrected Value 154** Default Value: 9a00h Format: U16
	47:32	**Forward Pixel Value 154** Default Value: 9a00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 154** Default Value: 9a00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 154** Default Value: 9a00h Format: U16
620..621	63:48	**Inverse R-ch Gamma Corrected Value 155** Default Value: 9b00h Format: U16
	47:32	**Inverse Pixel Value 155** Default Value: 9b00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 155** Default Value: 9b00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 155** Default Value: 9b00h Format: U16
622..623	63:48	**Forward R-ch Gamma Corrected Value 155** Default Value: 9b00h Format: U16
	47:32	**Forward Pixel Value 155** Default Value: 9b00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 155** Default Value: 9b00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 155** Default Value: 9b00h Format: U16
624..625	63:48	**Inverse R-ch Gamma Corrected Value 156** Default Value: 9c00h Format: U16
	47:32	**Inverse Pixel Value 156** Default Value: 9c00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 156** Default Value: 9c00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 156** Default Value: 9c00h Format: U16
626..627	63:48	**Forward R-ch Gamma Corrected Value 156** Default Value: 9c00h Format: U16
	47:32	**Forward Pixel Value 156** Default Value: 9c00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 156** Default Value: 9c00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 156** Default Value: 9c00h Format: U16
628..629	63:48	**Inverse R-ch Gamma Corrected Value 157** Default Value: 9d00h Format: U16
	47:32	**Inverse Pixel Value 157** Default Value: 9d00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 157** Default Value: 9d00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 157** Default Value: 9d00h Format: U16
630..631	63:48	**Forward R-ch Gamma Corrected Value 157** Default Value: 9d00h Format: U16
	47:32	**Forward Pixel Value 157** Default Value: 9d00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 157** Default Value: 9d00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 157** Default Value: 9d00h Format: U16
632..633	63:48	**Inverse R-ch Gamma Corrected Value 158** Default Value: 9e00h Format: U16
	47:32	**Inverse Pixel Value 158** Default Value: 9e00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 158** Default Value: 9e00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 158** Default Value: 9e00h Format: U16
634..635	63:48	**Forward R-ch Gamma Corrected Value 158** Default Value: 9e00h Format: U16
	47:32	**Forward Pixel Value 158** Default Value: 9e00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 158** Default Value: 9e00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 158** Default Value: 9e00h Format: U16
636..637	63:48	**Inverse R-ch Gamma Corrected Value 159** Default Value: 9f00h Format: U16
	47:32	**Inverse Pixel Value 159** Default Value: 9f00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 159** Default Value: 9f00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 159** Default Value: 9f00h Format: U16
638..639	63:48	**Forward R-ch Gamma Corrected Value 159** Default Value: 9f00h Format: U16
	47:32	**Forward Pixel Value 159** Default Value: 9f00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 159** Default Value: 9f00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 159** Default Value: 9f00h Format: U16
640..641	63:48	**Inverse R-ch Gamma Corrected Value 160** Default Value: a000h Format: U16
	47:32	**Inverse Pixel Value 160** Default Value: a000h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 160** Default Value: a000h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 160** Default Value: a000h Format: U16
642..643	63:48	**Forward R-ch Gamma Corrected Value 160** Default Value: a000h Format: U16
	47:32	**Forward Pixel Value 160** Default Value: a000h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 160** Default Value: a000h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 160** Default Value: a000h Format: U16
644..645	63:48	**Inverse R-ch Gamma Corrected Value 161** Default Value: a100h Format: U16
	47:32	**Inverse Pixel Value 161** Default Value: a100h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 161** Default Value: a100h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 161** Default Value: a100h Format: U16
646..647	63:48	**Forward R-ch Gamma Corrected Value 161** Default Value: a100h Format: U16
	47:32	**Forward Pixel Value 161** Default Value: a100h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 161** Default Value: a100h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 161** Default Value: a100h Format: U16
648..649	63:48	**Inverse R-ch Gamma Corrected Value 162** Default Value: a200h Format: U16
	47:32	**Inverse Pixel Value 162** Default Value: a200h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 162** Default Value: a200h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 162** Default Value: a200h Format: U16
650..651	63:48	**Forward R-ch Gamma Corrected Value 162** Default Value: a200h Format: U16
	47:32	**Forward Pixel Value 162** Default Value: a200h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 162** Default Value: a200h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 162** Default Value: a200h Format: U16
652..653	63:48	**Inverse R-ch Gamma Corrected Value 163** Default Value: a300h Format: U16
	47:32	**Inverse Pixel Value 163** Default Value: a300h Format: U16

Gamut_Expansion_Gamma_Correction

	31:16	**Inverse B-ch Gamma Corrected Value 163** Default Value: a300h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 163** Default Value: a300h Format: U16
654..655	63:48	**Forward R-ch Gamma Corrected Value 163** Default Value: a300h Format: U16
	47:32	**Forward Pixel Value 163** Default Value: a300h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 163** Default Value: a300h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 163** Default Value: a300h Format: U16
656..657	63:48	**Inverse R-ch Gamma Corrected Value 164** Default Value: a400h Format: U16
	47:32	**Inverse Pixel Value 164** Default Value: a400h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 164** Default Value: a400h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 164** Default Value: a400h Format: U16
658..659	63:48	**Forward R-ch Gamma Corrected Value 164** Default Value: a400h Format: U16
	47:32	**Forward Pixel Value 164** Default Value: a400h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 164** Default Value: a400h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 164** Default Value: a400h Format: U16
660..661	63:48	**Inverse R-ch Gamma Corrected Value 165** Default Value: a500h Format: U16
	47:32	**Inverse Pixel Value 165** Default Value: a500h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 165** Default Value: a500h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 165** Default Value: a500h Format: U16
662..663	63:48	**Forward R-ch Gamma Corrected Value 165** Default Value: a500h Format: U16
	47:32	**Forward Pixel Value 165** Default Value: a500h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 165** Default Value: a500h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 165** Default Value: a500h Format: U16
664..665	63:48	**Inverse R-ch Gamma Corrected Value 166** Default Value: a600h Format: U16
	47:32	**Inverse Pixel Value 166** Default Value: a600h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 166** Default Value: a600h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 166** Default Value: a600h Format: U16
666..667	63:48	**Forward R-ch Gamma Corrected Value 166** Default Value: a600h Format: U16
	47:32	**Forward Pixel Value 166** Default Value: a600h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 166** Default Value: a600h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 166** Default Value: a600h Format: U16
668..669	63:48	**Inverse R-ch Gamma Corrected Value 167** Default Value: a700h Format: U16
	47:32	**Inverse Pixel Value 167** Default Value: a700h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 167** Default Value: a700h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 167** Default Value: a700h Format: U16
670..671	63:48	**Forward R-ch Gamma Corrected Value 167** Default Value: a700h Format: U16
	47:32	**Forward Pixel Value 167** Default Value: a700h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 167** Default Value: a700h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 167** Default Value: a700h Format: U16
672..673	63:48	**Inverse R-ch Gamma Corrected Value 168** Default Value: a800h Format: U16
	47:32	**Inverse Pixel Value 168** Default Value: a800h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 168** Default Value: a800h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 168** Default Value: a800h Format: U16
674..675	63:48	**Forward R-ch Gamma Corrected Value 168** Default Value: a800h Format: U16
	47:32	**Forward Pixel Value 168** Default Value: a800h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 168** Default Value: a800h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 168** Default Value: a800h Format: U16
676..677	63:48	**Inverse R-ch Gamma Corrected Value 169** Default Value: a900h Format: U16
	47:32	**Inverse Pixel Value 169** Default Value: a900h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 169** Default Value: a900h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 169** Default Value: a900h Format: U16
678..679	63:48	**Forward R-ch Gamma Corrected Value 169** Default Value: a900h Format: U16
	47:32	**Forward Pixel Value 169** Default Value: a900h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 169** Default Value: a900h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 169** Default Value: a900h Format: U16
680..681	63:48	**Inverse R-ch Gamma Corrected Value 170** Default Value: aa00h Format: U16
	47:32	**Inverse Pixel Value 170** Default Value: aa00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 170** Default Value: aa00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 170** Default Value: aa00h Format: U16
682..683	63:48	**Forward R-ch Gamma Corrected Value 170** Default Value: aa00h Format: U16
	47:32	**Forward Pixel Value 170** Default Value: aa00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 170** Default Value: aa00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 170** Default Value: aa00h Format: U16
684..685	63:48	**Inverse R-ch Gamma Corrected Value 171** Default Value: ab00h Format: U16
	47:32	**Inverse Pixel Value 171** Default Value: ab00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 171** Default Value: ab00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 171** Default Value: ab00h Format: U16
686..687	63:48	**Forward R-ch Gamma Corrected Value 171** Default Value: ab00h Format: U16
	47:32	**Forward Pixel Value 171** Default Value: ab00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 171** Default Value: ab00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 171** Default Value: ab00h Format: U16
688..689	63:48	**Inverse R-ch Gamma Corrected Value 172** Default Value: ac00h Format: U16
	47:32	**Inverse Pixel Value 172** Default Value: ac00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 172** Default Value: ac00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 172** Default Value: ac00h Format: U16
690..691	63:48	**Forward R-ch Gamma Corrected Value 172** Default Value: ac00h Format: U16
	47:32	**Forward Pixel Value 172** Default Value: ac00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 172** Default Value: ac00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 172** Default Value: ac00h Format: U16
692..693	63:48	**Inverse R-ch Gamma Corrected Value 173** Default Value: ad00h Format: U16
	47:32	**Inverse Pixel Value 173** Default Value: ad00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 173** Default Value: ad00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 173** Default Value: ad00h Format: U16
694..695	63:48	**Forward R-ch Gamma Corrected Value 173** Default Value: ad00h Format: U16
	47:32	**Forward Pixel Value 173** Default Value: ad00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 173** Default Value: ad00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 173** Default Value: ad00h Format: U16
696..697	63:48	**Inverse R-ch Gamma Corrected Value 174** Default Value: ae00h Format: U16
	47:32	**Inverse Pixel Value 174** Default Value: ae00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 174** Default Value: ae00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 174** Default Value: ae00h Format: U16
698..699	63:48	**Forward R-ch Gamma Corrected Value 174** Default Value: ae00h Format: U16
	47:32	**Forward Pixel Value 174** Default Value: ae00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 174** Default Value: ae00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 174** Default Value: ae00h Format: U16
700..701	63:48	**Inverse R-ch Gamma Corrected Value 175** Default Value: af00h Format: U16
	47:32	**Inverse Pixel Value 175** Default Value: af00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 175** Default Value: af00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 175** Default Value: af00h Format: U16
702..703	63:48	**Forward R-ch Gamma Corrected Value 175** Default Value: af00h Format: U16
	47:32	**Forward Pixel Value 175** Default Value: af00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 175** Default Value: af00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 175** Default Value: af00h Format: U16
704..705	63:48	**Inverse R-ch Gamma Corrected Value 176** Default Value: b000h Format: U16
	47:32	**Inverse Pixel Value 176** Default Value: b000h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 176** Default Value: b000h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 176** Default Value: b000h Format: U16
706..707	63:48	**Forward R-ch Gamma Corrected Value 176** Default Value: b000h Format: U16
	47:32	**Forward Pixel Value 176** Default Value: b000h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 176** Default Value: b000h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 176** Default Value: b000h Format: U16
708..709	63:48	**Inverse R-ch Gamma Corrected Value 177** Default Value: b100h Format: U16
	47:32	**Inverse Pixel Value 177** Default Value: b100h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 177** Default Value: b100h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 177** Default Value: b100h Format: U16
710..711	63:48	**Forward R-ch Gamma Corrected Value 177** Default Value: b100h Format: U16
	47:32	**Forward Pixel Value 177** Default Value: b100h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 177** Default Value: b100h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 177** Default Value: b100h Format: U16
712..713	63:48	**Inverse R-ch Gamma Corrected Value 178** Default Value: b200h Format: U16
	47:32	**Inverse Pixel Value 178** Default Value: b200h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 178** Default Value: b200h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 178** Default Value: b200h Format: U16
714..715	63:48	**Forward R-ch Gamma Corrected Value 178** Default Value: b200h Format: U16
	47:32	**Forward Pixel Value 178** Default Value: b200h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 178** Default Value: b200h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 178** Default Value: b200h Format: U16
716..717	63:48	**Inverse R-ch Gamma Corrected Value 179** Default Value: b300h Format: U16
	47:32	**Inverse Pixel Value 179** Default Value: b300h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 179** Default Value: b300h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 179** Default Value: b300h Format: U16
718..719	63:48	**Forward R-ch Gamma Corrected Value 179** Default Value: b300h Format: U16
	47:32	**Forward Pixel Value 179** Default Value: b300h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 179** Default Value: b300h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 179** Default Value: b300h Format: U16
720..721	63:48	**Inverse R-ch Gamma Corrected Value 180** Default Value: b400h Format: U16
	47:32	**Inverse Pixel Value 180** Default Value: b400h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 180** Default Value: b400h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 180** Default Value: b400h Format: U16
722..723	63:48	**Forward R-ch Gamma Corrected Value 180** Default Value: b400h Format: U16
	47:32	**Forward Pixel Value 180** Default Value: b400h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 180** Default Value: b400h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 180** Default Value: b400h Format: U16
724..725	63:48	**Inverse R-ch Gamma Corrected Value 181** Default Value: b500h Format: U16
	47:32	**Inverse Pixel Value 181** Default Value: b500h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 181** Default Value: b500h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 181** Default Value: b500h Format: U16
726..727	63:48	**Forward R-ch Gamma Corrected Value 181** Default Value: b500h Format: U16
	47:32	**Forward Pixel Value 181** Default Value: b500h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 181** Default Value: b500h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 181** Default Value: b500h Format: U16
728..729	63:48	**Inverse R-ch Gamma Corrected Value 182** Default Value: b600h Format: U16
	47:32	**Inverse Pixel Value 182** Default Value: b600h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 182** Default Value: b600h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 182** Default Value: b600h Format: U16
730..731	63:48	**Forward R-ch Gamma Corrected Value 182** Default Value: b600h Format: U16
	47:32	**Forward Pixel Value 182** Default Value: b600h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 182** Default Value: b600h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 182** Default Value: b600h Format: U16
732..733	63:48	**Inverse R-ch Gamma Corrected Value 183** Default Value: b700h Format: U16
	47:32	**Inverse Pixel Value 183** Default Value: b700h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 183** Default Value: b700h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 183** Default Value: b700h Format: U16
734..735	63:48	**Forward R-ch Gamma Corrected Value 183** Default Value: b700h Format: U16
	47:32	**Forward Pixel Value 183** Default Value: b700h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 183** Default Value: b700h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 183** Default Value: b700h Format: U16
736..737	63:48	**Inverse R-ch Gamma Corrected Value 184** Default Value: b800h Format: U16
	47:32	**Inverse Pixel Value 184** Default Value: b800h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 184** Default Value: b800h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 184** Default Value: b800h Format: U16
738..739	63:48	**Forward R-ch Gamma Corrected Value 184** Default Value: b800h Format: U16
	47:32	**Forward Pixel Value 184** Default Value: b800h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 184** Default Value: b800h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 184** Default Value: b800h Format: U16
740..741	63:48	**Inverse R-ch Gamma Corrected Value 185** Default Value: b900h Format: U16
	47:32	**Inverse Pixel Value 185** Default Value: b900h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 185** Default Value: b900h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 185** Default Value: b900h Format: U16
742..743	63:48	**Forward R-ch Gamma Corrected Value 185** Default Value: b900h Format: U16
	47:32	**Forward Pixel Value 185** Default Value: b900h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 185** Default Value: b900h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 185** Default Value: b900h Format: U16
744..745	63:48	**Inverse R-ch Gamma Corrected Value 186** Default Value: ba00h Format: U16
	47:32	**Inverse Pixel Value 186** Default Value: ba00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 186** Default Value: ba00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 186** Default Value: ba00h Format: U16
746..747	63:48	**Forward R-ch Gamma Corrected Value 186** Default Value: ba00h Format: U16
	47:32	**Forward Pixel Value 186** Default Value: ba00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 186** Default Value: ba00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 186** Default Value: ba00h Format: U16
748..749	63:48	**Inverse R-ch Gamma Corrected Value 187** Default Value: bb00h Format: U16
	47:32	**Inverse Pixel Value 187** Default Value: bb00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 187** Default Value: bb00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 187** Default Value: bb00h Format: U16
750..751	63:48	**Forward R-ch Gamma Corrected Value 187** Default Value: bb00h Format: U16
	47:32	**Forward Pixel Value 187** Default Value: bb00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 187** Default Value: bb00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 187** Default Value: bb00h Format: U16
752..753	63:48	**Inverse R-ch Gamma Corrected Value 188** Default Value: bc00h Format: U16
	47:32	**Inverse Pixel Value 188** Default Value: bc00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 188** Default Value: bc00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 188** Default Value: bc00h Format: U16
754..755	63:48	**Forward R-ch Gamma Corrected Value 188** Default Value: bc00h Format: U16
	47:32	**Forward Pixel Value 188** Default Value: bc00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 188** Default Value: bc00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 188** Default Value: bc00h Format: U16
756..757	63:48	**Inverse R-ch Gamma Corrected Value 189** Default Value: bd00h Format: U16
	47:32	**Inverse Pixel Value 189** Default Value: bd00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 189** Default Value: bd00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 189** Default Value: bd00h Format: U16
758..759	63:48	**Forward R-ch Gamma Corrected Value 189** Default Value: bd00h Format: U16
	47:32	**Forward Pixel Value 189** Default Value: bd00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 189** Default Value: bd00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 189** Default Value: bd00h Format: U16
760..761	63:48	**Inverse R-ch Gamma Corrected Value 190** Default Value: be00h Format: U16
	47:32	**Inverse Pixel Value 190** Default Value: be00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 190** Default Value: be00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 190** Default Value: be00h Format: U16
762..763	63:48	**Forward R-ch Gamma Corrected Value 190** Default Value: be00h Format: U16
	47:32	**Forward Pixel Value 190** Default Value: be00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 190** Default Value: be00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 190** Default Value: be00h Format: U16
764..765	63:48	**Inverse R-ch Gamma Corrected Value 191** Default Value: bf00h Format: U16
	47:32	**Inverse Pixel Value 191** Default Value: bf00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 191** Default Value: bf00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 191** Default Value: bf00h Format: U16
766..767	63:48	**Forward R-ch Gamma Corrected Value 191** Default Value: bf00h Format: U16
	47:32	**Forward Pixel Value 191** Default Value: bf00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 191** Default Value: bf00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 191** Default Value: bf00h Format: U16
768..769	63:48	**Inverse R-ch Gamma Corrected Value 192** Default Value: c000h Format: U16
	47:32	**Inverse Pixel Value 192** Default Value: c000h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 192** Default Value: c000h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 192** Default Value: c000h Format: U16
770..771	63:48	**Forward R-ch Gamma Corrected Value 192** Default Value: c000h Format: U16
	47:32	**Forward Pixel Value 192** Default Value: c000h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 192** Default Value: c000h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 192** Default Value: c000h Format: U16
772..773	63:48	**Inverse R-ch Gamma Corrected Value 193** Default Value: c100h Format: U16
	47:32	**Inverse Pixel Value 193** Default Value: c100h Format: U16

Gamut_Expansion_Gamma_Correction

	31:16	**Inverse B-ch Gamma Corrected Value 193** Default Value: c100h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 193** Default Value: c100h Format: U16
774..775	63:48	**Forward R-ch Gamma Corrected Value 193** Default Value: c100h Format: U16
	47:32	**Forward Pixel Value 193** Default Value: c100h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 193** Default Value: c100h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 193** Default Value: c100h Format: U16
776..777	63:48	**Inverse R-ch Gamma Corrected Value 194** Default Value: c200h Format: U16
	47:32	**Inverse Pixel Value 194** Default Value: c200h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 194** Default Value: c200h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 194** Default Value: c200h Format: U16
778..779	63:48	**Forward R-ch Gamma Corrected Value 194** Default Value: c200h Format: U16
	47:32	**Forward Pixel Value 194** Default Value: c200h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 194** Default Value: c200h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 194** Default Value: c200h Format: U16
780..781	63:48	**Inverse R-ch Gamma Corrected Value 195** Default Value: c300h Format: U16
	47:32	**Inverse Pixel Value 195** Default Value: c300h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 195** Default Value: c300h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 195** Default Value: c300h Format: U16
782..783	63:48	**Forward R-ch Gamma Corrected Value 195** Default Value: c300h Format: U16
	47:32	**Forward Pixel Value 195** Default Value: c300h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 195** Default Value: c300h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 195** Default Value: c300h Format: U16
784..785	63:48	**Inverse R-ch Gamma Corrected Value 196** Default Value: c400h Format: U16
	47:32	**Inverse Pixel Value 196** Default Value: c400h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 196** Default Value: c400h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 196** Default Value: c400h Format: U16
786..787	63:48	**Forward R-ch Gamma Corrected Value 196** Default Value: c400h Format: U16
	47:32	**Forward Pixel Value 196** Default Value: c400h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 196** Default Value: c400h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 196** Default Value: c400h Format: U16
788..789	63:48	**Inverse R-ch Gamma Corrected Value 197** Default Value: c500h Format: U16
	47:32	**Inverse Pixel Value 197** Default Value: c500h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 197** Default Value: c500h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 197** Default Value: c500h Format: U16
790..791	63:48	**Forward R-ch Gamma Corrected Value 197** Default Value: c500h Format: U16
	47:32	**Forward Pixel Value 197** Default Value: c500h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 197** Default Value: c500h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 197** Default Value: c500h Format: U16
792..793	63:48	**Inverse R-ch Gamma Corrected Value 198** Default Value: c600h Format: U16
	47:32	**Inverse Pixel Value 198** Default Value: c600h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 198** Default Value: c600h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 198** Default Value: c600h Format: U16
794..795	63:48	**Forward R-ch Gamma Corrected Value 198** Default Value: c600h Format: U16
	47:32	**Forward Pixel Value 198** Default Value: c600h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 198** Default Value: c600h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 198** Default Value: c600h Format: U16
796..797	63:48	**Inverse R-ch Gamma Corrected Value 199** Default Value: c700h Format: U16
	47:32	**Inverse Pixel Value 199** Default Value: c700h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 199** Default Value: c700h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 199** Default Value: c700h Format: U16
798..799	63:48	**Forward R-ch Gamma Corrected Value 199** Default Value: c700h Format: U16
	47:32	**Forward Pixel Value 199** Default Value: c700h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 199** Default Value: c700h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 199** Default Value: c700h Format: U16
800..801	63:48	**Inverse R-ch Gamma Corrected Value 200** Default Value: c800h Format: U16
	47:32	**Inverse Pixel Value 200** Default Value: c800h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 200** Default Value: c800h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 200** Default Value: c800h Format: U16
802..803	63:48	**Forward R-ch Gamma Corrected Value 200** Default Value: c800h Format: U16
	47:32	**Forward Pixel Value 200** Default Value: c800h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 200** Default Value: c800h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 200** Default Value: c800h Format: U16
804..805	63:48	**Inverse R-ch Gamma Corrected Value 201** Default Value: c900h Format: U16
	47:32	**Inverse Pixel Value 201** Default Value: c900h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 201** Default Value: c900h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 201** Default Value: c900h Format: U16
806..807	63:48	**Forward R-ch Gamma Corrected Value 201** Default Value: c900h Format: U16
	47:32	**Forward Pixel Value 201** Default Value: c900h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 201** Default Value: c900h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 201** Default Value: c900h Format: U16
808..809	63:48	**Inverse R-ch Gamma Corrected Value 202** Default Value: ca00h Format: U16
	47:32	**Inverse Pixel Value 202** Default Value: ca00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 202** Default Value: ca00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 202** Default Value: ca00h Format: U16
810..811	63:48	**Forward R-ch Gamma Corrected Value 202** Default Value: ca00h Format: U16
	47:32	**Forward Pixel Value 202** Default Value: ca00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 202** Default Value: ca00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 202** Default Value: ca00h Format: U16
812..813	63:48	**Inverse R-ch Gamma Corrected Value 203** Default Value: cb00h Format: U16
	47:32	**Inverse Pixel Value 203** Default Value: cb00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 203** Default Value: cb00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 203** Default Value: cb00h Format: U16
814..815	63:48	**Forward R-ch Gamma Corrected Value 203** Default Value: cb00h Format: U16
	47:32	**Forward Pixel Value 203** Default Value: cb00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 203** Default Value: cb00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 203** Default Value: cb00h Format: U16
816..817	63:48	**Inverse R-ch Gamma Corrected Value 204** Default Value: cc00h Format: U16
	47:32	**Inverse Pixel Value 204** Default Value: cc00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 204** Default Value: cc00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 204** Default Value: cc00h Format: U16
818..819	63:48	**Forward R-ch Gamma Corrected Value 204** Default Value: cc00h Format: U16
	47:32	**Forward Pixel Value 204** Default Value: cc00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 204** Default Value: cc00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 204** Default Value: cc00h Format: U16
820..821	63:48	**Inverse R-ch Gamma Corrected Value 205** Default Value: cd00h Format: U16
	47:32	**Inverse Pixel Value 205** Default Value: cd00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 205** Default Value: cd00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 205** Default Value: cd00h Format: U16
822..823	63:48	**Forward R-ch Gamma Corrected Value 205** Default Value: cd00h Format: U16
	47:32	**Forward Pixel Value 205** Default Value: cd00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 205** Default Value: cd00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 205** Default Value: cd00h Format: U16
824..825	63:48	**Inverse R-ch Gamma Corrected Value 206** Default Value: ce00h Format: U16
	47:32	**Inverse Pixel Value 206** Default Value: ce00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 206** Default Value: ce00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 206** Default Value: ce00h Format: U16
826..827	63:48	**Forward R-ch Gamma Corrected Value 206** Default Value: ce00h Format: U16
	47:32	**Forward Pixel Value 206** Default Value: ce00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 206** Default Value: ce00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 206** Default Value: ce00h Format: U16
828..829	63:48	**Inverse R-ch Gamma Corrected Value 207** Default Value: cf00h Format: U16
	47:32	**Inverse Pixel Value 207** Default Value: cf00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 207** Default Value: cf00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 207** Default Value: cf00h Format: U16
830..831	63:48	**Forward R-ch Gamma Corrected Value 207** Default Value: cf00h Format: U16
	47:32	**Forward Pixel Value 207** Default Value: cf00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 207** Default Value: cf00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 207** Default Value: cf00h Format: U16
832..833	63:48	**Inverse R-ch Gamma Corrected Value 208** Default Value: d000h Format: U16
	47:32	**Inverse Pixel Value 208** Default Value: d000h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 208** Default Value: d000h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 208** Default Value: d000h Format: U16
834..835	63:48	**Forward R-ch Gamma Corrected Value 208** Default Value: d000h Format: U16
	47:32	**Forward Pixel Value 208** Default Value: d000h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 208** Default Value: d000h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 208** Default Value: d000h Format: U16
836..837	63:48	**Inverse R-ch Gamma Corrected Value 209** Default Value: d100h Format: U16
	47:32	**Inverse Pixel Value 209** Default Value: d100h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 209** Default Value: d100h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 209** Default Value: d100h Format: U16
838..839	63:48	**Forward R-ch Gamma Corrected Value 209** Default Value: d100h Format: U16
	47:32	**Forward Pixel Value 209** Default Value: d100h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 209** Default Value: d100h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 209** Default Value: d100h Format: U16
840..841	63:48	**Inverse R-ch Gamma Corrected Value 210** Default Value: d200h Format: U16
	47:32	**Inverse Pixel Value 210** Default Value: d200h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 210** Default Value: d200h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 210** Default Value: d200h Format: U16
842..843	63:48	**Forward R-ch Gamma Corrected Value 210** Default Value: d200h Format: U16
	47:32	**Forward Pixel Value 210** Default Value: d200h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 210** Default Value: d200h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 210** Default Value: d200h Format: U16
844..845	63:48	**Inverse R-ch Gamma Corrected Value 211** Default Value: d300h Format: U16
	47:32	**Inverse Pixel Value 211** Default Value: d300h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 211** Default Value: d300h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 211** Default Value: d300h Format: U16
846..847	63:48	**Forward R-ch Gamma Corrected Value 211** Default Value: d300h Format: U16
	47:32	**Forward Pixel Value 211** Default Value: d300h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 211** Default Value: d300h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 211** Default Value: d300h Format: U16
848..849	63:48	**Inverse R-ch Gamma Corrected Value 212** Default Value: d400h Format: U16
	47:32	**Inverse Pixel Value 212** Default Value: d400h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 212** Default Value: d400h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 212** Default Value: d400h Format: U16
850..851	63:48	**Forward R-ch Gamma Corrected Value 212** Default Value: d400h Format: U16
	47:32	**Forward Pixel Value 212** Default Value: d400h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 212** Default Value: d400h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 212** Default Value: d400h Format: U16
852..853	63:48	**Inverse R-ch Gamma Corrected Value 213** Default Value: d500h Format: U16
	47:32	**Inverse Pixel Value 213** Default Value: d500h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 213** Default Value: d500h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 213** Default Value: d500h Format: U16
854..855	63:48	**Forward R-ch Gamma Corrected Value 213** Default Value: d500h Format: U16
	47:32	**Forward Pixel Value 213** Default Value: d500h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 213** Default Value: d500h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 213** Default Value: d500h Format: U16
856..857	63:48	**Inverse R-ch Gamma Corrected Value 214** Default Value: d600h Format: U16
	47:32	**Inverse Pixel Value 214** Default Value: d600h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 214** Default Value: d600h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 214** Default Value: d600h Format: U16
858..859	63:48	**Forward R-ch Gamma Corrected Value 214** Default Value: d600h Format: U16
	47:32	**Forward Pixel Value 214** Default Value: d600h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 214** Default Value: d600h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 214** Default Value: d600h Format: U16
860..861	63:48	**Inverse R-ch Gamma Corrected Value 215** Default Value: d700h Format: U16
	47:32	**Inverse Pixel Value 215** Default Value: d700h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 215** Default Value: d700h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 215** Default Value: d700h Format: U16
862..863	63:48	**Forward R-ch Gamma Corrected Value 215** Default Value: d700h Format: U16
	47:32	**Forward Pixel Value 215** Default Value: d700h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 215** Default Value: d700h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 215** Default Value: d700h Format: U16
864..865	63:48	**Inverse R-ch Gamma Corrected Value 216** Default Value: d800h Format: U16
	47:32	**Inverse Pixel Value 216** Default Value: d800h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 216** Default Value: d800h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 216** Default Value: d800h Format: U16
866..867	63:48	**Forward R-ch Gamma Corrected Value 216** Default Value: d800h Format: U16
	47:32	**Forward Pixel Value 216** Default Value: d800h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 216** Default Value: d800h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 216** Default Value: d800h Format: U16
868..869	63:48	**Inverse R-ch Gamma Corrected Value 217** Default Value: d900h Format: U16
	47:32	**Inverse Pixel Value 217** Default Value: d900h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 217** Default Value: d900h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 217** Default Value: d900h Format: U16
870..871	63:48	**Forward R-ch Gamma Corrected Value 217** Default Value: d900h Format: U16
	47:32	**Forward Pixel Value 217** Default Value: d900h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 217** Default Value: d900h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 217** Default Value: d900h Format: U16
872..873	63:48	**Inverse R-ch Gamma Corrected Value 218** Default Value: da00h Format: U16
	47:32	**Inverse Pixel Value 218** Default Value: da00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 218** Default Value: da00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 218** Default Value: da00h Format: U16
874..875	63:48	**Forward R-ch Gamma Corrected Value 218** Default Value: da00h Format: U16
	47:32	**Forward Pixel Value 218** Default Value: da00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 218** Default Value: da00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 218** Default Value: da00h Format: U16
876..877	63:48	**Inverse R-ch Gamma Corrected Value 219** Default Value: db00h Format: U16
	47:32	**Inverse Pixel Value 219** Default Value: db00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 219** Default Value: db00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 219** Default Value: db00h Format: U16
878..879	63:48	**Forward R-ch Gamma Corrected Value 219** Default Value: db00h Format: U16
	47:32	**Forward Pixel Value 219** Default Value: db00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 219** Default Value: db00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 219** Default Value: db00h Format: U16
880..881	63:48	**Inverse R-ch Gamma Corrected Value 220** Default Value: dc00h Format: U16
	47:32	**Inverse Pixel Value 220** Default Value: dc00h Format: U16

Gamut_Expansion_Gamma_Correction

	31:16	**Inverse B-ch Gamma Corrected Value 220** Default Value: dc00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 220** Default Value: dc00h Format: U16
882..883	63:48	**Forward R-ch Gamma Corrected Value 220** Default Value: dc00h Format: U16
	47:32	**Forward Pixel Value 220** Default Value: dc00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 220** Default Value: dc00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 220** Default Value: dc00h Format: U16
884..885	63:48	**Inverse R-ch Gamma Corrected Value 221** Default Value: dd00h Format: U16
	47:32	**Inverse Pixel Value 221** Default Value: dd00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 221** Default Value: dd00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 221** Default Value: dd00h Format: U16
886..887	63:48	**Forward R-ch Gamma Corrected Value 221** Default Value: dd00h Format: U16
	47:32	**Forward Pixel Value 221** Default Value: dd00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 221** Default Value: dd00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 221** Default Value: dd00h Format: U16
888..889	63:48	**Inverse R-ch Gamma Corrected Value 222** Default Value: de00h Format: U16
	47:32	**Inverse Pixel Value 222** Default Value: de00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 222** Default Value: de00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 222** Default Value: de00h Format: U16
890..891	63:48	**Forward R-ch Gamma Corrected Value 222** Default Value: de00h Format: U16
	47:32	**Forward Pixel Value 222** Default Value: de00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 222** Default Value: de00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 222** Default Value: de00h Format: U16
892..893	63:48	**Inverse R-ch Gamma Corrected Value 223** Default Value: df00h Format: U16
	47:32	**Inverse Pixel Value 223** Default Value: df00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 223** Default Value: df00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 223** Default Value: df00h Format: U16
894..895	63:48	**Forward R-ch Gamma Corrected Value 223** Default Value: df00h Format: U16
	47:32	**Forward Pixel Value 223** Default Value: df00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 223** Default Value: df00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 223** Default Value: df00h Format: U16
896..897	63:48	**Inverse R-ch Gamma Corrected Value 224** Default Value: e000h Format: U16
	47:32	**Inverse Pixel Value 224** Default Value: e000h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 224** Default Value: e000h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 224** Default Value: e000h Format: U16
898..899	63:48	**Forward R-ch Gamma Corrected Value 224** Default Value: e000h Format: U16
	47:32	**Forward Pixel Value 224** Default Value: e000h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 224** Default Value: e000h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 224** Default Value: e000h Format: U16
900..901	63:48	**Inverse R-ch Gamma Corrected Value 225** Default Value: e100h Format: U16
	47:32	**Inverse Pixel Value 225** Default Value: e100h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 225** Default Value: e100h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 225** Default Value: e100h Format: U16
902..903	63:48	**Forward R-ch Gamma Corrected Value 225** Default Value: e100h Format: U16
	47:32	**Forward Pixel Value 225** Default Value: e100h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 225** Default Value: e100h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 225** Default Value: e100h Format: U16
904..905	63:48	**Inverse R-ch Gamma Corrected Value 226** Default Value: e200h Format: U16
	47:32	**Inverse Pixel Value 226** Default Value: e200h Format: U16

Gamut_Expansion_Gamma_Correction

	31:16	**Inverse B-ch Gamma Corrected Value 226** Default Value: e200h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 226** Default Value: e200h Format: U16
906..907	63:48	**Forward R-ch Gamma Corrected Value 226** Default Value: e200h Format: U16
	47:32	**Forward Pixel Value 226** Default Value: e200h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 226** Default Value: e200h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 226** Default Value: e200h Format: U16
908..909	63:48	**Inverse R-ch Gamma Corrected Value 227** Default Value: e300h Format: U16
	47:32	**Inverse Pixel Value 227** Default Value: e300h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 227** Default Value: e300h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 227** Default Value: e300h Format: U16
910..911	63:48	**Forward R-ch Gamma Corrected Value 227** Default Value: e300h Format: U16
	47:32	**Forward Pixel Value 227** Default Value: e300h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 227** Default Value: e300h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 227** Default Value: e300h Format: U16
912..913	63:48	**Inverse R-ch Gamma Corrected Value 228** Default Value: e400h Format: U16
	47:32	**Inverse Pixel Value 228** Default Value: e400h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 228** Default Value: e400h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 228** Default Value: e400h Format: U16
914..915	63:48	**Forward R-ch Gamma Corrected Value 228** Default Value: e400h Format: U16
	47:32	**Forward Pixel Value 228** Default Value: e400h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 228** Default Value: e400h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 228** Default Value: e400h Format: U16
916..917	63:48	**Inverse R-ch Gamma Corrected Value 229** Default Value: e500h Format: U16
	47:32	**Inverse Pixel Value 229** Default Value: e500h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 229** Default Value: e500h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 229** Default Value: e500h Format: U16
918..919	63:48	**Forward R-ch Gamma Corrected Value 229** Default Value: e500h Format: U16
	47:32	**Forward Pixel Value 229** Default Value: e500h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 229** Default Value: e500h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 229** Default Value: e500h Format: U16
920..921	63:48	**Inverse R-ch Gamma Corrected Value 230** Default Value: e600h Format: U16
	47:32	**Inverse Pixel Value 230** Default Value: e600h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 230** Default Value: e600h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 230** Default Value: e600h Format: U16
922..923	63:48	**Forward R-ch Gamma Corrected Value 230** Default Value: e600h Format: U16
	47:32	**Forward Pixel Value 230** Default Value: e600h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 230** Default Value: e600h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 230** Default Value: e600h Format: U16
924..925	63:48	**Inverse R-ch Gamma Corrected Value 231** Default Value: e700h Format: U16
	47:32	**Inverse Pixel Value 231** Default Value: e700h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 231** Default Value: e700h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 231** Default Value: e700h Format: U16
926..927	63:48	**Forward R-ch Gamma Corrected Value 231** Default Value: e700h Format: U16
	47:32	**Forward Pixel Value 231** Default Value: e700h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 231** Default Value: e700h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 231** Default Value: e700h Format: U16
928..929	63:48	**Inverse R-ch Gamma Corrected Value 232** Default Value: e800h Format: U16
	47:32	**Inverse Pixel Value 232** Default Value: e800h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 232** Default Value: e800h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 232** Default Value: e800h Format: U16
930..931	63:48	**Forward R-ch Gamma Corrected Value 232** Default Value: e800h Format: U16
	47:32	**Forward Pixel Value 232** Default Value: e800h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 232** Default Value: e800h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 232** Default Value: e800h Format: U16
932..933	63:48	**Inverse R-ch Gamma Corrected Value 233** Default Value: e900h Format: U16
	47:32	**Inverse Pixel Value 233** Default Value: e900h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 233** Default Value: e900h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 233** Default Value: e900h Format: U16
934..935	63:48	**Forward R-ch Gamma Corrected Value 233** Default Value: e900h Format: U16
	47:32	**Forward Pixel Value 233** Default Value: e900h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 233** Default Value: e900h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 233** Default Value: e900h Format: U16
936..937	63:48	**Inverse R-ch Gamma Corrected Value 234** Default Value: ea00h Format: U16
	47:32	**Inverse Pixel Value 234** Default Value: ea00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 234** Default Value: ea00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 234** Default Value: ea00h Format: U16
938..939	63:48	**Forward R-ch Gamma Corrected Value 234** Default Value: ea00h Format: U16
	47:32	**Forward Pixel Value 234** Default Value: ea00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 234** Default Value: ea00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 234** Default Value: ea00h Format: U16
940..941	63:48	**Inverse R-ch Gamma Corrected Value 235** Default Value: eb00h Format: U16
	47:32	**Inverse Pixel Value 235** Default Value: eb00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 235** Default Value: eb00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 235** Default Value: eb00h Format: U16
942..943	63:48	**Forward R-ch Gamma Corrected Value 235** Default Value: eb00h Format: U16
	47:32	**Forward Pixel Value 235** Default Value: eb00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 235** Default Value: eb00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 235** Default Value: eb00h Format: U16
944..945	63:48	**Inverse R-ch Gamma Corrected Value 236** Default Value: ec00h Format: U16
	47:32	**Inverse Pixel Value 236** Default Value: ec00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 236** Default Value: ec00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 236** Default Value: ec00h Format: U16
946..947	63:48	**Forward R-ch Gamma Corrected Value 236** Default Value: ec00h Format: U16
	47:32	**Forward Pixel Value 236** Default Value: ec00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 236** Default Value: ec00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 236** Default Value: ec00h Format: U16
948..949	63:48	**Inverse R-ch Gamma Corrected Value 237** Default Value: ed00h Format: U16
	47:32	**Inverse Pixel Value 237** Default Value: ed00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 237** Default Value: ed00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 237** Default Value: ed00h Format: U16
950..951	63:48	**Forward R-ch Gamma Corrected Value 237** Default Value: ed00h Format: U16
	47:32	**Forward Pixel Value 237** Default Value: ed00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 237** Default Value: ed00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 237** Default Value: ed00h Format: U16
952..953	63:48	**Inverse R-ch Gamma Corrected Value 238** Default Value: ee00h Format: U16
	47:32	**Inverse Pixel Value 238** Default Value: ee00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 238** Default Value: ee00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 238** Default Value: ee00h Format: U16
954..955	63:48	**Forward R-ch Gamma Corrected Value 238** Default Value: ee00h Format: U16
	47:32	**Forward Pixel Value 238** Default Value: ee00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 238** Default Value: ee00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 238** Default Value: ee00h Format: U16
956..957	63:48	**Inverse R-ch Gamma Corrected Value 239** Default Value: ef00h Format: U16
	47:32	**Inverse Pixel Value 239** Default Value: ef00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 239** Default Value: ef00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 239** Default Value: ef00h Format: U16
958..959	63:48	**Forward R-ch Gamma Corrected Value 239** Default Value: ef00h Format: U16
	47:32	**Forward Pixel Value 239** Default Value: ef00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 239** Default Value: ef00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 239** Default Value: ef00h Format: U16
960..961	63:48	**Inverse R-ch Gamma Corrected Value 240** Default Value: f000h Format: U16
	47:32	**Inverse Pixel Value 240** Default Value: f000h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 240** Default Value: f000h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 240** Default Value: f000h Format: U16
962..963	63:48	**Forward R-ch Gamma Corrected Value 240** Default Value: f000h Format: U16
	47:32	**Forward Pixel Value 240** Default Value: f000h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 240** Default Value: f000h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 240** Default Value: f000h Format: U16
964..965	63:48	**Inverse R-ch Gamma Corrected Value 241** Default Value: f100h Format: U16
	47:32	**Inverse Pixel Value 241** Default Value: f100h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 241** Default Value: f100h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 241** Default Value: f100h Format: U16
966..967	63:48	**Forward R-ch Gamma Corrected Value 241** Default Value: f100h Format: U16
	47:32	**Forward Pixel Value 241** Default Value: f100h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 241** Default Value: f100h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 241** Default Value: f100h Format: U16
968..969	63:48	**Inverse R-ch Gamma Corrected Value 242** Default Value: f200h Format: U16
	47:32	**Inverse Pixel Value 242** Default Value: f200h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 242** Default Value: f200h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 242** Default Value: f200h Format: U16
970..971	63:48	**Forward R-ch Gamma Corrected Value 242** Default Value: f200h Format: U16
	47:32	**Forward Pixel Value 242** Default Value: f200h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 242** Default Value: f200h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 242** Default Value: f200h Format: U16
972..973	63:48	**Inverse R-ch Gamma Corrected Value 243** Default Value: f300h Format: U16
	47:32	**Inverse Pixel Value 243** Default Value: f300h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 243** Default Value: f300h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 243** Default Value: f300h Format: U16
974..975	63:48	**Forward R-ch Gamma Corrected Value 243** Default Value: f300h Format: U16
	47:32	**Forward Pixel Value 243** Default Value: f300h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 243** Default Value: f300h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 243** Default Value: f300h Format: U16
976..977	63:48	**Inverse R-ch Gamma Corrected Value 244** Default Value: f400h Format: U16
	47:32	**Inverse Pixel Value 244** Default Value: f400h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 244** Default Value: f400h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 244** Default Value: f400h Format: U16
978..979	63:48	**Forward R-ch Gamma Corrected Value 244** Default Value: f400h Format: U16
	47:32	**Forward Pixel Value 244** Default Value: f400h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 244** Default Value: f400h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 244** Default Value: f400h Format: U16
980..981	63:48	**Inverse R-ch Gamma Corrected Value 245** Default Value: f500h Format: U16
	47:32	**Inverse Pixel Value 245** Default Value: f500h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 245** Default Value: f500h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 245** Default Value: f500h Format: U16
982..983	63:48	**Forward R-ch Gamma Corrected Value 245** Default Value: f500h Format: U16
	47:32	**Forward Pixel Value 245** Default Value: f500h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 245** Default Value: f500h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 245** Default Value: f500h Format: U16
984..985	63:48	**Inverse R-ch Gamma Corrected Value 246** Default Value: f600h Format: U16
	47:32	**Inverse Pixel Value 246** Default Value: f600h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 246** Default Value: f600h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 246** Default Value: f600h Format: U16
986..987	63:48	**Forward R-ch Gamma Corrected Value 246** Default Value: f600h Format: U16
	47:32	**Forward Pixel Value 246** Default Value: f600h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 246** Default Value: f600h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 246** Default Value: f600h Format: U16
988..989	63:48	**Inverse R-ch Gamma Corrected Value 247** Default Value: f700h Format: U16
	47:32	**Inverse Pixel Value 247** Default Value: f700h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 247** Default Value: f700h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 247** Default Value: f700h Format: U16
990..991	63:48	**Forward R-ch Gamma Corrected Value 247** Default Value: f700h Format: U16
	47:32	**Forward Pixel Value 247** Default Value: f700h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 247** Default Value: f700h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 247** Default Value: f700h Format: U16
992..993	63:48	**Inverse R-ch Gamma Corrected Value 248** Default Value: f800h Format: U16
	47:32	**Inverse Pixel Value 248** Default Value: f800h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 248** Default Value: f800h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 248** Default Value: f800h Format: U16
994..995	63:48	**Forward R-ch Gamma Corrected Value 248** Default Value: f800h Format: U16
	47:32	**Forward Pixel Value 248** Default Value: f800h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 248** Default Value: f800h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 248** Default Value: f800h Format: U16
996..997	63:48	**Inverse R-ch Gamma Corrected Value 249** Default Value: f900h Format: U16
	47:32	**Inverse Pixel Value 249** Default Value: f900h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 249** Default Value: f900h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 249** Default Value: f900h Format: U16
998..999	63:48	**Forward R-ch Gamma Corrected Value 249** Default Value: f900h Format: U16
	47:32	**Forward Pixel Value 249** Default Value: f900h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 249** Default Value: f900h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 249** Default Value: f900h Format: U16
1000..1001	63:48	**Inverse R-ch Gamma Corrected Value 250** Default Value: fa00h Format: U16
	47:32	**Inverse Pixel Value 250** Default Value: fa00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 250** Default Value: fa00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 250** Default Value: fa00h Format: U16
1002..1003	63:48	**Forward R-ch Gamma Corrected Value 250** Default Value: fa00h Format: U16
	47:32	**Forward Pixel Value 250** Default Value: fa00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 250** Default Value: fa00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 250** Default Value: fa00h Format: U16
1004..1005	63:48	**Inverse R-ch Gamma Corrected Value 251** Default Value: fb00h Format: U16
	47:32	**Inverse Pixel Value 251** Default Value: fb00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 251** Default Value: fb00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 251** Default Value: fb00h Format: U16
1006..1007	63:48	**Forward R-ch Gamma Corrected Value 251** Default Value: fb00h Format: U16
	47:32	**Forward Pixel Value 251** Default Value: fb00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 251** Default Value: fb00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 251** Default Value: fb00h Format: U16
1008..1009	63:48	**Inverse R-ch Gamma Corrected Value 252** Default Value: fc00h Format: U16
	47:32	**Inverse Pixel Value 252** Default Value: fc00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 252** Default Value: fc00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 252** Default Value: fc00h Format: U16
1010..1011	63:48	**Forward R-ch Gamma Corrected Value 252** Default Value: fc00h Format: U16
	47:32	**Forward Pixel Value 252** Default Value: fc00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 252** Default Value: fc00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 252** Default Value: fc00h Format: U16
1012..1013	63:48	**Inverse R-ch Gamma Corrected Value 253** Default Value: fd00h Format: U16
	47:32	**Inverse Pixel Value 253** Default Value: fd00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Inverse B-ch Gamma Corrected Value 253** Default Value: fd00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 253** Default Value: fd00h Format: U16
1014..1015	63:48	**Forward R-ch Gamma Corrected Value 253** Default Value: fd00h Format: U16
	47:32	**Forward Pixel Value 253** Default Value: fd00h Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 253** Default Value: fd00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 253** Default Value: fd00h Format: U16
1016..1017	63:48	**Inverse R-ch Gamma Corrected Value 254** Default Value: fe00h Format: U16
	47:32	**Inverse Pixel Value 254** Default Value: fe00h Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 254** Default Value: fe00h Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 254** Default Value: fe00h Format: U16
1018..1019	63:48	**Forward R-ch Gamma Corrected Value 254** Default Value: fe00h Format: U16
	47:32	**Forward Pixel Value 254** Default Value: fe00h Format: U16

Gamut_Expansion_Gamma_Correction		
	31:16	**Forward B-ch Gamma Corrected Value 254** Default Value: fe00h Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 254** Default Value: fe00h Format: U16
1020..1021	63:48	**Inverse R-ch Gamma Corrected Value 255** Default Value: ffffh Format: U16
	47:32	**Inverse Pixel Value 255** Default Value: ffffh Format: U16
	31:16	**Inverse B-ch Gamma Corrected Value 255** Default Value: ffffh Format: U16
	15:0	**Inverse G-ch Gamma Corrected Value 255** Default Value: ffffh Format: U16
1022..1023	63:48	**Forward R-ch Gamma Corrected Value 255** Default Value: ffffh Format: U16
	47:32	**Forward Pixel Value 255** Default Value: ffffh Format: U16
	31:16	**Forward B-ch Gamma Corrected Value 255** Default Value: ffffh Format: U16
	15:0	**Forward G-ch Gamma Corrected Value 255** Default Value: ffffh Format: U16

GGTT Entry

GGTT_ENTRY - GGTT Entry		
Size (in bits):	64	
Default Value:	0x00000000, 0x00000000	
The GPU accesses GGTT table entries as uncacheable.		
DWord	**Bit**	**Description**
0	63:54	**Reserved**
	53	**Reserved**
	52	**Reserved** Access: RO Format: MBZ
	51:46	**Reserved**
	45:12	**Address** Physical address of 4KB memory page referenced by this entry.
	11:5	**Reserved**
	4:2	**Function Number** The Number of the Function to which this page has been assigned. This field is ignored unless HW Graphics Virtualization (SRIOV) is enabled.
	1	**Ignored** *Ignored (h/w does not care about values behind ignored registers)*
	0	**Present** When set to 1, indicates that this Entry is Valid, and the corresponding page is Present in physical memory

GraphicsAddress63-0

GA63_0 - GraphicsAddress63-0		
Size (in bits):	64	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0..1	63:57	**Reserved** Access: RO Format: MBZ
	56:48	**Reserved** Access: RO Format: MBZ
	47:0	**GraphicsAddress47-0** Format: GraphicsAddress[47:0] GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and must be zero. For the specified product generations, use 48-bit addressing.

GraphicsAddress63-1

GA63_1 - GraphicsAddress63-1		
Size (in bits):	63	
Default Value:	0x00000000, 0x00000000	
This structure is intended to define the upper bits of the GraphicsAddress, when bit 0 is already defined in the referring register. So bit 0 of this structure should correspond to bit 1 of the full GraphicsAddress.		
DWord	**Bit**	**Description**
0..1	62:56	**Reserved** Access: RO Format: MBZ
	55:47	**Reserved** Access: RO Format: MBZ
	46:0	**GraphicsAddress47-1** Format: GraphicsAddress[47:1] GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and must be zero. Some GraphicsAddress fields only specify the upper address bits. For example GraphicsAddress[47:12] would be a 4KB page address. Bits 47:1 of a 48-bit GraphicsAddress. Look for bit 0 definition in the referring register.

GraphicsAddress63-2

GA63_2 - GraphicsAddress63-2		
Size (in bits):	62	
Default Value:	0x00000000, 0x00000000	
This structure is intended to define the upper bits of the GraphicsAddress, when bits 1:0 are already defined in the referring register. So bit 0 of this structure should correspond to bit 2 of the full GraphicsAddress.		
DWord	**Bit**	**Description**
0..1	61:55	**Reserved** Access: RO Format: MBZ
	54:46	**Reserved** Access: RO Format: MBZ
	45:0	**GraphicsAddress47-2** Format: GraphicsAddress[47:2] GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and must be zero. Some GraphicsAddress fields only specify the upper address bits. For example GraphicsAddress[47:12] would be a 4KB page address. Bits 47:2 of a 48-bit GraphicsAddress. Look for the definition of bits 1:0 in the referring register.

GraphicsAddress63-3

GA63_3 - GraphicsAddress63-3		
Size (in bits):	61	
Default Value:	0x00000000, 0x00000000	
This structure is intended to define the upper bits of the GraphicsAddress, when bits 2:0 are already defined in the referring register. So bit 0 of this structure should correspond to bit 3 of the full GraphicsAddress.		
DWord	**Bit**	**Description**
0..1	60:54	**Reserved** Access: RO Format: MBZ
	53:45	**Reserved** Access: RO Format: MBZ
	44:0	**GraphicsAddress47-3** Format: GraphicsAddress[47:3] GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and must be zero. Some GraphicsAddress fields only specify the upper address bits. For example GraphicsAddress[47:12] would be a 4KB page address. Bits 47:2 of a 48-bit GraphicsAddress. Look for the definition of bits 2:0 in the referring register.

GraphicsAddress63-4

GA63_4 - GraphicsAddress63-4		
Size (in bits):	60	
Default Value:	0x00000000, 0x00000000	
This structure is intended to define the upper bits of the GraphicsAddress, when bits 3:0 are already defined in the referring register. So bit 0 of this structure should correspond to bit 4 of the full GraphicsAddress.		
DWord	**Bit**	**Description**
0..1	59:53	**Reserved** Access: RO Format: MBZ
	52:44	**Reserved** Access: RO Format: MBZ
	43:0	**GraphicsAddress47-4** Format: GraphicsAddress[47:4] GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and must be zero. Some GraphicsAddress fields only specify the upper address bits. For example GraphicsAddress[47:12] would be a 4KB page address. Bits 47:4 of a 48-bit GraphicsAddress. Look for the definition of bits 3:0 in the referring register.

GraphicsAddress63-5

GA63_5 - GraphicsAddress63-5		
Size (in bits):	59	
Default Value:	0x00000000, 0x00000000	
This structure is intended to define the upper bits of the GraphicsAddress, when bits 4:0 are already defined in the referring register. So bit 0 of this structure should correspond to bit 5 of the full GraphicsAddress.		
DWord	**Bit**	**Description**
0..1	58:52	**Reserved** Access: RO Format: MBZ
	51:43	**Reserved** Access: RO Format: MBZ
	42:0	**GraphicsAddress47-5** Format: GraphicsAddress[47:5] GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and must be zero. Some GraphicsAddress fields only specify the upper address bits. For example GraphicsAddress[47:12] would be a 4KB page address. Bits 47:5 of a 48-bit GraphicsAddress. Look for the definition of bits 4:0 in the referring register.

GraphicsAddress63-6

GA63_6 - GraphicsAddress63-6		
Size (in bits):	58	
Default Value:	0x00000000, 0x00000000	
This structure is intended to define the upper bits of the GraphicsAddress, when bits 5:0 are already defined in the referring register. So bit 0 of this structure should correspond to bit 6 of the full GraphicsAddress.		
DWord	**Bit**	**Description**
0..1	57:51	**Reserved** Access: RO Format: MBZ
	50:42	**Reserved** Access: RO Format: MBZ
	41:0	**GraphicsAddress47-6** Format: GraphicsAddress[47:6] GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and must be zero. Some GraphicsAddress fields only specify the upper address bits. For example GraphicsAddress[47:12] would be a 4KB page address. Bits 47:6 of a 48-bit GraphicsAddress. Look for the definition of bits 5:0 in the referring register.

GraphicsAddress63-12

GA63_12 - GraphicsAddress63-12

Size (in bits): 52

Default Value: 0x00000000, 0x00000000

This structure is intended to define the upper bits of the GraphicsAddress, when bits 11:0 are already defined in the referring register. So bit 0 of this structure should correspond to bit 12 of the full GraphicsAddress.

DWord	Bit	Description
0..1	51:45	**Reserved** Access: RO Format: MBZ
	44:36	**Reserved** Access: RO Format: MBZ
	35:0	**GraphicsAddress47-12** Format: GraphicsAddress[47:12] GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and must be zero. Some GraphicsAddress fields only specify the upper address bits. For example GraphicsAddress[47:12] would be a 4KB page address. Bits 47:12 of a 48-bit GraphicsAddress. Look for the definition of bits 11:0 in the referring register.

GTC Interrupt Bit Definition

GTC Interrupt Bit Definition		
Size (in bits):	32	
Default Value:	0x00000000	
The GTC Interrupt Registers all share the same bit definitions from this table.		
DWord	**Bit**	**Description**
0	31	**GTC Lock Loss** GTC has lost lock with a remote GTC sink.The difference between the local and remote GTC has exceeded programmed threshold.
	30:22	**Reserved** Access: RO Format: MBZ
	21	**GTC Aux Rx Error USBC6** An aux channel error occurred during GTC transfer with remote GTC sink attached to this port.
	20	**GTC Update Complete USBC6** A hardware initiated GTC update has completed with a sink attached to this port.
	19	**GTC Aux Rx Error USBC5** An aux channel error occurred during GTC transfer with remote GTC sink attached to this port.
	18	**GTC Update Complete USBC5** A hardware initiated GTC update has completed with a sink attached to this port.
	17	**GTC Aux Rx Error USBC4** An aux channel error occurred during GTC transfer with remote GTC sink attached to this port.
	16	**GTC Update Complete USBC4** A hardware initiated GTC update has completed with a sink attached to this port.
	15	**GTC Aux Rx Error USBC3** An aux channel error occurred during GTC transfer with remote GTC sink attached to this port.
	14	**GTC Update Complete USBC3** A hardware initiated GTC update has completed with a sink attached to this port.
	13	**GTC Aux Rx Error USBC2** An aux channel error occurred during GTC transfer with remote GTC sink attached to this port.
	12	**GTC Update Complete USBC2** A hardware initiated GTC update has completed with a sink attached to this port.
	11	**GTC Aux Rx Error USBC1** An aux channel error occurred during GTC transfer with remote GTC sink attached to this port.
	10	**GTC Update Complete USBC1** A hardware initiated GTC update has completed with a sink attached to this port.
	9:6	**Reserved** Access: RO Format: MBZ

GTC Interrupt Bit Definition		
	5	**GTC Aux Rx Error DDIC** An aux channel error occurred during GTC transfer with remote GTC sink attached to this port.
	4	**GTC Update Complete DDIC** A hardware initiated GTC update has completed with a sink attached to this port.
	3	**GTC Aux Rx Error DDIB** An aux channel error occurred during GTC transfer with remote GTC sink attached to this port.
	2	**GTC Update Complete DDIB** A hardware initiated GTC update has completed with a sink attached to this port.
	1	**GTC Aux Rx Error DDIA** An aux channel error occurred during GTC transfer with remote GTC sink attached to this port.
	0	**GTC Update Complete DDIA** A hardware initiated GTC update has completed with a sink attached to this port.

GTPM Interrupt Vector

GTPM_INTR_VEC - GTPM Interrupt Vector		
Size (in bits):	16	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	15:14	**Reserved** Access: RO Format: MBZ
	13	**Unslice Frequency Control Up Interrupt** This interrupt is no longer used
	12	**Unslice Frequency Control Down Interrupt** This interrupt is no longer used
	11	**NFADFL Frequency Up Interrupt** This interrupt is no longer used
	10	**NFADFL Frequency Down Interrupt** This interrupt is no longer used
	9	**ARAT Interrupt** Always Running Apic Timer Interrupt. This interrupt is sent by GPM to GuCmicro-controller (for scheduling purposes). Host SW does not require this interrupt (so this interrupt is not sent to G-unit).
	8	**GTPM Engines Idle Interrupt**
	7	**GTPM Uncore to Core Trap Interrupt**
	6	**GTPM Render Frequency Downwards Timeout During RC6 Interrupt** This interrupt is no longer used
	5	**GTPM Render P-State Up Threshold Interrupt** This interrupt is no longer used
	4	**GTPM Render P-State Down Threshold Interrupt** This interrupt is no longer used
	3	**Spare 3** This interrupt is no longer used
	2	**GTPM Render Geyserville Up Evaluation Interval Interrupt** This interrupt is no longer used
	1	**GTPM Render Geyserville Down Evaluation Interval Interrupt** This interrupt is no longer used
	0	**Reserved** Access: RO Format: MBZ

GUC Interrupt Vector

GUC_INTR_VEC - GUC Interrupt Vector		
Size (in bits):	16	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	15	**GUC Interrupt to Host**
	14	**GUC Execution Error**
	13	**GUC Display Event Received**
	12	**GUC Semaphore Signaled**
	11	**IOMMU Sent Message to GUC**
	10	**GUC Doorbell Rang**
	9	**GUC DMA Done**
	8	**GUC Fatal Error**
	7	**GUC Notification Error**
	6	**GUC SW interrupt 6**
	5	**GUC SW interrupt 5**
	4	**GUC SW interrupt 4**
	3	**GUC SW interrupt 3**
	2	**GUC SW interrupt 2**
	1	**GUC SW interrupt 1**
	0	**GUC SW interrupt 0**

G-Unit Interrupt Vector

GUNIT_INTR_VEC - G-Unit Interrupt Vector		
Size (in bits): 16 Default Value: 0x00000000		
This interrupt is delivered to GuC to indicate a request for Function Level Reset of a specific Virtual Function in Virtualization mode. GuC uses this request to clean up internal queues.		
DWord	**Bit**	**Description**
0	15:8	**Reserved** Access: RO Format: MBZ
	7	**Function Level Reset Request For Virtual Function 7**
	6	**Function Level Reset Request For Virtual Function 6**
	5	**Function Level Reset Request For Virtual Function 5**
	4	**Function Level Reset Request For Virtual Function 4**
	3	**Function Level Reset Request For Virtual Function 3**
	2	**Function Level Reset Request For Virtual Function 2**
	1	**Function Level Reset Request For Virtual Function 1**
	0	**Function Level Reset Request For Virtual Function 0 (Physical Function)**

Half Precision Dual Source SIMD8 Message Data Payload Register

MDPR_DSH_SIMD8 - Half Precision Dual Source SIMD8 Message Data Payload Register

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:16	**Src0 Data1** Format: IEEE_FLOAT16 Specifies the source 0 slot 1 data in this payload register
	15:0	**Src0 Data0** Format: IEEE_FLOAT16 Specifies the source 0 slot 0 data in this payload register
1	31:16	**Src0 Data3** Format: IEEE_FLOAT16 Specifies the source 0 slot 3 data in this payload register
	15:0	**Src0 Data2** Format: IEEE_FLOAT16 Specifies the source 0 slot 2 data in this payload register
2	31:16	**Src0 Data5** Format: IEEE_FLOAT16 Specifies the source 0 slot 5 data in this payload register
	15:0	**Src0 Data4** Format: IEEE_FLOAT16 Specifies the source 0 slot 4 data in this payload register
3	31:16	**Src0 Data7** Format: IEEE_FLOAT16 Specifies the source 0 slot 7 data in this payload register
	15:0	**Src0 Data6** Format: IEEE_FLOAT16 Specifies the source 0 slot 6 data in this payload register

MDPR_DSH_SIMD8 - Half Precision Dual Source SIMD8 Message Data Payload Register		
4	31:16	**Src1 Data1** Format: IEEE_FLOAT16 Specifies the source 1 slot 1 data in this payload register
	15:0	**Src1 Data0** Format: IEEE_FLOAT16 Specifies the source 1 slot 0 data in this payload register
5	31:16	**Src1 Data3** Format: IEEE_FLOAT16 Specifies the source 1 slot 3 data in this payload register
	15:0	**Src1 Data2** Format: IEEE_FLOAT16 Specifies the source 1 slot 2 data in this payload register
6	31:16	**Src1 Data5** Format: IEEE_FLOAT16 Specifies the source 1 slot 5 data in this payload register
	15:0	**Src1 Data4** Format: IEEE_FLOAT16 Specifies the source 1 slot 4 data in this payload register
7	31:16	**Src1 Data7** Format: IEEE_FLOAT16 Specifies the source 1 slot 7 data in this payload register
	15:0	**Src1 Data6** Format: IEEE_FLOAT16 Specifies the source 1 slot 6 data in this payload register

Half Precision OM Replicated SIMD16 Render Target Data Payload

MDP_RTWH_M16REP - Half Precision OM Replicated SIMD16 Render Target Data Payload

Size (in bits):	512
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**oMask** Format: MDPR_OMASK Slots [15:0] oMask
1.0-1.7	255:0	**RGBA** Format: MDPR_H_RGBA RGBA for all slots [15:0]

Half Precision OM S0A SIMD8 Render Target Data Payload

MDP_RTWH_MA8 - Half Precision OM S0A SIMD8 Render Target Data Payload

Size (in bits): 1536

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Source 0 Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Source 0 Alpha
1.0-1.7	255:0	**oMask** Format: MDPR_OMASK Slots [7:0] oMask. Upper half ignored.
2.0-2.7	255:0	**Red** Format: MDPR_H_SIMD8 Slots [7:0] Red
3.0-3.7	255:0	**Green** Format: MDPR_H_SIMD8 Slots [7:0] Green
4.0-4.7	255:0	**Blue** Format: MDPR_H_SIMD8 Slots [7:0] Blue
5.0-5.7	255:0	**Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Alpha

Half Precision OM S0A SIMD16 Render Target Data Payload

MDP_RTWH_MA16 - Half Precision OM S0A SIMD16 Render Target Data Payload

Size (in bits): 1536

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Source 0 Alpha[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Source 0 Alpha
1.0-1.7	255:0	**oMask** Format: MDPR_OMASK Slots [15:0] oMask
2.0-2.7	255:0	**Red[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Red
3.0-3.7	255:0	**Green[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Green
4.0-4.7	255:0	**Blue[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Blue
5.0-5.7	255:0	**Alpha[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Alpha

Half Precision OM SIMD8 Dual Source Render Target Data Payload

MDP_RTWH_M8DS - Half Precision OM SIMD8 Dual Source Render Target Data Payload

Size (in bits): 1280

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**oMask** Format: MDPR_OMASK oMask for slots [7:0] and [15:8]. Operation selects upper or lower half.
1.0-1.7	255:0	**Red** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Red
2.0-2.7	255:0	**Green** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Green
3.0-3.7	255:0	**Blue** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Blue
4.0-4.7	255:0	**Alpha** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Alpha

Half Precision OM SIMD8 Render Target Data Payload

MDP_RTWH_M8 - Half Precision OM SIMD8 Render Target Data Payload

Size (in bits): 1280

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**oMask** Format: MDPR_OMASK Slots [7:0] oMask. Upper half ignored.
1.0-1.7	255:0	**Red** Format: MDPR_H_SIMD8 Slots [7:0] Red
2.0-2.7	255:0	**Green** Format: MDPR_H_SIMD8 Slots [7:0] Green
3.0-3.7	255:0	**Blue** Format: MDPR_H_SIMD8 Slots [7:0] Blue
4.0-4.7	255:0	**Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Alpha

Half Precision OM SIMD16 Render Target Data Payload

MDP_RTWH_M16 - Half Precision OM SIMD16 Render Target Data Payload

Size (in bits): 1280

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**oMask** Format: MDPR_OMASK Slots [15:0] oMask
1.0-1.7	255:0	**Red[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Red
2.0-2.7	255:0	**Green[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Green
3.0-3.7	255:0	**Blue[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Blue
4.0-4.7	255:0	**Alpha[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Alpha

Half Precision OS OM S0A SIMD8 Render Target Data Payload

MDP_RTWH_SMA8 - Half Precision OS OM S0A SIMD8 Render Target Data Payload

Size (in bits): 1792

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Source 0 Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Source 0 Alpha
1.0-1.7	255:0	**oMask** Format: MDPR_OMASK Slots [7:0] oMask. Upper half ignored.
2.0-2.7	255:0	**Red** Format: MDPR_H_SIMD8 Slots [7:0] Red
3.0-3.7	255:0	**Green** Format: MDPR_H_SIMD8 Slots [7:0] Green
4.0-4.7	255:0	**Blue** Format: MDPR_H_SIMD8 Slots [7:0] Blue
5.0-5.7	255:0	**Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Alpha

<table>
<tr><th colspan="3">MDP_RTWH_SMA8 - Half Precision OS OM S0A SIMD8 Render Target Data Payload</th></tr>
<tr><td>6.0-6.7</td><td>255:0</td><td>Stencil<table><tr><td>Format:</td><td>MDPR_STENCIL</td></tr></table>Slots [7:0] Stencil</td></tr>
</table>

Half Precision OS OM SIMD8 Dual Source Render Target Data Payload

MDP_RTWH_SM8DS - Half Precision OS OM SIMD8 Dual Source Render Target Data Payload		
Size (in bits):	1536	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**oMask** Format: **MDPR_OMASK** oMask for slots [7:0] and [15:8]. Operation selects upper or lower half.
1.0-1.7	255:0	**Red** Format: **MDPR_DSH_SIMD8** Slots[7:0] or [15:8] of Src0 and Src1 Red
2.0-2.7	255:0	**Green** Format: **MDPR_DSH_SIMD8** Slots[7:0] or [15:8] of Src0 and Src1 Green
3.0-3.7	255:0	**Blue** Format: **MDPR_DSH_SIMD8** Slots[7:0] or [15:8] of Src0 and Src1 Blue
4.0-4.7	255:0	**Alpha** Format: **MDPR_DSH_SIMD8** Slots[7:0] or [15:8] of Src0 and Src1 Alpha
5.0-5.7	255:0	**Stencil** Format: **MDPR_STENCIL** Slots [7:0] or [15:8] of Stencil

Half Precision OS OM SIMD8 Render Target Data Payload

MDP_RTWH_SM8 - Half Precision OS OM SIMD8 Render Target Data Payload

Size (in bits): 1536

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**oMask** Format: MDPR_OMASK Slots [7:0] oMask. Upper half ignored.
1.0-1.7	255:0	**Red** Format: MDPR_H_SIMD8 Slots [7:0] Red
2.0-2.7	255:0	**Green** Format: MDPR_H_SIMD8 Slots [7:0] Green
3.0-3.7	255:0	**Blue** Format: MDPR_H_SIMD8 Slots [7:0] Blue
4.0-4.7	255:0	**Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Alpha
5.0-5.7	255:0	**Stencil** Format: MDPR_STENCIL Slots [7:0] Stencil

Half Precision OS S0A SIMD8 Render Target Data Payload

MDP_RTWH_SA8 - Half Precision OS S0A SIMD8 Render Target Data Payload

Size (in bits): 1536

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Source 0 Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Source 0 Alpha
1.0-1.7	255:0	**Red** Format: MDPR_H_SIMD8 Slots [7:0] Red
2.0-2.7	255:0	**Green** Format: MDPR_H_SIMD8 Slots [7:0] Green
3.0-3.7	255:0	**Blue** Format: MDPR_H_SIMD8 Slots [7:0] Blue
4.0-4.7	255:0	**Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Alpha
5.0-5.7	255:0	**Stencil** Format: MDPR_STENCIL Slots [7:0] Stencil

Half Precision OS SIMD8 Dual Source Render Target Data Payload

MDP_RTWH_S8DS - Half Precision OS SIMD8 Dual Source Render Target Data Payload

Size (in bits): 1280

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Red** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Red
1.0-1.7	255:0	**Green** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Green
2.0-2.7	255:0	**Blue** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Blue
3.0-3.7	255:0	**Alpha** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Alpha
4.0-4.7	255:0	**Stencil** Format: MDPR_STENCIL Slots [7:0] or [15:8] of Stencil

Half Precision OS SIMD8 Render Target Data Payload

MDP_RTWH_S8 - Half Precision OS SIMD8 Render Target Data Payload

Size (in bits): 1280

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Red** Format: MDPR_H_SIMD8 Slots [7:0] Red
1.0-1.7	255:0	**Green** Format: MDPR_H_SIMD8 Slots [7:0] Green
2.0-2.7	255:0	**Blue** Format: MDPR_H_SIMD8 Slots [7:0] Blue
3.0-3.7	255:0	**Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Alpha
4.0-4.7	255:0	**Stencil** Format: MDPR_STENCIL Slots [7:0] Stencil

Half Precision OS SZ OM S0A SIMD8 Render Target Data Payload

MDP_RTWH_SZMA8 - Half Precision OS SZ OM S0A SIMD8 Render Target Data Payload

Size (in bits): 2048

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Source 0 Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Source 0 Alpha
1.0-1.7	255:0	**oMask** Format: MDPR_OMASK Slots [7:0] oMask. Upper half ignored.
2.0-2.7	255:0	**Red** Format: MDPR_H_SIMD8 Slots [7:0] Red
3.0-3.7	255:0	**Green** Format: MDPR_H_SIMD8 Slots [7:0] Green
4.0-4.7	255:0	**Blue** Format: MDPR_H_SIMD8 Slots [7:0] Blue
5.0-5.7	255:0	**Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Alpha

MDP_RTWH_SZMA8 - Half Precision OS SZ OM S0A SIMD8 Render Target Data Payload

6.0-6.7	255:0	**Source Depth** Format: **MDP_DW_SIMD8** Slots [7:0] Source Depth
7.0-7.7	255:0	**Stencil** Format: **MDPR_STENCIL** Slots [7:0] Stencil

Half Precision OS SZ OM SIMD8 Dual Source Render Target Data Payload

MDP_RTWH_SZM8DS - Half Precision OS SZ OM SIMD8 Dual Source Render Target Data Payload

Size (in bits): 1792

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**oMask** Format: MDPR_OMASK oMask for slots [7:0] and [15:8]. Operation selects upper or lower half.
1.0-1.7	255:0	**Red** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Red
2.0-2.7	255:0	**Green** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Green
3.0-3.7	255:0	**Blue** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Blue
4.0-4.7	255:0	**Alpha** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Alpha
5.0-5.7	255:0	**Source Depth** Format: MDP_DW_SIMD8 Slots [7:0] or [15:8] of Source Depth

MDP_RTWH_SZM8DS - Half Precision OS SZ OM SIMD8 Dual Source Render Target Data Payload		
6.0-6.7	255:0	**Stencil** Format: MDPR_STENCIL Slots [7:0] or [15:8] of Stencil

Half Precision OS SZ OM SIMD8 Render Target Data Payload

MDP_RTWH_SZM8 - Half Precision OS SZ OM SIMD8 Render Target Data Payload

Size (in bits):	1792
Default Value:	0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**oMask** Format: MDPR_OMASK Slots [7:0] oMask. Upper half ignored.
1.0-1.7	255:0	**Red** Format: MDPR_H_SIMD8 Slots [7:0] Red
2.0-2.7	255:0	**Green** Format: MDPR_H_SIMD8 Slots [7:0] Green
3.0-3.7	255:0	**Blue** Format: MDPR_H_SIMD8 Slots [7:0] Blue
4.0-4.7	255:0	**Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Alpha
5.0-5.7	255:0	**Source Depth** Format: MDP_DW_SIMD8 Slots [7:0] Source Depth

MDP_RTWH_SZM8 - Half Precision OS SZ OM SIMD8 Render Target Data Payload

6.0-6.7	255:0	**Stencil** Format: MDPR_STENCIL Slots [7:0] Stencil

Half Precision OS SZ S0A SIMD8 Render Target Data Payload

MDP_RTWH_SZA8 - Half Precision OS SZ S0A SIMD8 Render Target Data Payload

Size (in bits): 1792

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Source 0 Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Source 0 Alpha
1.0-1.7	255:0	**Red** Format: MDPR_H_SIMD8 Slots [7:0] Red
2.0-2.7	255:0	**Green** Format: MDPR_H_SIMD8 Slots [7:0] Green
3.0-3.7	255:0	**Blue** Format: MDPR_H_SIMD8 Slots [7:0] Blue
4.0-4.7	255:0	**Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Alpha
5.0-5.7	255:0	**Source Depth** Format: MDP_DW_SIMD8 Slots [7:0] Source Depth

MDP_RTWH_SZA8 - Half Precision OS SZ S0A SIMD8 Render Target Data Payload		
6.0-6.7	255:0	**Stencil** Format: MDPR_STENCIL Slots [7:0] Stencil

Half Precision OS SZ SIMD8 Dual Source Render Target Data Payload

MDP_RTWH_SZ8DS - Half Precision OS SZ SIMD8 Dual Source Render Target Data Payload

Size (in bits): 1536

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Red** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Red
1.0-1.7	255:0	**Green** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Green
2.0-2.7	255:0	**Blue** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Blue
3.0-3.7	255:0	**Alpha** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Alpha
4.0-4.7	255:0	**Source Depth** Format: MDP_DW_SIMD8 Slots [7:0] or [15:8] of Source Depth
5.0-5.7	255:0	**Stencil** Format: MDPR_STENCIL Slots [7:0] or [15:8] of Stencil

Half Precision OS SZ SIMD8 Render Target Data Payload

MDP_RTWH_SZ8 - Half Precision OS SZ SIMD8 Render Target Data Payload

Size (in bits): 1536

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Red** Format: MDPR_H_SIMD8 Slots [7:0] Red
1.0-1.7	255:0	**Green** Format: MDPR_H_SIMD8 Slots [7:0] Green
2.0-2.7	255:0	**Blue** Format: MDPR_H_SIMD8 Slots [7:0] Blue
3.0-3.7	255:0	**Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Alpha
4.0-4.7	255:0	**Source Depth** Format: MDP_DW_SIMD8 Slots [7:0] Source Depth
5.0-5.7	255:0	**Stencil** Format: MDPR_STENCIL Slots [7:0] Stencil

Half Precision Replicated Pixel Render Target Data Payload Register

MDPR_H_RGBA - Half Precision Replicated Pixel Render Target Data Payload Register

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:16	**Green** Format: U16 Specifies the value of all slots' green channel.
	15:0	**Red** Format: U16 Specifies the value of all slots' red channel.
1	31:16	**Alpha** Format: U16 Specifies the value of all slots' alpha channel.
	15:0	**Blue** Format: U16 Specifies the value of all slots' blue channel.
2..7	191:0	**Reserved** Access: RO Format: MBZ

Half Precision Replicated SIMD16 Render Target Data Payload

MDP_RTWH_16REP - Half Precision Replicated SIMD16 Render Target Data Payload

Size (in bits):	256
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**RGBA** Format: MDPR_H_RGBA RGBA for all slots [15:0]

Half Precision S0A SIMD8 Render Target Data Payload

MDP_RTWH_A8 - Half Precision S0A SIMD8 Render Target Data Payload

Size (in bits): 1280

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Source 0 Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Source 0 Alpha
1.0-1.7	255:0	**Red** Format: MDPR_H_SIMD8 Slots [7:0] Red
2.0-2.7	255:0	**Green** Format: MDPR_H_SIMD8 Slots [7:0] Green
3.0-3.7	255:0	**Blue** Format: MDPR_H_SIMD8 Slots [7:0] Blue
4.0-4.7	255:0	**Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Alpha

Half Precision S0A SIMD16 Render Target Data Payload

MDP_RTWH_A16 - Half Precision S0A SIMD16 Render Target Data Payload

Size (in bits): 1280

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Source 0 Alpha[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Source 0 Alpha
1.0-1.7	255:0	**Red[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Red
2.0-2.7	255:0	**Green[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Green
3.0-3.7	255:0	**Blue[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Blue
4.0-4.7	255:0	**Alpha[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Alpha

Half Precision SIMD8 Dual Source Render Target Data Payload

MDP_RTWH_8DS - Half Precision SIMD8 Dual Source Render Target Data Payload

Size (in bits): 1024

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Red** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Red
1.0-1.7	255:0	**Green** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Green
2.0-2.7	255:0	**Blue** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Blue
3.0-3.7	255:0	**Alpha** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Alpha

Half Precision SIMD8 Message Data Payload Register

MDPR_H_SIMD8 - Half Precision SIMD8 Message Data Payload Register

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:16	**Data1** Format: IEEE_FLOAT16 Specifies the slot 1 data in this payload register
	15:0	**Data0** Format: IEEE_FLOAT16 Specifies the slot 0 data in this payload register
1	31:16	**Data3** Format: IEEE_FLOAT16 Specifies the slot 3 data in this payload register
	15:0	**Data2** Format: IEEE_FLOAT16 Specifies the slot 2 data in this payload register
2	31:16	**Data5** Format: IEEE_FLOAT16 Specifies the slot 5 data in this payload register
	15:0	**Data4** Format: IEEE_FLOAT16 Specifies the slot 4 data in this payload register
3	31:16	**Data7** Format: IEEE_FLOAT16 Specifies the slot 7 data in this payload register
	15:0	**Data6** Format: IEEE_FLOAT16 Specifies the slot 6 data in this payload register

MDPR_H_SIMD8 - Half Precision SIMD8 Message Data Payload Register		
4..7	127:0	**Reserved** Access: RO Format: MBZ

Half Precision SIMD8 Render Target Data Payload

MDP_RTWH_8 - Half Precision SIMD8 Render Target Data Payload

Size (in bits): 1024

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Red** Format: MDPR_H_SIMD8 Slots [7:0] Red
1.0-1.7	255:0	**Green** Format: MDPR_H_SIMD8 Slots [7:0] Green
2.0-2.7	255:0	**Blue** Format: MDPR_H_SIMD8 Slots [7:0] Blue
3.0-3.7	255:0	**Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Alpha

Half Precision SIMD16 Message Data Payload Register

MDPR_H_SIMD16 - Half Precision SIMD16 Message Data Payload Register

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:16	**Data1** Format: IEEE_FLOAT16 Specifies the slot 1 data in this payload register
	15:0	**Data0** Format: IEEE_FLOAT16 Specifies the slot 0 data in this payload register
1	31:16	**Data3** Format: IEEE_FLOAT16 Specifies the slot 3 data in this payload register
	15:0	**Data2** Format: IEEE_FLOAT16 Specifies the slot 2 data in this payload register
2	31:16	**Data5** Format: IEEE_FLOAT16 Specifies the slot 5 data in this payload register
	15:0	**Data4** Format: IEEE_FLOAT16 Specifies the slot 4 data in this payload register
3	31:16	**Data7** Format: IEEE_FLOAT16 Specifies the slot 7 data in this payload register
	15:0	**Data6** Format: IEEE_FLOAT16 Specifies the slot 6 data in this payload register

MDPR_H_SIMD16 - Half Precision SIMD16 Message Data Payload Register

4	31:16	**Data9** Format: IEEE_FLOAT16 Specifies the slot 9 data in this payload register
	15:0	**Data8** Format: IEEE_FLOAT16 Specifies the slot 8 data in this payload register
5	31:16	**Data11** Format: IEEE_FLOAT16 Specifies the slot 11 data in this payload register
	15:0	**Data10** Format: IEEE_FLOAT16 Specifies the slot 10 data in this payload register
6	31:16	**Data13** Format: IEEE_FLOAT16 Specifies the slot 13 data in this payload register
	15:0	**Data12** Format: IEEE_FLOAT16 Specifies the slot 12 data in this payload register
7	31:16	**Data15** Format: IEEE_FLOAT16 Specifies the slot 15 data in this payload register
	15:0	**Data14** Format: IEEE_FLOAT16 Specifies the slot 14 data in this payload register

Half Precision SIMD16 Render Target Data Payload

MDP_RTWH_16 - Half Precision SIMD16 Render Target Data Payload

Size (in bits): 1024

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Red[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Red
1.0-1.7	255:0	**Green[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Green
2.0-2.7	255:0	**Blue[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Blue
3.0-3.7	255:0	**Alpha[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Alpha

Half Precision SZ OM S0A SIMD8 Render Target Data Payload

MDP_RTWH_ZMA8 - Half Precision SZ OM S0A SIMD8 Render Target Data Payload		
Size (in bits):	1792	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Source 0 Alpha** Format: **MDPR_H_SIMD8** Slots [7:0] Source 0 Alpha
1.0-1.7	255:0	**oMask** Format: **MDPR_OMASK** Slots [7:0] oMask. Upper half ignored.
2.0-2.7	255:0	**Red** Format: **MDPR_H_SIMD8** Slots [7:0] Red
3.0-3.7	255:0	**Green** Format: **MDPR_H_SIMD8** Slots [7:0] Green
4.0-4.7	255:0	**Blue** Format: **MDPR_H_SIMD8** Slots [7:0] Blue
5.0-5.7	255:0	**Alpha** Format: **MDPR_H_SIMD8** Slots [7:0] Alpha

MDP_RTWH_ZMA8 - Half Precision SZ OM S0A SIMD8 Render Target Data Payload

6.0-6.7	255:0	**Source Depth** Format: MDP_DW_SIMD8 Slots [7:0] Source Depth

Half Precision SZ OM S0A SIMD16 Render Target Data Payload

MDP_RTWH_ZMA16 - Half Precision SZ OM S0A SIMD16 Render Target Data Payload

Size (in bits): 2048

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Source 0 Alpha** Format: MDPR_H_SIMD16 Slots [15:0] Source 0 Alpha
1.0-1.7	255:0	**oMask** Format: MDPR_OMASK Slots [15:0] oMask
2.0-2.7	255:0	**Red** Format: MDPR_H_SIMD16 Slots [15:0] Red
3.0-3.7	255:0	**Green** Format: MDPR_H_SIMD16 Slots [15:0] Green
4.0-4.7	255:0	**Blue** Format: MDPR_H_SIMD16 Slots [15:0] Blue
5.0-5.7	255:0	**Alpha** Format: MDPR_H_SIMD16 Slots [15:0] Alpha

MDP_RTWH_ZMA16 - Half Precision SZ OM S0A SIMD16 Render Target Data Payload		
6.0-6.7	255:0	**Source Depth[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Source Depth
7.0-7.7	255:0	**Source Depth[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Source Depth

Half Precision SZ OM SIMD8 Dual Source Render Target Data Payload

MDP_RTWH_ZM8DS - Half Precision SZ OM SIMD8 Dual Source Render Target Data Payload		
Size (in bits):	1536	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**oMask** Format: **MDPR_OMASK** oMask for slots [7:0] and [15:8]. Operation selects upper or lower half.
1.0-1.7	255:0	**Red** Format: **MDPR_DSH_SIMD8** Slots[7:0] or [15:8] of Src0 and Src1 Red
2.0-2.7	255:0	**Green** Format: **MDPR_DSH_SIMD8** Slots[7:0] or [15:8] of Src0 and Src1 Green
3.0-3.7	255:0	**Blue** Format: **MDPR_DSH_SIMD8** Slots[7:0] or [15:8] of Src0 and Src1 Blue
4.0-4.7	255:0	**Alpha** Format: **MDPR_DSH_SIMD8** Slots[7:0] or [15:8] of Src0 and Src1 Alpha
5.0-5.7	255:0	**Source Depth** Format: **MDP_DW_SIMD8** Slots [7:0] or [15:8] of Source Depth

Half Precision SZ OM SIMD8 Render Target Data Payload

MDP_RTWH_ZM8 - Half Precision SZ OM SIMD8 Render Target Data Payload

Size (in bits): 1536

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**oMask** Format: MDPR_OMASK Slots [7:0] oMask. Upper half ignored.
1.0-1.7	255:0	**Red** Format: MDPR_H_SIMD8 Slots [7:0] Red
2.0-2.7	255:0	**Green** Format: MDPR_H_SIMD8 Slots [7:0] Green
3.0-3.7	255:0	**Blue** Format: MDPR_H_SIMD8 Slots [7:0] Blue
4.0-4.7	255:0	**Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Alpha
5.0-5.7	255:0	**Source Depth** Format: MDP_DW_SIMD8 Slots [7:0] Source Depth

Half Precision SZ OM SIMD16 Render Target Data Payload

MDP_RTWH_ZM16 - Half Precision SZ OM SIMD16 Render Target Data Payload

Size (in bits): 1792

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**oMask** Format: **MDPR_OMASK** Slots [15:0] oMask
1.0-1.7	255:0	**Red[15:0]** Format: **MDPR_H_SIMD16** Slots [15:0] Red
2.0-2.7	255:0	**Green[15:0]** Format: **MDPR_H_SIMD16** Slots [15:0] Green
3.0-3.7	255:0	**Blue[15:0]** Format: **MDPR_H_SIMD16** Slots [15:0] Blue
4.0-4.7	255:0	**Alpha[15:0]** Format: **MDPR_H_SIMD16** Slots [15:0] Alpha
5.0-5.7	255:0	**Source Depth[7:0]** Format: **MDP_DW_SIMD8** Slots [7:0] Source Depth

MDP_RTWH_ZM16 - Half Precision SZ OM SIMD16 Render Target Data Payload		
6.0-6.7	255:0	**Source Depth[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Source Depth

Half Precision SZ S0A SIMD8 Render Target Data Payload

MDP_RTWH_ZA8 - Half Precision SZ S0A SIMD8 Render Target Data Payload

Size (in bits): 1536

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Source 0 Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Source 0 Alpha
1.0-1.7	255:0	**Red** Format: MDPR_H_SIMD8 Slots [7:0] Red
2.0-2.7	255:0	**Green** Format: MDPR_H_SIMD8 Slots [7:0] Green
3.0-3.7	255:0	**Blue** Format: MDPR_H_SIMD8 Slots [7:0] Blue
4.0-4.7	255:0	**Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Alpha
5.0-5.7	255:0	**Source Depth** Format: MDP_DW_SIMD8 Slots [7:0] Source Depth

Half Precision SZ S0A SIMD16 Render Target Data Payload

MDP_RTWH_ZA16 - Half Precision SZ S0A SIMD16 Render Target Data Payload

Size (in bits): 1792

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Source 0 Alpha[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Source 0 Alpha
1.0-1.7	255:0	**Red[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Red
2.0-2.7	255:0	**Green[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Green
3.0-3.7	255:0	**Blue[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Blue
4.0-4.7	255:0	**Alpha[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Alpha
5.0-5.7	255:0	**Source Depth[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Source Depth

MDP_RTWH_ZA16 - Half Precision SZ S0A SIMD16 Render Target Data Payload		
6.0-6.7	255:0	**Source Depth[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Source Depth

Half Precision SZ SIMD8 Dual Source Render Target Data Payload

MDP_RTWH_Z8DS - Half Precision SZ SIMD8 Dual Source Render Target Data Payload

Size (in bits): 1280

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Red** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Red
1.0-1.7	255:0	**Green** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Green
2.0-2.7	255:0	**Blue** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Blue
3.0-3.7	255:0	**Alpha** Format: MDPR_DSH_SIMD8 Slots[7:0] or [15:8] of Src0 and Src1 Alpha
4.0-4.7	255:0	**Source Depth** Format: MDP_DW_SIMD8 Slots [7:0] or [15:8] of Source Depth

Half Precision SZ SIMD8 Render Target Data Payload

MDP_RTWH_Z8 - Half Precision SZ SIMD8 Render Target Data Payload

Size (in bits): 1280

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Red** Format: MDPR_H_SIMD8 Slots [7:0] Red
1.0-1.7	255:0	**Green** Format: MDPR_H_SIMD8 Slots [7:0] Green
2.0-2.7	255:0	**Blue** Format: MDPR_H_SIMD8 Slots [7:0] Blue
3.0-3.7	255:0	**Alpha** Format: MDPR_H_SIMD8 Slots [7:0] Alpha
4.0-4.7	255:0	**Source Depth** Format: MDP_DW_SIMD8 Slots [7:0] Source Depth

Half Precision SZ SIMD16 Render Target Data Payload

MDP_RTWH_Z16 - Half Precision SZ SIMD16 Render Target Data Payload

Size (in bits): 1536

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Red[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Red
1.0-1.7	255:0	**Green[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Green
2.0-2.7	255:0	**Blue[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Blue
3.0-3.7	255:0	**Alpha[15:0]** Format: MDPR_H_SIMD16 Slots [15:0] Alpha
4.0-4.7	255:0	**Source Depth[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Source Depth
5.0-5.7	255:0	**Source Depth[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Source Depth

Hardware-Detected Error Bit Definitions

Hardware-Detected Error Bit Definitions

Source: RenderCS

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:8	**Reserved** Access: RO Format: MBZ
	7	**Reserved**
	6:3	**Reserved** Access: RO Format: MBZ
	2	**Command Privilege Violation Error** This bit is set if a command classified as privileged is parsed in a non-privileged batch buffer. The command will be converted to a NOOP and parsing will continue.
	1	**Reserved** Access: RO Format: MBZ
	0	**Instruction Error** This bit is set when the Renderer Instruction Parser detects an error while parsing an instruction. Instruction errors include: • Client ID value (Bits 31:29 of the Header) is not supported (only MI, 2D and 3D are supported). • Defeatured MI Instruction Opcodes: Value: 1 — Name: — Description: Instruction Error detected **Programming Notes** This error indications cannot be cleared except by reset (i.e., it is a fatal error).

Hardware Status Page Layout

Hardware Status Page Layout	
Source:	BSpec
Size (in bits):	32768
Default Value:	0x00000000, 0x00000000,

Hardware Status Page Layout
0x00000000, 0x00000000,

Hardware Status Page Layout
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,

Hardware Status Page Layout

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:0	**Interrupt Status Register Storage** The content of the ISR register is written to this location whenever an "unmasked" bit of the ISR (as determined by the HWSTAM register) changes state.
1..3	95:0	**Reserved** Access: RO Format: MBZ

<table>
<tr><th colspan="3">Hardware Status Page Layout</th></tr>
<tr><td>4</td><td>31:0</td><td>Ring Head Pointer Storage
<table><tr><th>Description</th></tr><tr><td>The contents of the Ring Buffer Head Pointer register (register DWord 1) are written to this location either as result of an MI_REPORT_HEAD instruction or as the result of an "automatic report" (see RINGBUF registers).</td></tr></table></td></tr>
<tr><td>5..15</td><td>351:0</td><td>Reserved
<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>16..39</td><td>767:0</td><td>Context Status DWords
<table><tr><td>Format:</td><td>CONTEXT STATUS[12]</td></tr></table></td></tr>
<tr><td>40..46
These dwords are reserved.</td><td>223:0</td><td>Reserved
<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>47</td><td>31:0</td><td>Last Written Status Offset</td></tr>
<tr><td>48..1023</td><td>31231:0</td><td>General Purpose
<table><tr><td>Format:</td><td>U32[976]</td></tr></table>These locations can be used for general purpose via the MI_STORE_DATA_INDEX or MI_STORE_DATA_IMM instructions.</td></tr>
</table>

HCP_PAK_INSERT_OBJECT_INDIRECT_PAYLOAD

HCP_PAK_INSERT_OBJECT_INDIRECT_PAYLOAD

Source: VideoCS

Size (in bits): 128

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:0	**Indirect Payload Data Size in bits** Format: U32 Number of bits to be inserted. Not including those skipped bytes in the beginning. For VP9: the Data is always valid from start of cache-line, no offset is allowed.
1..2	63:0	**Indirect Payload Base Address** Format: SplitBaseAddress64ByteAligned 48-bit address of the indirect payload data in memory buffer. **Programming Notes** Payload must begin in a byte position, but the payload can be ended in a bit position.
3	31:0	**Indirect Payload Base Address** Format: MemoryAddressAttributes

HCP_REF_LIST_ENTRY

HCP_REF_LIST_ENTRY

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15	**bottom_field_flag** Format: U1 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15. Value / Name: 0 – Bottom Field 1 – Top Field Programming Notes: Not supported in encoder mode.
	14	**field_pic_flag** Format: U1 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15. Value / Name: 0 – Video Frame 1 – Video Field Programming Notes: Not supported in encoder mode.
	13	**LongTermReference** Format: U1 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15. Value / Name: 0 – Short term reference 1 – Long term reference

HCP_REF_LIST_ENTRY

	12	**luma_weight_lX_flag** Format: U1 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15. **Value** / **Name** 0 – Default weighted prediction for luma 1 – Explicit weighted prediction for Luma
	11	**chroma_weight_lX_flag** Format: U1 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15. **Value** / **Name** 0 – Default weighted prediction for Chroma 1 – Explicit weighted prediction for Chroma
	10:8	**list_entry_lX: Reference Picture Frame ID (RefAddr[0-7])** Format: U3 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15. The reference picture frame ID identifies the reference picture associated with the base addressdefined in **Reference Picture Address (RefAddr[0-7])** in the HCP_PIPE_BUF_ADDR_STATE command.
	7:0	**Reference Picture tb Value** Format: U8 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15. clip(-128,127, CurrentPOC - RefPOC), where RefPOC is the POC value of the reference picture. 8-bit signed. See the "Derivation process for temporal luma motion vector prediction" in the HEVC standard.

HCP_TILE_POSITION_IN_CTB

HCP_TILE_POSITION_IN_CTB		
Size (in bits): 32 Default Value: 0x00000000		
DWord	**Bit**	**Description**
0	31:24	**CtbPos3+i** Format: U8
	23:16	**CtbPos2+i** Format: U8
	15:8	**CtbPos1+i** Format: U8
	7:0	**CtbPos0+i** Format: U8

HCP_TILE_POSITION_IN_CTB_MSB

HCP_TILE_POSITION_IN_CTB_MSB		
Size (in bits):	64	
Default Value:	0x00000000, 0x00000000	
Added to support 16k picture size.		
DWord	**Bit**	**Description**
0..1	63:44	**Reserved** Access: RO Format: MBZ
	43:42	**Ctb position of tile 21 [9:8]** MSB 2 bits of CTB row position of tile row 21. **Programming Notes** Please note that this field is MBZ for columns
	41:40	**Ctb row position of tile column 20 [9:8]** MSB 2 bits of CTB row position of tile row 20. **Programming Notes** Please note that this field is MBZ for columns
	39:38	**Ctb row position of tile column 19 [9:8]** MSB 2 bits of CTB row or column position of tile row or column 19.
	37:36	**Ctb row position of tile column 18 [9:8]** MSB 2 bits of CTB row or column position of tile row or column 18.
	35:34	**Ctb row position of tile column 17 [9:8]** MSB 2 bits of CTB row or column position of tile row or column 17.
	33:32	**Ctb row position of tile column 16 [9:8]** MSB 2 bits of CTB row or column position of tile row or column 16.
	31:30	**Ctb row position of tile column 15 [9:8]** MSB 2 bits of CTB row or column position of tile row or column 15.
	29:28	**Ctb row position of tile column 14 [9:8]** MSB 2 bits of CTB row or column position of tile row or column 14.
	27:26	**Ctb row position of tile column 13 [9:8]** MSB 2 bits of CTB row or column position of tile row or column 13.
	25:24	**Ctb row position of tile column 12 [9:8]** MSB 2 bits of CTB row or column position of tile row or column 12.
	23:22	**Ctb row position of tile column 11 [9:8]** MSB 2 bits of CTB row or column position of tile row or column 11.
	21:20	**Ctb row position of tile column 10 [9:8]** MSB 2 bits of CTB row or column position of tile row or column 10.
	19:18	**Ctb row position of tile column 9 [9:8]** MSB 2 bits of CTB row or column position of tile row or column 9.

HCP_TILE_POSITION_IN_CTB_MSB		
	17:16	**Ctb row position of tile column 8 [9:8]** MSB 2 bits of CTB row or column position of tile row or column 8.
	15:14	**Ctb row position of tile column 7 [9:8]** MSB 2 bits of CTB row or column position of tile row or column 7.
	13:12	**Ctb row position of tile column 6 [9:8]** MSB 2 bits of CTB row or column position of tile row or column 6.
	11:10	**Ctb row position of tile column 5 [9:8]** MSB 2 bits of CTB row or column position of tile row or column 5.
	9:8	**Ctb row position of tile column 4 [9:8]** MSB 2 bits of CTB row or column position of tile row or column 4.
	7:6	**Ctb row position of tile column 3 [9:8]** MSB 2 bits of CTB row or column position of tile row or column 3.
	5:4	**Ctb row position of tile column 2 [9:8]** MSB 2 bits of CTB row or column position of tile row or column 2.
	3:2	**Ctb row position of tile column 1 [9:8]** MSB 2 bits of CTB row or column position of tile row or column 1.
	1:0	**Ctb row position of tile column 0 [9:8]** MSB 2 bits of CTB row or column position of tile row or column 0.

HCP_WEIGHTOFFSET_CHROMA_ENTRY

HCP_WEIGHTOFFSET_CHROMA_ENTRY

Source: VideoCS
Size (in bits): 32
Default Value: 0x00000000

DWord	Bit	Description
0	31:24	**ChromaOffsetLX [i][1]** Where X is the RefPicListNum and i is the list entry number 0 through 15. DW 18corresponds to i=0, DW 33 corresponds to i=15. Valid only if explicit weighted prediction for chroma is enabled, otherwise must be zero. **Programming Notes** This (combined with its MSbyte below) shall be in the range of -WpOffsetHalfRangeC to (WpOffsetHalfRangeC - 1), inclusive WpOffsetHalfRangeC = 1 « (high_precision_offsets_enabled_flag ? (BitDepthC - 1) : 7)
	23:16	**delta_chroma_weight_lX[i][1]** Format: S7 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW 18 corresponds to i=0, DW 33 corresponds to i=15. Valid only if explicit weighted prediction for chroma is enabled, otherwise must be zero. **Programming Notes** This shall be in the range of 128 to 127, inclusive
	15:8	**ChromaOffsetLX[i][0]** Where X is the RefPicListNum and i is the list entry number 0 through 15. DW 18 corresponds to i=0, DW 33 corresponds to i=15. Valid only if explicit weighted prediction for chroma is enabled, otherwise must be zero. **Programming Notes** This (combined with its MSbyte below) shall be in the range of -WpOffsetHalfRangeC to (WpOffsetHalfRangeC - 1), inclusive WpOffsetHalfRangeC = 1 « (high_precision_offsets_enabled_flag ? (BitDepthC - 1) : 7)
	7:0	**delta_chroma_weight_lX[i][0]** Format: S7 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW 18 corresponds to i=0, DW 33 corresponds to i=15. Valid only if explicit weighted prediction for chroma is enabled, otherwise must be zero.

HCP_WEIGHTOFFSET_CHROMA_ENTRY		
		Programming Notes
		This shall be in the range of 128 to 127, inclusive

HCP_WEIGHTOFFSET_CHROMA_EXT_ENTRY

HCP_WEIGHTOFFSET_CHROMA_EXT_ENTRY		
Source:	VideoCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:24	**ChromaOffsetLX[i+1][1] MSByte** To support 4:4:4, the chroma offsetisextended into 16-bit. In order to keep SW back compatible, the most significant byte is programmed here. **Programming Notes** This is only MSByte portion of ChromaOffsetLX. Please refer to LSB section for available range.
	23:16	**ChromaOffsetLX[i][1] MSByte** To support 4:4:4, the chroma offsetisextended into 16-bit. In order to keep SW back compatible, the most significant byte is programmed here. **Programming Notes** This is only MSByte portion of ChromaOffsetLX. Please refer to LSB section for available range.
	15:8	**ChromaOffsetLX[i+1][0] MSByte** To support 4:4:4, the chroma offsetisextended into 16-bit. In order to keep SW back compatible, the most significant byte is programmed here. **Programming Notes** This is only MSByte portion of ChromaOffsetLX. Please refer to LSB section for available range.
	7:0	**ChromaOffsetLX[i][0] MSByte** To support 4:4:4, the chroma offsetisextended into 16-bit. In order to keep SW back compatible, the most significant byte is programmed here. **Programming Notes** This is only MSByte portion of ChromaOffsetLX. Please refer to LSB section for available range.

HCP_WEIGHTOFFSET_LUMA_ENTRY

HCP_WEIGHTOFFSET_LUMA_ENTRY		
Source:	VideoCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:24	**luma_offset_lX[i] MSByte** To support 4:4:4, the luma offsetisextended into 16-bit. In order to keep SW back compatible, the most significant byte is programmed here. **Programming Notes** This is only MSByte portion of luma_offset_lX. Please refer to LSB section for available range.
	23:16	**Reserved** Access: RO Format: MBZ
	15:8	**luma_offset_lX[i]** Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15. Valid only if explicit weighted prediction for luma is enabled, otherwise must be zero. **Programming Notes** This (combined with it MSbyte above) shall be in the range of$WpOffsetHalfRange_Y$ to $WpOffsetHalfRange_Y$1,where $WpOffsetHalfRange_Y = 1 \ll (high\_precision\_offsets\_enabled\_flag ? (BitDepth_Y 1) : 7)$
	7:0	**delta_luma_weight_lX[i]** Format: S7 Where X is the RefPicListNum and i is the list entry number 0 through 15. DW2 corresponds to i=0, DW17 corresponds to i=15. Valid only if explicit weighted prediction for luma is enabled, otherwise must be zero. **Programming Notes** When luma_weight_l0_flag[i] is equal to 1, the value of delta_luma_weight_l0[i] shall be in the range of 128 to 127, inclusive.

Header

Header		
Size (in bits):	16	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	15:12	**SWSB[7:4]** Format: SWSB[7:4] This field specify the Software Scoreboard information.
	11:8	**SWSB[3:0]** Format: SWSB[3:0] This field specify the Software Scoreboard information.
	7	**Reserved** Format: MBZ
	6:0	**Opcode** Format: EU_OPCODE This field determines the operation performed by the instruction.

Header Forbidden Message Descriptor Control Field

MDC_MHF - Header Forbidden Message Descriptor Control Field

Size (in bits): 1

Default Value: 0x00000000

DWord	Bit	Description
0	0	**Message Header Present** Indicates the message forbids a message header.

Value	Name	Description
0h	No **[Default]**	Message header is not present
1h	Reserved	Not used

Header Present Message Descriptor Control Field

MDC_MHP - Header Present Message Descriptor Control Field

Size (in bits): 1

Default Value: 0x00000000

DWord	Bit	Description
0	0	**Message Header Present** Format: Boolean Specifies if the message uses the optional message header. Value / Name / Description: 0h / No / Message header is not present 1h / Yes / Message header is present

Header Required Message Descriptor Control Field

MDC_MHR - Header Required Message Descriptor Control Field

Size (in bits): 1

Default Value: 0x00000001

DWord	Bit	Description
0	0	**Message Header Present** Format: Boolean Indicates the message requires a message header. Value / Name / Description 0h / Reserved / Not used 1h / Yes **[Default]** / Message header is present

HEVC_ARBITRATION_PRIORITY

HEVC_ARBITRATION_PRIORITY

Size (in bits): 2

Default Value: 0x00000000

This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.

DWord	Bit	Description
0	1:0	**Priority** Format: U2 Value / Name: 00b — Highest priority 01b — Second highest priority 10b — Third highest priority 11b — Lowest priority

HEVC_VP9_RDOQ_LAMBDA_FIELDS

HEVC_VP9_RDOQ_LAMBDA_FIELDS		
Source:	VideoCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:16	**LambdaValue1** Lambda value for Intra/Inter Luma/Chroma component of QP=1, 3, , 61, 63...73,75 (odd number) For 12-bit video, the QP range has extended to include 65 to 75. for HEVC
	15:0	**LambdaValue0** Lambda value for Intra/Inter Luma/Chroma component of QP=0, 2, , 60, 62,...72,74(even number) For 12-bit video, the QP range has extended to include 64to 74. for HEVC

HUC_VIRTUAL_ADDR_REGION

HUC_VIRTUAL_ADDR_REGION		
Source:	VideoCS	
Size (in bits):	96	
Default Value:	0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0..2	95:64	**HUC Surface (VirtualAddrRegion[0-15])** Format: **MemoryAddressAttributes**
	63:0	**HUC Surface Base Address (VirtualAddrRegion[0-15])** Format: **SplitBaseAddress4KByteAligned** HUC surface base address for each virtual address region [0-15].

HW Generated BINDING_TABLE_STATE

HW Generated BINDING_TABLE_STATE		
Size (in bits):	16	
Default Value:	0x00000000	
The binding table binds surfaces to logical resource indices used by shaders and other compute engine kernels. The HW generated Binding_Table_State have different format than the SW generated Binding_Table_State. The HW generated Binding_Table_State is stored as an array of 256 elements, each of which contains one word as defined here. The start of each element is spaced one word apart. The first element of the binding table is aligned to a 64-byte boundary. Binding table indexes beyond 256 will automatically be mapped to entry 0 by the HW, w/ the exception of any messages which support the special indexes 240 through 255, inclusive.		
DWord	**Bit**	**Description**
0	15:0	**Surface State Pointer** Format: SurfaceStateOffset[21:6]

Hword 1 Block Data Payload

MDP_HW1 - Hword 1 Block Data Payload		
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Hword** Format: U256 Specifies the Hword data

Hword 2 Block Data Payload

MDP_HW2 - Hword 2 Block Data Payload		
Size (in bits):	512	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Hword0** Format: U256 Specifies the Hword data for element 0
1.0-1.7	255:0	**Hword1** Format: U256 Specifies the Hword data for element 1

Hword 4 Block Data Payload

MDP_HW4 - Hword 4 Block Data Payload		
Size (in bits):	1024	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Hword0** Format: U256 Specifies the Hword data for element 0
1.0-1.7	255:0	**Hword1** Format: U256 Specifies the Hword data for element 1
2.0-2.7	255:0	**Hword2** Format: U256 Specifies the Hword data for element 2
3.0-3.7	255:0	**Hword3** Format: U256 Specifies the Hword data for element 3

Hword 8 Block Data Payload

MDP_HW8 - Hword 8 Block Data Payload		
Size (in bits):	2048	
Default Value:	0x00000000, 0x00000000	

DWord	Bit	Description
0.0-0.7	255:0	**Hword0** Format: U256 Specifies the Hword data for element 0
1.0-1.7	255:0	**Hword1** Format: U256 Specifies the Hword data for element 1
2.0-2.7	255:0	**Hword2** Format: U256 Specifies the Hword data for element 2
3.0-3.7	255:0	**Hword3** Format: U256 Specifies the Hword data for element 3
4.0-4.7	255:0	**Hword4** Format: U256 Specifies the Hword data for element 4
5.0-5.7	255:0	**Hword5** Format: U256 Specifies the Hword data for element 5

MDP_HW8 - Hword 8 Block Data Payload		
6.0-6.7	255:0	**Hword6** Format: U256 Specifies the Hword data for element 6
7.0-7.7	255:0	**Hword7** Format: U256 Specifies the Hword data for element 7

Hword Channel Mode Message Header Control

MHC_A64_CMODE - Hword Channel Mode Message Header Control

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:0	**Reserved** Access: RO Format: MBZ

Hword Register Blocks Message Descriptor Control Field

MDC_DB_HW - Hword Register Blocks Message Descriptor Control Field

Size (in bits): 2

Default Value: 0x00000000

DWord	Bit	Description
0	1:0	**Register Blocks** Specifies the number of Hword blocks to be read or written

Value	Name	Description
00h	HW1	1 Hword register
01h	HW2	2 Hword registers
02h	HW4	4 Hword registers
03h	HW8	8 Hword registers

Ignored Message Header

MH_IGNORE - Ignored Message Header		
Source:	EuSubFunctionDataPort0	
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
Some messages require a message header or have an optional message header, but do not use any information in the header.		
DWord	**Bit**	**Description**
0..7	255:0	**Reserved** Access: RO Format: MBZ

IndirectOperand

IndirectOperand		
Size (in bits):	14	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	13:10	**AddrSubRegNum** Format: AddrSubRegNum
	9:0	**AddrImm** Format: S9 This field defines a 10-bit signed integer offset in units of byte, only used with the Indirect Addressing Mode. In that addressing mode, the Address Immediate Offset value is added to an address subregister value to determine the operand's address in the GRF.

Inline Data Description for MFD_AVC_BSD_Object

Inline Data Description for MFD_AVC_BSD_Object

Source: VideoCS

Size (in bits): 96

Default Value: 0x00000000, 0x00000000, 0x00000000

This structure includes all the required Slice Header parameters and error handling settings for AVC_BSD_OBJECT Command (DW3..DW5).

DWord	Bit	Description
0	31	**Concealment Method** This field specifies the method used for concealment when error is detected. If set, a copy from collocated macroblock location is performed from the concealment reference indicated by the ConCeal_Pic_Id field. If it is not set, a copy from the current picture is performed using Intra 16x16 Prediction method. Value / Name / Description: 0 / / Intra 16x16 Prediction 1 / / Inter P Copy
	30	**Init Current MB Number** When set, the current Slice_Start_MB_Num, Slice_MB_Start_Hor_Pos and Slice_MB_Start_Vert_Pos fields will be used to initialize the Current_MB_Number register.This effectively disables the concealment capability.
	29	**Intra PredMode (4x4/8x8 Luma) Error Control Bit** This field controls if AVC decoder will fix Intra Prediction Mode if the decoded value is incorrect according to MB position Value / Name / Description: 0 / / AVC decoder will detect and fix IntraPredMode (4x4/8x8 Luma) Errors. 1 / / AVC decoder will NOT detect IntraPredMode (4x4/8x8 Luma) Errors. The wrong IntraPredMode value will be retaind.
	28:27	**MB Error Concealment B Temporal Prediction mode** These two bits control how the reference L0/L1 are overridden in B temporal slice. Value / Name / Description: 00b / **[Default]** / Both Reference Indexes L0/L1 are forced to 0 during Concealment 01b / / Only Reference Index L1 is forced to 0; Reference Index L0 is forced to -1 10b / / Only Reference Index L0 is forced to 0; Reference Index L1 is forced to -1 11b / Reserved / Invalid
	26	**Reserved** Access: RO Format: MBZ

Inline Data Description for MFD_AVC_BSD_Object

25 — MB Error Concealment B Temporal Motion Vectors Override Enable Flag

During MB Error Concealment on B slice with Temporal Direct Prediction, motion vectors are forced to 0 to improve image quality.This bit can be set to preserve the original weight prediction.

Value	Name	Description
0	**[Default]**	Predicted Motion Vectors are used during MB Concealment
1		Motion Vectors are Overridden to 0 during MB Concealment

24 — MB Error Concealment B Temporal Weight Prediction Disable Flag

During MB Error Concealment on B slice with Temporal Direct Prediction, weight prediction is disabled to improve image quality.This bit can be set to preserve the original weight prediction.

Value	Name	Description
0	**[Default]**	Weight Prediction is Disabled during MB Concealment
1		Weight Prediction will not be overridden during MB Concealment

23:22 — Reserved

Access:	RO
Format:	MBZ

21:16 — Concealment Picture ID

This field identifies the picture in the reference list to be used for concealment. This field is only valid if **Concealment Method** is Inter P Copy.

Bit Filed	Value	Defenition
21	0	Frame Picture
21	1	Field picture
20:16	All	Frame Store Index[4:0]

15 — Reserved

Access:	RO
Format:	MBZ

14 — BSD Premature Complete Error Handling

BSD Premature Complete Error occurs in situation where the Slice decode is completed but there are still data in the bitstream.

Value	Name	Description
1		Set the interrupt to the driver (provide MMIO registers for MB address R/W)
0		Ignore the error and continue (masked the interrupt), assume the hardware automatically performs the error handling

13 — Reserved

Access:	RO
Format:	MBZ

Inline Data Description for MFD_AVC_BSD_Object

12 — MPR Error (MV out of range) Handling

Software must follow the action for each Value as follow:

Value	Name	Description
1		Set the interrupt to the driver (provide MMIO registers for MB address R/W)
0		Ignore the error and continue (masked the interrupt), assume the hardware automatically performs the error handling

11 — Reserved

Access:	RO
Format:	MBZ

10 — Entropy Error Handling

Software must follow the action for each Value as follow:

Value	Name	Description
1		Set the interrupt to the driver (provide MMIO registers for MB address R/W).
0		Ignore the error and continue (masked the interrupt), assume the hardware automatically perform the error handling.

9 — Reserved

Access:	RO
Format:	MBZ

8 — MB Header Error Handling

Software must follow the action for each Value as follow:

Value	Name	Description
1		Set the interrupt to the driver (provide MMIO registers for MB address R/W).
0		Ignore the error and continue (masked the interrupt), assume the hardware automatically perform the error concealment.

7:6 — MB Error Concealment B Spatial Prediction mode

These two bits control how the reference L0/L1 are overridden in B spatial slice.

Value	Name	Description
00b	**[Default]**	Both Reference Indexes L0/L1 are forced to 0 during Concealment
01b		Only Reference Index L1 is forced to 0; Reference Index L0 is forced to -1
10b		Only Reference Index L0 is forced to 0; Reference Index L1 is forced to -1
11b	Reserved	Invalid

5 — Reserved

Access:	RO
Format:	MBZ

Inline Data Description for MFD_AVC_BSD_Object

	4	**MB Error Concealment B Spatial Motion Vectors Override Disable Flag** During MB Error Concealment on B slice with Spatial Direct Prediction, motion vectors are forced to 0 to improve image quality. This bit can be set to use the predicted motion vectors instead. This bit does not affect normal decoded MB.

Value	Name	Description
0	**[Default]**	Motion Vectors are Overridden to 0 during MB Concealment
1		Predicted Motion Vectors are used during MB Concealment

	3	**MB Error Concealment B Spatial Weight Prediction Disable Flag** During MB Error Concealment on B slice with Spatial Direct Prediction, weight prediction is disabled to improve image quality.This bit can be set to preserve the original weight prediction. This bit does not affect normal decoded MB.

Value	Name	Description
0	**[Default]**	Weight Prediction is Disabled during MB Concealment.
1		Weight Prediction will not be overridden during MB Concealment.

	2	**Reserved**

Access:	RO
Format:	MBZ

	1	**MB Error Concealment P Slice Motion Vectors Override Disable Flag** During MB Error Concealment on P slice, motion vectors are forced to 0 to improve image quality. This bit can be set to use the predicted motion vectors instead. This bit does not affect normal decoded MB.

Value	Name	Description
0	**[Default]**	Motion Vectors are Overridden to 0 during MB Concealment
1		Predicted Motion Vectors are used during MB Concealment

	0	**MB Error Concealment P Slice Weight Prediction Disable Flag** During MB Error Concealment on P slice, weight prediction is disabled to improve image quality.This bit can be set to preserve the original weight prediction. This bit does not affect normal decoded MB.

Value	Name	Description
0	**[Default]**	Weight Prediction is Disabled during MB Concealment.
1		Weight Prediction will not be overridden during MB Concealment.

1	31:16	**First MB Byte Offset of Slice Data or Slice Header**

Programming Notes
MFX supports only DXVA2 Long and Short Format.

	15:8	**Reserved**

Access:	RO
Format:	MBZ

Inline Data Description for MFD_AVC_BSD_Object

	7	**Fix Prev Mb Skipped** Enables an alternative method for decoding mb_skipped, to cope with an encoder that codes a skipped MB as a direct MB with no coefficient.
	6:5	**Reserved** Access: RO Format: MBZ
	4	**Emulation Prevention Byte Present** 0 – H/W needs to perform Emulation Byte Removal 1 – H/W does not need to perform Emulation Byte Removal
	3	**LastSlice Flag** It is needed for both error concealment at the end of a picture (so, no more phantom slice). It is also needed to know to set the last MB in a picture correctly. 1 – If the current Slice to be decoded is the very last slice of the current picture. 0 – If the current Slice to be decoded is any slice other than the very last slice of the current picture
	2:0	**First Macroblock (MB)Bit Offset** Exists If: //AVC Long Format Only Format: U3 This field provides the bit offset of the first macroblock of the Slice in the first byte of the input compressed bitstream.
2	31	**I Slice Concealment Mode** This field controls how AVC decoder handle MB concealment in I Slice 1 – Intra Concealment 0 – Inter Concealment **Programming Notes** If this field is set to "0" (Inter Concealment), driver must provide a valid reference picture (programmed using "Concealment Reference Picture" field) for concealment reference picture. In this mode, weight prediction is disabled and motion vectors are forced to 0 as well.
	30	**Reserved** Access: RO Format: MBZ

Inline Data Description for MFD_AVC_BSD_Object

29:24 **Concealment Reference Picture + Field Bit**

Format:	U6

This field provides the concealment reference picture for hardware to conceal in case driver wants to specify one concealment picture. This field matches with the DPB order sent to hardware. This field applies to all I/P/B slices

Bit Filed	Value	Defenition
29	MBZ	is reserved for future expansion
28:25	All	Reference PIcture Number
24	All	Field Bit(if the current picture is a field picture [Frame picture must be 0]

23 **P Slice Concealment Mode**

This field controls how AVC decoder handle MB concealment in P Slice

Value	Name
1	Intra Concealment
0	Inter Concealment

22:19 **Reserved**

Access:	RO
Format:	MBZ

18:16 **P Slice Inter Concealment Mode**

This field controls how AVC decoder select reference picture for Concealment in P Slice.

Value	Name	Description
000b		Top of Reference List L0 (Use top entry of Reference List L0)
001b		Driver Specified Concealment Reference
010b		Predicted Reference (Use reference picture predicted using P-Skip Algorithm)
011b		Temporal Closest (Using POC to select the closest forward picture)[For L0: Closest POC smaller than current POC]
100b		First Long Term Picture in Reference List L0 (If no long term picture available, use Temporal Closest Picture)
101b-111b	Reserved	

15 **B Slice Concealment Mode**

This field controls how AVC decoder handle MB concealment in B Slice

Value	Name
1	Intra Concealment
0	Inter Concealment

Inline Data Description for MFD_AVC_BSD_Object

14	**Reserved** Access: RO Format: MBZ
13:12	**B Slice Inter Direct Type Concealment Mode** AVC decoder can use Spatial or Temporal Direct for B Skip/Direct. This field determine can override the mode on how AVC decoder handles MB concealment in B slice.

Value	Name	Description
00b		Use Default Direct Type (slice programmed direct type)
01b		Forced to Spatial Direct Only
10b		Forced to Temporal Direct Only
11b		Spatial Direct without Temporal Componenet (MovingBlock information)

11	**Reserved** Access: RO Format: MBZ
10:8	**B Slice Spatial Inter Concealment Mode** This field controls how AVC decoder select reference picture for Spatial Inter Concealment in B Slice.

Value	Name	Description
000b		Top of Reference List L0/L1 (Use top entry of Reference List L0/L1).
001b		Driver Specified Concealment Reference
011b		Temporal Closest (Using POC to select the closest forward picture)[For L0: Closest POC smaller than current POC][For L1: Closest POC larger than current POC]
100b		" First Long Term Picture in Reference List L0/L1 (If no long term picture available, use Temporal Closest Picture)
101b-111b	Reserved	

7	**Reserved** Access: RO Format: MBZ
6:4	**B Slice Temporal Inter Concealment Mode** This field controls how AVC decoder select reference picture for Temporal Inter Concealment in B Slice

Value	Name	Description
000b		Top of Reference List L0/L1 (Use top entry of Reference List L0/L1)
001b		Driver Specified Concealment Reference
010b		Predicted Reference (Use reference picture predicted using B-Skip Algorithm)
011b		" Temporal Closest (Using POC to select the closest forward picture)[For L0:

Inline Data Description for MFD_AVC_BSD_Object

		Closest POC smaller than current POC][For L1: Closest POC larger than current POC]
100b		First Long Term Picture in Reference List L0/L1(If no long term picture available, use Temporal Closest Picture)
101b-111b	Reserved	

3:2 Reserved

Access:	RO
Format:	MBZ

1 Intra 8x8/4x4 Prediction Error Concealment Control Bit

This field controls if AVC goes into MB concealment mode (next MB) when an error is detected on Intra8x8/4x4 Prediction Mode (these 2 modes have fixed coding so it may not affect the bitstream.

Value	Name	Description
0		AVC decoder will NOT go into MB concealment when Intra8x8/4x4 Prediction mode is incorrect.
1		AVC decoder will go into MB concealment when Intra8x8/4x4 Prediction mode is incorrect.

0 Intra Prediction Error Control Bit (applied to Intra16x16/Intra8x8/Intra4x4 Luma and Chroma)

This field controls if AVC decoder will fix Intra Prediction Mode if the decoded value is incorrect according to MB position.

Value	Name	Description
0		AVC decoder will detect and fix Intra Prediction Mode Errors.
1		AVC decoder will retain the Intra Prediction value decoded from bitstream.

Inline Data Description in MPEG2-IT Mode

Inline Data Description in MPEG2-IT Mode

Source:	VideoCS
Size (in bits):	192
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

The content in this command is similar to that in the MEDIA_OBJECT command in IS mode described in the Media Chapter.
Each MFD_IT_OBJECT command corresponds to the processing of one macroblock. Macroblock parameters are passed in as inline data and the non-zero DCT coefficient data for the macroblock is passed in as indirect data. Inline data starts at dword 7 of MFD_IT_OBJECT command. There are 7 dwords total.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:28</td><td>Motion Vertical Field Select
A bit-wise representation of a long [2][2] array as defined in #167;6.3.17.2 of the ISO/IEC 13818-2 (see also #167;7.6.4).
<table>
<tr><th>Bit</th><th>MVector[r]</th><th>MVector[s]</th><th>MotionVerticalFieldSelect Index</th></tr>
<tr><td>28</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>29</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>30</td><td>1</td><td>0</td><td>2</td></tr>
<tr><td>31</td><td>1</td><td>1</td><td>3</td></tr>
</table>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>Top Field</td><td>The prediction is taken from the top reference field.</td></tr>
<tr><td>1</td><td>Bottom Field</td><td>The prediction is taken from the bottom reference field.</td></tr>
</table></td></tr>
<tr><td></td><td>27:26</td><td>Reserved
<table>
<tr><td>Access:</td><td>RO</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>25:24</td><td>Motion Type
When combined with the destination picture type (field or frame) this Motion Type field indicates the type of motion to be applied to the macroblock. See ISO/IEC 13818-2 #167;6.3.17.1, Tables 6-17, 6-18. In particular, the device supports dual-prime motion prediction (11) in both frame and field picture type.
<table>
<tr><th>Value</th><th>Destination = Frame
Picture_Structure = 11</th><th>Destination = Field
Picture_Structure != 11</th></tr>
<tr><td>'00'</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>'01'</td><td>Field</td><td>Field</td></tr>
<tr><td>'10'</td><td>Frame</td><td>16x8</td></tr>
<tr><td>'11'</td><td>Dual-Prime</td><td>Dual-Prime</td></tr>
</table></td></tr>
</table>

Inline Data Description in MPEG2-IT Mode

23:22 Reserved

Access:	RO
Format:	MBZ

21 DCT Type

This field specifies the DCT type of the current macroblock. The kernel should ignore this field when processing Cb/Cr data. See ISO/IEC 13818-2 #167;6.3.17.1. This field is zero if Coded Block Pattern is also zero (no coded blocks present).

Value	Name	Description
0	MC_FRAME_DCT	Macroblock is frame DCT coded
1	MC_FIELD_DCT	Macroblock is field DCT coded

20:19 Reserved

Access:	RO
Format:	MBZ

18 Macroblock Motion Backward

This field specifies if the backward motion vector is active. See ISO/IEC 13818-2 Tables B-2 through B-4.

Value	Name
0	No backward motion vector
1	Use backward motion vector(s)

17 Macroblock Motion Forward

This field specifies if the forward motion vector is active. See ISO/IEC 13818-2 Tables B-2 through B-4.

Value	Name
0	No forward motion vector
1	Use forward motion vector(s)

16 Macroblock Intra Type

This field specifies if the current macroblock is intra-coded. When set, Coded Block Pattern is ignored and no prediction is performed (i.e., no motion vectors are used). See ISO/IEC 13818-2 Tables B-2 through B-4.

Value	Name
0	Non-intra macroblock
1	Intra macroblock

15:12 Reserved

Access:	RO
Format:	MBZ

Inline Data Description in MPEG2-IT Mode

	11:6	**Coded Block Pattern** Format: Enable[6] Bit 11: Y0 Bit 10: Y1 Bit 9: Y2 Bit 8: Y3 Bit 7: Cb4 Bit 6: Cr5
	5:4	**Reserved** Access: RO Format: MBZ
	3	**LastMBInRow** This field indicates the last MB in each row
	2:0	**Reserved** Access: RO Format: MBZ
1	31:16	**Reserved** Access: RO Format: MBZ
	15:8	**VertOrigin** Vertical Origin In unit of macroblocks relative to the current picture (frame or field).
	7:0	**HorzOrigin** Horizontal Origin in unit of macroblocks.
2	31:16	**Motion Vectors - Field 0, Forward, Vertical Component** Each vector component is a 16-bit two's-complement value. The vector is relative to the current macroblock location. According to ISO/IEC 13818-2 Table 7-8, the valid range of each vector component is [-2048, +2047.5], implying a format of s11.1. However, it should be noted that motion vector values are sign extended to 16 bits.
	15:0	**Motion Vectors - Field 0, Forward, Horizontal Component**
3	31:16	**Motion Vectors - Field 0, Backward, Vertical Component**
	15:0	**Motion Vectors - Field 0, Backward, Horizontal Component**
4	31:16	**Motion Vectors - Field 1, Forward, Vertical Component**
	15:0	**Motion Vectors - Field 1, Forward, Horizontal Component**
5	31:16	**Motion Vectors - Field 1, Backward, Vertical Component**
	15:0	**Motion Vectors - Field 1, Backward, Horizontal Component**

Inline Data Description - VP8 PAK OBJECT

Inline Data Description - VP8 PAK OBJECT

Source:	VideoCS
Size (in bits):	384
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

This structure corresponds to Dw3..6 of MFX_VP8_PAK_OBJECT Command.

DWord	Bit	Description
0	31:23	**Reserved** Access: RO Format: MBZ
	22:20	**MV Format(Motion Vector Size)** Exists If: //IntraMbFlag = 0 This field specifies the size and format of the output motion vectors. Value / Name / Description: 000b / Intra MB / No Motion vectors 100b / Inter Predict MB (Unpacked Motion Vector Mode) / Sixteen Motion Vectors Per MacroBlock Others / Reserved / **Programming Notes** This field MBZ, when the **IntraMbFlag = 1**.
	19:18	**SegmentID** Format: U2 Segment number 0-3
	17	**Enable Coeff Clamp** Value / Name / Description: 1 / / Magnitude of coefficients of the current MB is clamped based on the clamping matrix after quantization 0 / / No Clamping
	16:14	**Reserved** Access: RO Format: MBZ
	13	**Intra MB Flag** This field specifies whether the current macroblock is an Intra (I) Macroblock. For Key pictures (IsKyeFrameFlag DW2, bit[5] of MFX_VP8_PIC_STATE), this field must be set to 1.

Inline Data Description - VP8 PAK OBJECT

Value	Name
0h	INTER (Inter MacroBlock)
1h	INTRA (Intra MacroBlock)

Programming Notes
For I-picture MB (Intra MB Flag =1), this field must be set to 1.

12:11 **RefPicSelect**

This field specifies which reference pic (among Last Frame, Golden Frame and Alt Frame) is selected for the current macroblock when Intra MB Flag = 0.

Value	Name
00b	Last Frame
01b	Golden Frame
10b	Alt Frame

10:8 **MB Type 3-Bits - Inter/Intra MB**

MB Type 3Bits [10:8] specifiesInterMB MV mode configurations: 16x16 or 2 16x8 or 4 8x8 or 16 4x4 when Intra MB Flag = 0 and bit [8] = IntraMB mode configurations: 4x4 or 16x16 when Intra MB Flag = 1

Value	Name	Description
000b	16x16	**Inter MB**Only DW 6 bits 3:0 are used to indicate MVMode, MVMode can't be split
001b	2 16x8 (mv_Top Bottom)	**Inter MB [10:8]**Split MV is inferred. DW5 bits[3:0] are used for MVMode for first 16x8 partition, DW6 bits[3:0] are used for MVMode for second 16x8 partition.
010b	2 8 x16 (mv_left_right)	**Inter MB [10:8]**Split MV is inferred. DW5 bits[3:0] are used for MVMode for first 8x16 partition, DW5 bits[11:8] are used for MVMode for second 8x16 partition.
011b	4 8x8 (mv_quarters)	**Inter MB [10:8]**Split MV is inferred. DW5 bits[3:0] are used for MVMode for first 8x8 partition. DW5 bits[11:8] are used for MvMode for second 8x8 partition. DW6 bits[3:0] are used for MVMode for third 8x8 partition. DW6 bits[11:8] are used for MVMode for fourth 8x8 partition.
100b	16 4x4 (mv_16)	**Inter MB [10:8]**Split MV is inferred. There are 16 partitions. Each Sub-block uses 4 bits in DW6 and DW7.
0b	16x16	**Intra MB [8]**Only DW5, bits[3:0] are used for Y mode. For B_PRED, "16 4x4" should be used which implies B_PRED mode.
1b	16 4x4	**Intra MB [8]**All bits in DW5 and DW6 are used to represent B_PRED modes (Bmodes) in each sub-blocks.

7:6 **Reserved**

Access:	RO
Format:	MBZ

Inline Data Description - VP8 PAK OBJECT

	5:4	**MB UV Mode** **Value** / **Name** 0 – DC_PRED 1 – V_PRED 2 – H_PRED 3 – TM_PRED
	3	**Reserved** Access: RO Format: MBZ
	2	**Skip MB Flag** This field is equivalent to mb_skip_flag in VP8 spec. **Programming Notes** By setting this field to 1, it forces an Inter MacroBlock to be encoded as a skipped MacroBlock
	1:0	**Reserved** Access: RO Format: MBZ
1	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**MbYCnt (Vertical Origin)** Format: U8 This field specifies the vertical origin of current macroblock in the destination picture in units of macroblocks.
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**MbXCnt (Horizontal Origin)** Format: U8 This field specifies the horizontal origin of current macroblock in the destination picture in units of macroblocks.
2	31:28	**B Mode for SubBlock7 (Y mode for the macroblock in non-B mode)** For Y-Mode and B-Mode Assignments refer to the assignment lists below this table.
	27:24	**B Mode for SubBlock6 (Y mode for the macroblock in non-B mode)** For Y-Mode and B-Mode Assignments refer to the assignment lists below this table.
	23:20	**B Mode for SubBlock5 (Y mode for the macroblock in non-B mode)** For Y-Mode and B-Mode Assignments refer to the assignment lists below this table.

Inline Data Description - VP8 PAK OBJECT

	19:16	**B Mode for SubBlock4 (Y mode for the macroblock in non-B mode)** For Y-Mode and B-Mode Assignments refer to the assignment lists below this table.
	15:12	**B Mode for SubBlock3 (Y mode for the macroblock in non-B mode)** For Y-Mode and B-Mode Assignments refer to the assignment lists below this table.
	11:8	**B Mode for SubBlock2 (Y mode for the macroblock in non-B mode)** For Y-Mode and B-Mode Assignments refer to the assignment lists below this table.
	7:4	**B Mode for SubBlock1 (Y mode for the macroblock in non-B mode)** For Y-Mode and B-Mode Assignments refer to the assignment lists below this table.
	3:0	**B Mode for SubBlock0 (Y mode for the macroblock in non-B mode)** For Y-Mode and B-Mode Assignments refer to the assignment lists below this table.
3	31:28	**B Mode for SubBlock15 (Y mode for the macroblock in non-B mode)** For Y-Mode and B-Mode Assignments refer to the assignment lists below this table.
	27:24	**B Mode for SubBlock14(Y mode for the macroblock in non-B mode)** For Y-Mode and B-Mode Assignments refer to the assignment lists below this table.
	23:20	**B Mode for SubBlock13(Y mode for the macroblock in non-B mode)** For Y-Mode and B-Mode Assignments refer to the assignment lists below this table.
	19:16	**B Mode for SubBlock12(Y mode for the macroblock in non-B mode)** For Y-Mode and B-Mode Assignments refer to the assignment lists below this table.
	15:12	**B Mode for SubBlock11(Y mode for the macroblock in non-B mode)** For Y-Mode and B-Mode Assignments refer to the assignment lists below this table.
	11:8	**B Mode for SubBlock10 (Y mode for the macroblock in non-B mode)** For Y-Mode and B-Mode Assignments refer to the assignment lists below this table.
	7:4	**B Mode for SubBlock9 (Y mode for the macroblock in non-B mode)** For Y-Mode and B-Mode Assignments refer to the assignment lists below this table.
	3:0	**B Mode for SubBlock8 (Y mode for the macroblock in non-B mode)** For Y-Mode and B-Mode Assignments refer to the assignment lists below this table.
4	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**MV Y FWD 0** Format: S13 The value of the y component of this motion vector for FWD block 0. Max value +/-1024 full pel (+/- 8192 1/8th pel) precision
	15:14	**Reserved** Access: RO Format: MBZ

Inline Data Description - VP8 PAK OBJECT

	13:0	**MV X FWD 0** Format: S13 The value of the x component of this motion vector for FWD block 0. Max value +/-1024 full pel (+/- 8192 1/8th pel) precision
5	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**MV Y FWD 1** Format: S13 The value of the y component of this motion vector for FWD block 1.
	15:14	**Reserved** Access: RO Format: MBZ
	13:0	**MV X FWD 1** Format: S13 The value of the x component of this motion vector for FWD block 1.
6	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**MV Y FWD 2** Format: S13 The value of the y component of this motion vector for FWD block 2.
	15:14	**Reserved** Access: RO Format: MBZ
	13:0	**MV X FWD 2** Format: S13 The value of the x component of this motion vector for FWD block 2.
7	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**MV Y FWD 3** Format: S13 The value of the y component of this motion vector for FWD block 3.

Inline Data Description - VP8 PAK OBJECT

	15:14	**Reserved** Access: RO Format: MBZ
	13:0	**MV X FWD 3** Format: S13 The value of the x component of this motion vector for FWD block 3.
8	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**MV Y BWD 0** Format: S13 The value of the y component of this motion vector for BWD block 0.
	15:14	**Reserved** Access: RO Format: MBZ
	13:0	**MV X BWD 0** Format: S13 The value of the x component of this motion vector for BWD block 0.
9	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**MV Y BWD 1** Format: S13 The value of the y component of this motion vector for BWD block 1.
	15:14	**Reserved** Access: RO Format: MBZ
	13:0	**MV X BWD 1** Format: S13 The value of the x component of this motion vector for BWD block 1.
10	31:30	**Reserved** Access: RO Format: MBZ

Inline Data Description - VP8 PAK OBJECT

	29:16	**MV Y BWD 2** Format: S13 The value of the y component of this motion vector for BWD block 2.
	15:14	**Reserved** Access: RO Format: MBZ
	13:0	**MV X BWD 2** Format: S13 The value of the x component of this motion vector for BWD block 2.
11	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**MV Y BWD 3** Format: S13 The value of the y component of this motion vector for BWD block 3.
	15:14	**Reserved** Access: RO Format: MBZ
	13:0	**MV X BWD 3** Format: S13 The value of the x component of this motion vector for BWD block 3.

INTERFACE_DESCRIPTOR_DATA

INTERFACE_DESCRIPTOR_DATA

Source: RenderCS

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:6	**Kernel Start Pointer** Format: InstructionBaseOffset[31:6] Specifies the 64-byte aligned address offset of the first instruction in the kernel. This pointer is relative to the **Instruction Base Address.**
	5:0	**Reserved** Access: RO Format: MBZ
1	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Kernel Start Pointer High** Format: InstructionBaseOffset[47:32] This field specifies the high 16 bits of starting address of the Kernel Pointer.
2	31:21	**Reserved** Access: RO Format: MBZ
	20	**Thread Preemption disable** This field specifies whether, when dispatched, the thread is allowed to stop in middle on receiving mid-thread pre-emption request. Value / Name / Description 0h / Disable **[Default]** / Thread is pre-empted on receiving pre-emption indication. 1h / Enable / Thread is preempted only in case of page-fault.
	19	**Denorm Mode** This field specifies how Float denormalized numbers are handles in the dispatched thread. Value / Name / Description 0h / Ftz / Float denorms will be flushed to zero when appearing as inputs; denorms will never come out of instructions. Double precision float and half precision float numbers are not flushed to zero. 1h / SetByKernel / Denorms will be handled in by kernel.

INTERFACE_DESCRIPTOR_DATA

18 Single Program Flow

Specifies whether the kernel program has a single program flow (SIMDnxm with m = 1) or multiple program flows (SIMDnxm with m > 1).

Value	Name
0h	Multiple
1h	Single

17 Thread Priority

Specifies the priority of the thread for dispatch.

Value	Name
0h	Normal Priority
1h	High Priority

16 Floating Point Mode

Specifies the floating point mode used by the dispatched thread.

Value	Name
0h	IEEE-754
1h	Alternate

15:14 Reserved

Access:	RO
Format:	MBZ

13 Illegal Opcode Exception Enable

Format:	Enable

This bit gets loaded into EU CR0.1[12] (note the bit # difference). See *Exceptions and ISA Execution Environment*.

12 Reserved

Access:	RO
Format:	MBZ

11 Mask Stack Exception Enable

Format:	Enable

This bit gets loaded into EU CR0.1[11]. See *Exceptions and ISA Execution Environment*.

10:8 Reserved

Access:	RO
Format:	MBZ

7 Software Exception Enable

Format:	Enable

This bit gets loaded into EU CR0.1[13] (note the bit # difference). See *Exceptions and ISA Execution Environment*.

INTERFACE_DESCRIPTOR_DATA

	6:0	**Reserved** Access: RO Format: MBZ
3	31:5	**Sampler State Pointer** Format: DynamicStateOffset[31:5]SAMPLER_STATE Specifies the 32-byte aligned address offset of the sampler state table. This pointer is relative to the **Dynamic State Base Address.***This field is ignored for child threads.*
	4:2	**Sampler Count** Format: U3 Specifies how many samplers (in multiples of 4) the kernel uses. Used only for prefetching the associated sampler state entries. *This field is ignored for child threads. If this field is not zero, sampler state is prefetched for the first instance of a root thread upon the startup of the media pipeline.*

Value	Name
[0,4]	
0h	No samplers used
1h	Between 1 and 4 samplers used
2h	Between 5 and 8 samplers used
3h	Between 9 and 12 samplers used
4h	Between 13 and 16 samplers used

	1:0	**Reserved** Access: RO Format: MBZ
4	31:16	**Reserved** Access: RO Format: MBZ
	15:5	**Binding Table Pointer**

Format:	SurfaceStateOffset[15:5]SW Generated BINDING_TABLE_STATE*256 When HW Binding Table Alignment is alignment is clear.
Format:	SurfaceStateOffset[16:6]SW Generated BINDING_TABLE_STATE*256 [] When Binding Table Pool is Enabled.
Format:	SurfaceStateOffset[18:8]SW Generated BINDING_TABLE_STATE*256 When HW Binding Table Alignment is alignment set to 512KB size

Description
Specifies a pointer offset into the binding table. *This field is ignored for child threads.* • When Binding Table Pool is Disabled, this pointer is relative to the **Surface State Base Address**.

- When Binding Table Pool is Enabled, this pointer is relative to the **Binding Table PoolBase Address**.

4:0 **Binding Table Entry Count**

Format:	U5

Specifies how many binding table entries the kernel uses. Used only for prefetching of the binding table entries and associated surface state.*This field is ignored for child threads.If this field is not zero, binding table and surface state are prefetched for the first instance of a root thread upon the startup of the media pipeline.*

Value	Name
[0,31]	

Programming Notes
The maximum number of prefetched binding table entries is limited to 31. For kernels using a large number of binding table entries, it may be wise to set this field to zero to avoid prefetching too many entries and thrashing the state cache.

5 31:16 **Constant/Indirect URB Entry Read Length**

Format:	U16

Specifies the amount of URB data read and passed in the thread payload for the Constant orIndirect URB entry, in 8-DW register increments. A value 0 means that no Constant or IndirectURB Entry will be loaded. The Constant URB Entry Read Offset field will then be ignored.In GPGPU mode this describes how much data is delivered in a single dispatch. Multiple dispatchesin a thread group will deliver constant data offset by this value. The total amount of constantdata is (Constant URB Read Length * Number of Threads in GPGPU Thread Group + Cross-ThreadConstant Data Read Length).

If **Cross-Thread Constant Data Read Length** for Indirect is greater than 0,then this field must also be greater than 0. The allowed combinations are:

Constant/Indirect URB Entry Read Length**Entry Read Length**	Cross-Thread Constant **Data Read Length**	Notes
=0	=0	No Payload
>0	=0	Per-thread payload only
>0	>0	Both kinds of payload
=0	>0	Only for CURBE payloads

Value	Name
[0,63]	

INTERFACE_DESCRIPTOR_DATA

15:0 Constant URB Entry Read Offset

Format:	U16

Specifies the offset (in 8-DW units) at which Constant URB data is to be read from the URB before being included in the thread payload.

Value	Name	Description
[0,1983]		Indicating [0,1983] 256-bit register increments. ROB has 64KB of storage; 2048 entries. However, lowest 64 entries are reserved for VFE/TS to store interface descriptor data. Hence, (URB Entry Read Offset + Read Length) shall not exceed 1984.

6 31:24 Reserved

Access:	RO
Format:	MBZ

23:22 Rounding Mode

Format:	U2

Value	Name	Description
00b	RTNE **[Default]**	Round to Nearest Even
01b	RU	Round toward +Infinity
10b	RD	Round toward -Infinity
11b	RTZ	Round toward Zero

21 Barrier Enable

Format:	Enable

This field specifies whether the thread group requires a barrier. If not, it can be dispatched without allocating one.

20:16 Shared Local Memory Size

Format:	U5

This field indicates how much Shared Local Memory the thread group requires. The amount is specified in 4k blocks, but only powers of 2 are allowed: 0, 4k, 8k, 16k, 32k and 64k per half-slice.

uses a different encoding to allow encodings for the new 1k and 2k SLM sizes.

Value	Name	Description
0	Encodes 0K	No SLM used
1	Encodes 1K	
2	Encodes 2K	
3	Encodes 4K	
4	Encodes 8K	

5	Encodes 16K	
6	Encodes 32K	
7	Encodes 64K	

15 — Reserved

Access:	RO
Format:	MBZ

14:13 — Over Dispatch Control

Format:	U2

Over Dispatch Control that enables and controls the amount of GPGPU thread over dispatch.

Value	Name
00b	None **[Default]**
11b	Normal
01b	Low
10b	High

Restriction
Set this over dispatch control to None if Mid Thread Preemption is used.

12:10 — Reserved

Access:	RO
Format:	MBZ

9:0 — Number of Threads in GPGPU Thread Group

Format:	U10

Specifies the number of threads that are in this thread group.

Value	Name	Description
[1,112]		The minimum value is 1, while the maximum value is the number of threads in two subslices for local barriers. See vol1b Configurations for the number of threads per subslice for different products.

7 — 31:8 Reserved

Access:	RO
Format:	MBZ

7:0 — Cross-Thread Constant Data Read Length

Format:	U8

Specifies the amount of constant data in CURBE in 8-DW register increments which will be sent to every thread in the thread group in addition to the per thread ids specified by **Constant URB Entry Read Length**.

Value	Name
[0,127]	

INTERRUPT

INTERRUPT

Size (in bits): 128

Default Value: 0x00000000, 0xFFFFFFFF, 0x00000000, 0x00000000

See the Interrupt Definition Tables to find the source event for each interrupt bit.There are multiple instances of this register format.

DWord	Bit	Description
0	31:0	**ISR** Access: RO These are the Interrupt Status Register Bits.This field contains the non-persistent values of the interrupt status bits.The IMR selects which of these interrupt conditions are reported in the persistent IIR **Value** / **Name** 0b – Condition Doesn't exist 1b – Condition Exists **Restriction** Some inputs to this register are short pulses. Do not use this register to sample these conditions.
1	31:0	**IMR** Access: R/W These are the Interrupt Mask Register Bits.This field contains a bit mask which selects which interrupt bits from the ISR are reported in the IIR. **Value** / **Name** FFFFFFFFh – All interrupts masked **[Default]** 0b – Not Masked 1b – Masked
2	31:0	**IIR** Access: R/WC These are the Interrupt Identity Register Bits.This field holds the persistent values of the interrupt bits from the ISR which are unmasked by the IMR.The IER enables an interrupt to be generated when the corresponding bit in the IIR becomes set. A disabled interrupt will still appear in the IIR.Bits set in this register will remain set (persist) until the interrupt condition is cleared by writing a '1' to the appropriate bits. **Value** / **Name** 0b – Condition Not Detected 1b – Condition Detected

<table>
<tr><th colspan="3">INTERRUPT</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>For each bit, the IIR can store a second pending interrupt if two or more of the same interrupt conditions occur before the first condition is cleared. Upon clearing the first interrupt, the IIR bit will momentarily go low, then return high to indicate there is second interrupt pending.</td></tr>
</table>
</td></tr>
<tr><td>3</td><td>31:0</td><td>IER
<table>
<tr><td>Access:</td><td>R/W</td></tr>
</table>
These are the Interrupt Enable Register Bits.The field enables an interrupt to be generated when the corresponding bit in the IIR becomes set. A disabled interrupt will still appear in the IIR.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0b</td><td>Disabled</td></tr>
<tr><td>1b</td><td>Enabled</td></tr>
</table>
</td></tr>
</table>

Invalidate After Read Message Descriptor Control Field

MDC_IAR - Invalidate After Read Message Descriptor Control Field		
Size (in bits): 1		
Default Value: 0x00000000		
DWord	**Bit**	**Description**
0	0	**Reserved** Access: RO Format: MBZ

JPEG

JPEG		
Source:	VideoCS	
Size (in bits):	16	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	15:5	**Reserved** Access: RO Format: MBZ
	4	**Inconsistent VLD SE Error** This flag indicates an inconsistent SE coded in the bit-stream. Bit-stream does not match any entries in the hauffman table.
	3	**Extra Block Error** This flag indicates extra block coded within an ECS data boundary.
	2	**Missing block Error** This flag indicates one or more blocks are missing within an ECS data boundary.
	1	**Extra ECS Error** This flag indicates extra ECS' coded in the bit-stream SCAN payload data.
	0	**Missing ECS Error** This flag indicates one or more ECS' are missing from the bit-stream SCAN payload data.

LNCF_POWER_CONTEXT

LNCF_POWER_CONTEXT []				
Description	**MMIO Offset/Command**	**Unit**	**# of DW**	**Address Offset(DWord)**
LNCFCMOCS0	0xB020		2	0x0000
LNCFCMOCS1	0xB024		2	0x0002
LNCFCMOCS2	0xB028		2	0x0004
LNCFCMOCS3	0xB02C		2	0x0006
LNCFCMOCS4	0xB030		2	0x0008
LNCFCMOCS5	0xB034		2	0x000A
LNCFCMOCS6	0xB038		2	0x000C
LNCFCMOCS7	0xB03C		2	0x000E
LNCFCMOCS8	0xB040		2	0x0010
LNCFCMOCS9	0xB044		2	0x0012
LNCFCMOCS10	0xB048		2	0x0014
LNCFCMOCS11	0xB04C		2	0x0016
LNCFCMOCS12	0xB050		2	0x0018
LNCFCMOCS13	0xB054		2	0x001A
LNCFCMOCS14	0xB058		2	0x001C
LNCFCMOCS15	0xB05C		2	0x001E
LNCFCMOCS16	0xB060		2	0x0020
LNCFCMOCS17	0xB064		2	0x0022
LNCFCMOCS18	0xB068		2	0x0024
LNCFCMOCS19	0xB06C		2	0x0026
LNCFCMOCS20	0xB070		2	0x0028
LNCFCMOCS21	0xB074		2	0x002A
LNCFCMOCS22	0xB078		2	0x002C
LNCFCMOCS23	0xB07C		2	0x002E
LNCFCMOCS24	0xB080		2	0x0030
LNCFCMOCS25	0xB084		2	0x0032
LNCFCMOCS26	0xB088		2	0x0034
LNCFCMOCS27	0xB08C		2	0x0036
LNCFCMOCS28	0xB090		2	0x0038
LNCFCMOCS29	0xB094		2	0x003A
LNCFCMOCS30	0xB098		2	0x003C
LNCFCMOCS31	0xB09C		2	0x003E
SCRATCH_LNCF2	0xB0A0		2	0x0040

LNCF_POWER_CONTEXT []				
Description	**MMIO Offset/Command**	**Unit**	**# of DW**	**Address Offset(DWord)**
SCRATCH_LNCF3	0xB0A8		2	0x0042
L3NODELOCK	0xB0AC		2	0x0044
LSN_SLCVC	0xB0C8		2	0x0046
LSN_MISC	0xB0CC		2	0x0048
LSN_ARBPRI0	0xB0D0		2	0x004A
LSN_ARBPRI1	0xB0D4		2	0x004C
GAFCFG0	0xD800		2	0x004E
GAFCFG1	0xD804		2	0x0050

LOD Message Address Payload Control

MACD_LOD - LOD Message Address Payload Control

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:4	**Reserved** Access: RO Format: MBZ
	3:0	**LOD** Format: U4 Specifies the LOD for this slot. Value / Name / Description: [0,14] / / representing LOD

Lower Oword Block Data Payload

MDP_OW1L - Lower Oword Block Data Payload		
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.3	127:0	**Oword** Format: U128 Specifies the upper Oword data element
0.4-0.7	127:0	**Reserved** Access: RO Format: MBZ

LPFC_POWER_CONTEXT

LPFC_POWER_CONTEXT []				
Description	**MMIO Offset/Command**	**Unit**	**# of DW**	**Address Offset(DWord)**
L3LPFCLOCK	0xB460		2	0x0000
SCRATCH_LPFC1	0xB474		2	0x0002

LRI Data Entry

LRI_DATA - LRI Data Entry

Source:	RenderCS
Size (in bits):	64
Default Value:	0x00000000, 0x00000000

Each LRI command header is followed by LRI_DATA entries. Each of these entries is a pair of Dwords: the MMIO register address and the data to be written.

DWord	Bit	Description
0..1	63:55	**Reserved** Access: RO Format: MBZ
	54:32	**MMIO** Format: U23 **Programming Notes** Bits [1:0] MBZ
	31:0	**Data** Format: U32

MacroOperand

MacroOperand		
Size (in bits):	14	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	13:6	**RegNum** Format: U8 This field provide the register number for the operand. For a GRF register, is the part of a register address that aligns to a 256-bit (32-byte) boundary. For an ARF register, this field is encoded such that MSBs identify the architecture register type and LSBs provide the register number. An ARF register can only be destination or Source 0. Any Source 1 or Source 2 operands cannot be ARF registers. RegNum and SubRegNum together provide the byte-aligned address for the origin of a register region. RegNum provides bits 12:5 of that address. This field applies to both source and destination operands.
	5	**Reserved** Access: RO Format: MBZ
	4:1	**SpecialAccNum** This field specifies the accumulator numbers used by the IEEE macro instructions (madm and math.invm/math.rsqtm).The 8 special accumulators, acc2 to acc9 are encoded consecutively from 0000b to 0111b and noacc, indicating no special accumulator used is encoded as 1000b
	0	**RegFile** This field indicate whether Architecture register file or General register file are selected. **Value** / **Name** / **Description** 0 / ARF / Architecture Register File.Only allowed for Source 0 or Destination. 1 / GRF / General Register File. Allowed for any Source or Destination.

Manageability Engine Interrupt Vector

CSME_INTR_VEC - Manageability Engine Interrupt Vector

Size (in bits): 16

Default Value: 0x00000000

DWord	Bit	Description
0	15:2	**Reserved** Access: RO Format: MBZ
	1	**CSME Response** Format: U1 CSME sets this bit in the interrupt when responding to GuC initiated transaction for: • Response to wake up request from GuC • Payload message sent toto ME_MESG, ME_DATA for a GuC request
	0	**CSME Request** Format: U1 CSME sets this bit in the interrupt when CSME initiates the transaction for: • CSME to GuC wake up request • Payload message sent toto ME_MESG, ME_DATA for CMSE initiated request

MEDIA_SURFACE_STATE

MEDIA_SURFACE_STATE

Exists If:	//([MessageType] == 'Deinterlace') OR ([MessageType] == 'Sample_8x8')
Size (in bits):	256
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

This is the SURFACE_STATE used by only deinterlace, sample_8x8, and VME messages.

DWord	Bit	Description
0	31:30	**Rotation** Value / Name: 00b — No Rotation or 0 Degree 01b — 90 Degree Rotation 10b — 180 Degree Rotation 11b — 270 Degree Rotation **Programming Notes** Rotation is only supported only with AVS function messages and not with HDC direct write and 16x8 AVS messages.
	29:27	**Reserved** Access: RO Format: MBZ
	26:20	**X Offset** Exists If: //[Surface Format] is one of Planar Formats Format: U7 This field specifies the horizontal offset in pixels from the **Surface Base Address** to the start (origin) of the surface.This field effectively loosens the alignment restrictions on the origin of tiled surfaces. Previously, tiled surface origin was (by definition) located at the base address, and thus needed to satisfy the 4KB base address alignment restriction. Now the origin can be specified at a finer (4-wide x 4-high pixel) resolution. Value / Name / Description: [0,127] — In multiples of 4 (low 2 bits missing) **Programming Notes** For linear surfaces and Packed Formats, this field must be zero. For **Surface Format** with 8 bits per element, this field must be a multiple of 16. For **Surface Format** with 16 bits per element, this field must be a multiple of 8.

MEDIA_SURFACE_STATE

	19:16	**Y Offset** Exists If: //[Surface Format] is one of Planar Formats Format: U4 This field specifies the vertical offset in rows from the **Surface Base Address** to the start of the surface. (See additional description in the **X Offset** field) **Value** / **Name** / **Description** [0,15] / / In multiples of 4 (low two bits missing) **Programming Notes** For linear surfaces and Packed Formats, this field must be zero.
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Reserved** Access: RO Format: MBZ
1	31:18	**Height** Format: U14-1 This field specifies the height of the surface in units of pixels. For PLANAR surface formats, this field indicates the height of the Y (luma) plane. **Value** / **Name** / **Description** [0,16383] / / representing heights [1,16384] **Programming Notes** Height (field value + 1) must be a multiple of 2 for PLANAR_420 surfaces. If Vertical Line Stride is 1, this field indicates the height of the field, not the height of the frame.
	17:4	**Width** Format: U14-1 This field specifies the width of the surface in units of pixels. For PLANAR surface formats, this field indicates the width of the Y (luma) plane. **Value** / **Name** / **Description** [0,16383] / / representing widths [1,16383] **Programming Notes** • The Width specified by this field multiplied by the pixel size in bytes must be less than or equal to the surface pitch (specified in bytes via the Surface Pitch field). • Width (field value + 1) must be a multiple of 2 for PLANAR_420, PLANAR_422, and all YCRCB_* and Y16_UNORM surfaces, and must be a multiple of 4 for PLANAR_411 and Y8_UNORM_VA surfaces.

		• For deinterlace messages, the Width (field value + 1) must be a multiple of 8. Width (field value + 1) must be a multiple of 2 for PLANAR_420_16 For Y16_UNORM format width should be in multiple of 2
	3:2	**Picture Structure** Specifies the encoding of the current picture. Value: 00b – Frame Picture Value: 01b – Top Field Picture Value: 10b – Bottom Field Picture Value: 11b – Invalid, not allowed
	1:0	**Cr(V)/Cb(U) Pixel Offset V Direction** Default Value: 0 Format: U0.2 Specifies the distance to the U/V values with respect to the even numbered Y channels in the V direction Programming Notes: This field is ignored for all formats except for PLANAR_420_8 and PLANAR_420_16 This offset has been increased from 2 bits to 3 bits to support U1.2 format, and the MSB bit is added as Pixel Offset V Direction MSB in DWord 2. Valid values for the combined field range from 0 to 4.
2	31:27	**Surface Format** Specifies the format of the surface. All of the Y and G channels will use table 0 and all of the Cr/Cb/R/B channels will use table 1.

Value	Name	Description
0	YCRCB_NORMAL	
1	YCRCB_SWAPUVY	
2	YCRCB_SWAPUV	
3	YCRCB_SWAPY	
4	PLANAR_420_8	
8	R10G10B10A2_UNORM	Sample_8x8 only
9	R8G8B8A8_UNORM	Sample_8x8 AVS only
10	R8B8_UNORM (CrCb)	Sample_8x8 AVS only
11	R8_UNORM (Cr/Cb)	Sample_8x8 AVS only
12	Y8_UNORM	Sample_8x8 AVS only
13	A8Y8U8V8_UNORM	Sample_8x8 AVS only
14	B8G8R8A8_UNORM	Sample_8x8 AVS only
15	R16G16B16A16	Sample_8x8 AVS only

18	PLANAR_422_8	Sample_8x8 AVS only
23	PLANAR_420_16	Sample_8x8 AVS only
24	R16B16_UNORM (CrCb)	Sample_8x8 AVS only
25	R16_UNORM (Cr/Cb)	Sample_8x8 AVS only
26	Y16_UNORM	Sample_8x8 AVS only
Others	Reserved	

26 **Interleave Chroma**

Format:	Enable

This field indicates that the chroma fields are interleaved in a single plane rather than stored as two separate planes. This field is only used for PLANAR surface formats.

25 **Cr(V)/Cb(U) Pixel Offset U Direction**

Default Value:	0
Format:	U0.1

Specifies the distance to the U/V values with respect to the even numbered Y channels in the U direction

Programming Notes
This field is must be zero for all formats except PLANAR_420_16, PLANAR_420_8, PLANAR_422_8, YCRCB_NORMAL, YCRCB_SWAPUVY, YCRCB_SWAPUV, YCRCB_SWAPY.

24 **Cr(V)/Cb(U) Pixel Offset V Direction MSB**

Default Value:	0
Format:	U1

Specifies the distance to the U/V values with respect to the even numbered Y channels in the V direction

Programming Notes
This field is must be zero for all formats except?PLANAR_420_16 and PLANAR_420_8
This offset has been increased from 2 bits to 3 bits as U1.2 format and this bit is used in conjunction with the bits in the Cr(V)/Cb(U) Pixel Offset V Direction field in DWord 1, which contain the rest of the bits for offset V-direction. Valid values for the combined field range from 0 to 4.

23 **Memory Compression Type**

Specifies the type of memory compression used.

Value	Name
0	Media Compression
1	Render Compression

22 **Memory Compression Enable**

Format:	Enable

This surface may contain compressed or compressible pixels. Memory compression will be attempted for writes to this surface. Reads from this surface will check for compressed data.

Programming Notes
The compression control must have 0 value for non-tileY modes.
Please refer to vol1a Memory Data Formats chapter -- section Media Memory Compression for more details, including format restrictions.

21 **Address Control**

Value	Name	Description
0	CLAMP	Clamp
1	MIRROR	Mirror

20:3 **Surface Pitch**

Format:	U18-1

This field specifies the surface pitch in (#Bytes - 1).

Value	Name	Description
[0,262143]		For other linear surfaces: representing [1B, 256KB]
[511, 262143]		For X-tiled surface: representing [512B, 256KB] = [1 tile, 512 tiles]
[127, 262143]		For Y-tiled surfaces: representing [128B, 256KB] = [1 tile, 2048 tiles]

Programming Notes
For tiled surfaces, the pitch must be a multiple of the tile widthIf Half Pitch for Chroma is set, this field must be a multiple of two tile widths for tiled surfaces, or a multiple of 2 bytes for linear surfaces.The Surface Pitches of current picture and reference picture should be declared as the identical type in VDI mode with identical Height, Width and Format.
If Media Memory Compression is enabled, the following max pitch size restriction must be honored. For larger resolution, Media Memory compression Must be disabled. Tiling Mode Pixel Format Max Frame Width (bytes) Max Frame Width (pixels) Max Pitch (bytes) Legacy 4K 8bpp 16k 16k 16k + 127 16bpp 16k 8k 16k + 127 32bpp 16k 4k 16k + 127 64bpp 16k 2k 16k + 127 128bpp 16k 1k 16k + 127 TileYF 8bpp 8k 8k 8k + 63 16bpp 16k 8k 16k + 127 32bpp 16k 4k 16k + 127 64bpp 16k 2k 16k + 255 128bpp 16k 1k 16k + 255 TileYS 8bpp 16k 16k 16k + 255 16bpp 16k 8k 16k + 511 32bpp 16k 4k 16k + 511 64bpp 16k 2k 16k + 1023 128bpp 16k 1k 16k + 1023

2 **Half Pitch for Chroma**

Format:	Enable

This field indicates that the chroma plane(s) will use a pitch equal to half the value specified in the Surface Pitch field. This field is only used for PLANAR surface formats.

Programming Notes
Must be Zero as this field is not used.

1:0 **Tile Mode**

Format:	U2

This field specifies the type of memory tiling (Linear, WMajor, XMajor, or YMajor) employed to tile this surface. See Memory Interface Functions for details on memory tiling and restrictions.

Value	Name	Description

		0h	TILEMODE_LINEAR	Linear mode (no tiling)
		1h	Reserved	Reserved
		2h	TILEMODE_XMAJOR	X major tiling
		3h	TILEMODE_YMAJOR	Y major tiling

Programming Notes

- Refer to *Memory Data Formats* for restrictions on TileMode direction for the various buffer types. (Of particular interest is the fact that YMAJOR tiling is not supported for display/overlay buffers).
- The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this field.
- Linear surfaces can be mapped to Main Memory (uncached) or System Memory (cacheable, snooped). Tiled (X/Y/W) surfaces can only be mapped to Main Memory.

3	31:30	**Reserved**	
		Access:	RO
		Format:	MBZ
	29:16	**X Offset for U(Cb)**	
		Format:	U14
		For non planar surfaces this field specifies the horizontal offset in pixels from the Surface Base Address to the start (origin) of the surface.	
		For Planar surfaces this field specifies the horizontal offset in pixels from the Y-plane origin to the start (origin) of the U(Cb) plane or the interleaved UV plane if Interleave Chroma is enabled. Resultant X-offset = 'X-offset of the surface (Y-plane)' + 'X offset for U(Cb)'	
		For TileYS and TileYF this offset should be integral multiple of Tile width of Luma plane.	
		Programming Notes	
		For PLANAR_420 and PLANAR_422 surface formats, this field must indicate an even number of pixels.	
	15:14	**Reserved**	
		Access:	RO
		Format:	MBZ
	13:0	**Y Offset for U(Cb)**	
		Format:	U14
		For non planar surfaces this field specifies the vertical offset in pixels from the Surface Base Address to the start (origin) of the surface.	
		For Planar surfaces this field specifies the vertical offset in rows from the Y-plane origin to the start (origin) of the U(Cb) plane or the interleaved UV plane if Interleave Chroma is enabled.	

		Resultant X-offset = 'Y-offset of the surface (Y-plane)' + 'Y offset for U(Cb)' For TileYS and TileYF this offset should be integral multiple of Tile width of Luma plane. **Programming Notes** This field must be aligned by 4 bit[1:0] = 00 This field must be aligned by 4 bit[1:0] = 00 for all format besides PLANAR_420_*
4	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**X Offset for V(Cr)** Exists If: //([Surface Format] is one of planar) AND ([Interleave Chroma] == '0') Format: U14 For Planar surfaces this field specifies the horizontal offset in pixels from the Y-plane origin to the start (origin) of the V(Cb) plane. Resultant X-offset = 'X-offset of the surface (Y-plane)' + 'X offset for V(Cb)' For TileYS and TileYF this offset should be integral multiple of Tile width of Luma plane. **Programming Notes** For PLANAR_420 and PLANAR_422 surface formats, this field must indicate an even number of pixels.
	15	**Reserved** Access: RO Format: MBZ
	14:0	**Y Offset for V(Cr)** Exists If: //([Surface Format] is one of planar) AND ([Interleave Chroma] == '0') Format: U15 For Planar surfaces this field specifies the vertical offset in rows from the Y-plane origin to the start (origin) of the V(Cb) plane. Resultant Y-offset = 'Y-offset of the surface (Y-plane)' + 'Y offset for V(Cb)' For TileYS and TileYF this offset should be integral multiple of Tile width of Luma plane. **Programming Notes** This field must indicate a multiple of 4 (bit 0 & 1 = 00).
5	31	**Vertical Line Stride** Format: U1 For Surfaces accessed via the sample_8x8 message:Specifies number of lines (0 or 1) to skip between logically adjacent lines - provides support of interleaved (field) surfaces as textures.For

		Other Surfaces:Vertical Line Stride must be zero.
	30	**Vertical Line Stride Offset** Format: U1 For Surfaces accessed via the sample_8x8 message: Specifies the offset of the initial line from the beginning of the buffer. For Other Surfaces: Vertical Line Stride Offset must be zero. **Programming Notes** This field must be set to 0 if Vertical Line Stride is 0.
	29:20	**Reserved** Access: RO Format: MBZ
	19:18	**Tiled Resource Mode** Format: U2 **For Sampling Engine, Render Target, and Typed/Untyped Surfaces:** This field specifies the tiled resource mode. **For other surfaces:** This field is ignored. **Value** / **Name** / **Description** 0h / TRMODE_NONE / No tiled resource 1h / TRMODE_TILEYF / 4KB tiled resources 2h / TRMODE_TILEYS / 64KB tiled resources 3h / Reserved / **Programming Notes** If **Tile Mode** is not set to TILEMODE_YMAJOR, this field must be set to TRMODE_NONE. If this field is not set to TRMODE_NONE, the **Surface Format** must be one with 8, 16, 32, 64, or 128 bits per element, or one of the compressed texture modes (BC*, ETC*, EAC*, ASTC*). Additionally, YCRCB* formats are supported and treated as 16 bits per element, and the PLANAR_420_8 and PLANAR_422_8 formats are supported and treated as 8 bits per element on the Y plane and 16 bits per element on the UV plane (if **Interleave Chroma** is enabled) or 8 bits per element on the U and V planes (if **Interleave Chroma** is disabled.
	17:7	**Reserved** Access: RO Format: MBZ
	6:0	**Surface Memory Object Control State** Default Value: 0h DefaultVaueDesc Format: **MEMORY_OBJECT_CONTROL_STATE** This 7-bit field is used in various state commands and indirect state objects to define cacheability and other attributes related to memory objects.

MEDIA_SURFACE_STATE

6	31:0	**Surface Base Address** Format: GraphicsAddress[31:0] Specifies the low 32 bits of the byte-aligned base address of the surface. **Programming Notes** For SURFTYPE_BUFFER render targets, this field specifies the base address of first element of the surface. The surface is interpreted as a simple array of that single element type. The address must be naturally-aligned to the element size (e.g., a buffer containing R32G32B32A32_FLOAT elements must be 16-byte aligned).For SURFTYPE_BUFFER non-rendertarget surfaces, this field specifies the base address of the first element of the surface, computed in software by adding the surface base address to the byte offset of the element in the buffer.Mipmapped, cube and 3D sampling engine surfaces are stored in a 'monolithic' (fixed) format, and only require a single address for the base texture.Linear render target surface base addresses must be element-size aligned, for non-YUV surface formats, or a multiple of 2 element-sizes for YUV surface formats. Other linear surfaces have no alignment requirements (byte alignment is sufficient.)Linear depth buffer surface base addresses must be 64-byte aligned. Note that while render targets (color) can be SURFTYPE_BUFFER, depth buffers cannot.Tiled surface base addresses must be 4KB-aligned. Note that only the offsets from Surface Base Address are tiled, Surface Base Address itself is not transformed using the tiling algorithm.For tiled surfaces, the actual start of the surface can be offset from the Surface Base Address by the X Offset and Y Offset fields.Certain message types used to access surfaces have more stringent alignment requirements. Please refer to the specific message documentation for additional restrictions.
7	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Surface Base Address High** Format: GraphicsAddress[47:32] Specifies the high 16 bits of the byte-aligned base address of the surface. Refer to Surface Base Address [31:0] for programming notes applying to this field.

MEMORY_OBJECT_CONTROL_STATE

MEMORY_OBJECT_CONTROL_STATE		
Size (in bits):	7	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	6:1	**Index to MOCS Tables** The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers.The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime. **Programming Notes** When an access is made through Data Port and the index to MOCS[6:1] = [48,59] (decimal), that surface or stateless memory access can be cached in HDC L1 cache. Accesses made through Data Port with MOCS[6:1] < 48 or > 59 will bypass the HDC L1 cache. This bypass is useful when software wants to ensure that Data Port access are coherent with the L3 memory.
	0	**Reserved**

MemoryAddressAttributes

MemoryAddressAttributes

Size (in bits): 32

Default Value: 0x00000000

This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. It defines the attributes for VDBOX addresses.

DWord	Bit	Description
0	31:15	**Reserved** Access: RO Format: MBZ
	14:13	**Base Address - Tiled Resource Mode** Format: U2 **For Media Surfaces:** This field specifies the tiled resource mode. Value / Name / Description: 00b / TRMODE_NONE / TileY resources 01b / TRMODE_TILEYF / 4KB tiled resources 10b / TRMODE_TILEYS / 64KB tiled resources 11b / Reserved /
	12	**Base Address - Row Store Scratch Buffer Cache Select** Format: U1 Description: This field controls if the Row Store is going to store inside Media Cache (rowstore cache) or to LLC. When this is programmed to "1" (going to Media Cache), the corresponding base address will be programmed with the starting position in the media cache. The programming table is in "Buffer Size Requirement Page" in HEVC section Value / Name / Description: 0 / / Buffer going to LLC. 1 / / Buffer going to Internal Media Storage.
	11	**Reserved** Access: RO Format: MBZ
	10	**Compression Type** Indicates if buffer is render/media compressed.

MemoryAddressAttributes

		Value / **Name** 0 — Media Compression Enable **[Default]** 1 — Render Compression Enable
	9	**Base Address - Memory Compression Enable** Format: Enable Memory compression will be attempted for this surface.
	8:7	**Base Address - Arbitration Priority Control** Format: HEVC_ARBITRATION_PRIORITY
	6:1	**Base Address - Index to Memory Object Control State (MOCS) Tables** Format: U6 The index to define the L3 and system cache memory properties. The details of the controls arefurther defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Relatedcontrol registers can be updated during runtime.
	0	**Reserved**

Merged Media Block Message Header Control

MHC_MBM_CONTROL - Merged Media Block Message Header Control

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:30	**Message Mode** Specifies the Media Block Read message is Normal subtype. Value / Name / Description: 00h / Normal / The Block Height and Block Width fields are specified in this Dword. The Mask is ignofed by a media block read message. Others / Reserved / Reserved.
	29	**Reserved** Access: RO Format: MBZ
	28:24	**Sub-Register Offset** Format: U5 Provides the sub-register offset in unit of bytes of a Merged Media Block Read message. This field is ignored (reserved) for a media block write message. Range = [0, 28]. Only a multiple of BasePitch, including 0, is valid. **Programming Notes** Sub-Register Offset and Register Pitch Control allow software to assembly multiple media block reads directly into a shared GRF register set. For example, if both are set to zero, the read data are written to GRF registers, aligning to the least significant bits of the first register, and the register pitch is equal to the next power-of-2 that is greater than or equal to the Block Width. If Register Pitch Control is non-zero, multiple media block read messages sharing the same Register Pitch Control but with different Sub-Register Offset can fill in the same set of GRF registers with media block data line interleaved. **Restriction** For the Sampler Cache Data, this field must be zero. BasePitch is defined as the next the power-of-2 that is greater than or equal to the Block Width. Minimum BasePitch is 1 DWord. Sub-Register Offset must be aligned to BasePitch (therefore will be a multiple of DWords as well). When Register Pitch Control = 0, Sub-Register Offset must align to BasePitch*Block Height. ensuring the output fits in a single GRF register. In general (and specifically when Sub-Register Offset is greater than 0), when the resulting data will cross a GRF register boundary, the data must be placed symmetrically between GRF registers.

MHC_MBM_CONTROL - Merged Media Block Message Header Control

	23:22	**Reserved** Access: RO Format: MBZ
	21:16	**Block Height** Format: U6 Height in rows of block being accessed. Range = [0,63] representing 1 to 64 rows **Restriction** If Block Width (bytes), then Maximum Block Height (rows) is constrained by (# Dwords width) * (# rows) <= 64 Dwords.
	15:10	**Reserved** Access: RO Format: MBZ
	9:8	**Register Pitch Control** Format: U2 Controls the register pitch for a Merged Media Block Read message. This field is ignored (reserved) for a media block write message. Register Pitch Control is only allowed to be non-zero when Block Width is a multiple of DWords. Restriction : For the Sampler Cache Data, this field must be zero.

Value	Name	Description
0h	RPC_1 **[Default]**	1 Block
1h	RPC_2	2 Blocks
3h	RPC_4	4 Blocks

Restriction
BasePitch is defined as the next the power-of-2 that is greater than or equal to the Block Width. The effective register pitch (RPC*BasePitch)+SRO must be less than or equal to 32 bytes (to fit in a single GRF register).

	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**Block Width** Format: U6 Width in bytes of the block being accessed. Range = [0,31] representing 1 to 32 Bytes.

Message Descriptor - Render Target Write

Message Descriptor - Render Target Write

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Data Format** Format: U1 Value / Name / Description: 0 – Single Precision – 32b 1 – Half Precision – 16b **Programming Notes** This field is applicable for Render Target Write Messages ONLY.
	29:14	**Reserved** Access: RO Format: MBZ
	13	**Per-Sample PS outputs enable** This bit must not be set when Render Target is not bound to pixel-shader OR when Render Target is not multisampled. This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. By setting this bit, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot. When Render Target is multisampled and this bit is reset, Render Target outputs color, depth(optional) and stencil(optional) at pixel frequency. It should be noted that the latter case is applicable for only per-pixel PS invocation.
	12	**Last Render Target Select** This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. **Programming Notes** In general, when threads are not launched by 3D FF, this bit must be zero.
	11	**Slot Group Select** This field selects whether slots 15:0 or slots 31:16 are used for bypassed data. Bypassed data includes the antialias alpha, multisample coverage mask, and if the header is not

present also includes the X/Y addresses and pixel enables. For 8- and 16-pixel dispatches, SLOTGRP_LO must be selected on every message. For 32-pixel dispatches, this field must be set correctly for each message based on which slots are currently being processed.

Value	Name	Description
0	SLOTGRP_LO	choose bypassed data for slots 15:0
1	SLOTGRP_HI	choose bypassed data for slots 31:16

Programming Notes
For SIMD8 Image Write message thsi field MBZ.

10:8 **Message Type**

This field specifies the type of render target message. For the SIMD8_DUALSRC_xx messages, the low bit indicates which slots to use for the pixel enables, X/Y addresses, and oMask.

Value	Name	Description
000b	SIMD16	SIMD16 single source message
001b	SIMD16_REPDATA	SIMD16 single source message with replicated data
010b	SIMD8_DUALSRC_LO	SIMD8 dual source message, use slots 7:0
011b	SIMD8_DUALSRC_HI	SIMD8 dual source message, use slots 15:8
100b	SIMD8_LO	SIMD8 single source message, use slots 7:0
111b	SIMD16_REPDATA_TM	It's only supported when accessing *Tiled Memory*. Using this Message Type to access linear *(Untiled)* memory is UNDEFINED.

Programming Notes
the above slots indicated are within the 16 slots selected by **Slot Group Select**. If SLOTGRP_HI is selected, the SIMD8 message types above reference slots 23:16 or 31:24 instead of 7:0 or 15:8, respectively.
SIMD16_REPDATA message must not be used in SIMD8 pixel-shaders.

7:0 **Reserved**

Access:	RO
Format:	MBZ

Message Descriptor - Sampling Engine

Message Descriptor - Sampling Engine

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31	**EOT**
	30	**Return Format** Format: U1 **Value / Name / Description** 0 / 32-bit / Return data is 32b 1 / 16-bit / Return data is 16b **Programming Notes** This field must be set to 32 for resinfo, LOD and sampleinfo messages.
	29	**SIMD Mode[2]** Format: U1 This field is the upper bit of the 3-bit SIMD Mode field.
	28:25	**Message Length** Format: U4 This field specifies the number of 256-bit GRF registers starting from (src) to be sent out on the request message payload. **Value / Name** [1,15] /
	24:20	**Response Length** Format: U5 This field indicates the number of 256-bit registers expected in the message response. **Value / Name** [0,16] / **Programming Notes** A value 0 indicates that the request message does not expect any response. The largest response supported is 16 GRF registers. This field must be programmed to 0 for a surface which is marked as a Procedural Texture (AMFS)

Message Descriptor - Sampling Engine

19 **Header Present**

Format:	Enable

Specifies whether the message includes a header phase. If the header is not present (this field is zero), all of the fields normally contained in the header are assumed to be 0.

If the header is not present, in some cases the **Write Channel Mask** fields are set according to the Response Length.

SIMD16 or**Return Format** Float16

Response Length	Mask R	Mask G	Mask B	Mask A
1	0	1	1	1
2	0	0	1	1
3	0	0	0	1
4	0	0	0	0

Else

Response Length	Mask R	Mask G	Mask B	Mask A
2	0	1	1	1
4	0	0	1	1
6	0	0	0	1
8	0	0	0	0

18:17 **SIMD Mode[1:0]**

Format:	U2

Description

Specifies the SIMD mode of the message being sent.

A third bit SIMD Mode[2] is added to this field (bit 29 of message descriptor). Encodings now range from 0-7. SIMD Mode[2:0] SIMD
000 Reserved
001 SIMD8
010 SIMD16
011 SIMD32/64
100 Reserved
101 SIMD8H
110 SIMD16H
111 Reserved

16:12 **Message Type**

Format:	U5

Specifies the type of message being sent. For more details, please refer to **Message Format** section for the definition of these 5 bits..

Message Descriptor - Sampling Engine

	11:8	**Sampler Index** Format: U4 Specifies the index into the sampler state table. Ignored for ld, resinfo, sampleinfo, and cache_flush type messages. **Value** / **Name** [0,15] /
	7:0	**Binding Table Index** Format: U8 Specifies the index into the **binding table**. Ignored for cache_flush type messages. Values of 255 and 253 indicate stateless. 254 indicates SLM. 252 indicates bindless. **Value** / **Name** [0,255] /

MFD_MPEG2_BSD_OBJECT Inline Data Description

MFD_MPEG2_BSD_OBJECT Inline Data Description

Source:	VideoCS
Size (in bits):	64
Default Value:	0x00000000, 0x00000000

DW0..1 corresponds to DW3..4 of the MFD_MPEG2_BSD_OBJECT.

DWord	Bit	Description
0	31:24	**Slice Horizontal Position** Format: U8 This field indicates the horizontal position of the first macroblock in the slice.
	23:16	**Slice Vertical Position** Format: U8 This field indicates the vertical position of the first macroblock in the slice.
	15:8	**Macroblock Count** Format: U8 This field indicates the number of macroblocks in the slice, including skipped macroblocks.
	7	**Slice Concealment Override Bit** This bit forces hardware to handle the current slice in Conceal or Deocde Mode. If this bit is set to one, VIN will force the current slice to do concealment or to decode from bitstream regardless if the slice boundary has errors or not. Value / Name / Description: 1h — VIN will use driver-provided "Slice Concealment Type" regardless of valid slice boundary 0h — Driver must program "Slice Concealment Type" to '0'. VIN will set "Slice Concealment Type" depending if the slice boundary has error or not
	6	**Slice Concealment Type Bit** This bit can be forced by driver ("Slice Concealment Override Bit") or set by VINunit depending on slice boundary errors. Value / Name / Description: 1h — VMD will conceal all MBs of the slice regardless of bitstream. (If driver does not force the value of this bit, VIN will set this bit depending on slice boundary error. If the next slice position of the current slice is out-of-bound or the same or earlier than the current slice start position, VIN will set this bit for the next slice) 0h — VMD will decode MBs from the bitstream until the bitstream is run-out. Then VMD will conceal the remaining MBs.

MFD_MPEG2_BSD_OBJECT Inline Data Description

		Programming Notes VIN can turn this bit from 0 to 1 internally if "Slice Concealment Disable Bit" is "0" and VIN detects slice boundary errors.
	5	**Last Pic Slice** This bit is added to support error concealment at the end of a picture. **Value** / **Name** / **Description** 1h / / The current Slice is the last Slice of the entire picture 0h / / The current Slice is not the last Slice of current picture
	4	**Reserved**
	3	**Is Last MB** **Value** / **Name** / **Description** 1h / / The current MB is the last MB in the current Slice 0h / / The current MB is not the last MB in the current Slice
	2:0	**First Macroblock Bit Offset** Format: U3 This field provides the bit offset of the first macroblock in the first byte of the input bitstream.
1	31:29	**Reserved** Access: RO Format: MBZ
	28:24	**Quantizer Scale Code** Format: U5 This field sets the quantizer scale code of the inverse quantizer. It remains in effect until changed by a decoded quantizer scale code in a macroblock. This field is decoded from the slice header by host software.
	23:17	**Reserved** Access: RO Format: MBZ
	16:8	**Next Slice Vertical Position** Format: U9 This field indicates the vertical position (in macroblock units) of the first macroblock in the next slice. **Programming Notes** This field is primarily used for error concealment. In the case that current slice is the last slice, this field should set to the height of the picture (field picture will be in height of field) (since y-direction is zero-based numbering).

MFD_MPEG2_BSD_OBJECT Inline Data Description		
	7:0	**Next Slice Horizontal Position** Format: U8 This field indicates the horizontal position (in macroblock units) of the first macroblock in the next slice. **Programming Notes** This field is primarily used for error concealment. In the case that current slice is the last slice, this field should set 0.

MFX_REFERENCE_PICTURE_BASE_ADDR

MFX_REFERENCE_PICTURE_BASE_ADDR

Source: VideoCS
Size (in bits): 64
Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0..1	63:48	**Reserved** Access: RO Format: MBZ
	47:32	**MFX Reference Picture Address [n] High** Format: GraphicsAddress[47:32] This field is for the upper range of Reference Picture Addresses
	31:6	**MFX Reference Picture Address [n]** Format: GraphicsAddress[31:6] Specifies the 64 byte aligned reference frame buffer addresses for the motion compensation operation in AVC//MPEG2. AVC can specify up to 16 YUV frame-based surfaces for both forward and backward references, i.e. L0+L1 total = 16 max. Any entry can be assigned to L0 or L1 or both lists.But VC1 and MPEG2, worst case, can use up to 2 YUV frame-based surfaces for both forward and backward references: • P-MB : RefAddr[0] - temporal closest previous field of a reference frame (can be the current frame) • RefAddr[1]- next temporal closest previous field of a reference frame (must be different from the current frame) It is a variant (without the LongTermRefPic specification) of the RefFrameList[16] defined in AVC DXVA Spec. RefAddr[0-15] is indexed by frame_storeID »1. It is not a packed list, i.e. invalid entries can scatter among the list. All invalid addresses must be set to a valid address RefAddr[0] by the driver. The same applies to VC1 and MPEG2. **Programming Notes** AVC: Always specifies all 16 addresses even some of them are not needed as indicated by the max num of active reference pictures. This is done for preventing data corruption (error, fault condition, etc.) by having all the references being set to a legal location.
	5:0	**Reserved** Access: RO Format: MBZ

MPEG2

MPEG2		
Source:	VideoCS	
Size (in bits):	16	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	15:6	**Reserved** Access: RO Format: MBZ
	5	**Missing EOB Error** This flag indicates missing EOB SEs coded in the bit-stream. Missing EOBs are concealed to match CBP of the error MB.
	4	**Inconsistent starting position Error - overlapping MBs** This flag indicates two slices overlapping one another by one or more MBs. Duplicate MBs decoded off the second slice shall be discarded.
	3	**Slice out-of-bound Error** This flag indicates a slice is running beyond the width of the picture. Out-of-bound MBs shall be discarded.
	2	**Premature frame end Error** This flag indicates missing slices/MBs coded in the bit-stream of a frame. One or more MBs are concealed to reach end of picture.
	1	**Inconsistent starting position Error - Missing MBs** This flag indicates one or more MBs are being concealed due to inconsistent MB starting and ending positions between slices.
	0	**MB Concealment Flag** . Each pulse from this flag indicates one MB is concealed by hardware.

MSAA Sample Number Message Address Control

MACD_MSAA_SN - MSAA Sample Number Message Address Control

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:4	**Reserved** Access: RO Format: MBZ
	3:0	**Sample Number** Format: U4 Specifies the sample number for the slot. If the sample number is larger than the **Number of Multisamples** in the Surface State, then the access is out of bounds.

MsgDesc

MsgDesc

Source: EuIsa
Size (in bits): 32
Default Value: 0x00000000

DWord	Bit	Description
0	31	**Reserved** Format: MBZ
	30	**Data Format** Format: U1 **Description** This field specifies the width of data read from sampler or written to render target. This field specifies the width of data read from written to render target. **Value** / **Name** 0b / 32bit 1b / 16bit
	29	**SIMD Mode[2]** Format: MBZ This field is the upper bit of the 3-bit SIMD Mode Field. Refer to the SIMD Mode[1:0] Field for encodings
	28:25	**Message Length** This field specifies the number of GRF registers starting from <Src0.RegNum> to be sent out on the request message payload. Valid value ranges from 1 to 15. A value of 0 is considered erroneous. **Value** / **Name** 1-15 / Number of Registers
	24:20	**Response Length** This field indicates the number of GRF registers expected in the message response. The valid value ranges from 0 to 16. A value 0 indicates that the request message does not expect any response. The largest response supported is 16 GRF registers. **Value** / **Name** 0-16 / Number of Registers

MsgDesc		
	19	**Header Present** Format: Enable If set, indicates that the message includes a header. Depending on the target shared function, this field may be restricted to either enabled or disabled. Refer to the specific shared function section for details.
	18:0	**Function Control** This field is intended to control the target function unit. Refer to the section on the specific target function unit for details on the contents of this field.

Named Barrier State

NAMED_BAR_STATE - Named Barrier State

Size (in bits): 32

Default Value: 0x00000000

Hardware state of one "named barrier", which is part of the payload of Save_Barrier and Restore_Barrier messages.

DWord	Bit	Description
0	31:20	**Reserved** Format: MBZ
	19	**Designated Thread** Designated thread for WMTP Save. **Programming Notes** This field is ignored for Restore_Barrier message payload. **Restriction** This field is valid only for Logical Barrier ID 0.
	18:11	**Number of Consumers** Format: U8 Specifies the number of consumer threads in the barrier.
	10:3	**Number of Producers** Format: U8 Specifies the number of producer threads in the barrier.
	2:1	**Barrier Type** Format: U2 Value / Name: 0 Producer_Consumer 1 Producer_Only 2 Consumer_Only
	0	**Valid** Indicates this barrier's signal is set for the thread.

No Event Data Payload

MDP_NO_EVENT - No Event Data Payload		
Source:	EuSubFunctionGateway	
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0..7	255:0	**Reserved** Access: RO Format: MBZ

Normal Media Block Message Header

MH_MB - Normal Media Block Message Header

Source:	EuSubFunctionDataPort1
Size (in bits):	256
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:0	**X Offset** Format: S31 X offset (in bytes) of the upper left corner of the block into the surface. **Programming Notes** Must be DWord aligned (Bits 1:0 MBZ) for the write form of the message.
1	31:0	**Y Offset** Format: S31 Y offset (in rows) of the upper left corner of the block into the surface.
2	31:0	**Normal Media Block Message Control** Format: MHC_MB_CONTROL Specifies the Normal message subtype and additional input parameters.
3	31:0	**Mask** Format: U32 The Mask is ignored by the Normal Media Block message: all Dwords are always returned on reads, and always enabled to be written on writes.
4	31:0	**FFTID** Format: MHC_FFTID Fixed Function Thread ID
5..7	95:0	**Reserved** Access: RO Format: MBZ

Normal Media Block Message Header Control

MHC_MB_CONTROL - Normal Media Block Message Header Control

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:30	**Message Mode** **Description** Specifies the interpretation of M0.3 (Pixel or Byte Mask). For the Sampler Cache Data Port, this field is ignored, behaving as if always set to NORMAL. Specifies the interpretation of M0.3 (Pixel or Byte Mask). **Value / Name / Description** 00h — Normal — The Block Height and Block Width fields are specified in this Dword. The Mask is ignored by a media block read message and behaves as if it is set to all ones for a media block write message. Others — Reserved — Reserved. **Programming Notes** The Media Block Read message is Normal subtype when both Sub-Register Offset and Register Pitch Control are zero. The Media Block Read message is Merged subtype when either Sub-Register Offset or Register Pitch Control are non-zero.
	29	**Reserved** Access: RO Format: MBZ
	28:24	**Sub-Register Offset** Default Value: 0 Format: U5 The sub-register offset must be 0 for Normal Media Block Read message subtype. This field is ignored (reserved) for a media block write message.
	23:22	**Reserved** Access: RO Format: MBZ
	21:16	**Block Height** Format: U6 Height in rows of block being accessed. Range = [0,63] representing 1 to 64 rows

MHC_MB_CONTROL - Normal Media Block Message Header Control

		Restriction If Block Width (bytes), then Maximum Block Height (rows) is constrained by (# Dwords width) * (# rows) <= 64 Dwords.
	15:10	**Reserved** Access: RO Format: MBZ
	9:8	**Register Pitch Control** Default Value: 0 Format: U2 The register pitch must be 0 for a Normal Media Block Read message. This field is ignored (reserved) for a media block write message.
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**Block Width** Format: U6 **Description** Width in bytes of the block being accessed. For normal Media Block Writes, Range = [0,63] representing 1 to 64 Bytes. For normal Media Block Reads and for masked and merged Media Block messages, Range = [0,31] representing 1 to 32 Bytes. **Programming Notes** Must be DWord aligned for the write form of the message.

oMask Message Data Payload Register

MDPR_OMASK - oMask Message Data Payload Register

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:16	**oMask1** Format: U16 oMask for Pixels [15:0] of Slot 1. Not used for Slot Group HI.
	15:0	**oMask0** Format: U16 oMask for Pixels [15:0] of Slot 0. Not used for Slot Group HI.
1	31:16	**oMask3** Format: U16 oMask for Pixels [15:0] of Slot 3. Not used for Slot Group HI.
	15:0	**oMask2** Format: U16 oMask for Pixels [15:0] of Slot 2. Not used for Slot Group HI.
2	31:16	**oMask5** Format: U16 oMask for Pixels [15:0] of Slot 5. Not used for Slot Group HI.
	15:0	**oMask4** Format: U16 oMask for Pixels [15:0] of Slot 4. Not used for Slot Group HI.
3	31:16	**oMask7** Format: U16 oMask for Pixels [15:0] of Slot 7. Not used for Slot Group HI.
	15:0	**oMask6** Format: U16 oMask for Pixels [15:0] of Slot 6. Not used for Slot Group HI.

MDPR_OMASK - oMask Message Data Payload Register		
4	31:16	**oMask9** Format: U16 oMask for Pixels [15:0] of Slot 9. Used only if Slot Group HI or SIMD16.
	15:0	**oMask8** Format: U16 oMask for Pixels [15:0] of Slot 8. Used only if Slot Group HI or SIMD16.
5	31:16	**oMask11** Format: U16 oMask for Pixels [15:0] of Slot 11. Used only if Slot Group HI or SIMD16.
	15:0	**oMask10** Format: U16 oMask for Pixels [15:0] of Slot 10. Used only if Slot Group HI or SIMD16.
6	31:16	**oMask13** Format: U16 oMask for Pixels [15:0] of Slot 13. Used only if Slot Group HI or SIMD16.
	15:0	**oMask12** Format: U16 oMask for Pixels [15:0] of Slot 12. Used only if Slot Group HI or SIMD16.
7	31:16	**oMask15** Format: U16 oMask for Pixels [15:0] of Slot 15. Used only if Slot Group HI or SIMD16.
	15:0	**oMask14** Format: U16 oMask for Pixels [15:0] of Slot 14. Used only if Slot Group HI or SIMD16.

OM Replicated SIMD16 Render Target Data Payload

MDP_RTW_M16REP - OM Replicated SIMD16 Render Target Data Payload

Size (in bits):	512
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**oMask** Format: MDPR_OMASK Slots [15:0] oMask
1.0-1.7	255:0	**RGBA** Format: MDPR_RGBA RGBA for all slots [15:0]

OM S0A SIMD8 Render Target Data Payload

MDP_RTW_MA8 - OM S0A SIMD8 Render Target Data Payload		
Size (in bits):	1536	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Source 0 Alpha** Format: MDP_DW_SIMD8 Slots [7:0] Source 0 Alpha
1.0-1.7	255:0	**oMask** Format: MDPR_OMASK Slots [7:0] oMask. Upper half ignored.
2.0-2.7	255:0	**Red** Format: MDP_DW_SIMD8 Slots [7:0] Red
3.0-3.7	255:0	**Green** Format: MDP_DW_SIMD8 Slots [7:0] Green
4.0-4.7	255:0	**Blue** Format: MDP_DW_SIMD8 Slots [7:0] Blue
5.0-5.7	255:0	**Alpha** Format: MDP_DW_SIMD8 Slots [7:0] Alpha

OM S0A SIMD16 Render Target Data Payload

MDP_RTW_MA16 - OM S0A SIMD16 Render Target Data Payload

Size (in bits): 2816

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Source 0 Alpha[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Source 0 Alpha
1.0-1.7	255:0	**Source 0 Alpha[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Source 0 Alpha
2.0-2.7	255:0	**oMask** Format: MDPR_OMASK Slots [15:0] oMask
3.0-3.7	255:0	**Red[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Red
4.0-4.7	255:0	**Red[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Red

MDP_RTW_MA16 - OM S0A SIMD16 Render Target Data Payload		
5.0-5.7	255:0	**Green[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Green
6.0-6.7	255:0	**Green[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Green
7.0-7.7	255:0	**Blue[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Blue
8.0-8.7	255:0	**Blue[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Blue
9.0-9.7	255:0	**Alpha[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Alpha
10.0-10.7	255:0	**Alpha[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Alpha

OM SIMD8 Dual Source Render Target Data Payload

MDP_RTW_M8DS - OM SIMD8 Dual Source Render Target Data Payload	
Size (in bits):	2304
Default Value:	0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**oMask** Format: **MDPR_OMASK** oMask for slots [7:0] and [15:8]. Operation selects upper or lower half.
1.0-1.7	255:0	**Src0 Red** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src0 Red
2.0-2.7	255:0	**Src0 Green** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src0 Green
3.0-3.7	255:0	**Src0 Blue** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src0 Blue
4.0-4.7	255:0	**Src0 Alpha** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src0 Alpha
5.0-5.7	255:0	**Src1 Red** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src1 Red

MDP_RTW_M8DS - OM SIMD8 Dual Source Render Target Data Payload

6.0-6.7	255:0	**Src1 Green** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src1 Green
7.0-7.7	255:0	**Src1 Blue** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src1 Blue
8.0-8.7	255:0	**Src1 Alpha** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src1 Alpha

OM SIMD8 Render Target Data Payload

MDP_RTW_M8 - OM SIMD8 Render Target Data Payload		
Size (in bits):	1280	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**oMask** Format: MDPR_OMASK Slots [7:0] oMask. Upper half ignored.
1.0-1.7	255:0	**Red** Format: MDP_DW_SIMD8 Slots [7:0] Red
2.0-2.7	255:0	**Green** Format: MDP_DW_SIMD8 Slots [7:0] Green
3.0-3.7	255:0	**Blue** Format: MDP_DW_SIMD8 Slots [7:0] Blue
4.0-4.7	255:0	**Alpha** Format: MDP_DW_SIMD8 Slots [7:0] Alpha

OM SIMD16 Render Target Data Payload

MDP_RTW_M16 - OM SIMD16 Render Target Data Payload		
Size (in bits):	2304	
Default Value:	0x00000000, 0x00000000	

DWord	Bit	Description
0.0-0.7	255:0	**oMask** Format: **MDPR_OMASK** Slots [15:0] oMask
1.0-1.7	255:0	**Red[7:0]** Format: **MDP_DW_SIMD8** Slots [7:0] Red
2.0-2.7	255:0	**Red[15:8]** Format: **MDP_DW_SIMD8** Slots [15:8] Red
3.0-3.7	255:0	**Green[7:0]** Format: **MDP_DW_SIMD8** Slots [7:0] Green
4.0-4.7	255:0	**Green[15:8]** Format: **MDP_DW_SIMD8** Slots [15:8] Green
5.0-5.7	255:0	**Blue[7:0]** Format: **MDP_DW_SIMD8** Slots [7:0] Blue

MDP_RTW_M16 - OM SIMD16 Render Target Data Payload		
6.0-6.7	255:0	**Blue[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Blue
7.0-7.7	255:0	**Alpha[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Alpha
8.0-8.7	255:0	**Alpha[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Alpha

OS OM S0A SIMD8 Render Target Data Payload

MDP_RTW_SMA8 - OS OM S0A SIMD8 Render Target Data Payload		
Size (in bits):	1792	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Source 0 Alpha** Format: MDP_DW_SIMD8 Slots [7:0] Source 0 Alpha
1.0-1.7	255:0	**oMask** Format: MDPR_OMASK Slots [7:0] oMask. Upper half ignored.
2.0-2.7	255:0	**Red** Format: MDP_DW_SIMD8 Slots [7:0] Red
3.0-3.7	255:0	**Green** Format: MDP_DW_SIMD8 Slots [7:0] Green
4.0-4.7	255:0	**Blue** Format: MDP_DW_SIMD8 Slots [7:0] Blue
5.0-5.7	255:0	**Alpha** Format: MDP_DW_SIMD8 Slots [7:0] Alpha

MDP_RTW_SMA8 - OS OM S0A SIMD8 Render Target Data Payload		
6.0-6.7	255:0	**Stencil** Format: MDPR_STENCIL Slots [7:0] Stencil

OS OM SIMD8 Dual Source Render Target Data Payload

MDP_RTW_SM8DS - OS OM SIMD8 Dual Source Render Target Data Payload

Size (in bits): 2560

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**oMask** Format: MDPR_OMASK oMask for slots [7:0] and [15:8]. Operation selects upper or lower half.
1.0-1.7	255:0	**Src0 Red** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src0 Red
2.0-2.7	255:0	**Src0 Green** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src0 Green
3.0-3.7	255:0	**Src0 Blue** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src0 Blue
4.0-4.7	255:0	**Src0 Alpha** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src0 Alpha

MDP_RTW_SM8DS - OS OM SIMD8 Dual Source Render Target Data Payload		
5.0-5.7	255:0	**Src1 Red** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src1 Red
6.0-6.7	255:0	**Src1 Green** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src1 Green
7.0-7.7	255:0	**Src1 Blue** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src1 Blue
8.0-8.7	255:0	**Src1 Alpha** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src1 Alpha
9.0-9.7	255:0	**Stencil** Format: **MDPR_STENCIL** Slots [7:0] or [15:8] of Stencil

OS OM SIMD8 Render Target Data Payload

MDP_RTW_SM8 - OS OM SIMD8 Render Target Data Payload

Size (in bits): 1536

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**oMask** Format: MDPR_OMASK Slots [7:0] oMask. Upper half ignored.
1.0-1.7	255:0	**Red** Format: MDP_DW_SIMD8 Slots [7:0] Red
2.0-2.7	255:0	**Green** Format: MDP_DW_SIMD8 Slots [7:0] Green
3.0-3.7	255:0	**Blue** Format: MDP_DW_SIMD8 Slots [7:0] Blue
4.0-4.7	255:0	**Alpha** Format: MDP_DW_SIMD8 Slots [7:0] Alpha
5.0-5.7	255:0	**Stencil** Format: MDPR_STENCIL Slots [7:0] Stencil

OS S0A SIMD8 Render Target Data Payload

MDP_RTW_SA8 - OS S0A SIMD8 Render Target Data Payload

Size (in bits): 1536

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Source 0 Alpha** Format: **MDP_DW_SIMD8** Slots [7:0] Source 0 Alpha
1.0-1.7	255:0	**Red** Format: **MDP_DW_SIMD8** Slots [7:0] Red
2.0-2.7	255:0	**Green** Format: **MDP_DW_SIMD8** Slots [7:0] Green
3.0-3.7	255:0	**Blue** Format: **MDP_DW_SIMD8** Slots [7:0] Blue
4.0-4.7	255:0	**Alpha** Format: **MDP_DW_SIMD8** Slots [7:0] Alpha
5.0-5.7	255:0	**Stencil** Format: **MDPR_STENCIL** Slots [7:0] Stencil

OS SIMD8 Dual Source Render Target Data Payload

MDP_RTW_S8DS - OS SIMD8 Dual Source Render Target Data Payload		
Size (in bits):	2304	
Default Value:	0x00000000, 0x00000000	

DWord	Bit	Description
0.0-0.7	255:0	**Src0 Red** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src0 Red
1.0-1.7	255:0	**Src0 Green** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src0 Green
2.0-2.7	255:0	**Src0 Blue** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src0 Blue
3.0-3.7	255:0	**Src0 Alpha** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src0 Alpha
4.0-4.7	255:0	**Src1 Red** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src1 Red
5.0-5.7	255:0	**Src1 Green** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src1 Green

MDP_RTW_S8DS - OS SIMD8 Dual Source Render Target Data Payload

6.0-6.7	255:0	**Src1 Blue** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src1 Blue
7.0-7.7	255:0	**Src1 Alpha** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src1 Alpha
8.0-8.7	255:0	**Stencil** Format: MDPR_STENCIL Slots [7:0] or [15:8] of Stencil

OS SIMD8 Render Target Data Payload

MDP_RTW_S8 - OS SIMD8 Render Target Data Payload

Size (in bits): 1280

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Red** Format: MDP_DW_SIMD8 Slots [7:0] Red
1.0-1.7	255:0	**Green** Format: MDP_DW_SIMD8 Slots [7:0] Green
2.0-2.7	255:0	**Blue** Format: MDP_DW_SIMD8 Slots [7:0] Blue
3.0-3.7	255:0	**Alpha** Format: MDP_DW_SIMD8 Slots [7:0] Alpha
4.0-4.7	255:0	**Stencil** Format: MDPR_STENCIL Slots [7:0] Stencil

OS SZ OM S0A SIMD8 Render Target Data Payload

MDP_RTW_SZMA8 - OS SZ OM S0A SIMD8 Render Target Data Payload

Size (in bits): 2048

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Source 0 Alpha** Format: MDP_DW_SIMD8 Slots [7:0] Source 0 Alpha
1.0-1.7	255:0	**oMask** Format: MDPR_OMASK Slots [7:0] oMask. Upper half ignored.
2.0-2.7	255:0	**Red** Format: MDP_DW_SIMD8 Slots [7:0] Red
3.0-3.7	255:0	**Green** Format: MDP_DW_SIMD8 Slots [7:0] Green
4.0-4.7	255:0	**Blue** Format: MDP_DW_SIMD8 Slots [7:0] Blue
5.0-5.7	255:0	**Alpha** Format: MDP_DW_SIMD8 Slots [7:0] Alpha

MDP_RTW_SZMA8 - OS SZ OM S0A SIMD8 Render Target Data Payload

6.0-6.7	255:0	**Source Depth** Format: **MDP_DW_SIMD8** Slots [7:0] Source Depth
7.0-7.7	255:0	**Stencil** Format: **MDPR_STENCIL** Slots [7:0] Stencil

OS SZ OM SIMD8 Dual Source Render Target Data Payload

MDP_RTW_SZM8DS - OS SZ OM SIMD8 Dual Source Render Target Data Payload

Size (in bits):	2816
Default Value:	0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**oMask** Format: **MDPR_OMASK** oMask for slots [7:0] and [15:8]. Operation selects upper or lower half.
1.0-1.7	255:0	**Src0 Red** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src0 Red
2.0-2.7	255:0	**Src0 Green** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src0 Green
3.0-3.7	255:0	**Src0 Blue** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src0 Blue
4.0-4.7	255:0	**Src0 Alpha** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src0 Alpha

MDP_RTW_SZM8DS - OS SZ OM SIMD8 Dual Source Render Target Data Payload

5.0-5.7	255:0	**Src1 Red** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src1 Red
6.0-6.7	255:0	**Src1 Green** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src1 Green
7.0-7.7	255:0	**Src1 Blue** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src1 Blue
8.0-8.7	255:0	**Src1 Alpha** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src1 Alpha
9.0-9.7	255:0	**Source Depth** Format: MDP_DW_SIMD8 Slots [7:0] or [15:8] of Source Depth
10.0-10.7	255:0	**Stencil** Format: MDPR_STENCIL Slots [7:0] or [15:8] of Stencil

OS SZ OM SIMD8 Render Target Data Payload

MDP_RTW_SZM8 - OS SZ OM SIMD8 Render Target Data Payload

Size (in bits): 1792

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**oMask** Format: MDPR_OMASK Slots [7:0] oMask. Upper half ignored.
1.0-1.7	255:0	**Red** Format: MDP_DW_SIMD8 Slots [7:0] Red
2.0-2.7	255:0	**Green** Format: MDP_DW_SIMD8 Slots [7:0] Green
3.0-3.7	255:0	**Blue** Format: MDP_DW_SIMD8 Slots [7:0] Blue
4.0-4.7	255:0	**Alpha** Format: MDP_DW_SIMD8 Slots [7:0] Alpha
5.0-5.7	255:0	**Source Depth** Format: MDP_DW_SIMD8 Slots [7:0] Source Depth
6.0-6.7	255:0	**Stencil** Format: MDPR_STENCIL Slots [7:0] Stencil

OS SZ S0A SIMD8 Render Target Data Payload

MDP_RTW_SZA8 - OS SZ S0A SIMD8 Render Target Data Payload

Size (in bits): 1792

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Source 0 Alpha** Format: MDP_DW_SIMD8 Slots [7:0] Source 0 Alpha
1.0-1.7	255:0	**Red** Format: MDP_DW_SIMD8 Slots [7:0] Red
2.0-2.7	255:0	**Green** Format: MDP_DW_SIMD8 Slots [7:0] Green
3.0-3.7	255:0	**Blue** Format: MDP_DW_SIMD8 Slots [7:0] Blue
4.0-4.7	255:0	**Alpha** Format: MDP_DW_SIMD8 Slots [7:0] Alpha
5.0-5.7	255:0	**Source Depth** Format: MDP_DW_SIMD8 Slots [7:0] Source Depth
6.0-6.7	255:0	**Stencil** Format: MDPR_STENCIL Slots [7:0] Stencil

OS SZ SIMD8 Dual Source Render Target Data Payload

MDP_RTW_SZ8DS - OS SZ SIMD8 Dual Source Render Target Data Payload

Size (in bits):	2560
Default Value:	0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Src0 Red** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src0 Red
1.0-1.7	255:0	**Src0 Green** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src0 Green
2.0-2.7	255:0	**Src0 Blue** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src0 Blue
3.0-3.7	255:0	**Src0 Alpha** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src0 Alpha
4.0-4.7	255:0	**Src1 Red** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src1 Red
5.0-5.7	255:0	**Src1 Green** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src1 Green

MDP_RTW_SZ8DS - OS SZ SIMD8 Dual Source Render Target Data Payload

6.0-6.7	255:0	**Src1 Blue** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src1 Blue
7.0-7.7	255:0	**Src1 Alpha** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src1 Alpha
8.0-8.7	255:0	**Source Depth** Format: **MDP_DW_SIMD8** Slots [7:0] or [15:8] of Source Depth
9.0-9.7	255:0	**Stencil** Format: **MDPR_STENCIL** Slots [7:0] or [15:8] of Stencil

OS SZ SIMD8 Render Target Data Payload

MDP_RTW_SZ8 - OS SZ SIMD8 Render Target Data Payload

Size (in bits): 1536

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Red** Format: **MDP_DW_SIMD8** Slots [7:0] Red
1.0-1.7	255:0	**Green** Format: **MDP_DW_SIMD8** Slots [7:0] Green
2.0-2.7	255:0	**Blue** Format: **MDP_DW_SIMD8** Slots [7:0] Blue
3.0-3.7	255:0	**Alpha** Format: **MDP_DW_SIMD8** Slots [7:0] Alpha
4.0-4.7	255:0	**Source Depth** Format: **MDP_DW_SIMD8** Slots [7:0] Source Depth
5.0-5.7	255:0	**Stencil** Format: **MDPR_STENCIL** Slots [7:0] Stencil

Oword 2 Block Data Payload

MDP_OW2 - Oword 2 Block Data Payload		
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.3	127:0	**Oword0** Format: U128 Specifies the Oword data for block element 0
0.4-0.7	127:0	**Oword1** Format: U128 Specifies the Oword data for block element 1

Oword 4 Block Data Payload

MDP_OW4 - Oword 4 Block Data Payload		
Size (in bits):	512	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Data[1:0]** Format: **MDCR_OW** Specifies the Oword data for block elements [1:0]
1.0-1.7	255:0	**Data[3:2]** Format: **MDCR_OW** Specifies the Oword data for block elements [3:2]

Oword 8 Block Data Payload

MDP_OW8 - Oword 8 Block Data Payload		
Size (in bits):	1024	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Data[1:0]** Format: MDCR_OW Specifies the Oword data for block elements [1:0]
1.0-1.7	255:0	**Data[3:2]** Format: MDCR_OW Specifies the Oword data for block elements [3:2]
2.0-2.7	255:0	**Data[5:4]** Format: MDCR_OW Specifies the Oword data for block elements [5:4]
3.0-3.7	255:0	**Data[7:6]** Format: MDCR_OW Specifies the Oword data for block elements [7:6]

Oword A64 SIMD8 Atomic Operation CMPWR16B Message Data Payload

MDP_A64_AOP8_OW2 - Oword A64 SIMD8 Atomic Operation CMPWR16B Message Data Payload		
Size (in bits):	2048	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Slot[1:0] Src0** Format: MDCR_OW Specifies the Slot [1:0] Source 0 data
1.0-1.7	255:0	**Slot[3:2] Src0** Format: MDCR_OW Specifies the Slot [3:2] Source 0 data
2.0-2.7	255:0	**Slot[5:4] Src0** Format: MDCR_OW Specifies the Slot [5:4] Source 0 data
3.0-3.7	255:0	**Slot[7:6] Src0** Format: MDCR_OW Specifies the Slot [7:6] Source 0 data
4.0-4.7	255:0	**Slot[1:0] Src1** Format: MDCR_OW Specifies the Slot [1:0] Source 1 data
5.0-5.7	255:0	**Slot[3:2] Src1** Format: MDCR_OW Specifies the Slot [3:2] Source 1 data

MDP_A64_AOP8_OW2 - Oword A64 SIMD8 Atomic Operation CMPWR16B Message Data Payload		
6.0-6.7	255:0	**Slot[5:4] Src1** Format: **MDCR_OW** Specifies the Slot [5:4] Source 1 data
7.0-7.7	255:0	**Slot[7:6] Src1** Format: **MDCR_OW** Specifies the Slot [7:6] Source 1 data

Oword Data Blocks Message Descriptor Control Field

MDC_DB_OW - Oword Data Blocks Message Descriptor Control Field

Size (in bits): 3

Default Value: 0x00000000

DWord	Bit	Description
0	2:0	**Data Blocks** Specifies the number of Oword blocks to be read or written

Value	Name	Description
00h	OW1L	1 Oword, read into or written from the low 128 bits of the destination register
01h	Reserved	Reserved
02h	OW2	2 Owords
03h	OW4	4 Owords
04h	OW8	8 Owords
Others	Reserved	Ignored

Oword Data Payload Register

MDCR_OW - Oword Data Payload Register		
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.3	127:0	**Oword0** Format: U128 Specifies the slot 0 data in this payload register
0.4-0.7	127:0	**Oword1** Format: U128 Specifies the slot 1 data in this payload register

Oword Dual Data Blocks Message Descriptor Control Field

MDC_DB_OWD - Oword Dual Data Blocks Message Descriptor Control Field

Size (in bits): 2

Default Value: 0x00000000

DWord	Bit	Description
0	1:0	**OW Dual Data Blocks** Specifies the number of Oword Blocks to be read or written

Value	Name	Description
00h	OWD1	1 Hword register, 2 Owords
02h	OWD4	4 Hword registers, 8 Owords
Others	Reserved	Ignored

Performance Counter Report Format 101b

Performance Counter Report Format 101b		
Size (in bits):	2048	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0	31:0	**RPT_ID**
1	31:0	**TIME_STAMP**
2	31:0	**CTX_ID**
3	31:0	**GPU_TICKS**
4	31:0	**A-Cntr 0 (low dword)**
5	31:0	**A-Cntr 1 (low dword)**
6	31:0	**A-Cntr 2 (low dword)**
7	31:0	**A-Cntr 3 (low dword)**
8	31:0	**A-Cntr 4 (low dword)**
9	31:0	**A-Cntr 5 (low dword)**
10	31:0	**A-Cntr 6 (low dword)**
11	31:0	**A-Cntr 7 (low dword)**
12	31:0	**A-Cntr 8 (low dword)**
13	31:0	**A-Cntr 9 (low dword)**
14	31:0	**A-Cntr 10 (low dword)**
15	31:0	**A-Cntr 11 (low dword)**
16	31:0	**A-Cntr 12 (low dword)**
17	31:0	**A-Cntr 13 (low dword)**
18	31:0	**A-Cntr 14 (low dword)**
19	31:0	**A-Cntr 15 (low dword)**
20	31:0	**A-Cntr 16 (low dword)**
21	31:0	**A-Cntr 17 (low dword)**
22	31:0	**A-Cntr 18 (low dword)**

Performance Counter Report Format 101b		
23	31:0	**A-Cntr 19 (low dword)**
24	31:0	**A-Cntr 20 (low dword)**
25	31:0	**A-Cntr 21 (low dword)**
26	31:0	**A-Cntr 22 (low dword)**
27	31:0	**A-Cntr 23 (low dword)**
28	31:0	**A-Cntr 24 (low dword)**
29	31:0	**A-Cntr 25 (low dword)**
30	31:0	**A-Cntr 26 (low dword)**
31	31:0	**A-Cntr 27 (low dword)**
32	31:0	**A-Cntr 28 (low dword)**
33	31:0	**A-Cntr 29 (low dword)**
34	31:0	**A-Cntr 30 (low dword)**
35	31:0	**A-Cntr 31 (low dword)**
36	31:0	**A-Cntr 32 (low dword)**
37	31:0	**A-Cntr 33 (low dword)**
38	31:0	**A-Cntr 34 (low dword)**
39	31:0	**A-Cntr 35 (low dword)**
40	31:24	**High byte of A3**
	23:16	**High byte of A2**
	15:8	**High byte of A1**
	7:0	**High byte of A0**
41	31:24	**High byte of A7**
	23:16	**High byte of A6**
	15:8	**High byte of A5**
	7:0	**High byte of A4**
42	31:24	**High byte of A11**
	23:16	**High byte of A10**
	15:8	**High byte of A9**
	7:0	**High byte of A8**
43	31:24	**High byte of A15**
	23:16	**High byte of A14**
	15:8	**High byte of A13**
	7:0	**High byte of A12**
44	31:24	**High byte of A19**
	23:16	**High byte of A18**
	15:8	**High byte of A17**

Performance Counter Report Format 101b		
	7:0	**High byte of A16**
45	31:24	**High byte of A23**
	23:16	**High byte of A22**
	15:8	**High byte of A21**
	7:0	**High byte of A20**
46	31:24	**High byte of A27**
	23:16	**High byte of A26**
	15:8	**High byte of A25**
	7:0	**High byte of A24**
47	31:24	**High byte of A31**
	23:16	**High byte of A30**
	15:8	**High byte of A29**
	7:0	**High byte of A28**
48	31:0	**B-Cntr 0**
49	31:0	**B-Cntr 1**
50	31:0	**B-Cntr 2**
51	31:0	**B-Cntr 3**
52	31:0	**B-Cntr 4**
53	31:0	**B-Cntr 5**
54	31:0	**B-Cntr 6**
55	31:0	**B-Cntr 7**
56	31:0	**C-Cntr 0**
57	31:0	**C-Cntr 1**
58	31:0	**C-Cntr 2**
59	31:0	**C-Cntr 3**
60	31:0	**C-Cntr 4**
61	31:0	**C-Cntr 5**
62	31:0	**C-Cntr 6**
63	31:0	**C-Cntr 7**

Per Thread Scratch Space Message Header Control

MHC_PTSS - Per Thread Scratch Space Message Header Control

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:4	**Reserved** Access: RO Format: MBZ
	3:0	**Per Thread Scratch Space** Format: U4 Specifies the amount of scratch space allowed to be used by this thread for messages in which the Binding Table Index is Stateless model, otherwise this field is ignored.The data port will use this to bounds check scratch space messages. Value range = [0,11] represents [1KB, 2MB] in powers of two. **Programming Notes** Writes out of bounds will be ignored. Reads out of bounds will return 0.

PIXEL_HASH_TABLE_1BIT_32ENTRY

PIXEL_HASH_TABLE_1BIT_32ENTRY		
Size (in bits):	32	
Default Value:	0x00000000	
Description		
2-way pixel hashing table. Table is 32-entries:8X,4Y in [Y][X] format. Each entry is a single bit that indicates which sub-slice hardware block the indicated xy pixel block is mapped.		
pixelhash_id maps to color-pipe. A value of 0 indicates the larger color-pipe, or first enabled color-pipe if both enabled color-pipes are balanced		

DWord	Bit	Description
0	31:24	**Pixel Hashing Table Entries y[3]x[7:0]** Format: U8 Indicates the pixelhash_id for the pixel block that has y=3 and x=7..0
	23:16	**Pixel Hashing Table Entries y[2]x[7:0]** Format: U8 Indicates the pixelhash_id for the pixel block that has y=2 and x=7..0
	15:8	**Pixel Hashing Table Entries y[1]x[7:0]** Format: U8 Indicates the pixelhash_id for the pixel block that has y=1 and x=7..0
	7:0	**Pixel Hashing Table Entries y[0]x[7:0]** Format: U8 Indicates the pixelhash_id for the pixel block that has y=0 and x=7..0

PIXEL_HASH_TABLE_1BIT_64ENTRY

PIXEL_HASH_TABLE_1BIT_64ENTRY		
Size (in bits):	64	
Default Value:	0x00000000, 0x00000000	
Description		
2-way pixel hashing table. Table is 64-entries:8X,8Y in [Y][X] format. Each entry is a single bit that indicates which sub-slice hardware block the indicated xy pixel block is mapped.		
pixelhash_id maps to color-pipe. A value of 0 indicates the larger color-pipe, or first enabled color-pipe if both enabled color-pipes are balanced		

DWord	Bit	Description
0	31:24	**Pixel Hashing Table Entries y[3]x[7:0]** Format: U8 Indicates the pixelhash_id for the pixel block that has y=3 and x=7..0
	23:16	**Pixel Hashing Table Entries y[2]x[7:0]** Format: U8 Indicates the pixelhash_id for the pixel block that has y=2 and x=7..0
	15:8	**Pixel Hashing Table Entries y[1]x[7:0]** Format: U8 Indicates the pixelhash_id for the pixel block that has y=1 and x=7..0
	7:0	**Pixel Hashing Table Entries y[0]x[7:0]** Format: U8 Indicates the pixelhash_id for the pixel block that has y=0 and x=7..0
1	31:24	**Pixel Hashing Table Entries y[7]x[7:0]** Format: U8 Indicates the pixelhash_id for the pixel block that has y=7 and x=7..0
	23:16	**Pixel Hashing Table Entries y[6]x[7:0]** Format: U8 Indicates the pixelhash_id for the pixel block that has y=6 and x=7..0
	15:8	**Pixel Hashing Table Entries y[5]x[7:0]** Format: U8 Indicates the pixelhash_id for the pixel block that has y=5 and x=7..0
	7:0	**Pixel Hashing Table Entries y[4]x[7:0]** Format: U8 Indicates the pixelhash_id for the pixel block that has y=4 and x=7..0

PIXEL_HASH_TABLE_1BIT_128ENTRY

PIXEL_HASH_TABLE_1BIT_128ENTRY		
Size (in bits):	128	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000	
Description		
2-way pixel hashing table. Table is 128-entries:16X,8Y in [Y][X] format. Each entry is a single bit that indicates which sub-slice hardware block the indicated xy pixel block is mapped.		
pixelhash_id maps to color-pipe. A value of 0 indicates the larger color-pipe, or first enbled color-pipe if both enabled color-pipes are balanced		

DWord	Bit	Description
0	31:16	**Pixel Hashing Table Entries y[1]x[15:0]** Format: U16 Indicates the pixelhash_id for the pixel block that has y=1 and x=15..0
	15:0	**Pixel Hashing Table Entries y[0]x[15:0]** Format: U16 Indicates the pixelhash_id for the pixel block that has y=0 and x=15..0
1	31:16	**Pixel Hashing Table Entries y[3]x[15:0]** Format: U16 Indicates the pixelhash_id for the pixel block that has y=3 and x=15..0
	15:0	**Pixel Hashing Table Entries y[2]x[15:0]** Format: U16 Indicates the pixelhash_id for the pixel block that has y=2 and x=15..0
2	31:16	**Pixel Hashing Table Entries y[5]x[15:0]** Format: U16 Indicates the pixelhash_id for the pixel block that has y=5 and x=15..0
	15:0	**Pixel Hashing Table Entries y[4]x[15:0]** Format: U16 Indicates the pixelhash_id for the pixel block that has y=4 and x=15..0
3	31:16	**Pixel Hashing Table Entries y[7]x[15:0]** Format: U16 Indicates the pixelhash_id for the pixel block that has y=7 and x=15..0
	15:0	**Pixel Hashing Table Entries y[6]x[15:0]** Format: U16 Indicates the pixelhash_id for the pixel block that has y=6 and x=15..0

PIXEL_HASH_TABLE_2BIT_64ENTRY

PIXEL_HASH_TABLE_2BIT_64ENTRY

Size (in bits): 128

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000

Description
3-wayor 4-way pixel hashing table. Table is 64-entries:8X,8Y in [Y][X] format. Each entry is two bits that indicates which sub-slice hardware block the indicated xy pixel block is mapped.
pixelhash_id maps to color-pipe. A value of 0 indicates the largest color-pipe, or first enbled color-pipe if all enabled color-pipes are balanced. A value of 2 indicates the smallest color-pipe, or last enbled color-pipe if all enabled color-pipes are balanced.

DWord	Bit	Description
0	31:30	**Pixel Hashing Table Entry y[1]x[7]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=7 and y=1
	29:28	**Pixel Hashing Table Entry y[1]x[6]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=6 and y=1
	27:26	**Pixel Hashing Table Entry y[1]x[5]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=5 and y=1
	25:24	**Pixel Hashing Table Entry y[1]x[4]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=4 and y=1
	23:22	**Pixel Hashing Table Entry y[1]x[3]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=3 and y=1
	21:20	**Pixel Hashing Table Entry y[1]x[2]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=2 and y=1
	19:18	**Pixel Hashing Table Entry y[1]x[1]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=1 and y=1

PIXEL_HASH_TABLE_2BIT_64ENTRY

	17:16	**Pixel Hashing Table Entry y[1]x[0]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=0 and y=1
	15:14	**Pixel Hashing Table Entry y[0]x[7]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=7 and y=0
	13:12	**Pixel Hashing Table Entry y[0]x[6]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=6 and y=0
	11:10	**Pixel Hashing Table Entry y[0]x[5]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=5 and y=0
	9:8	**Pixel Hashing Table Entry y[0]x[4]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=4 and y=0
	7:6	**Pixel Hashing Table Entry y[0]x[3]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=3 and y=0
	5:4	**Pixel Hashing Table Entry y[0]x[2]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=2 and y=0
	3:2	**Pixel Hashing Table Entry y[0]x[1]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=1 and y=0
	1:0	**Pixel Hashing Table Entry y[0]x[0]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=0 and y=0
1	31:30	**Pixel Hashing Table Entry y[3]x[7]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=7 and y=3

PIXEL_HASH_TABLE_2BIT_64ENTRY		
	29:28	**Pixel Hashing Table Entry y[3]x[6]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=6 and y=3
	27:26	**Pixel Hashing Table Entry y[3]x[5]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=5 and y=3
	25:24	**Pixel Hashing Table Entry y[3]x[4]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=4 and y=3
	23:22	**Pixel Hashing Table Entry y[3]x[3]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=3 and y=3
	21:20	**Pixel Hashing Table Entry y[3]x[2]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=2 and y=3
	19:18	**Pixel Hashing Table Entry y[3]x[1]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=1 and y=3
	17:16	**Pixel Hashing Table Entry y[3]x[0]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=0 and y=3
	15:14	**Pixel Hashing Table Entry y[2]x[7]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=7 and y=2
	13:12	**Pixel Hashing Table Entry y[2]x[6]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=6 and y=2
	11:10	**Pixel Hashing Table Entry y[2]x[5]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=5 and y=2

PIXEL_HASH_TABLE_2BIT_64ENTRY		
	9:8	**Pixel Hashing Table Entry y[2]x[4]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=4 and y=2
	7:6	**Pixel Hashing Table Entry y[2]x[3]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=3 and y=2
	5:4	**Pixel Hashing Table Entry y[2]x[2]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=2 and y=2
	3:2	**Pixel Hashing Table Entry y[2]x[1]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=1 and y=2
	1:0	**Pixel Hashing Table Entry y[2]x[0]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=0 and y=2
2	31:30	**Pixel Hashing Table Entry y[5]x[7]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=7 and y=5
	29:28	**Pixel Hashing Table Entry y[5]x[6]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=6 and y=5
	27:26	**Pixel Hashing Table Entry y[5]x[5]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=5 and y=5
	25:24	**Pixel Hashing Table Entry y[5]x[4]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=4 and y=5
	23:22	**Pixel Hashing Table Entry y[5]x[3]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=3 and y=5

PIXEL_HASH_TABLE_2BIT_64ENTRY		
	21:20	**Pixel Hashing Table Entry y[5]x[2]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=2 and y=5
	19:18	**Pixel Hashing Table Entry y[5]x[1]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=1 and y=5
	17:16	**Pixel Hashing Table Entry y[5]x[0]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=0 and y=5
	15:14	**Pixel Hashing Table Entry y[4]x[7]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=7 and y=4
	13:12	**Pixel Hashing Table Entry y[4]x[6]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=6 and y=4
	11:10	**Pixel Hashing Table Entry y[4]x[5]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=5 and y=4
	9:8	**Pixel Hashing Table Entry y[4]x[4]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=4 and y=4
	7:6	**Pixel Hashing Table Entry y[4]x[3]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=3 and y=4
	5:4	**Pixel Hashing Table Entry y[4]x[2]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=2 and y=4
	3:2	**Pixel Hashing Table Entry y[4]x[1]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=1 and y=4

PIXEL_HASH_TABLE_2BIT_64ENTRY		
	1:0	**Pixel Hashing Table Entry y[4]x[0]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=0 and y=4
3	31:30	**Pixel Hashing Table Entry y[7]x[7]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=7 and y=7
	29:28	**Pixel Hashing Table Entry y[7]x[6]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=6 and y=7
	27:26	**Pixel Hashing Table Entry y[7]x[5]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=5 and y=7
	25:24	**Pixel Hashing Table Entry y[7]x[4]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=4 and y=7
	23:22	**Pixel Hashing Table Entry y[7]x[3]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=3 and y=7
	21:20	**Pixel Hashing Table Entry y[7]x[2]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=2 and y=7
	19:18	**Pixel Hashing Table Entry y[7]x[1]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=1 and y=7
	17:16	**Pixel Hashing Table Entry y[7]x[0]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=0 and y=7
	15:14	**Pixel Hashing Table Entry y[6]x[7]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=7 and y=6

PIXEL_HASH_TABLE_2BIT_64ENTRY		
	13:12	**Pixel Hashing Table Entry y[6]x[6]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=6 and y=6
	11:10	**Pixel Hashing Table Entry y[6]x[5]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=5 and y=6
	9:8	**Pixel Hashing Table Entry y[6]x[4]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=4 and y=6
	7:6	**Pixel Hashing Table Entry y[6]x[3]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=3 and y=6
	5:4	**Pixel Hashing Table Entry y[6]x[2]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=2 and y=6
	3:2	**Pixel Hashing Table Entry y[6]x[1]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=1 and y=6
	1:0	**Pixel Hashing Table Entry y[6]x[0]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=0 and y=6

PIXEL_HASH_TABLE_2BIT_128ENTRY

PIXEL_HASH_TABLE_2BIT_128ENTRY

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

Description

3-wayor 4-way pixel hashing table. Table is 128-entries:16X,8Y in [Y][X] format. Each entry is two bits that indicates which sub-slice hardware block the indicated xy pixel block is mapped.

pixelhash_id maps to color-pipe. A value of 0 indicates the largest color-pipe, or first enbled color-pipe if all enabled color-pipes are balanced. A value of 2 indicates the smallest color-pipe, or last enbled color-pipe if all enabled color-pipes are balanced.

DWord	Bit	Description
0	31:30	**Pixel Hashing Table Entry y[0]x[15]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=15 and y=0
	29:28	**Pixel Hashing Table Entry y[0]x[14]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=14 and y=0
	27:26	**Pixel Hashing Table Entry y[0]x[13]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=13 and y=0
	25:24	**Pixel Hashing Table Entry y[0]x[12]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=12 and y=0
	23:22	**Pixel Hashing Table Entry y[0]x[11]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=11 and y=0
	21:20	**Pixel Hashing Table Entry y[0]x[10]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=10 and y=0
	19:18	**Pixel Hashing Table Entry y[0]x[9]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=9 and y=0

PIXEL_HASH_TABLE_2BIT_128ENTRY		
	17:16	**Pixel Hashing Table Entry y[0]x[8]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=8 and y=0
	15:14	**Pixel Hashing Table Entry y[0]x[7]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=7 and y=0
	13:12	**Pixel Hashing Table Entry y[0]x[6]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=6 and y=0
	11:10	**Pixel Hashing Table Entry y[0]x[5]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=5 and y=0
	9:8	**Pixel Hashing Table Entry y[0]x[4]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=4 and y=0
	7:6	**Pixel Hashing Table Entry y[0]x[3]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=3 and y=0
	5:4	**Pixel Hashing Table Entry y[0]x[2]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=2 and y=0
	3:2	**Pixel Hashing Table Entry y[0]x[1]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=1 and y=0
	1:0	**Pixel Hashing Table Entry y[0]x[0]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=0 and y=0
1	31:30	**Pixel Hashing Table Entry y[1]x[15]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=15 and y=1

PIXEL_HASH_TABLE_2BIT_128ENTRY

	29:28	**Pixel Hashing Table Entry y[1]x[14]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=14 and y=1
	27:26	**Pixel Hashing Table Entry y[1]x[13]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=13 and y=1
	25:24	**Pixel Hashing Table Entry y[1]x[12]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=12 and y=1
	23:22	**Pixel Hashing Table Entry y[1]x[11]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=11 and y=1
	21:20	**Pixel Hashing Table Entry y[1]x[10]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=10 and y=1
	19:18	**Pixel Hashing Table Entry y[1]x[9]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=9 and y=1
	17:16	**Pixel Hashing Table Entry y[1]x[8]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=8 and y=1
	15:14	**Pixel Hashing Table Entry y[1]x[7]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=7 and y=1
	13:12	**Pixel Hashing Table Entry y[1]x[6]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=6 and y=1
	11:10	**Pixel Hashing Table Entry y[1]x[5]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=5 and y=1

PIXEL_HASH_TABLE_2BIT_128ENTRY		
	9:8	**Pixel Hashing Table Entry y[1]x[4]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=4 and y=1
	7:6	**Pixel Hashing Table Entry y[1]x[3]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=3 and y=1
	5:4	**Pixel Hashing Table Entry y[1]x[2]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=2 and y=1
	3:2	**Pixel Hashing Table Entry y[1]x[1]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=1 and y=1
	1:0	**Pixel Hashing Table Entry y[1]x[0]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=0 and y=1
2	31:30	**Pixel Hashing Table Entry y[2]x[15]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=15 and y=2
	29:28	**Pixel Hashing Table Entry y[2]x[14]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=14 and y=2
	27:26	**Pixel Hashing Table Entry y[2]x[13]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=13 and y=2
	25:24	**Pixel Hashing Table Entry y[2]x[12]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=12 and y=2
	23:22	**Pixel Hashing Table Entry y[2]x[11]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=11 and y=2

PIXEL_HASH_TABLE_2BIT_128ENTRY

	21:20	**Pixel Hashing Table Entry y[2]x[10]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=10 and y=2
	19:18	**Pixel Hashing Table Entry y[2]x[9]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=9 and y=2
	17:16	**Pixel Hashing Table Entry y[2]x[8]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=8 and y=2
	15:14	**Pixel Hashing Table Entry y[2]x[7]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=7 and y=2
	13:12	**Pixel Hashing Table Entry y[2]x[6]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=6 and y=2
	11:10	**Pixel Hashing Table Entry y[2]x[5]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=5 and y=2
	9:8	**Pixel Hashing Table Entry y[2]x[4]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=4 and y=2
	7:6	**Pixel Hashing Table Entry y[2]x[3]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=3 and y=2
	5:4	**Pixel Hashing Table Entry y[2]x[2]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=2 and y=2
	3:2	**Pixel Hashing Table Entry y[2]x[1]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=1 and y=2

PIXEL_HASH_TABLE_2BIT_128ENTRY		
	1:0	**Pixel Hashing Table Entry y[2]x[0]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=0 and y=2
3	31:30	**Pixel Hashing Table Entry y[3]x[15]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=15 and y=3
	29:28	**Pixel Hashing Table Entry y[3]x[14]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=14 and y=3
	27:26	**Pixel Hashing Table Entry y[3]x[13]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=13 and y=3
	25:24	**Pixel Hashing Table Entry y[3]x[12]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=12 and y=3
	23:22	**Pixel Hashing Table Entry y[3]x[11]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=11 and y=3
	21:20	**Pixel Hashing Table Entry y[3]x[10]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=10 and y=3
	19:18	**Pixel Hashing Table Entry y[3]x[9]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=9 and y=3
	17:16	**Pixel Hashing Table Entry y[3]x[8]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=8 and y=3
	15:14	**Pixel Hashing Table Entry y[3]x[7]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=7 and y=3

PIXEL_HASH_TABLE_2BIT_128ENTRY

	13:12	**Pixel Hashing Table Entry y[3]x[6]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=6 and y=3
	11:10	**Pixel Hashing Table Entry y[3]x[5]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=5 and y=3
	9:8	**Pixel Hashing Table Entry y[3]x[4]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=4 and y=3
	7:6	**Pixel Hashing Table Entry y[3]x[3]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=3 and y=3
	5:4	**Pixel Hashing Table Entry y[3]x[2]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=2 and y=3
	3:2	**Pixel Hashing Table Entry y[3]x[1]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=1 and y=3
	1:0	**Pixel Hashing Table Entry y[3]x[0]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=0 and y=3
4	31:30	**Pixel Hashing Table Entry y[4]x[15]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=15 and y=4
	29:28	**Pixel Hashing Table Entry y[4]x[14]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=14 and y=4
	27:26	**Pixel Hashing Table Entry y[4]x[13]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=13 and y=4

PIXEL_HASH_TABLE_2BIT_128ENTRY

	25:24	**Pixel Hashing Table Entry y[4]x[12]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=12 and y=4
	23:22	**Pixel Hashing Table Entry y[4]x[11]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=11 and y=4
	21:20	**Pixel Hashing Table Entry y[4]x[10]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=10 and y=4
	19:18	**Pixel Hashing Table Entry y[4]x[9]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=9 and y=4
	17:16	**Pixel Hashing Table Entry y[4]x[8]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=8 and y=4
	15:14	**Pixel Hashing Table Entry y[4]x[7]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=7 and y=4
	13:12	**Pixel Hashing Table Entry y[4]x[6]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=6 and y=4
	11:10	**Pixel Hashing Table Entry y[4]x[5]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=5 and y=4
	9:8	**Pixel Hashing Table Entry y[4]x[4]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=4 and y=4
	7:6	**Pixel Hashing Table Entry y[4]x[3]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=3 and y=4

PIXEL_HASH_TABLE_2BIT_128ENTRY

	5:4	**Pixel Hashing Table Entry y[4]x[2]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=2 and y=4
	3:2	**Pixel Hashing Table Entry y[4]x[1]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=1 and y=4
	1:0	**Pixel Hashing Table Entry y[4]x[0]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=0 and y=4
5	31:30	**Pixel Hashing Table Entry y[5]x[15]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=15 and y=5
	29:28	**Pixel Hashing Table Entry y[5]x[14]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=14 and y=5
	27:26	**Pixel Hashing Table Entry y[5]x[13]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=13 and y=5
	25:24	**Pixel Hashing Table Entry y[5]x[12]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=12 and y=5
	23:22	**Pixel Hashing Table Entry y[5]x[11]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=11 and y=5
	21:20	**Pixel Hashing Table Entry y[5]x[10]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=10 and y=5
	19:18	**Pixel Hashing Table Entry y[5]x[9]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=9 and y=5

PIXEL_HASH_TABLE_2BIT_128ENTRY		
	17:16	**Pixel Hashing Table Entry y[5]x[8]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=8 and y=5
	15:14	**Pixel Hashing Table Entry y[5]x[7]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=7 and y=5
	13:12	**Pixel Hashing Table Entry y[5]x[6]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=6 and y=5
	11:10	**Pixel Hashing Table Entry y[5]x[5]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=5 and y=5
	9:8	**Pixel Hashing Table Entry y[5]x[4]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=4 and y=5
	7:6	**Pixel Hashing Table Entry y[5]x[3]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=3 and y=5
	5:4	**Pixel Hashing Table Entry y[5]x[2]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=2 and y=5
	3:2	**Pixel Hashing Table Entry y[5]x[1]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=1 and y=5
	1:0	**Pixel Hashing Table Entry y[5]x[0]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=0 and y=5
6	31:30	**Pixel Hashing Table Entry y[6]x[15]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=15 and y=6

PIXEL_HASH_TABLE_2BIT_128ENTRY

	29:28	**Pixel Hashing Table Entry y[6]x[14]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=14 and y=6
	27:26	**Pixel Hashing Table Entry y[6]x[13]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=13 and y=6
	25:24	**Pixel Hashing Table Entry y[6]x[12]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=12 and y=6
	23:22	**Pixel Hashing Table Entry y[6]x[11]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=11 and y=6
	21:20	**Pixel Hashing Table Entry y[6]x[10]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=10 and y=6
	19:18	**Pixel Hashing Table Entry y[6]x[9]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=9 and y=6
	17:16	**Pixel Hashing Table Entry y[6]x[8]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=8 and y=6
	15:14	**Pixel Hashing Table Entry y[6]x[7]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=7 and y=6
	13:12	**Pixel Hashing Table Entry y[6]x[6]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=6 and y=6
	11:10	**Pixel Hashing Table Entry y[6]x[5]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=5 and y=6

PIXEL_HASH_TABLE_2BIT_128ENTRY		
	9:8	**Pixel Hashing Table Entry y[6]x[4]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=4 and y=6
	7:6	**Pixel Hashing Table Entry y[6]x[3]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=3 and y=6
	5:4	**Pixel Hashing Table Entry y[6]x[2]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=2 and y=6
	3:2	**Pixel Hashing Table Entry y[6]x[1]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=1 and y=6
	1:0	**Pixel Hashing Table Entry y[6]x[0]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=0 and y=6
7	31:30	**Pixel Hashing Table Entry y[7]x[15]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=15 and y=7
	29:28	**Pixel Hashing Table Entry y[7]x[14]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=14 and y=7
	27:26	**Pixel Hashing Table Entry y[7]x[13]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=13 and y=7
	25:24	**Pixel Hashing Table Entry y[7]x[12]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=12 and y=7
	23:22	**Pixel Hashing Table Entry y[7]x[11]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=11 and y=7

PIXEL_HASH_TABLE_2BIT_128ENTRY

	21:20	**Pixel Hashing Table Entry y[7]x[10]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=10 and y=7
	19:18	**Pixel Hashing Table Entry y[7]x[9]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=9 and y=7
	17:16	**Pixel Hashing Table Entry y[7]x[8]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=8 and y=7
	15:14	**Pixel Hashing Table Entry y[7]x[7]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=7 and y=7
	13:12	**Pixel Hashing Table Entry y[7]x[6]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=6 and y=7
	11:10	**Pixel Hashing Table Entry y[7]x[5]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=5 and y=7
	9:8	**Pixel Hashing Table Entry y[7]x[4]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=4 and y=7
	7:6	**Pixel Hashing Table Entry y[7]x[3]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=3 and y=7
	5:4	**Pixel Hashing Table Entry y[7]x[2]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=2 and y=7
	3:2	**Pixel Hashing Table Entry y[7]x[1]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=1 and y=7

PIXEL_HASH_TABLE_2BIT_128ENTRY		
	1:0	**Pixel Hashing Table Entry y[7]x[0]** Format: U2 Indicates the pixelhash_id for the pixel block that has x=0 and y=7

Pixel Sample Mask Render Target Message Header Control

MHC_RT_PSM - Pixel Sample Mask Render Target Message Header Control

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:16	**Dispatched Pixel/Sample Enables** Format: U16 One bit per pixel (or sample within pixel) indicating which pixels/samples were originally enabled when the thread was dispatched. The Dispatched Pixel/Sample Enables must be unmodified from the ones sent when the pixel shader thread was initiated. If the Dispatched Pixel/Sample Enables are modified, behavior is undefined. **Programming Notes** When operating in PER_SAMPLE mode these bits correspond to samples, not pixels. Each subspan slot (4 bits) corresponds to a specific sample location for the subspan. Note that in NUMSAMPLES_1 mode, a pixel and sample are synonomous. When operating in PER_PIXEL mode, this field is ignored, and instead the SampleEnableMask (obtained via bypass) are used to clear the Depth Scoreboard.
	15:0	**Pixel/Sample Enables** Format: U16 Specifies which pixels/samples are still lit based on kill instruction activity in the pixel shader. This mask is AND'd with the Dispatched Pixel/Sample Enables mask, and that is used to control actual accesses to the color buffer. Pixels/samples will be dropped on masked writes, and the GRF is not modified for masked reads. **Programming Notes** When operating in PER_SAMPLE mode these bits correspond to samples, not pixels, as the PS is run per-sample. Each subspan slot (4 bits) corresponds to a specific sample location for the subspan. When operating in PER_PIXEL mode, these bits still correspond to pixels, as the PS is run per-pixel. Each pixel's mask bit is replicated according to Number of Multisamples and combined with other masks to control writes to the multisample locations.

PPHWSP_LAYOUT

PPHWSP_LAYOUT - PPHWSP_LAYOUT	
Size (in bits):	32672
Default Value:	0x00000000, 0x00000000,

PPHWSP_LAYOUT - PPHWSP_LAYOUT

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,

PPHWSP_LAYOUT - PPHWSP_LAYOUT
0x00000000, 0x00000000,

PPHWSP_LAYOUT - PPHWSP_LAYOUT

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000

DWord	Bit	Description
0..3	127:0	**Reserved** Access: RO Format: MBZ

PPHWSP_LAYOUT - PPHWSP_LAYOUT

4	31:0	**Ring Head Pointer Storage** Description: The contents of the Ring Buffer Head Pointer register (register DWord 1) are written to this location either as result of an MI_REPORT_HEAD instruction or as the result of an automatic report (see RINGBUF registers).
5..15	351:0	**Reserved** Access: RO Format: MBZ
16	0	**Cumulative Context Run Time** This has the cumulative run time of the context on HW. HW reports CTX_TIMESTAMP to this location on a context switch. This value is written after the context save is complete. The value that is saved in the context image does not include the time between the saving of the cumulative value to context to the time we complete the save. If required for the value to always increment and not take the context save into consideration, driver must look at the value in the context image.
17	31:1	**Reserved** Access: RO Format: MBZ
	0	**Element Switch** Value / Name / Description: 0 — Indicates the context is not submitted as the first element in the execlist. 1 — Indicates the corresponding context has been submitted as first element of the execlist. Preempt Request Received Timestamp is the time when the pending execlist has been submitted to HW. Note that across multiple submissions a given context could be first or second element of an execlist. This bit will get set if the context has been submitted as the first element in the execlist.
18..19	63:0	**Preempt Request Received Timestamp** TIMESTAMP register sampled on preemption request is reported.
20..21	63:0	**Context Restore Complete Timestamp** TIMESTAMP register sampled on context restore complete is reported.
22..23	63:0	**Context Save Finished Timestamp** TIMESTAMP register sampled on context save completion is reported.
24..27	127:0	**MI_SEMAPHORE_WAIT** MI_SEMAPHORE_WAIT command on which the context got switched out due to semaphore wait. This field is only valid and must be looked at when the context switch reason in context status buffer is stated as Wait on Semaphore.

PPHWSP_LAYOUT - PPHWSP_LAYOUT		
28..31	127:0	**Reserved** Access: RO Format: MBZ
32..33 This field describes the most recent context switch status of the corresponding context.	63:0	**Context Switch Status Qword**
34..1020	31583:0	**Reserved** Access: RO Format: MBZ

Predicate Barrier Message Data Payload

MDP_PREDICATE_BARRIER - Predicate Barrier Message Data Payload

Source:	EuSubFunctionGateway
Size (in bits):	256
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

This response message is sent back only if the Gateway Barrier Message specifies that this is a predicated barrier. This response is written to the GRF writeback location, and the response length specified in the send message to the EU must be 1.

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Predicated Barrier Mask Sum** Format: U16 This field is a sum of the predicate mask bits sent by each thread. This field (and the DW containing it) is not written if the barrier is not marked as a predicated barrier. The kernel should compare this field to 0 for the predicated OR function and compare it to the workgroup size for the predicated AND function.
1..7	223:0	**Reserved** Access: RO Format: MBZ

Qword Data Payload Register

MDCR_QW - Qword Data Payload Register		
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.1	63:0	**Qword0** Format: U64 Specifies the slot 0 data in this payload register
0.2-0.3	63:0	**Qword1** Format: U64 Specifies the slot 1 data in this payload register
0.4-0.5	63:0	**Qword2** Format: U64 Specifies the slot 2 data in this payload register
0.6-0.7	63:0	**Qword3** Format: U64 Specifies the slot 3 data in this payload register

Qword SIMD8 Atomic Operation CMPWR8B Message Data Payload

MDP_AOP8_QW2 - Qword SIMD8 Atomic Operation CMPWR8B Message Data Payload

Size (in bits): 1024

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Slot[7:0] Src0[31:0]** Format: MDCR_DW Specifies the lower 32-bits of Slot [7:0] Source 0 data
1.0-1.7	255:0	**Slot[7:0] Src0[63:32]** Format: MDCR_DW Specifies the upper 32-bits of Slot [7:0] Source 0 data
2.0-2.7	255:0	**Slot[7:0] Src1[31:0]** Format: MDCR_DW Specifies the lower 32-bits of Slot [7:0] Source 1 data
3.0-3.7	255:0	**Slot[7:0] Src1[63:32]** Format: MDCR_DW Specifies the upper 32-bits of Slot [7:0] Source 1 data

Qword SIMD8 Atomic Operation CMPWR Message Data Payload

MDP_A64_AOP8_QW2 - Qword SIMD8 Atomic Operation CMPWR Message Data Payload

Size (in bits): 1024

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Slot[3:0] Src0** Format: MDCR_QW Specifies the Slot [3:0] Source 0 data
1.0-1.7	255:0	**Slot[7:4] Src0** Format: MDCR_QW Specifies the Slot [7:4] Source 0 data
2.0-2.7	255:0	**Slot[3:0] Src1** Format: MDCR_QW Specifies the Slot [3:0] Source 1 data
3.0-3.7	255:0	**Slot[7:4] Src1** Format: MDCR_QW Specifies the Slot [7:4] Source 1 data

Qword SIMD8 Atomic Operation Return Data Message Data Payload

MDP_AOP8_QW1 - Qword SIMD8 Atomic Operation Return Data Message Data Payload

Size (in bits): 512

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Slot[7:0] Qword[31:0]** Format: MDCR_DW Specifies the lower 32-bits of Slot [7:0] Return data
1.0-1.7	255:0	**Slot[7:0] Qword[63:32]** Format: MDCR_DW Specifies the upper 32-bits of Slot [7:0] Return data

Qword SIMD8 Data Payload

MDP_QW_SIMD8 - Qword SIMD8 Data Payload		
Size (in bits):	512	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Data[3:0]** Format: **MDCR_QW** Specifies the Slot [3:0] data
1.0-1.7	255:0	**Data[7:4]** Format: **MDCR_QW** Specifies the Slot [7:4] data

Qword SIMD16 Atomic Operation CMPWR8B Message Data Payload

MDP_AOP16_QW2 - Qword SIMD16 Atomic Operation CMPWR8B Message Data Payload

Size (in bits): 2048

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Slot[7:0] Src0[31:0]** Format: MDCR_DW Specifies the lower 32-bits of Source 0 data for Slot [7:0]
1.0-1.7	255:0	**Slot[15:8] Src0[31:0]** Format: MDCR_DW Specifies the lower 32-bits Source 0 data for Slot [15:8]
2.0-2.7	255:0	**Slot[7:0] Src0[63:32]** Format: MDCR_DW Specifies the upper 32-bits of Source 0 data for Slot [7:0]
3.0-3.7	255:0	**Slot[15:8] Src0[63:32]** Format: MDCR_DW Specifies the upper 32-bits Source 0 data for Slot [15:8]
4.0-4.7	255:0	**Slot[7:0] Src1[31:0]** Format: MDCR_DW Specifies the lower 32-bits of Source 1 data for Slot [7:0]
5.0-5.7	255:0	**Slot[15:8] Src1[31:0]** Format: MDCR_DW Specifies the lower 32-bits Source 1 data for Slot [15:8]

MDP_AOP16_QW2 - Qword SIMD16 Atomic Operation CMPWR8B Message Data Payload		
6.0-6.7	255:0	**Slot[7:0] Src1[63:32]** Format: **MDCR_DW** Specifies the upper 32-bits of Source 1 data for Slot [7:0]
7.0-7.7	255:0	**Slot[15:8] Src1[63:32]** Format: **MDCR_DW** Specifies the upper 32-bits Source 1 data for Slot [15:8]

Qword SIMD16 Atomic Operation Return Data Message Data Payload

MDP_AOP16_QW1 - Qword SIMD16 Atomic Operation Return Data Message Data Payload

Size (in bits): 1024

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Slot[7:0] Qword[31:0]** Format: MDCR_DW Specifies the lower 32-bits of Return data for Slot [7:0]
1.0-1.7	255:0	**Slot[15:8] Qword[31:0]** Format: MDCR_DW Specifies the lower 32-bits of Return data for Slot [15:8]
2.0-2.7	255:0	**Slot[7:0] Qword[63:32]** Format: MDCR_DW Specifies the upper 32-bits of Return data for Slot [7:0]
3.0-3.7	255:0	**Slot[15:8] Qword[63:32]** Format: MDCR_DW Specifies the upper 32-bits of Return data for Slot [15:8]

Qword SIMD16 Data Payload

MDP_QW_SIMD16 - Qword SIMD16 Data Payload		
Size (in bits):	1024	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Data[3:0]** Format: MDCR_QW Specifies the Slot [3:0] data
1.0-1.7	255:0	**Data[7:4]** Format: MDCR_QW Specifies the Slot [7:4] data
2.0-2.7	255:0	**qw11_qw8** Format: MDCR_QW Specifies the Slot [11:8] data
3.0-3.7	255:0	**qw15_qw12** Format: MDCR_QW Specifies the Slot [15:12] data

Read-Only Data Port Message Types

MT_DP_RO - Read-Only Data Port Message Types

Source:	EuSubFunctionReadOnlyDataPort
Size (in bits):	5
Default Value:	0x00000000

Description
Lists all the Message Types in a Read-Only Data Port Message Descriptor [18:14].Read operations from the Constant Cache and Sampler Cache are encoded in the Read-Only Data Port.Many of the operations are also implemented in Data Port 0, and those operations use the same Message Header.
Lists all the Message Types in a Read-Only Data Port Message Descriptor [18:14].Read operations from the Constant Cache are encoded in the Read-Only Data Port.Many of the operations are also implemented in Data Port 0, and those operations use the same Message Header.

DWord	Bit	Description
0	4:0	**Message Type** Specifies type of message

Value	Name	Description
00h	MT_CC_OWB **[Default]**	Oword Block Read Constant Cache message
01h	MT_CC_OWUB	Unaligned Oword Block Read Constant Cache message
03h	MT_CC_DWS	Dword Scattered Read Constant Cache message
04h	MT_SC_OWUB	Unaligned Oword Block Read Sampler Cache message
05h	MT_SC_MB	Media Block Read Sampler Cache message
06h	MT_RSI	Read Surface Info message
Others	Reserved	Ignored

Read Surface Info 32-Bit Address Payload

MAP32B_RSI - Read Surface Info 32-Bit Address Payload		
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0	31:0	**U** Format: U32 Specifies the U channel address offset.
0.1	31:0	**V** Format: U32 Specifies the V channel address offset.
0.2	31:0	**R** Format: U32 Specifies the R channel address offset.
0.3	31:0	**LOD** Format: MACD_LOD Specifies the LOD.
0.4-0.7	127:0	**Reserved** Access: RO Format: MBZ

Read Surface Info Address Payload

DP_ASTATE_INFO_PAYLOAD - Read Surface Info Address Payload

Size (in bits): 128

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0.0	31:0	**U** Format: U32 Specifies the U channel address offset.
0.1	31:0	**V** Format: U32 Specifies the V channel address offset.
0.2	31:0	**R** Format: U32 Specifies the R channel address offset.
0.3	31:0	**LOD** Format: MACD_LOD Specifies the LOD.

Read Surface Info Data Payload

MDP_RSI - Read Surface Info Data Payload

Size (in bits): 512

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.5	191:0	**Reserved** Access: RO Format: MBZ
0.6-0.7	63:0	**Instruction Base Address** Format: GraphicsAddress[63:0] Instruction Base Address from STATE_BASE_ADDRESS, extended to 64-bit format. **Programming Notes** The 48-bit address is returned in a 64-bit address in canonical form. **Restriction** The Instruction Base Address returned is incorrect if the thread is from CCS queue.
1.0	31:0	**Width** Format: U32 Surface Width generally computed from RENDER_SURFACE_STATE Width (stored as width minus 1). The value is 0 for NULL surface, and in all other cases (Width+1) » LOD. Surface Width from RENDER_SURFACE_STATE (U14), zero extended to 32 bits.
1.1	31:0	**Height** Format: U32 Surface Height, generally computed from RENDER_SURFACE_STATE Height (stored as height minus 1). The value for a 1D array is RENDER_SUFACE_STATE's (Depth + 1). The value for 1D non-array, BUFFER, and NULL surface is 0. In all other case, the value is (Height + 1) » LOD.
1.2	31:0	**Depth** Format: U32 Surface Depth, generally computed from RENDER_SURFACE_STATE Depth (which is stored depth minus 1). If 2D Array or Cube Array surface, value is the (Depth+1). If 3D surface, value is (Depth+1) » LOD. In all other case, the value is 0.
1.3	31:0	**MIP Count** Format: U32 MIP Count from RENDER_SURFACE_STATE, range [0, 14], zero extended to 32 bits.

MDP_RSI - Read Surface Info Data Payload

1.4	31:0	**Surface Type** Format: U32 Surface Type from RENDER_SURFACE_STATE, zero extended to 32 bits

Value	Name	Description
0h	SURFTYPE_1D	1-dimensional map or array of maps
1h	SURFTYPE_2D	2-dimensional map or array of maps
2h	SURFTYPE_3D	3-dimensional map (volumetric) of maps
3h	SURFTYPE_CUBE	Cube map or array of cube maps
4h	SURFTYPE_BUFFER	Element in a buffer
7h	SURTYPE_NULL	Null surface
Others	Reserved	Reserved

1.5	31:0	**Surface Format** Format: U32 Surface Format from RENDER_SURFACE_STATE (U9), zero extended to 32 bits.
1.6-1.7	63:0	**Reserved** Access: RO Format: MBZ

RENDER_SURFACE_STATE

RENDER_SURFACE_STATE

Exists If:	//[MessageType] != 'Sample_8x8'
Size (in bits):	512
Default Value:	0x0001C000, 0x80000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

This is the normal surface state used by all messages that use SURFACE_STATE except those that use MEDIA_SURFACE_STATE.

DWord	Bit	Description
0	31:29	**Surface Type** This field defines the type of the surface.

Value	Name	Description
0h	SURFTYPE_1D	Defines a 1-dimensional map or array of maps
1h	SURFTYPE_2D	Defines a 2-dimensional map or array of maps
2h	SURFTYPE_3D	Defines a 3-dimensional (volumetric) map
3h	SURFTYPE_CUBE	Defines a cube map or array of cube maps
4h	SURFTYPE_BUFFER	Defines an element in a buffer
6h	Reserved	
7h	SURFTYPE_NULL	Defines a null surface

Programming Notes

A null surface is used in instances where an actual surface is not bound. When a write message is generated to a null surface, no actual surface is written to. When a read message (including any sampling engine message) is generated to a null surface, the result is all zeros. Note that a null surface type is allowed to be used with all messages, even if it is not specifically indicated as supported. All of the remaining fields in surface state are ignored for null surfaces, with the following exceptions:

- **Width, Height, Depth, LOD, Render Target View Extent, Minimum Array element** fields must match for all surfaces that compose the current render target.

All sampling engine and data port messages support null surfaces with the above behavior, even if not mentioned as specifically supported, except for the following:

- The **Surface Type** of a surface used as a render target (accessed via the Data Port's Render Target Write message) must be the same as the **Surface Type** of all other render targets and of the depth / stencil buffer (defined in 3DSTATE_DEPTH_BUFFER, 3DSTATE_STENCIL_BUFFER), unless either the stencil buffer, depth buffer or render targets are SURFTYPE_NULL.

	28	**Surface Array** Format: Enable This field, if enabled, indicates that the surface is an array. **Programming Notes** If this field is *enabled*, the **Surface Type** must be SURFTYPE_1D, SURFTYPE_2D, or SURFTYPE_CUBE. If this field is *disabled* and **Surface Type** is SURFTYPE_1D, SURFTYPE_2D, or SURFTYPE_CUBE, the **Depth** field must be set to zero. To avoid performance issues with the sampler texture cache this bit should not be set unless the depth of the arrayed surface is greater than 1 (Depth field >1).
	27	**Reserved** Access: RO Format: MBZ
	26:18	**Surface Format** Format: SURFACE_FORMAT This field specifies the format of the surface or element within this surface. This field is ignored for all data port messages other than the render target message and streamed vertex buffer write message. Some forms of the media block messages use the surface format. **Programming Notes** If **Number of Multisamples** is set to a value other than MULTISAMPLECOUNT_1, this field cannot be set to the following formats: • Any compressed texture format (BC*, DXT*, FXT*, ETC*, EAC*) • Any YCRCB* format If **ASTC_Enable** is set to 1: • ASTC_Profile: ASTC_Profile can be freely set to 1 or 0. • ASTC_BlockDimention: Bit [25] will be set to 0 if ASTC_Profile is LDR-profile since only 2D blocks are supported in LDR profile. 2D HDR-profile can be supported where the ASTC_Profile is set at 1 (Full-Profile) and ASTC_BlockDimention is set as 0 (2D). This field cannot be a YUV (YCRCB*) or compressed (BC*, DXT*, FXT*, ETC*, EAC*) format if the **Surface Type** is SURFTYPE_BUFFER. This field cannot be a compressed (BC*, DXT*, FXT*, ETC*, EAC*) format if the **Surface Type** is SURFTYPE_1D.
	17:16	**Surface Vertical Alignment** **Description** **For Sampling Engine and Render Target Surfaces:** This field specifies the vertical alignment requirement in elements for the surface. Refer to the "Memory Data Formats" chapter for details on how this field changes the layout of the surface in memory. An *element* is defined as a pixel in uncompressed surface formats, and as a compression block in compressed surface formats. For MSFMT_DEPTH_STENCIL type multisampled surfaces, an element is a sample.

This field is used for 2D, CUBE, and 3D surface alignment when Tiled Resource Mode is TRMODE_NONE (Tiled Resource Mode is disabled). This field is ignored for 1D surfaces and also when Tiled Resource Mode is not TRMODE_NONE (e.g. Tiled Resource Mode is enabled).
See the appropriate Alignment table in the "Surface Layout and Tiling" section under Common Surface Formats for the table of alignment values for Tiled Resources.

The vertical alignment field (VALIGN) is also used to determine the start of the surface in memory. All surface are vertically aligned to VALIGN rows or VALIGN rows times the block height for block-compressed surface formats.

For other surfaces: This field is ignored.

Value	Name	Description
0h	Reserved	Reserved
1h	VALIGN_4 **[Default]**	Vertical alignment factor j = 4
2h	VALIGN_8	Vertical alignment factor j = 8
3h	VALIGN_16	Vertical alignment factor j = 16

Programming Notes

This field is intended to be set to VALIGN_4 if the surface was rendered as a depth buffer, for a multisampled (4x) render target, or for a multisampled (8x) render target, since these surfaces support only alignment of 4. Use of VALIGN_4 for other surfaces is supported, but increases memory usage.

This field is intended to be set to VALIGN_8 only if the surface was rendered as a stencil buffer, since stencil buffer surfaces support only alignment of 8. If set to VALIGN_8, Surface Format must be R8_UINT.
This field should also be set to VALIGN_8 if the surface was rendered as a D16_UNORM depth buffer, for render target of 1X/4X/16X.

For uncompressed surfaces, the units of "j" are rows of pixels on the physical surface. For compressed texture formats, the units of "j" are in compression blocks, thus each increment in "j" is equal to h pixels, where h is the height of the compression block in pixels.
See**Surface Layout and Tiling**for a full description of how i and j parameters are used to determine horizontal and vertical offset to the start of a MIP.

15:14 **Surface Horizontal Alignment**

Description

For Sampling Engine and Render Target Surfaces: This field specifies the horizontal alignment requirement for the surface.

This field is used for alignment when LOD >= Mip Tail Start LOD
This field is ignored when Tiled Resource Mode is not TRMODE_NONE (i.e. Tiled Resources are enabled). See the "Surface Layout and Tiling" section under Common Surface Formats for the table of alignment values for Tile Resources.

For other surfaces: This field is ignored.

Value	Name	Description
0h	Reserved	Reserved
1h	HALIGN_4	Horizontal alignment factor j = 4
2h	HALIGN_8	Horizontal alignment factor j = 8
3h	HALIGN_16 **[Default]**	Horizontal alignment factor j = 16

Programming Notes
See **Alignment Unit Size** For restrictions on Horizontal Alignment.
When Auxiliary Surface Mode is set to AUX_MCS_LCE or AUX_CCS_E, HALIGN_16 must be used unless contradicts **Alignment Unit Size** restrictions.

13:12 **Tile Mode**

Description
This field specifies the type of memory tiling (Linear, WMajor, XMajor, or YMajor) employed to tile this surface. See *Memory Interface Functions* for details on memory tiling and restrictions.
This field specifies the type of memory tiling (Linear, XMajor, or YMajor) employed to tile this surface. See *Memory Interface Functions* for details on memory tiling and restrictions.

Value	Name	Description	Programming Notes
0h	LINEAR	Linear mode (no tiling)	
1h	Reserved	Reserved	
2h	XMAJOR	X major tiling	
3h	YMAJOR	Y major tiling	

Programming Notes
• Refer to *Memory Data Formats* for restrictions on *TileMode* direction for the various buffer types. (Of particular interest is the fact that YMAJOR tiling is not supported for display/overlay buffers). • The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this field. • Linear surfaces can be mapped to Main Memory (uncached) or System Memory (cacheable, snooped). Tiled (X/Y/W) surfaces can only be mapped to Main Memory. • If **Surface Type** is SURFTYPE_BUFFER, this field must be TILEMODE_LINEAR • If **Number of Multisamples** is not MULTISAMPLECOUNT_1, this field must be YMAJOR.
If **Surface Type** is SURFTYPE_1D this field must be TILEMODE_LINEAR, unless **Sampler Legacy 1D Map Layout Disable** is set to 0, in which case TILEMODE_YMAJOR is also allowed. **Tiled Resource Mode** must be set to TRMODE_NONE for these cases.
If **Surface Format** is ASTC*, this field must be TILEMODE_YMAJOR.

11 **Vertical Line Stride**

Format:	U1

For 2D Non-Array Surfaces accessed via the Sampling Engine or Data Cache Data Port: Specifies number of lines (0 or 1) to skip between logically adjacent lines - provides support of interleaved (field) surfaces as textures.
For Other Surfaces: Vertical Line Stride must be zero.

Programming Notes
This bit must not be set if the surface format is a compressed type (BCn*, FXT1, ETC*, EAC*).
This bit must not be set if the **Auxiliary Surface Mode** is not AUX_NONE.
If this bit is set on a sampling engine surface, the mip mode filter must be set to MIPFILTER_NONE

10 **Vertical Line Stride Offset**

Format:	U1

For 2D Non-Array Surfaces accessed via the Sampling Engine or Data Cache Data Port: Specifies the offset of the initial line from the beginning of the buffer. Ignored when Vertical **Line Stride** is 0.
For Other Surfaces: Vertical Line Stride Offset must be zero.

9 **Sampler L2 Out of Order Mode Disable**

Format:	Disable

If disabled this will forced formats which would have bypassed the L2 and been filled into the L1 out of order to be cached in the L2 and send in order to the L1. In general that is any format which is expanded 1:4, 1:2 in L1 or not expanded at all. This would include all lossless compressed cases.

Programming Notes
This bit must be set for the following surface types: BC2_UNORM BC3_UNORM BC5_UNORM BC5_SNORM BC7_UNORM

8 **Render Cache Read Write Mode**
For Surfaces accessed via the Data Port to Render Cache: This field specifies the way Render Cache treats a write request. If unset, Render Cache allocates a write-only cache line for a write miss. If set, Render Cache allocates a read-write cache line for a write miss.
For Surfaces accessed via the Sampling Engine or Data Port to Texture Cache or Data Cache: This field is reserved : MBZ

Value	Name	Description
0h	Write-Only Cache	Allocating write-only cache for a write miss
1h	Read-Write Cache	Allocating read-write cache for a write miss

Programming Notes
This field is provided for performance optimization for Render Cache read/write accesses.

7:6 **Media Boundary Pixel Mode**

For 2D Non-Array Surfaces accessed via the Data Port Media Block Read Message or Data Port Transpose Read message: This field enables control of which rows are returned on vertical out-of-bounds reads using the Data Port Media Block Read Message or Data Port Transpose Read message. In the description below, frame mode refers to **Vertical Line Stride** = 0, field mode is **Vertical Line Stride** = 1 in which only the even or odd rows are addressable. The frame refers to the entire surface, while the field refers only to the even or odd rows within the surface.

For Other Surfaces: Reserved : MBZ

Value	Name	Description
0h	NORMAL_MODE	The row returned on an out-of-bound access is the closest row in the frame or field. Rows from the opposite field are never returned.
1h	Reserved	
2h	PROGRESSIVE_FRAME	The row returned on an out-of-bound access is the closest row in the frame, even if in field mode.
3h	INTERLACED_FRAME	In field mode, the row returned on an out-of-bound access is the closest row in the field. In frame mode, even out-of-bound rows return the nearest even row while odd out-of-bound rows return the nearest odd row.

5 **Cube Face Enable - Negative X**

Format:	Enable

For SURFTYPE_CUBE Surfaces accessed via the Sampling Engine: This field enable the individual face of a cube map. Enabling a face indicates that the face is present in the cube map, while disabling it indicates that that face is represented by the texture map's border color. Refer to Memory Data Formats for the correlation between faces and the cube map memory layout. Note that storage for disabled faces must be provided.

Programming Notes
When TEXCOORDMODE_CLAMP is used when accessing a cube map, this field must be programmed to 1b (face enabled).
This field must be programmed to 1h (enabled) whenever **Surface Type** is programmed to **SURFTYPE_CUBE**

4 **Cube Face Enable - Positive X**

Format:	Enable

For SURFTYPE_CUBE Surfaces accessed via the Sampling Engine: This field enable the individual face of a cube map. Enabling a face indicates that the face is present in the cube map, while disabling it indicates that that face is represented by the texture map's border color. Refer to Memory Data Formats for the correlation between faces and the cube map memory layout. Note that storage for disabled faces must be provided.

Programming Notes
When TEXCOORDMODE_CLAMP is used when accessing a cube map, this field must be programmed to 1b (face enabled).
This field must be programmed to 1h (enabled) whenever **Surface Type** is programmed to **SURFTYPE_CUBE**

3 **Cube Face Enable - Negative Y**

Format:	Enable

For SURFTYPE_CUBE Surfaces accessed via the Sampling Engine: This field enable the individual face of a cube map. Enabling a face indicates that the face is present in the cube map, while disabling it indicates that that face is represented by the texture map's border color. Refer to Memory Data Formats for the correlation between faces and the cube map memory layout. Note that storage for disabled faces must be provided.

Programming Notes
When TEXCOORDMODE_CLAMP is used when accessing a cube map, this field must be programmed to 1b (face enabled).
This field must be programmed to 1h (enabled) whenever **Surface Type** is programmed to **SURFTYPE_CUBE**

2 **Cube Face Enable - Positive Y**

Format:	Enable

For SURFTYPE_CUBE Surfaces accessed via the Sampling Engine: This field enable the individual face of a cube map. Enabling a face indicates that the face is present in the cube map, while disabling it indicates that that face is represented by the texture map's border color. Refer to Memory Data Formats for the correlation between faces and the cube map memory layout. Note that storage for disabled faces must be provided.

Programming Notes
When TEXCOORDMODE_CLAMP is used when accessing a cube map, this field must be programmed to 1b (face enabled).
This field must be programmed to 1h (enabled) whenever **Surface Type** is programmed to **SURFTYPE_CUBE**

1 **Cube Face Enable - Negative Z**

Format:	Enable

For SURFTYPE_CUBE Surfaces accessed via the Sampling Engine: This field enable the individual face of a cube map. Enabling a face indicates that the face is present in the cube map, while disabling it indicates that that face is represented by the texture map's border color. Refer to Memory Data Formats for the correlation between faces and the cube map memory layout. Note that storage for disabled faces must be provided.

Programming Notes
When TEXCOORDMODE_CLAMP is used when accessing a cube map, this field must be programmed to 1b (face enabled).
This field must be programmed to 1h (enabled) whenever **Surface Type** is programmed to **SURFTYPE_CUBE**

0 **Cube Face Enable - Positive Z**

Format:	Enable

For SURFTYPE_CUBE Surfaces accessed via the Sampling Engine: This field enable the individual face of a cube map. Enabling a face indicates that the face is present in the cube map, while disabling it indicates that that face is represented by the texture map's border color. Refer to Memory Data Formats for the correlation between faces and the cube map memory layout. Note

that storage for disabled faces must be provided.

Programming Notes
When TEXCOORDMODE_CLAMP is used when accessing a cube map, this field must be programmed to 1b (face enabled).
This field must be programmed to 1h (enabled) whenever **Surface Type** is programmed to **SURFTYPE_CUBE**

1 | 31 **Enable Unorm Path in Color Pipe**

Format:	Enable

Enables Unorm Path (fixed Point Conversion of floating point for fill and blend in DAPRSS) in color Pipe.

Value	Name	Description
1	ENABLE **[Default]**	Enables Unorm Path in Color Pipe.
0	DISABLE	Disables Unorm path in Color Pipe.

Programming Notes
This bit should never be programmed to 0

30:24 **Memory Object Control State**

Format:	MEMORY_OBJECT_CONTROL_STATE

Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).

23:19 **Base Mip Level**

Format:	U4.1

Range: [0.0, 14.0]
Specifies which mip level is considered the "base" level when determining mag-vs-min filter and selecting the "base" mip level.

Programming Notes
This field also exists in SAMPLER_STATE. If both fields are zero, the Base Mip Level is zero. If one is nonzero, Base Mip Level is the nonzero field. It is illegal to have both Base Mip Level fields nonzero.

18 **Corner Texel Mode**

Format:	Enable

This field, when ENABLED, indicates when a surface is using corner texel-mode for sampling.
Corner Texel Mode is ignored for Planar YUV/YCrCb surface formats.
Corner Texel Mode is ignored for sample_8X8 and sample_unorm messasge types.
Corner Texel Mode is not supported with Non-Normalized Coordinates
Does not support legacy sampler features set0 See legacy sampler page for more details

Value	Name	Description
0h	Disable **[Default]**	When programmed to 0h, Corner Texel Mode is disabled. This means texel references are shifted a half-texel from the upper-right corner of the texture map which is the standard texel reference mode.
1h	Enable	When programmed to 1h, Corner Texel Mode is enabled. The location of a sampled texel on a texture map is shifted a half-texel to the upper-left, meaning texel (0,0) isin the exact upper-left corner of the surface.

17 **Double Fetch Disable**

Format:	Disable

This bit is intended to disable the "double fetch" of adjacent cache-lines in most all cases. Double fetch is a performance mode, but for some surface types for formats it may be lower performance due to fetching unused cache-lines.

Value	Name	Description
0h	Enable **[Default]**	When programmed to 0h, double fetches are allowed.
1h	Disable	When programmed to 1h, double fetches are disabled.

16 **Reserved**

Access:	RO
Format:	MBZ

15 **Sample Tap Discard Disable**

This bit forces sample tap discard filter mode to be disabled for this surface state. This bit must be set for surfaces which are no Alpha Channel such as R8G8B8_UNORM.

Value	Name	Description
0h	DISABLE **[Default]**	When programmed to 0h, Sample Tap Discard filter mode is allowed and is not disabled by this bit. This bit is ignored if Sample Tap Discard is not enabled in the Sampler State.
1h	ENABLE	When programmed to 1h, Sample Tap Discard filter mode will be disabled even if enabled through Sampler State

Programming Notes
This bit must be set for all Planar YUV surface formats (e.g. PLANAR_420_8, PLANAR_420_16)

14:0 **Surface QPitch**

Format:	U17[16:2]

The interpretation of this field is dependent on Surface Type as follows:

- SURFTYPE_1D: distance in *pixels* between array slices
- SURFTYPE_2D/CUBE: distance in *rows* between array slices. For Quilted Textures this field specifies the distance in rows between *quilt* slices. For compressed texture formats, one row contains a complete compression block vertically.
- SURFTYPE_3D: distance in *rows* between R-slices [**Note:** these *rows* are only in the vertical dimension without considering the depth dimension]. For compressed texture formats, one row contains a complete compression block vertically.

- Other surface types: field is ignored

Value	Name	Description
[0,32767]		1 is added to the value and it is multiplied by 4 to determine the height in rows.

Programming Notes
For Surface Type 1D: This field must be set to an integer multiple of the **Surface Horizontal Alignment** **For Surface Type 2D, CUBE:** This field must be set to an integer multiple of the **Surface Vertical Alignment** **For Surface Type 3D:** *Tile Mode != Linear:* This field must be set to an integer multiple of the tile height (2^Cv) *Tile Mode == Linear:* This field must be set to an integer multiple of the Surface Vertical Alignment **Note:** for compressed textures (BC*, FXT1, ETC*, EAC*), this field is in units of rows of compression blocks.
Software must ensure that this field is set to a value sufficiently large such that the array slices in the surface do not overlap. Refer to the Memory Data Formats section for information on how surfaces are stored in memory.

2 | 31 **Depth/Stencil Resource**

Description
This bit field, when set, indicates if the resource is created as Depth/Stencil resource.

Programming Notes
SW must set this bit for any resource that was created with Depth/Stencil resource flag. Setting this bit allows HW to properly interpret the data-layout for various cases. For any resource that's created without Depth/Stencil resource flag, it must be reset.

30 **Reserved**

Access:	RO
Format:	MBZ

29:16 **Height**

Format:	U14-1

This field specifies the height of the surface, minus 1. If the surface is MIP-mapped, this field contains the height of the base MIP level. For buffers, this field specifies a portion of the buffer size.

Value	Name	Description	Exists If
[0,0]		must be zero	[Surface Type] == 'SURFTYPE_1D'
[0,8191]		height of surface - 1 (y/v dimension) If Vertical Line Stride is set for a 2D surface, then the maximum allowed height is 2^8.	([SurfaceType] == 'SURFTYPE_2D') && ([VerticalLineStride] == 1)
[0,16383]		height of surface - 1 (y/v	([SurfaceType] == 'SURFTYPE_2D') &&

	dimension)	([VerticalLineStride] == 0)
[0,2047]	height of surface -1 (y/v)	[SurfaceType]=='SURFTYPE_3D'
[0,16383]	height of surface - 1 (y/v dimension)	[SurfaceType] == 'SURFTYPE_CUBE'
[0,16383]	contains bits [20:7] of the number of entries in the buffer - 1	[SurfaceType] == 'SURFTYPE_BUFFER'

Programming Notes

For typed buffer and structured buffer surfaces, the number of entries in the buffer ranges from 1 to 2^{27}. For raw buffer surfaces, the number of entries in the buffer is the number of bytes which can range from 1 to 2^{30}. After subtracting one from the number of entries, software must place the fields of the resulting 27-bit value into the **Height, Width**, and **Depth** fields as indicated, right-justified in each field. Unused upper bits must be set to zero.

If **Vertical Line Stride** is 1, this field indicates the height of the field, not the height of the frame

The **Height** of a render target must be the same as the **Height** of the other render targets and the depth buffer (defined in 3DSTATE_DEPTH_BUFFER), unless **Surface Type** is SURFTYPE_1D or SURFTYPE_2D with **Depth** = 0 (non-array) and **LOD** = 0 (non-mip mapped).

If this surface in memory is accessed with Vertical Line Stride set to both 0 and 1, this field must be an even value when Vertical Line Stride is 0.

If Media Pixel Boundary Mode is not set to NORMAL_MODE, this field must be an even value.

If Surface Format is PLANAR*, this field must be a multiple of 4

15:14 **Reserved**

Access:	RO
Format:	MBZ

13:0 **Width**

Format:	U14-1

This field specifies the width of the surface, minus 1. If the surface is MIP-mapped, this field specifies the width of the base MIP level. The width is specified in units of pixels or texels. For buffers, this field specifies a portion of the buffer size.

For surfaces accessed with the Media Block Read/Write message, this field is in units of DWords.

For surfaces accessed with the Transpose Read Message, this field is in units of DWords.

Value	Name	Description	Exists If
[0,16383]		width of surface - 1 (x/u dimension)	[SurfaceType] == 'SURFTYPE_1D'
[0,16383]		width of surface - 1 (x/u dimension)	[SurfaceType] == 'SURFTYPE_2D'
[0,16383]		width of surface - 1 (x/u dimension)	[SurfaceType] == 'SURFTYPE_3D'
[0,2047]		width of surface - 1 (x/u dimension)	[SurfaceType]=='SURFTYPE_3D'
[0,16383]		width of surface - 1 (x/u dimension)	[SurfaceType] == 'SURFTYPE_CUBE'

		[0,127]		contains bits [6:0] of the number of entries in the buffer - 1	[SurfaceType] == 'SURFTYPE_BUFFER'

Programming Notes

- For surface types other than SURFTYPE_BUFFER. The Width specified by this field must be less than or equal to the surface pitch (specified in bytes via the Surface Pitch field).
- For cube maps, Width must be set equal to the Height.
- The **Width** of a render target must be the same as the **Width** of the other render target(s) and the depth buffer (defined in 3DSTATE_DEPTH_BUFFER), unless **Surface Type** is SURFTYPE_1D or SURFTYPE_2D with **Depth** = 0 (non-array) and **LOD** = 0 (non-mip mapped).
- The **Width** of a render target with YUV surface format must be a multiple of 2.
- For SURFTYPE_BUFFER: The low two bits of this field must be 11 if the Surface Format is RAW (the size of the buffer must be a multiple of 4 bytes).

If **Surface Format** is PLANAR*, this field must be a multiple of 2

A known issue exists if a primitive is rendered to the first 2 rows and last 2 columns of a 16K width surface. If any geometry is drawn inside this square it will be copied to column X=2 and X=3 (arrangement on Y position will stay the same). If any geometry exceeds the boundaries of this 2x2 region it will be drawn normally. The issue also only occurs if the surface has TileMode != Linear

3 | 31:21 | **Depth**

Format:	U11-1

This field specifies the total number of levels, minus 1, for a volume texture or the number of array elements, minus 1, allowed to be accessed starting at the **Minimum Array Element** for arrayed surfaces. If the volume texture is MIP-mapped, this field specifies the depth of the base MIP level. For buffers, this field specifies a portion of the buffer size.

Value	Name	Description	Exists If
[0,2047]		number of array elements - 1	[SurfaceType] == 'SURFTYPE_1D'
[0,2047]		number of array elements - 1	[SurfaceType] == 'SURFTYPE_2D'
[0,2047]		depth of surface - 1 (z/r dimension)	[SurfaceType] == 'SURFTYPE_3D'
[0,340]		number of array elements - 1 [see programming notes for range]	[SurfaceType] == 'SURFTYPE_CUBE'
[0,2047]		contains bits [31:21] of the number of entries in the buffer - 1	[SurfaceType] == SURFTYPE_BUFFER

Programming Notes

The **Depth** of a render target must be the same as the **Depth** of the other render target(s) and of the depth buffer (defined in 3DSTATE_DEPTH_BUFFER).

	For SURFTYPE_CUBE: For Sampling Engine Surfaces and Typed Data Port Surfaces, the range of this field is [0,340], indicating the number of cube array elements (equal to the number of underlying 2D array elements divided by 6). For other surfaces, this field must be zero.
	For SURFTYPE_1D, 2D, and CUBE: The range of this field is reduced by one for each increase from zero of **Minimum Array Element**. For example, if **Minimum Array Element** is set to 1024 on a 2D surface, the range of this field is reduced to [0,1023].

20 Tile Address Mapping Mode

Format:	U1

This field is used to select between Tile Address Mapping mode and for TileYs and TileYf.

Value	Name	Description
0h	Tile Address Mapping Mode	This field should always be programmed to 0h. Tiling mode is the standard tile layout for 3D.
1h	Tile Address Mapping Mode (for Standard Tiling)	This field should NOT be programmed to 1h as the Tiling Address Mapping mode is not supported by SW.

Programming Notes
Tile Address Mapping Mode must be set to Tile Address Mapping Mode when surface type is SURFTYPE_3D.

19 Standard Tiling Mode Extensions

It changes in the MIP Tail Packing. When enabled (programmed to 1h), MIP Tail packing for Volumetric and 1D are changed as defined in the Surface Layout and Tiling section.
This bit controls enabling of some Standard Tiling extensions:

Value	Name	Description
0h	Disable **[Default]**	When programmed to 0h, the extensions to support Standard Tiling are disabled. Behavior reverts to Miptail packing.
1h	Enable	When programmed to 1h, the changes to support Standard Tiling Extensions are enabled. See the Surface Layout and Tiling section for details.

18 Reserved

Format:	MBZ

17:0 Surface Pitch

Format:	U18-1

Description
Surface Pitch Range: • For surfaces of type SURFTYPE_BUFFER: [0,262143] -> [1B, 256KB] • For other linear surfaces: [0, 262143] -> [1B, 256KB] • For X-tiled surface: [511, 262143] -> [512B, 256KB] = [1 tile, 512 tiles]

- For Y-tiled surfaces: [127, 262143]->[128B, 256KB] = [1 tile, 2048 tiles]
- For W-tiled surfaces: [127, 262143]->[128B, 256KB] = [1 tile, 2048 tiles]
- For TileYF and TileYS surfaces, the range is dependent on the Cu parameter (refer to *Memory Data Formats* section for the definition of the Cu parameter depending on the case). The range in bytes is [2^{Cu}-1,262143] -> [(2^{Cu})B,256KB] = [1 tile, 256KB/(2^{Cu}) tiles]

This field specifies the surface pitch in (#Bytes - 1).
For surfaces of type SURFTYPE_BUFFER, this field indicates the size of the structure.

Programming Notes

- For linear *render target* surfaces and surfaces accessed with the typed data port messages, the pitch must be a multiple of the element size for non-YUV surface formats. Pitch must be a multiple of 2 * element size for YUV surface formats.
- For untyped data port messages, which are only supported with **Surface Type** SURFTYPE_BUFFER, the pitch must match the number of bytes implied by Surface Format..
- For linear surfaces with **Surface Type** of SURFTYPE_BUFFER and **Surface Format** RAW, the pitch must be 1 byte.
- For other linear surfaces, the pitch can be any multiple of bytes.
- For tiled surfaces, the pitch must be a multiple of the tile width.

- The width of a tile depends on the surface format if Tiled Resource Enable is enabled. Refer to the Tiled Resource Enable field to determine which sub-mode applies to the surface format in use, and determine the Cu parameter from the Surface Layout section. The tile width is equal to 2^Cu bytes.
- For surfaces of type SURFTYPE_1D, this field is ignored.

The following table indicates the maximum byte width, frame width, and pitch size allowed when memory compression is on.

Tiling Mode	Pixel Format	Max Frame Width (bytes)	Max Frame Width (pixels)	Max Pitch (bytes)
Legacy 4K	8bpp	16k	16k	16k + 127
	16bpp	16k	8k	16k + 127
	32bpp	16k	4k	16k + 127
	64bpp	16k	2k	16k + 127
	128bpp	16k	1k	16k + 127
TileYF	8bpp	8k	8k	8k + 63
	16bpp	16k	8k	16k + 127
	32bpp	16k	4k	16k + 127
	64bpp	16k	2k	16k + 255
	128bpp	16k	1k	16k + 255
TileYS	8bpp	16k	16k	16k + 255

	16bpp	16k	8k	16k + 511
	32bpp	16k	4k	16k + 511
	64bpp	16k	2k	16k + 1023
	128bpp	16k	1k	16k + 1023

4 — 31 — Decompress In L3

Value	Name	Description
0h	Disable **[Default]**	When this field is set to 0h, the associated compressible surface, when accessed by sampler and data-port, can be compressed in L3. If the surface is not compressible, this bit field is ignored.
1h	Enable	When this field is set to 1h, the associated compressible surface, when accessed by sampler and data-port, will be uncompressed in L3. If the surface is not compressible, this bit field is ignored.

30:29 — Render Target And Sample Unorm Rotation

Description
For Render Target Surfaces: This field specifies the rotation of this render target surface when being written to memory.
For sample_unorm Messages: This field specifies the rotation of the data returned by sampler for sample_unorm message.
For Other Surfaces:This field is ignored.

Value	Name	Description
0h	0DEG	No rotation (0 degrees)
1h	90DEG	Rotate by 90 degrees
2h	180DEG	Rotate by 180 degrees [for sample_unorm message]
3h	270DEG	Rotate by 270 degrees

Programming Notes

Programming Notes for Render Target Surfaces only

- Rotation is not supported for render targets of any type other than simple, non-mip-mapped, non-array 2D surfaces. The surface must be using tiled with X major.
- **Width** and **Height** fields apply to the dimensions of the surface before rotation.
- For 90 and 270 degree rotated surfaces, the **Height** (rather than the **Width**) must be less than or equal to the **Surface Pitch** (specified in bytes).
- For 90 and 270 degree rotated surfaces, the actual **Height** and **Width** of the surface in pixels (not the field value which is decremented) must both be even.

Rotation is supported only for surfaces with the following surface formats: B5G6R5_UNORM, B5G6R5_UNORM_SRGB, R8G8B8A8_UNORM, R8G8B8A8_UNORM_SRGB, B8G8R8[A|X]8_UNORM, B8G8R8[A|X]8_UNORM_SRGB, B10G10R10[A|X]2_UNORM, R10G10B10A2_UNORM,

		R16G16B16A16_FLOAT, R16G16B16X16_FLOAT
	28:18	**Minimum Array Element** Format: U11 Range 1D/2D/cube surfaces: [0,2047] 3D surfaces: [0,2047] **For Sampling Engine, Render Target, and Typed 1D, 2D and 3D Surfaces:** This field indicates the minimum array element that can be accessed as part of this surface. This field is added to the delivered array index before it is used to address the surface. For sampling 3D surfaces, this fieldis an offset in "R" slices into the LOD. **For Sampling Engine Cube Surfaces:** This field indicates the minimum array element in units of "cube faces" in the underlying 2D surface array that can be accessed as part of this surface (the cube array index is multiplied by 6 to compute this value, although Minimum Array Element is not restricted to only multiples of 6). This field is added to the delivered (array index)*6 before it is used to address the surface. **For all other surface types:** This field should be zero.
	17:7	**Render Target View Extent** Format: U11-1 Range [0,2047] to indicate extent of [1,2048] **For Render Target and Typed Dataport 3D Surfaces:** This field indicates the extent of the accessible 'R' coordinates minus 1 on the LOD currently being rendered to. **For Render Target and Typed Dataport 1D and 2D Surfaces:** This field must be set to the same value as the Depth field. **For Other Surfaces:** This field is ignored.
	6	**Multisampled Surface Storage Format** This field indicates the storage format of the multisampled surface. **Value** / **Name** / **Description** 0h / MSS / Multsampled surface was/is rendered as a render target 1h / DEPTH_STENCIL / Multisampled surface was rendered as a depth or stencil buffer **Programming Notes** • All multisampled render target surfaces must have this field set to MSFMT_MSS • IF this field is MSFMT_DEPTH_STENCIL, the only sampling engine messages allowed are "ld2dms", "resinfo", and "sampleinfo". • This field is ignored if **Number of Multisamples** is MULTISAMPLECOUNT_1
	5:3	**Number of Multisamples** This field indicates the number of multisamples on the surface. **Value** / **Name** 0h / MULTISAMPLECOUNT_1

		1h: MULTISAMPLECOUNT_2 2h: MULTISAMPLECOUNT_4 3h: MULTISAMPLECOUNT_8 4h: MULTISAMPLECOUNT_16 5h-7h: Reserved **Programming Notes** If this field is any value other than MULTISAMPLECOUNT_1, the **Surface Type** must be SURFTYPE_2DThis field must be set to MULTISAMPLECOUNT_1 unless the surface is a Sampling Engine surface or Render Target surface.
	2:0	**Multisample Position Palette Index** This field indicates the index into the sample position palette that the multisampled surface is using. This field is only used as a return value for the sampleinfo message, and is otherwise not used by hardware. **Value** [0,7] — **Name**
5	31:25	**X Offset** Format: U9[8:2] This field specifies the horizontal offset in pixels from the **Surface Base Address** to the start (origin) of the surface. This field effectively loosens the alignment restrictions on the origin of tiled surfaces. Previously, tiled surface origin was (by definition) located at the base address, and thus needed to satisfy the 4KB base address alignment restriction. Now the origin can be specified at a finer (4-wide x 4-high pixel) resolution. Format: `PixelOffset[8:2]` **Value** [0,127] — **Name** — **Description**: This value is multiplied by 4 to determine X Offset in pixels. **Programming Notes** • For linear surfaces, this field must be zero. • For surfaces accessed with the *Data Port Media Block Read/Write* message, the pixel size is assumed to be 32 bits in width. • For surfaces accessed with the **Data Port Transpose Read message**, the pixel size is assumed to be 32 bits in width. • For **Surface Format** with other than 8, 16, 32, 64, or 128 bits per pixel, this field must be zero. • If **Render Target Rotation** is set to other than RTROTATE_0DEG, this field must be zero.

- If **Surface Type** not SURFTYPE_2D, this field must be zero.
- If **MIP Count** is not zero, this field must be zero.
- If **Number of Multisamples** is not MULTISAMPLECOUNT_1, this field must be zero.
- If **Surface Array** is enabled, this field must be zero.
- If **Auxiliary Surface Mode** is not AUX_NONE, this field must be zero.
- If **Surface Vertical Alignment** is VALIGN_8, this field must be a multiple of 8.
- For **Surface Format** with 8 bits per element, this field must be a multiple of 16.
- For **Surface Format** with 16 bits per element, this field must be a multiple of 8.

If Surface Format is PLANAR_420_16 or PLANAR_420_8 this field must be zero

- If **Tiled Resource Mode** is not TRMODE_NONE, this field must be zero.

24 **Reserved**

Access:	RO
Format:	MBZ

23:21 **Y Offset**

Format:	U5[4:2]

This field specifies the vertical offset in rows from the **Surface Base Address** to the start of the surface.(See additional description in the **X Offset** field.)

Format:

```
RowOffset[4:2]
```

Value	Name	Description
[0,7]		This value is multiplied by 4 to determine the actual Y Offset in Rows.

Programming Notes

- For linear surfaces, this field must be zero.
- For render targets in which the **Render Target Array Index** is not zero, this field must be zero.
- For **Surface Format** with other than 8, 16, 32, 64, or 128 bits per pixel, this field must be zero.
- If **Render Target Rotation** is set to other than RTROTATE_0DEG, this field must be zero.
- If **Surface Type** not SURFTYPE_2D, this field must be zero.
- If **MIP Count** is not zero, this field must be zero.
- If **Number of Multisamples** is not MULTISAMPLECOUNT_1, this field must be zero.
- If **Surface Array** is enabled, this field must be zero.
- If **Auxiliary Surface Mode** is not AUX_NONE, this field must be zero.

If Surface Format is PLANAR_420_16 or PLANAR_420_8 this field must be zero

- If **Tiled Resource Mode** is not TRMODE_NONE, this field must be zero.

This field must be zero if Surface Format is Planar and the U and V planes are half-pitch (e.g. YV12 format).

If VERTICAL STRIDE field is 1h, then the Y Offset must be a multiple of 8. Therefore, the lsb of this field must be 0.

20 **EWA Disable For Cube**

Format:	Disable

Specifies if EWA mode for LOD quality improvement needs to be disabled for cube maps.

Value	Name	Description
0h	Enable **[Default]**	EWA is enabled for cube maps
1h	Disable	EWA is disabled for cube maps

Programming Notes

This field indicates if EWA mode for LOD quality improvement needs to be disabled for cube maps.By default EWA would be on for cube maps hence this field must be 0. If there is any spec violation seen with EWA on cube maps then this field must be set to 1 to disable EWA for cubes.

19:18 **Tiled Resource Mode**

For Sampling Engine, Render Target, and Typed/Untyped Surfaces: This field specifies the tiled resource mode.**For other surfaces:**This field is ignored.

Value	Name	Description	Exists If
0h	NONE	No tiled resource	
1h	4KB	4KB tiled resources	[SurfaceType] == 'SURFTYPE_1D'
2h	64KB	64KB tiled resources	[SurfaceType] == 'SURFTYPE_1D'
1h	TILEYF	4KB tiled resources	[SurfaceType] != 'SURFTYPE_1D'
2h	TILEYS	64KB tiled resources	[SurfaceType] != 'SURFTYPE_1D'
3h	Reserved		

Programming Notes

If **Tile Mode** is not set to TILEMODE_YMAJOR, this field must be set to TRMODE_NONE, unless the Surface Type is SURFTYPE_1D.

If this field is not set to TRMODE_NONE, the **Surface Format** must be one with 8, 16, 32, 64, or 128 bits per element, or one of the compressed texture modes (BC*, ETC*, EAC*, ASTC*). Additionally, YCRCB* formats are supported and treated as 16 bits per element, and the PLANAR_420_8 format is support and treated as 8 bits per element on the Y plane and 16 bits per element on the UV plane (if **Separate UV Plane Enable** is disabled) or 8 bits per element on the U and V planes (if **Separate UV Plane Enable** is enabled).

If this field is set to TRMODE_NONE, the surface cannot contain any null pages unless **Surface Type** is BUFFER. A BUFFER surface with null pages must have **Surface Base Address** and **Surface**

Pitch set to an integer multiple of the element size, and **Surface Format** must be one with 8, 16, 32, 64, or 128 bits per element.

If **Surface Format** is PLANAR, the surface cannot contain any null pages.

If RENDER_SURFACE_STATE.Surface Type = 3D, and RENDER_SURFACES_STATE.Auxiliary Surface Mode != AUX_NONE , this field must be set to TILEYS

17:15 **Reserved**

Access:	RO
Format:	MBZ

14 **Coherency Type**

Specifies the type of coherency maintained for this surface.

Value	Name	Description
0h	GPU coherent	Surface memory is kept coherent with GPU threads using GPU read/write ordering rules. Surface memory is backed by system memory but is not kept coherent with CPU (LLC).
1h	IA coherent	Surface memory is kept coherent with CPU (LLC).

Programming Notes

This field may optionally be 1 (IA coherent) for messages sent to SFID_DP_DC0 or SFID_DP_DC1 or SFID_DP_DC2. This field is typically set to 0 (GPU coherent) if the context is operating in a non-SVM legacy mode (for example, Ring Buffer or a Execlist using 32-bit Virtual Address Legacy Context PPGTT32).

13:12 **Reserved**

Access:	RO
Format:	MBZ

11:8 **Mip Tail Start LOD**

Format:	U4

Description

For Sampling Engine, Render Target, and Typed Surfaces: This field indicates which LOD is the first one in the MIP tail if **Tiled Resource Mode** is not TRMODE_NONE. The MIP tail has a different layout than the rest of the surface. Refer to the *Memory Data Formats* section for more details. **For other surfaces:** This field is ignored.

Programming Notes

If **Tiled Resource Mode** is not TRMODE_NONE, this field must be set to ensure that mips within the mip tail do not overlap given the storage algorithms given in the Memory Data Formats section.

If **Tiled Resource Mode** is not TRMODE_NONE, to disable the Mip Tail this field must be set to a mip that larger than those present in the surface (i.e. 15). This is recommended for non-mip-

mapped surfaces.

The following table indicates the *maximum* size of the mip that is set to be the Mip Tail Start LOD for various cases:

Surface Type	Tiling Mode	#MS	Bits Per Element				
			8	16	32	64	128
1D	64KB	1	16384	8192	4096	2048	1024
	4KB	1	1024	512	256	128	64
2D/ CUBE	TIleYS	1	128x256	128x128	64x128	64x64	32x64
		2	128x128	128x64	64x64	64x32	32x32
		4	64x128	64x64	32x64	32x32	16x32
		8	64x64	64x32	32x32	32x16	16x16
		16	32x64	32x32	16x32	16x16	8x16
	TIleYF	1x	32x64	32x32	16x32	16x16	8x16
		2x	16x32	16x16	8x16	8x8	4x8
		4x	16x16	16x8	8x8	8x4	4x4
		8x	16x16	16x8	8x8	8x4	4x4
		16x	8x4	8x4	4x4	2x4	1x4
3D	TIleYS	1	32x32x32	16x32x32	16x32x16	16x16x16	8x16x16
	TileYF	1	16x8x16	8x8x16	8x8x8	8x4x8	4x4x8

7:4 **Surface Min LOD**

Format:	U4

For Sampling Engine and Typed Surfaces: This field indicates the most detailed LOD that can be accessed as part of this surface. This field is added to the delivered LOD (*sample_l*, *ld*, or *resinfo* message types) before it is used to address the surface.
For Other Surfaces: This field is ignored.

3:0 **MIP Count / LOD**

Format:	**Sampling Engine and Typed Surfaces:** U4 in (LOD units - 1) **Render Target Surfaces:** U4 in LOD units
Range	**Sampling Engine and Typed Surfaces:** [0,14] representing [1,15] MIP levels **Render Target Surfaces:** [0,14] representing LOD **Other Surfaces:** [0]

For Sampling Engine and Typed Surfaces:
This field indicates the number of MIP levels allowed to be accessed starting at **Surface Min LOD**, which must be less than or equal to the number of MIP levels actually stored in memory for this surface. For sample* messages, the mip map access is clamped to be between the mipmap specified by the integer bits of the Min LOD and the ceiling of the value specified here. For ld* messages, out-of-bounds behavior results for LODs outside of the range specified in this field.
For Render Target Surfaces: This field defines the MIP level that is currently being rendered into. This is the absolute MIP level on the surface and is not relative to the **Surface Min LOD** field, which is ignored for render target surfaces.

For Other Surfaces: This field is reserved : MBZ

Programming Notes
The **LOD** of a render target must be the same as the **LOD** of the other render target(s) and of the depth buffer (defined in 3DSTATE_DEPTH_BUFFER). For render targets with YUV surface formats, the **LOD** must be zero. For sampling engine surfaces with PLANAR* surface format, **MIP Count** must be zero.

6 | 31 **Separate UV Plane Enable**

Exists If:	([Surface Format] == 'PLANAR')
Format:	Enable

If enabled, this field indicates that the U and V are present as separate planes. If disabled, the UV data is interleaved on a single plane.

Programming Notes
See the section "Planar Memory Organization" for a description of how the size and location of the chroma planes (U and V) are calculated.

30 **Half Pitch for Chroma**

Exists If:	([Surface Format] == 'PLANAR')

This bit enables support for half-pitch chroma planes for Planar YUV surfaces. It is ignored for Non-Planar surfaces. For planar surfaces it allows the chroma planes to be one-half the width of a the Y (Luma) plane.
For example, should be set to 0h for NV12 surfaces.
Must be set to 1h for YV12 surfaces.

Value	Name	Description
0h	Disable **[Default]**	Setting this bit to 0h (default) causes Chroma planes to be treated as full width (same as Y plane).
1h	Enable	Setting this bit to 1h causes Chroma planes (U and V) to be treated as half the width of the Luma (Y) plane.

30:16 **Auxiliary Surface QPitch**

Exists If:	([Surface Format] != 'PLANAR')
Format:	U17[16:2]

This field specifies the distance in rows between array slices on the auxiliary surface.

Value	Name	Description
[0h,7FFFh]		1 is added to the value and it is multiplied by 4 to determine the actual QPitch in rows

Programming Notes
This field must be set to an integer multiple of the **Surface Vertical Alignment**
Software must ensure that this field is set to a value sufficiently large such that the array slices in the auxiliary surface do not overlap. Refer to the Memory Data Formats section for information on how surfaces are stored in memory.
Auxiliary surface parameters in the Surface State is only meant for Multisampling Control Surface.

	This field should be programmed when Multisampling is enabled. In all other cases HW will ignore this value.
29:16	**X Offset for U or UV Plane** Exists If: ([Surface Format] == 'PLANAR') Format: U14 This field specifies the horizontal offset in pixels from the **Surface Base Address** to the start (origin) of the U plane or interleaved UV plane, depending on the setting of **Separate UV Plane Enable**. **Programming Notes** This field must be a multiple of 4 (bits 1:0 MBZ). If **Tiled Resource Mode** is enabled, this field must be a multiple of the tile width in pixels. **Auxiliary Surface Mode** is forced to AUX_NONE.
15	**YUV Interpolation Enable** Format: Enable **Description** This bit controls whether a Non-Planar YUV4:2:2 and Planar YUV4:2:0 surface use interpolated or replicated U and V channels for input to the Sampler filter. Programming to 1h causes interpolation of U and V channels. In this case the chrominance for odd pixels is computed by an interpolation between adjacent even pixels. Programming to 0h causes the chrominance to be copied from the pixel to the left. **Value** / **Name** / **Description** 0h / Disable **[Default]** / Programming to 0h causes the sampler to replicate U and V channels. This will lead to lower quality in certain cases where the YUV surface is being filtered (e.g. linear). 1h / Enable / Programming to 1h causes the sampler to interpolate the U and V channels between the horizontally neighboring pixels. This will improve image quality if the surface is being filtered.
14	**Reserved** Access: RO Format: MBZ
13:0	**Y Offset for U or UV Plane** Exists If: ([Surface Format] == 'PLANAR') Format: U14 This field specifies the vertical offset in rows from the **Surface Base Address** to the start (origin) of the U plane or interleaved UV plane, depending on the setting of **Separate UV Plane Enable**. **Programming Notes** For surfaces where **Surface Format** = PLANAR* and **Separate UV Plane** is Enabled, the Y Offset must be programmed in multiples of **half-rows**. For example, for a surface where Y is physically

followed by U and then V in memory, the Y Offset to U plane would be (2*Y-Height). For all other PLANAR YUV formats this is programmed in multiples of full rows.

For all format besides PLANAR_420_* This field must be a multiple of 4 (bits 1:0 MBZ). For formats PLANAR_420_* with separate chroma planes (e.g. YV12) this field must be multiple of 4 if U plane is the first chroma plane after the Y (luma) plane. It can be a multiple of 2 if it is the second chroma plane in memory. For formats PLANAR_420_* with interleaved chroma planes (e.g. NV12) this field can be multiple of 2.

If **Tiled Resource Mode** is enabled, this field must be a multiple of the tile height in rows.

Auxiliary Surface Mode is forced to AUX_NONE.

Workaround
For formats PLANAR_420_* when this field is not a multiple of 4 the Out-of-Bounds Supression check must be disabled to avoid false out of bound detection.

11:3 **Auxiliary Surface Pitch**

Exists If:	([Surface Format] != 'PLANAR')
Format:	U9-1

This field specifies the Auxiliary surface pitch in (#Tiles - 1).

Value	Name	Description
[0, 511]		-> [1 tile, 512 tiles]

2:0 **Auxiliary Surface Mode**

Exists If:	([Surface Format] != 'PLANAR')
Format:	U3

Specifies what type of surface the Auxiliary surface is. The Auxiliary surface has its own base address and pitch, but otherwise shares or overrides other fields set for the primary surface, detailed in the programming notes below.

Value	Name	Description
0h	AUX_NONE	No Auxiliary surface is used
1h	AUX_CCS_D	To be programmed only for Procedural Texture (PT). This mode is not valid for Render Targets.
2h	AUX_APPEND	The Auxiliary surface is an append buffer
3h	Reserved	
4h	AUX_MCS_LCE	[] Enables lossless compression on the top of MSAA compression for RTs with Number of Multisamples not equal to MULTISAMPLECOUNT_1. CCS is tiling format is linear. MCS tiling format is always tiledY.
5h	AUX_CCS_E	If Number of multisamples = 1, programming this value means lossless compression is enabled for that surface.Auxiliary surface is a CCS with linear tiling. AUX_CCS_E cannot be programmed for Procedural Texture surfaces

		If Number of multisamples > 1, programming this value means MSAA compression is enabled for that surface. Auxillary surafce is MSC with tile y.
6h	Reserved	
7h	Reserved	

Programming Notes
The CCS surface shares **Depth, Surface Type, Surface Array, Surface Min LOD, MIP Count / LOD, Surface Object Control State, Resource Min LOD**, and **Minimum Array Element** with the primary surface. **X & Y Offset** are set to zero for the purpose of accessing the Auxiliary surface. CCS Height and Width are scaled. CCS is always linear. MCS is always tile-y.
CCS surfaces are linear with **Tiled Resource Mode** of TRMODE_NONE regardless of the tile mode of the primary surface, and **Mip Tail Start LOD** is ignored for these surfaces.
The CCS Auxiliary surface for **Number of Multisamples** > 1 uses **Surface Horizontal Alignment** of 16 and **Surface Vertical Alignment** of 4 regardless of the primary surface's values for these fields.
If **Number of Multisamples** is MULTISAMPLECOUNT_1, AUX_CCS_E setting is only allowed if **Surface Format** is supported for Render Target Compression. This setting enables render target compression.
If **Number of Multisamples** is MULTISAMPLECOUNT_1, value of AUX_CCS_E is only allowed value if **Surface Format** is supported for Render Target Compression. If **Number of Multisamples** is other than MULTISAMPLECOUNT_1, value of AUX_MCS_LCE is only allowed if **Surface Format** is supported for Render Target Compression.
If **Number of Multisamples** is MULTISAMPLECOUNT_1, and if **Tiled Resource Mode** is NOT TRMODE_NONE, then, if CCS tile is NULL, Render Target Tiles represented by that CCS tile are assumed to be NULL by HW.
When Number of Multisamples > 1, and when MSAA compression is disabled, there is no way to enable lossless compression.
SW can enable lossless compression for MSRTs, by setting this the field to AUX_MCS_LCE for Render Targets with **Number of Multisamples** *not* equal to MULTISAMPLECOUNT_1. This value can not be progammed when **Number of Multisamples** equal to MULTISAMPLECOUNT_1.
Programming AUX_CCS_D is not allowed for all surfaces EXCEPT Procedural Texture (PT).
For an 8 bpp surface with NUM_MULTISAMPLES = 1, Surface Width not multiple of 64 pixels and more than 1 mip level in the view, Fast Clear is not supported when AUX_CCS_E is set in this field.
For Surface Type = SURFACETYPE_3D, Surface Format = 64bpp or 128bpp and Tiling Format = TiledYs, AUX_MODE must not be programmed to enable lossless compression or attaching CCS i.e. AUX_MODE must be programmed to AUX_NONE. except when Procedural Texture Enable bit is set. For Procedural Textures, Surface format TYS, 3D, 64/128 bpp is supported

ValidValue	Valu	Description	Msaa	CCS	Color	MCS	Msaa

	e		Mode	Aux Present	Compression Enabled	Aux Present	Compression Enabled
AUX_NONE	0	No Auxiliary surface is used					
AUX_CCS_D	1	To be programmed only for Procedural Texture (PT). This mode is not valid for Render Targets.	NA	True	False	False	False
AUX_APPEND	2	The Auxiliary surface is an append buffer	NA	-	-	-	-
RESERVED	3	-	-	-	-	-	-
AUX_MCS_LCE	4	Enables lossless compresion on the top of MSAA compression for RTs with **Number of Multisamples** *not* equal to MULTISAMPLECOUNT_1.	nX	True	True	True	True
AUX_CCS_E	5	If Number of multisamples = 1, programming this value means lossless compression is enabled for that surface. Auxiliary surface is a CCS with linear tiling.	1X	True	True	False	False
AUX_CCS_E	5	If Number of multisamples > 1, programming this value means MSAA compression is enabled for that surface. Auxillary surface is MSC with tiley.	nX	False	False	True	True

*nX = 2x, 4x, 8x, 16x

7	31	**Memory Compression Mode** Default Value: 0 Horizontal

Description
MBZ: Only Horizontal mode is supported.

30 **Memory Compression Enable**

Format:	Enable

This surface may contain compressed or compressible pixels. Memory compression will be attempted for writes to this surface. Reads from this surface will check for compressed data.

Programming Notes
Please refer to vol1a Memory Data Formats chapter section Media Memory Compression for more details, including format restrictions.

29:28 **Reserved**

Access:	RO
Format:	MBZ

27:25 **Shader Channel Select Red**

Format:	Shader Channel Select

Specifies which surface channel is read or written in the Red shader channel.

Programming Notes
For reads, the Shader Channel Select defines which surface channel is filled into the shader channel. If the Shader Channel Select is SCS_ZERO or SCS_ONE, then the shader channel takes on that fixed value, regardless of the surface channel's presence or value. Otherwise, if the Shader Channel Select is SCS_RED, SCS_GREEN, SCS_BLUE or SCS_ALPHA, then that surface channel is read into this shader channel. If the same Surface Channel Select appears for multiple shader channels, then that surface channel is replicated in each of those shader channels. If a Shader Channel Select is specified that is not present in the surface format, the value filled into the shader channel is undefined. Programs should always use SCS_ZERO or SCS_ONE for missing surface channels.
For writes, the Shader Channel Select defines which surface channel is written from the shader channel. If the Shader Channel Select is not present in the surface format or is SCS_ZERO or SCS_ONE, then the shader channel is not written to the surface. Otherwise, if the Shader Channel Select is SCS_RED, SCS_GREEN, SCS_BLUE or SCS_ALPHA, the shader channel is written to that surface channel. If more than one Shader Channel Select is set to the same surface channel, only the first shader channel in RGBA order will be written the surface channel, and subsequent shader channels with that Shader Channel Select are not written to the surface. If any surface channels are present but not specified by the Shader Channel Select, those surface channels are undefined (and might be written to zero). Programs should always specify the Shader Channel Select to cover all present surface channels.
Each Shader Channel Select must be set to the same surface channel (R = SCS_RED, G = SCS_GREEN, B = SCS_BLUE, A = SCS_ALPHA) if the surface is accessed via the sampler's sample_unorm* or sample_8x8 messages.
The Shader Channel Select fields do not affect the following sampling engine message types: resinfo, sampleinfo, LOD, and ld_mcs. These messages behave as if each Shader Channel Select is set to the same color surface channel.

For the sampling engine *gather4** messages, the Gather4 Source Channel Select field in the message header defines which channel's Shader Channel Select is used to select the surface channel to be sampled. Other Shader Channel Select fields are ignored.

For the sampling engine *sample*_c* and *gather4*_c* messages, the compare operation always occurs on the red channel from the surface regardless of the setting of the Shader Channel Select fields.

Restriction

For all Render Target and HDC messages, the Surface Channel Select is restricted to a subset of combinations that ensures, when reading the surface and then writing that value under the same Surface Channel Select, the identical value is put in memory. Any combination of Surface Channel Selects and channel surface format that do not guarantee this isomorphic property are not supported and may produce undefined results.
The isomorphism property is guaranteed by these restrictions:

- When the channel (R, G, B, A) is present in the surface format, then that channel (SCS_RED, SCS_GREEN, SCS_BLUE, SCS_ALPHA) is present exactly once in the 4 Surface Channel Selects (Red, Green, Blue, Alpha).
- For channels not present in the surface format, the corresponding Surface Channel Select is either SCS_ZERO or SCS_ONE.
- The Surface Channel Select for present channels either preserves or swaps the order of the surface channels. This guarantees write_swizzled(read_swizzled(x)) = x.

Render Target messages do not support swapping of colors with alpha. The Red, Green, or Blue Shader Channel Selects do not support SCS_ALPHA. The Shader Channel Select Alpha does not support SCS_RED, SCS_GREEN, or SCS_BLUE.

24:22 **Shader Channel Select Green**

Format:	Shader Channel Select

See **Shader Channel Select Red** for details.

21:19 **Shader Channel Select Blue**

Format:	Shader Channel Select

See **Shader Channel Select Red** for details.

18:16 **Shader Channel Select Alpha**

Format:	Shader Channel Select

See **Shader Channel Select Red** for details.

Programming Notes

For Render Target, this field MUST be programmed to value = SCS_ALPHA.

15:14 **Reserved**

Access:	RO
Format:	MBZ

	13:12	**Reserved** Access: RO Format: MBZ
	11:0	**Resource Min LOD** Format: U4.8 **For Sampling Engine Surfaces:**This field indicates the most detailed LOD that is present in the resource underlying the surface. Refer to the "LOD Computation Pseudocode" section for the use of this field. **For Other Surfaces:**This field is ignored. Value: [0,14] — Name: **Programming Notes** This field must be zero if the **ChromaKey Enable** is enabled in the associated sampler.
8..9	63:0	**Surface Base Address** Format: GA63_0 Specifies the byte-aligned base address of the surface. **Programming Notes** • For SURFTYPE_BUFFER render targets, this field specifies the base address of first element of the surface. The surface is interpreted as a simple array of that single element type. The address must be naturally-aligned to the element size (e.g., a buffer containing R32G32B32A32_FLOAT elements must be 16-byte aligned). • For SURFTYPE_BUFFER non-rendertarget surfaces, this field specifies the base address of the first element of the surface, computed in software by adding the surface base address to the byte offset of the element in the buffer. The base address must be aligned to element size. • Linear depth buffer surface base addresses must be 64-byte aligned. Note that while render targets (color) can be SURFTYPE_BUFFER, depth buffers cannot. • Mipmapped surfaces are stored in a "monolithic" (fixed) format, and only require a single address for the base MIP. All other MIPs are positioned relative to the base MIP. • The Base Address for linear (non-tiled) render target surfaces and surfaces accessed with the typed surface read/write data port messages must be element-size aligned for Non-YUV surface formats, or a multiple of 2 element-sizes for YUV surface formats. • Other linear (non-tiled) surfaces have no alignment requirements (byte alignment is sufficient). • For tiled surfaces, the actual start of the surface can be offset from the Surface Base Address by the X Offset and Y Offset fields. Tiles are inherently page-aligned (4K or 64K). • Certain message types used to access surfaces have more stringent alignment requirements. Please refer to the specific data-port message documentation for additional restrictions.

		Tiled surface base addresses must be 4KB-aligned. Note that only the offsets from Surface Base Address are tiled, Surface Base Address itself is not transformed using the tiling algorithm.
10..11	63:12	**Auxiliary Surface Base Address** Exists If: ([Surface Format] != 'PLANAR') AND [Memory Compression Enable] == 0 Format: GraphicsAddress[63:12] Specifies the 4kbyte-aligned base address of the Auxiliary surface associated with the primary surface specified in other SURFACE_STATE fields.
	61:48	**X Offset for V Plane** Exists If: ([Surface Format] == 'PLANAR') Format: U14 This field specifies the horizontal offset in pixels from the **Surface Base Address** to the start (origin) of the V plane. **Programming Notes** This field must be a multiple of 4 (bits 1:0 MBZ). If **Tiled Resource Mode** is enabled, this field must be a multiple of the tile width in pixels. This field is ignored if **Separate UV Plane Enable** is disabled.
	45:32	**Y Offset for V Plane** Exists If: ([Surface Format] == 'PLANAR') Format: U14 This field specifies the vertical offset in rows from the **Surface Base Address** to the start (origin) of the V plane. **Programming Notes** For surfaces where **Surface Format** = PLANAR* and **Separate UV Plane** is Enabled, the Y Offset must be programmed in multiples of **half-rows**. For example, for a surface where Y is physically followed by U and then V in memory, the Y Offset to V plane would be (2*Y-Height+ U-Height). For all other PLANAR YUV formats this is programmed in multiples of full rows (e.g Y-Height + U-Height). For all format besides PLANAR_420_* This field must be a multiple of 4 (bits 1:0 MBZ). For formats PLANAR_420_* this field must be multiple of 4 if U plane is the first chroma plane after the Y (luma) plane. It can be a multiple of 2 if it is the second chroma plane. For formats PLANAR_420_* when this field is not a multiple of 4 the Out-of-Bounds Suppression check must be disabled to avoid false out of bound detection. If **Tiled Resource Mode** is enabled, this field must be a multiple of the tile height in rows. This field is ignored if **Separate UV Plane Enable** is disabled.
	11	**Procedural Texture** This bit, when set, indicates that the associated surface is a procedural texture which is used for AMFS. This bit can be ENABLED for the following surface types: SURFTYPE_2D arrayed / non-arrayed, SURFTYPE_3D non-arrayed, SURFTYPE_CUBE arrayed/ non arrayed, and surftype = NULL. This bit can be set for the pixel formats that are supported has typed UAVs as per the DX spec. Therefore,

writes from only HDC are supported to Procedural Textures.
This bit cannot be ENABLED for the following surface types: SURFTYPE_3D arrayed, SURFTYPE_BUFFER

Programming Notes
This bit cannot be set when surface walk (tiling mode) is legacy Y This bit cannot be set when Tiled Resource Mode = TileYS and LOD >= MIP tail LOD

10 **Clear Value Address Enable**

Format:	Enable

This field enables HW Managed Clear Value Layout for the Surface State. If this bit is enabled, Clear Value Address is present instead of explicit clear values.

Value	Name	Description
0h	Disable **[Default]**	Clear values are present in the surface state explicitly.
1h	Enable	Clear value Address is present instead of explicit clear values.

Programming Notes
Clear values can only be enabled for sampled surface formats which are supported for clear in the Pixel Data Port. See Render Target Surfaces section of **Pixel Data Port** for a list of surface types supported . This bit has to be programmed to 1 if clear buffer is attached to the surface or if AUX_MODE is AUX_CCS_E. No support for explicit clear values. Only hw managed clear values are supported.

9:5 **Quilt Height**

Format:	U5

This field specifies the height of a quilted texture in units of quilt slices. Refer to the section on Quilted Textures for more details.

Value	Name	Description
[0,31]		representing height of quilt - 1 (y/v dimension)

Programming Notes

Programming Notes
• Only power-of-2 **Quilt Height** and **Quilt Width** values are allowed: (1,2,4,8,16,32) mapping to (0,1,3,7,15,31) values in the fields.
• A surface is defined as a quilted texture if either **Quilt Height** or **Quilt Width** is nonzero (actual field value, not the incremented value).
• A quilted texture
  • is only supported by the sampling engine (other shared functions will ignore the **Quilt Width** and **Quilt Height** field, behaving as if they are set to zero).
  • must have a **Surface Type** of SURFTYPE_2D.
  • must have **Number of Multisamples** set to NUMSAMPLES_1.
  • must have **Vertical Line Stride** set to 0.
  • must have **Auxiliary Surface Mode** set to AUX_NONE.

- **Depth** indicates the array dimension of the quilted texture if **Surface Array** is enabled. The valid range of **Depth** is [0, 2048 / (QuiltWidth * QuiltHeight) - 1], i.e. the total number of underlying array slices including quilt slices cannot exceed 2048.
- cannot be accessed with any ld* message type or using a sampler with the **Non-Normalized Coordinate Enable** field enabled.

Quilted surfaces are not supported and this field must be programmed to 0h

4:0 **Quilt Width**

Format:	U5

This field specifies the width of a quilted texture in units of quilt slices. Refer to the section on Quilted Textures for more details.

Value	Name	Description
[0,31]		representing width of quilt - 1 (x/u dimension)

Programming Notes

Quilted surfaces are not supported and this field must be programmed to 0h

12 31:6 **Clear Address Low**

Format:	GraphicsAddress[31:6]

Description

For Sampling Engine Surfaces and Render Targets: Specifies the lower bits of Graphics Address where clear value is stored in
The memory layout of the clear color pointed to by this address is a value stored in the lower-order bytes of a 64-byte cache-line.
The clear color will be formatted as 32-bit IEEE Floating-point per channel, 32-bit UINT per channel, 32-bit SINT per channel, or SRGB depending on the surface type (e.g. R32G32B32A32_UINT surfaces assume use 32-bit UINT for clear color). 3D Sampler will always fetch clear color from the location 16-bytes above this address, where the clear color, converted to native surface format, will be stored.
For Sampling Engine Surfaces and Render Targets with Depth Surfaces: Specifies the lower bits of Graphics Address where the depth clear value is stored.
The memory format is IEEE 32 bit float. The numeric range is required to match the numeric range limitations of 3DSTATE_CLEAR_PARAMS:Depth Clear Value.
3D Sampler will always fetch clear depth from the location 16-bytes above this address, where the clear depth, converted to native surface format by software, will be stored.

For D24X8 depth surfaces (R24_UNORM_X8_TYPELESS), the format of the data at this location shall be UNORM24_X8 rather than a 32-bit format.

5 **Reserved**

Access:	RO
Format:	MBZ

	4:0	**Reserved** Access: RO Format: MBZ
13	31	**Reserved** Access: RO Format: MBZ
	30:16	**Reserved** Access: RO Format: MBZ
	15:0	**Clear Address High** Format: GraphicsAddress[47:32] For Sampling Engine Surfaces and Render Targets with Auxiliary Surface Mode set to AUX_CCS: Specifies the higher bits of Graphics Address where clear value is stored from RGBA (R in the LSB and A in the MSB - in that order) For Depth Surfaces: Specifies the higher bits of Graphics Address.
14	31:0	**Reserved** Access: RO Format: MBZ
15	31:0	**Reserved** Access: RO Format: MBZ

Render Data Port Message Types

MT_DP_RT - Render Data Port Message Types

Source:	EuSubFunctionRenderDataPort
Size (in bits):	5
Default Value:	0x0000000C

Lists all the Message Types in a Render Data Port Message Descriptor [18:14].

DWord	Bit	Description
0	4	**Reserved** Access: RO Format: MBZ
	3:0	**Message Type** Specifies type of message

Value	Name	Description
0Ch	MT_RTW **[Default]**	Render Target Write message
0Dh	MT_RTR	Render Target Read message
Others	Reserved	Ignored

Render Engine Interrupt Vector

RENDER_INTR_VEC - Render Engine Interrupt Vector

Size (in bits): 16

Default Value: 0x00000000

DWord	Bit	Description
0	15	**Catastrophic Error** This interrupt signals that a unrecoverable errorduring the engine processing. When Memory interface signals this error, the Command Streamer will stop parsing any more instructions. Scheduler is expected to reset the engine to evict the context
	14	**EU Restart Interrupt** EU Restart Interrupt is generated by the GA fabric, and not by Render Command Streamer. GA routes this interrupt to GuC independently of Command Stream.
	13	**Context Stall** Command streamer will generate a Context Stall interrupt when a high priority context gets stalled due to the other command streamer executing a normal priority or lowpriority context is "Run Alone" mode OR Command streamer will generate a Context Stall interrupt when a high priority context gets stalled while procuring run alone mode.
	12	**Reserved** Access: RO Format: MBZ
	11	**CS Wait On Semaphore**
	10	**Spare 10**
	9	**CS TR Invalid Tile Detection**
	8	**CS Context Switch Interrupt**
	7	**Legacy Context Per Process Page Fault Interrupt** This Fault interrupt is only delivered to the Host SW (not to GuC). Fault interrupt is generated by GA fabric, not by the CS This interrupt is for handling Legacy Page Fault. When Fault Repair Mode is enabled, Interrupt mask register value is not looked at to generate interrupt due to page fault. Please refer to vol1c "page fault support" section for more details. In Advanced (PRQ) Fault Interface is done through GUC interface.
	6	**CS Watchdog Counter Expired**
	5	**Spare 5**
	4	**CS PIPE_CONTROL Notify**
	3	**CS Error Interrupt**
	2	**Spare 2**

RENDER_INTR_VEC - Render Engine Interrupt Vector		
	1	**Reserved**
	0	**CS MI User Interrupt**

Render Target Index Message Header Control

MHC_RT_RTI - Render Target Index Message Header Control

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:3	**Reserved** Access: RO Format: MBZ
	2:0	**Render Target Index** Format: U3 Specifies the render target index that will be used to select blend state from BLEND_STATE.

Render Target Message Header

MH_RT - Render Target Message Header

Size (in bits): 512

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.0	31:0	**Render Target Controls 0** Format: MHC_RT_C0 Specifies controls for Render Target Write and Read messages.
0.1-0.1	31:0	**Color Calculator State Pointer** Format: MHC_RT_CCSP For Render Target Write message, specifies the HWORD-aligned GeneralStateOffset for Color State. Ignored by Render Target Read message.
0.2-0.2	31:0	**Render Target Index** Format: MHC_RT_RTI For Render Target Write message, specifies the render target index used to select blend state from BLEND_STATE. Ignored by Render Target Read message.
0.3-0.4	63:0	**Reserved** Access: RO Format: MBZ
0.5-0.5	31:0	**Color Code** Format: MHC_RT_CC Hardware uses to track synchronizing events and free resources on thread completion.
0.6-0.7	63:0	**Reserved** Access: RO Format: MBZ
1.0-1.0	31:0	**Reserved** Access: RO Format: MBZ
1.1-1.1	31:0	**Poly 0** Format: MHC_RT_POLY Poly Information

MH_RT - Render Target Message Header		
1.2-1.2	31:0	**Subspan 0** Format: **MHC_RT_SUBSPAN** Upper left corner of subspan 0
1.3-1.3	31:0	**Subspan 1** Format: **MHC_RT_SUBSPAN** Upper left corner of subspan 1
1.4-1.4	31:0	**Subspan 2** Format: **MHC_RT_SUBSPAN** Upper left corner of subspan 2
1.5-1.5	31:0	**Subspan 3** Format: **MHC_RT_SUBSPAN** Upper left corner of subspan 3
1.6-1.6	31:0	**Poly 1** Format: **MHC_RT_POLY** Poly Information for second poly when dual-SIMD8 dispatch
1.7-1.7	31:0	**Pixel Sample Enables** Format: **MHC_RT_PSM** Pixel Sample Enables

Render Target Message Header Control

MHC_RT_C0 - Render Target Message Header Control

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:15	**Reserved** Access: RO Format: MBZ
	14	**Stencil Present to Render Target** Format: Enable For Render Target Write message, indicates that computed stencil is included in the message. Must be zero for Render Target Read message.
	13	**Source Depth Present to Render Target** Format: Enable For Render Target Write Message, indicates that source depth data is included in the message. Must be zero for Render Target Read message.
	12	**oMask to Render Target** Format: Enable For Render Target Write message, indicates that oMask data is present in the message and is to be used to mask off samples. Must be zero for Render Target Read message.
	11	**Source0 Alpha Present to Render Target** Format: Enable For Render Target Write message, indicates that Source0 Alpha (aka o0.a) data is included in RTWrite message. If present, these alpha values are used as inputs to AlphaTest and AlphaToCoverage functions. This is required to meet the API rules when writing to multiple render targets (MRTs). Must be zero for Render Target Read message. **Programming Notes** This bit should not be set when write to RT0, though sending and using redundant alpha will provide the correct results (at lower performance). This bit is not supported on Dual-Source Blend message types, as source0 alpha is already included in those messages. This bit is not supported on replicated data message types.
	10	**Reserved** Access: RO Format: MBZ

MHC_RT_C0 - Render Target Message Header Control

	9:6	**Sample Index** Format: U4 When pixel shader is dispatched in per-pixel mode with Per-Sample PS Enable bit set, this field indicates the index of a sample referenced by per-sample RT read or RT write messages. Range = [0, 15].
	5:0	**Reserved** Access: RO Format: MBZ

Render Target Message Header Poly

MHC_RT_POLY - Render Target Message Header Poly

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30:27	**Viewport Index** Format: U4 For Render Target Write message, specifies the index of the viewport currently being used. Range = [0,15] Ignored by Render Target Read message.
	26:16	**Render Target Array Index** Format: U11 Specifies the array index to be used for the following surface types: SURFTYPE_1D: specifies the array index. Range = [0,511] SURFTYPE_2D: specifies the array index. Range = [0,511] SURFTYPE_3D: specifies the Z or R coordinate. Range = [0,2047] SURFTYPE_BUFFER: must be zero. SURFTYPE_CUBE: specifies the face identifier. Mapping (0,+x) (1,-x) (2,+y) (3,-y) (4,+z) (5,-z). **Programming Notes** The Render Target Array Index used by hardware for access to the Render Target is overridden with the Minimum Array Element defined in SURFACE_STATE if it is out of the range between Minimum Array Element and Depth. For cube surfaces, a depth value of 5 is used for this determination.
	15	**Front/Back Facing Polygon** Format: U1 Determines whether the polygon is front or back facing. Used by the render cache to determine which stencil test state to use. Value / Name / Description 0h / Front facing / All 1h / Back facing / All
	14:9	**Reserved** Access: RO Format: MBZ
	8:6	**Starting Sample Pair Index** Format: U3 Indicates the index of the first sample pair of the dispatch. Range = [0,3]

MHC_RT_POLY - Render Target Message Header Poly		
	5:0	**Reserved** Access: RO Format: MBZ

Replicated Pixel Render Target Data Payload Register

MDPR_RGBA - Replicated Pixel Render Target Data Payload Register

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:0	**Red** Format: U32 Specifies the value of all slots' red channel.
1	31:0	**Green** Format: U32 Specifies the value of all slots' green channel.
2	31:0	**Blue** Format: U32 Specifies the value of all slots' blue channel.
3	31:0	**Alpha** Format: U32 Specifies the value of all slots' alpha channel.
4..7	127:0	**Reserved** Access: RO Format: MBZ

Replicated SIMD16 Render Target Data Payload

MDP_RTW_16REP - Replicated SIMD16 Render Target Data Payload		
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**RGBA** Format: MDPR_RGBA RGBA for all slots [15:0]

Reversed SIMD Mode 2 Message Descriptor Control Field

MDC_SM2R - Reversed SIMD Mode 2 Message Descriptor Control Field

Size (in bits): 1

Default Value: 0x00000000

DWord	Bit	Description
0	0	**SIMD Mode** Format: Boolean Specifies the SIMD mode of the message (number of slots processed) Value / Name / Description 00h / SIMD16 / SIMD16 01h / SIMD8 / SIMD8

S0A SIMD8 Render Target Data Payload

MDP_RTW_A8 - S0A SIMD8 Render Target Data Payload

Size (in bits): 1280

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Source 0 Alpha** Format: MDP_DW_SIMD8 Slots [7:0] Source 0 Alpha
1.0-1.7	255:0	**Red** Format: MDP_DW_SIMD8 Slots [7:0] Red
2.0-2.7	255:0	**Green** Format: MDP_DW_SIMD8 Slots [7:0] Green
3.0-3.7	255:0	**Blue** Format: MDP_DW_SIMD8 Slots [7:0] Blue
4.0-4.7	255:0	**Alpha** Format: MDP_DW_SIMD8 Slots [7:0] Alpha

S0A SIMD16 Render Target Data Payload

MDP_RTW_A16 - S0A SIMD16 Render Target Data Payload	
Size (in bits):	2560
Default Value:	0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Source 0 Alpha[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Source 0 Alpha
1.0-1.7	255:0	**Source 0 Alpha[15:7]** Format: MDP_DW_SIMD8 Slots [15:8] Source 0 Alpha
2.0-2.7	255:0	**Red[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Red
3.0-3.7	255:0	**Red[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Red
4.0-4.7	255:0	**Green[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Green
5.0-5.7	255:0	**Green[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Green

MDP_RTW_A16 - S0A SIMD16 Render Target Data Payload		
6.0-6.7	255:0	**Blue[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Blue
7.0-7.7	255:0	**Blue[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Blue
8.0-8.7	255:0	**Alpha[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Alpha
9.0-9.7	255:0	**Alpha[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Alpha

SAMPLER_BORDER_COLOR_STATE

SAMPLER_BORDER_COLOR_STATE

Size (in bits): 128

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000

Description

The interpretation of the border color depends on the Texture Border Color Mode field in SAMPLER_STATE as follows:

- DX9 mode: The border color is 8-bit UNORM format, regardless of the surface format chosen. For surface formats with one or more channels missing (i.e. R5G6R5_UNORM is missing the alpha channel), the value from the border color, if selected, will be used even for the missing channels.
- DX10/OGL mode: the format of the border color depends on the format of the surface being sampled. If the map format is UINT, then the border color format is R32G32B32A32_UINT. If the map format is SINT, then the border color format is R32G32B32A32_SINT. Otherwise, the border color format is R32G32B32A32_FLOAT.For surface formats with one or more channels missing, the value from the border color is not used for the missing channels, resulting in these channels resulting in the overall default value (0 for colors and 1 for alpha) regardless of whether border color is chosen. The surface formats with "L" and "I" have special behavior with respect to the border color. The border color value used for the replicated channels (RGB for "L" formats and RGBA for "I" formats) comes from the red channel of border color. In these cases, the green and blue channels, and also alpha for "I", of the border color are ignored. The format of this state depends on the Texture Border Color Mode field.

If the Texture Border Color Mode field in SAMPLER_STATE is set to DX9 and the MMIO register bit "Enable Missing Alpha Format Fix" is set in register E194h, then the interpretation of the border color format depends of the format of the surface being sampled:

1. If the map format is UINT, border color is R8G8B8A8_UINT
2. If the map format is SINT, border color is R8G8B8A8_SINT
3. Otherwise, border color is R8G8B8A8_UNORM

Programming Notes

- DX9 mode is not supported for surfaces with more than 16 bits in any channel, other than 32-bit float formats which are supported.
- The conditions under which this color is used depend on the **Surface Type** - 1D/2D/3D surfaces use the border color when the coordinates extend beyond the surface extent; cube surfaces use the border color for "empty" (disabled) faces.
- The border color itself is accessed through the texture cache hierarchy rather than the state cache hierarchy. Thus, if the border color is changed in memory, the texture cache must be invalidated and the state cache does not need to be invalidated.
- MAPFILTER_MONO: The border color is ignored. Border color is fixed at a value of 0 by hardware.

- The border color itself is accessed through the texture cache hierarchy rather than the state cache hierarchy. Thus, if the border color is changed in memory, the texture cache must be invalidated and the

SAMPLER_BORDER_COLOR_STATE

state cache does not need to be invalidated.

If the Texture Border Color Mode field in SAMPLER_STATE is set to DX9 and the MMIO register bit "Enable Missing Alpha Format Fix" is set in register E194h and a surface with format SINT is being sampled, then each channel of the border color must be 00h or 01h (0 or 1 in SINT encoding).

DWord	Bit	Description
0	31:24	**Border Color Alpha** Exists If: Structure[SAMPLER_STATE][Texture Border Color Mode] == 'DX9' Format: UNORM8 Texture Border Color Mode = DX9
	31:0	**Border Color Red - (DX10/0GL)** Exists If: Structure[SAMPLER_STATE][Texture Border Color Mode] == 'DX10/0GL' Format: IEEE_FLOAT Texture Border Color Mode = DX10/OGL
	23:16	**Border Color Blue** Exists If: Structure[SAMPLER_STATE][Texture Border Color Mode] == 'DX9' Format: UNORM8 Texture Border Color Mode = DX9
	15:8	**Border Color Green** Exists If: Structure[SAMPLER_STATE][Texture Border Color Mode] == 'DX9' Format: UNORM8 Texture Border Color Mode = DX9
	7:0	**Border Color Red - (DX9)** Exists If: Structure[SAMPLER_STATE][Texture Border Color Mode] == 'DX9' Format: UNORM8 Texture Border Color Mode = DX9
1	31:0	**Border Color Green** Format: IEEE_FLOAT Texture Border Color Mode = DX10/OGL
2	31:0	**Border Color Blue** Format: IEEE_FLOAT Texture Border Color Mode = DX10/OGL

SAMPLER_BORDER_COLOR_STATE		
3	31:0	**Border Color Alpha** Format: IEEE_FLOAT Texture Border Color Mode = DX10/OGL

SAMPLER_INDIRECT_STATE_BORDER_COLOR

SAMPLER_INDIRECT_STATE_BORDER_COLOR

Size (in bits): 128

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000

Description

This structure is a one version of the SAMPLER_INDIRECT_STATE structure, suitable for many needs.
An instance of this structure is pointed to by the **Indirect State Pointer** field in SAMPLER_STATE.
The interpretation of the border color depends on the **Texture Border Color Mode** field in SAMPLER_STATE as follows:

- In **8BIT** mode, the border color is 8-bit UNORM format, regardless of the surface format chosen. For surface formats with one or more channels missing (i.e. R5G6R5_UNORM is missing the alpha channel), the value from the border color, if selected, will be used *even for the missing channels*.
- In **OGL** mode, the format of the border color is R32G32B32A32_FLOAT, R32G32B32A32_SINT, or R32G32B32A32_UINT, depending on the surface format chosen. For surface formats with one or more channels missing, the value from the border color is not used for the missing channels, resulting in these channels resulting in the overall default value (0 for colors and 1 for alpha) regardless of whether border color is chosen. The surface formats with "L" and "I" have special behavior with respect to the border color. The border color value used for the replicated channels (RGB for "L" formats and RGBA for "I" formats) comes from the *red* channel of border color. In these cases, the green and blue channels, and also alpha for "I", of the border color are ignored.

Programming Notes

- 8BIT mode is not supported for surfaces with more than 16 bits in any channel, other than 32-bit float formats which are supported.
- The conditions under which this color is used depend on the **Surface Type** - 1D/2D/3D surfaces use the border color when the coordinates extend beyond the surface extent; cube surfaces use the border color for "empty" (disabled) faces.
- The border color itself is accessed through the texture cache hierarchy rather than the state cache hierarchy. Thus, if the border color is changed in memory, the texture cache must be invalidated and the state cache does not need to be invalidated.
- MAPFILTER_MONO: The border color is ignored. Border color is fixed at a value of 0 by hardware.

DWord	Bit	Description
0	31:24	**Border Color Alpha As U8** Exists If: //Structure[SAMPLER_STATE][Texture Border Color Mode] == '8BIT' Format: U8
	31:0	**Border Color Red As S31** Exists If: //Structure[SAMPLER_STATE][Texture Border Color Mode] == 'OGL' AND (Structure[RENDER_SURFACE_STATE][Surface Format]Property[IsSigned]=='true' Format: S31

SAMPLER_INDIRECT_STATE_BORDER_COLOR

		Format: U32
		Format: IEEE_FLOAT
	23:16	**Border Color Blue As U8**
		Exists If: //Structure[SAMPLER_STATE][Texture Border Color Mode] == '8BIT'
		Format: U8
	15:8	**Border Color Green As U8**
		Exists If: //Structure[SAMPLER_STATE][Texture Border Color Mode] == '8BIT'
		Format: U8
	7:0	**Border Color Red As U8**
		Exists If: //Structure[SAMPLER_STATE][Texture Border Color Mode] == '8BIT'
		Format: U8
1	31:0	**Border Color Green As S31**
		Exists If: //Structure[SAMPLER_STATE][Texture Border Color Mode] == 'OGL' AND (Structure[RENDER_SURFACE_STATE][Surface Format]Property[IsSigned]=='true'
		Format: S31
		Format: U32
		Format: IEEE_FLOAT
2	31:0	**Border Color Blue As S31**
		Exists If: //Structure[SAMPLER_STATE][Texture Border Color Mode] == 'OGL' AND (Structure[RENDER_SURFACE_STATE][Surface Format]Property[IsSigned]=='true'
		Format: S31
		Format: U32
		Format: IEEE_FLOAT
3	31:0	**Border Color Alpha As S31**
		Exists If: //Structure[SAMPLER_STATE][Texture Border Color Mode] == 'OGL' AND (Structure[RENDER_SURFACE_STATE][Surface Format]Property[IsSigned]=='true'
		Format: S31
		Format: U32
		Format: IEEE_FLOAT

SAMPLER_INDIRECT_STATE

SAMPLER_INDIRECT_STATE

Size (in bits):	512
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

Description

Note: There are three variations of this structure, defined separately because their payloads have different lengths. Currently only SAMPLER_INDIRECT_STATE_BORDER_COLOR is fully defined.
This structure is pointed to by **Indirect State Pointer** (SAMPLER_STATE).
The interpretation of the border color depends on the **Texture Border Color Mode** field in SAMPLER_STATE as follows:

- In **8BIT** mode, the border color is 8-bit UNORM format, regardless of the surface format chosen. For surface formats with one or more channels missing (i.e. R5G6R5_UNORM is missing the alpha channel), the value from the border color, if selected, will be used *even for the missing channels*.
- In **OGL** mode, the format of the border color is R32G32B32A32_FLOAT, R32G32B32A32_SINT, or R32G32B32A32_UINT, depending on the surface format chosen. For surface formats with one or more channels missing, the value from the border color is not used for the missing channels, resulting in these channels resulting in the overall default value (0 for colors and 1 for alpha) regardless of whether border color is chosen. The surface formats with "L" and "I" have special behavior with respect to the border color. The border color value used for the replicated channels (RGB for "L" formats and RGBA for "I" formats) comes from the *red* channel of border color. In these cases, the green and blue channels, and also alpha for "I", of the border color are ignored.

The format of this state depends on the **Texture Border Color Mode** field.

Programming Notes

- 8BIT mode is not supported for surfaces with more than 16 bits in any channel, other than 32-bit float formats which are supported.
- The conditions under which this color is used depend on the **Surface Type** - 1D/2D/3D surfaces use the border color when the coordinates extend beyond the surface extent; cube surfaces use the border color for "empty" (disabled) faces.
- The border color itself is accessed through the texture cache hierarchy rather than the state cache hierarchy. Thus, if the border color is changed in memory, the texture cache must be invalidated and the state cache does not need to be invalidated.
- MAPFILTER_MONO: The border color is ignored. Border color is fixed at a value of 0 by hardware.

- The conditions under which this color is used depend on the **Surface Type**- 1D/2D/3D surfaces use the border color when the coordinates extend beyond the surface extent; cube surfaces use the border color for "empty" (disabled) faces.
- The border color itself is accessed through the texture cache hierarchy rather than the state cache hierarchy. Thus, if the border color is changed in memory, the texture cache must be invalidated and the

SAMPLER_INDIRECT_STATE

state cache does not need to be invalidated.

DWord	Bit	Description
0	31:24	**Border Color Alpha** Exists If: //Structure[SAMPLER_STATE][Texture Border Color Mode] == '8BIT' Format: UNORM8 Texture Border Color Mode = 8BIT
	31:0	**Border Color Red** Exists If: //Structure[RENDER_SURFACE_STATE][Surface Format]Property[IsSigned]=='true' Format: S31 (2's complement) for all SINT surface formats Format: U32 for all UINT surface formats Format: IEEE_FLOAT for all other surface formats
	23:16	**Border Color Blue** Exists If: //Structure[SAMPLER_STATE][Texture Border Color Mode] == '8BIT' Format: UNORM8 Texture Border Color Mode = 8BIT
	15:8	**Border Color Green** Exists If: //Structure[SAMPLER_STATE][Texture Border Color Mode] == '8BIT' Format: UNORM8 Texture Border Color Mode = 8BIT
	7:0	**Border Color Red** Exists If: //Structure[SAMPLER_STATE][Texture Border Color Mode] == '8BIT' Format: UNORM8 Texture Border Color Mode = 8BIT
1	31:0	**Border Color Green** Exists If: //Structure[RENDER_SURFACE_STATE][Surface Format]Property[IsSigned]=='true' Format: IEEE_FLOAT Format: S31 Format: U32
2	31:0	**Border Color Blue** Exists If: //Structure[RENDER_SURFACE_STATE][Surface Format]Property[IsSigned]=='true' Format: IEEE_FLOAT Format: S31 Format: U32

<table>
<tr><th colspan="3">SAMPLER_INDIRECT_STATE</th></tr>
<tr><td>3</td><td>31:0</td><td>Border Color Alpha
<table>
<tr><td>Exists If:</td><td>//Structure[RENDER_SURFACE_STATE][Surface Format]Property[IsSigned]=='true'</td></tr>
<tr><td>Format:</td><td>IEEE_FLOAT</td></tr>
<tr><td>Format:</td><td>S31</td></tr>
<tr><td>Format:</td><td>U32</td></tr>
</table></td></tr>
<tr><td>4..15</td><td>383:0</td><td>Reserved
<table>
<tr><td>Access:</td><td>RO</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
</table>

SAMPLER_STATE_8x8_AVS

SAMPLER_STATE_8x8_AVS	
Size (in bits):	8960
Default Value:	0x00000000, 0x00000000,

SAMPLER_STATE_8x8_AVS

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

Description
ExistsIf = AVS

DWord	Bit	Description
0..2	95:0	**Reserved** Access: RO Format: MBZ
3	31:30	**Reserved** Access: RO Format: MBZ
	29:28	**Enable 8-tap filter** **Adaptive Filtering (Mode = 11) ExistsIf:** R10G10B10A2_UNORM R8G8B8A8_UNORM (AYUV also) R8B8G8A8_UNORM B8G8R8A8_UNORM R16G16B16A16 *Enable 8-tap Filtering on UV channel (Mode = 10) ExistsIf:* R16B16_UNORM, R16_UNORM **Enable 8-tap Filtering on UV channel (Mode = 10) ExistsIf:** R10G10B10A2_UNORMR8G8B8A8_UNORM (AYUV also)R8B8_UNORM (CrCb)R8_UNORMR8B8G8A8_UNORMB8G8R8A8_UNORMR16G16B16A16Y8_UNORM

Value	Name	Description
00b		4-tap filter is only done on all channels.
01b		Enable 8-tap Adaptive filter on G-channel. 4-tap filter on other channels.
10b		8-tap filter is done on all channels (UV-ch uses the Y-coefficients)
11b		Enable 8-tap Adaptive filter all channels (UV-ch uses the Y-coefficients).

Programming Notes
For 00 and 10, are applicable for RGB surfaces only or surface without Y-ch.In case it is a YUV surface it will default to adaptive mode automatically which is 01 and 11 respectively. Alpha channel is always bi-linear filter irrespective of the above modes.
Mode 01 and 00 are legacy support and are supported on all surface formats.
When Mode is 10 and Surface format is Y8_UNORM, Bypass X/Y Adaptive Filtering must be 1, and Default Sharp Level must be 255

SAMPLER_STATE_8x8_AVS

	27:0	**Reserved** Access: RO Format: MBZ
4	31:12	**Reserved** Access: RO Format: MBZ
	11	**Shuffle_OutputWriteback for sample_8x8** 0 – Writeback same as Original Sample_8x8 1 – Writeback of Sample_8x8 Is Modified to Suite Sample_Unorm
	10:0	**Reserved** Access: RO Format: MBZ
5..15	351:0	**Reserved** Access: RO Format: MBZ
16..151	4351:0	**Filter Coefficient[0..16]** Format: SAMPLER_STATE_8x8_AVS_COEFFICIENTS[17]
152	31:24	**Default Sharpness Level** Format: U8 When adaptive scaling is off, determines the balance between sharp and smooth scalers. 0 – **[Default]** Contribute 1 from the smooth scalar 255 – Contribute 1 from the sharp scalar
	23:16	**Max Derivative 4 Pixels** Format: U8 Used in adaptive filtering to specify the lower boundary of the smooth 4 pixel area.
	15:8	**Max Derivative 8 Pixels** Format: U8 Used in adaptive filtering to specify the lower boundary of the smooth 8 pixel area.
	7	**Reserved** Access: RO Format: MBZ

SAMPLER_STATE_8x8_AVS

	6:4	**Transition Area with 4 Pixels** Format: U3 Used in adaptive filtering to specify the width of the transition area for the 4 pixel calculation.
	3	**Reserved** Access: RO Format: MBZ
	2:0	**Transition Area with 8 Pixels** Format: U3 Used in adaptive filtering to specify the width of the transition area for the 8 pixel calculation.
153	31:23	**Reserved** Access: RO Format: MBZ
	22	**Bypass X Adaptive Filtering** Format: Disable When disabled, the X direction will use **Default Sharpness Level** to blend between the smooth and sharp filters rather than the calculated value.

Value	Name	Description
1	Disble	Disable X Adaptive Filtering
0	Enable	Enable X Adaptive Filtering

	21	**Bypass Y Adaptive Filtering** Format: Disable When disabled, the Y direction will use **Default Sharpness Level** to blend between the smooth and sharp filters rather than the calculated value.

Value	Name	Description
1	Disble	Disable Y Adaptive Filtering
0	Enable	Enable Y Adaptive Filtering

	20:2	**Reserved** Access: RO Format: MBZ
	1	**Adaptive Filter for all channels** Format: Enable Only to be enabled if 8-tap Adaptive filter mode is on, eElse it should be disabled.

Value	Name	Description
1	Enable	Enable Adaptive Filter on UV/RB Channels
0	Disble	Disable Adaptive Filter on UV/RB Channels

SAMPLER_STATE_8x8_AVS

	0	**RGB Adaptive** Format: Enable This should be always set to 0 for YUV input and can be enabled/disabled for RGB input.This should be enabled only if we enable 8-tap adaptive filter for RGB input. **Value** / **Name** / **Description** 1 / Enable / Enable the RGB Adaptive filter using the equation (Y=(R+2G+B)»2) 0 / Disble / Disable the RGB Adaptive equation and use G-Ch directly for adaptive filter
154..159	191:0	**Reserved** Access: RO Format: MBZ
160..279	3839:0	**Filter Coefficient[17..31]** Format: SAMPLER_STATE_8x8_AVS_COEFFICIENTS[15]

SAMPLER_STATE_8x8_AVS_COEFFICIENTS

SAMPLER_STATE_8x8_AVS_COEFFICIENTS		
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
ExistsIf = AVS && (Function_mode = 0)		
DWord	**Bit**	**Description**
0	31:24	**Table 0Y Filter Coefficient[n,1]** Format: S1.6 **Range:** [-2, +2)
	23:16	**Table 0X Filter Coefficient[n,1]** Format: S1.6 **Range:** [-2, +2)
	15:8	**Table 0Y Filter Coefficient[n,0]** Format: S1.6 **Range:** [-2, +2) **Programming Notes** If the format is R10G10B10A2_UNORM or R8G8B8A8_UNORM, this field MBZ.
	7:0	**Table 0X Filter Coefficient[n,0]** Format: S1.6 **Range:** [-2, +2) **Programming Notes** If the format is R10G10B10A2_UNORM or R8G8B8A8_UNORM, this field MBZ.
1	31:24	**Table 0Y Filter Coefficient[n,3]** Format: S1.6 **Range:** [-2.0, +2.0)
	23:16	**Table 0X Filter Coefficient[n,3]** Format: S1.6 **Range:** [-2.0, +2.0)
	15:8	**Table 0Y Filter Coefficient[n,2]** Format: S1.6 **Range:** [-2.0, +2.0)

SAMPLER_STATE_8x8_AVS_COEFFICIENTS

	7:0	**Table 0X Filter Coefficient[n,2]** Format: S1.6 **Range:** [-2.0, +2.0)
2	31:24	**Table 0Y Filter Coefficient[n,5]** Format: S1.6 **Range:** [-2.0, +2.0)
	23:16	**Table 0X Filter Coefficient[n,5]** Format: S1.6 **Range:** [-2.0, +2.0)
	15:8	**Table 0Y Filter Coefficient[n,4]** Format: S1.6 **Range:** [-2.0, +2.0) **Programming Notes** If the format is R10G10B10A2_UNORM or R8G8B8A8_UNORM, this field MBZ.
	7:0	**Table 0X Filter Coefficient[n,4]** Format: S1.6 **Range:** [-2.0, +2.0) **Programming Notes** If the format is R10G10B10A2_UNORM or R8G8B8A8_UNORM, this field MBZ.
3	31:24	**Table 0Y Filter Coefficient[n,7]** Format: S1.6 **Range:** [-2, +2)
	23:16	**Table 0X Filter Coefficient[n,7]** Format: S1.6 **Range:** [-2, +2)
	15:8	**Table 0Y Filter Coefficient[n,6]** Format: S1.6 **Range:** [-2, +2)
	7:0	**Table 0X Filter Coefficient[n,6]** Format: S1.6 **Range:** [-2, +2)

SAMPLER_STATE_8x8_AVS_COEFFICIENTS

4	31:24	**Table 1X Filter Coefficient[n,3]** Format: S1.6 **Range:** [-2.0, +2.0)
	23:16	**Table 1X Filter Coefficient[n,2]** Format: S1.6 **Range:** [-2.0, +2.0)
	15:0	**Reserved** Access: RO Format: MBZ
5	31:16	**Reserved** Access: RO Format: MBZ
	15:8	**Table 1X Filter Coefficient[n,5]** Format: S1.6 **Range:** [-2.0, +2.0)
	7:0	**Table 1X Filter Coefficient[n,4]** Format: S1.6 **Range:** [-2.0, +2.0)
6	31:24	**Table 1Y Filter Coefficient[n,3]** Format: S1.6 **Range:** [-2.0, +2.0)
	23:16	**Table 1Y Filter Coefficient[n,2]** Format: S1.6 **Range:** [-2.0, +2.0)
	15:0	**Reserved** Access: RO Format: MBZ
7	31:16	**Reserved** Access: RO Format: MBZ
	15:8	**Table 1Y Filter Coefficient[n,5]** Format: S1.6 **Range:** [-2.0, +2.0)

SAMPLER_STATE_8x8_AVS_COEFFICIENTS		
	7:0	**Table 1Y Filter Coefficient[n,4]** Format: S1.6 **Range:** [-2.0, +2.0)

SAMPLER_STATE

SAMPLER_STATE

Exists If:	//(MessageType != 'Deinterlace') && (MessageType != 'Sample_8x8')
Size (in bits):	128
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000

This is the normal sampler state used by all messages that use SAMPLER_STATE except sample_8x8 and deinterlace. The sampler state is stored as an array of up to 16 elements, each of which contains the dwords described here. The start of each element is spaced 4 dwords apart. The first element of the sampler state array is aligned to a 32-byte boundary.

DWord	Bit	Description
0	31	**Sampler Disable** Format: Disable This field allows the sampler to be disabled. If disabled, all output channels will return 0.
	30	**CPS LOD Compensation Enable** Format: Enable This field, if enabled, causes derivatives used to compute LOD to be adjusted by scale factors for coarse pixel shading. The adjustment only occurs if the following are all true: • This field is enabled • **CPS Message LOD Compensation Enable** in the message header is enabled The scale.x and scale.y factors are computed in hardware and delivered to the sampler at thread dispatch time. The following adjustments generate new derivatives as follows: $\frac{du}{dx} = \frac{du}{dx} * scale.x \quad \frac{dv}{dx} = \frac{dv}{dx} * scale.x \quad \frac{dr}{dx} = \frac{dr}{dx} * scale.x$ $\frac{du}{dy} = \frac{du}{dy} * scale.y \quad \frac{dv}{dy} = \frac{dv}{dy} * scale.y \quad \frac{dr}{dy} = \frac{dr}{dy} * scale.y$
	29	**Texture Border Color Mode** For some surface formats, the 32 bit border color is decoded differently based on the border color mode. In addition, the default value of channels not included in the surface may be affected by this field. Refer to the "Sampler Output Channel Mapping" table for the values of these channels, and for surface formats that may only support one of these modes. Also refer to the definition of SAMPLER_BORDER_COLOR_STATE for more details on the behavior of the two modes defined by this field. **Value** / **Name** / **Description** 0h / OGL / New mode for interpreting the border color 1h / 8BIT / Earlier mode for interpreting the border color

Programming Notes
This must not be set to 8BIT if there are null tiles in use
This field is required to be the same for every message over a period of time. A flush of the sampler cache must occur before a message with the opposite state of this field is delivered.
This field must be set to 8BIT mode when used with surfaces that have Surface Format P4A4_UNORM or A4P4_UNORM.
This field must be set to OGL mode when used with surfaces that have Surface Format YCRCB_SWAPUV or YCRCB_SWAPY.
This field must be set to OGL mode if **Surface Format** for the associated surface is UINT OR SINT except when setting BORDER COLOR RED/GREEN/BLUE and ALPHA to 0
This field must be set to OGL mode if REDUCTION_MINIMUM or REDUCTION_MAXIMUM or message type is sample_min or sample_max.
Undefined behavior if 8BIT mode is used with any feature added. See Legacy sampler feature page for details

28:27 **LOD PreClamp Mode**

This field determines whether the computed LOD is clamped to [max,min] mip levelbefore the mag-vs-min determination is performed.
PRECLAMP_OGL: LOD pre-clamped to **Min LOD** and **Max LOD**
OpenGL API currently clamps LOD to the **Min LOD** and **Max LOD** (from Sampler State)prior to performing min/mag determination, and therefore it is expected that an OpenGL driverwould need to set this field to PRECLAMP_OGL.

Value	Name	Description
0h	NONE	LOD PreClamp disabled
1h	Reserved	
2h	OGL	LOD PreClamp enabled (OGL mode)

26 **Low Quality Cube Corner Mode Enable**

Format:	U1

This bit, when set to 1, forces sampler to use low-quality filtering for Cube Corners with texel replication which is not compatible with DirectX
When cleared to 0 (default), the sampler will use a high-quality filtering for Cube Corners with 3-way texel averaging.

Value	Name	Description
0h	Disable **[Default]**	Disables low-quality Cube Corner mode
1h	Enable	Enables low-quality Cube Corner mode

25:22 **Reserved**

Access:	RO
Format:	MBZ

SAMPLER_STATE

21:20 **Mip Mode Filter**

Format:	U2

This field determines if and how mip map levels are chosen and/or combined when texture filtering.

Value	Name	Description
0h	NONE	Disable mip mapping - force use of the mipmap level corresponding to Min LOD.
1h	NEAREST	Nearest, Select the nearest mip map
2h	Reserved	
3h	LINEAR	Linearly interpolate between nearest mip maps (combined with linear min/mag filters this is analogous to "Trilinear" filtering).

Programming Notes
MIPFILTER_LINEAR is not supported for surface formats that do not support "Sampling Engine Filtering" as indicated in the Surface Formats table unless using the sample_c message type or minimum/maximum operation.
MIP Mode Filter must be set to NONE for Planar YUV surfaces.
Mip Mode Filter must be set to MIPFILTER_NONE or MIPFILTER_NEAREST if Surface Format for the associated surface is UINT or SINT. However, all settings of this field are allowed with UINT/SINT if a minimum, maximum or gather4 operation is being performed.

19:17 **Mag Mode Filter**

Format:	U3

This field determines how texels are sampled/filtered when a texture is being "magnified" (enlarged). For volume maps, this filter mode selection also applies to the 3rd (inter-layer) dimension.

Value	Name	Description
0h	NEAREST	Sample the nearest texel
1h	LINEAR	Bilinearly filter the 4 nearest texels
2h	ANISOTROPIC	Perform an "anisotropic" filter on the chosen mip level
4h-5h	Reserved	
6h	Reserved	
7h	Reserved	

Programming Notes
Only MAPFILTER_NEAREST and MAPFILTER_LINEAR are supported for surfaces of type SURFTYPE_3D.
Only MAPFILTER_NEAREST is supported for surface formats that do not support "Sampling Engine Filtering" as indicated in the Surface Formats table unless using the sample_c message type or minimum/maximum operation.
MAPFILTER_ANISOTROPIC will be converted to linear if enabled for cube maps with the

	TEXCOORDMODE_CUBE addressing mode.
	MAPFILTER_ANISOTROPIC will be overridden to MAPFILTER_LINEAR when using a sample_l or sample_l_c message type or when Force LOD to Zero is set in the message header.
	Both Mag Mode Filter and Min Mode Filter must be set to MAPFILTER_NEAREST if Surface Format for the associated surface is UINT or SINT. However, all settings of this field are allowed with UINT/SINT if a minimum, maximum or gather4 operation is being performed.

16:14 **Min Mode Filter**

Format:	U3

This field determines how texels are sampled/filtered when a texture is being "minified" (shrunk). For volume maps, this filter mode selection also applies to the 3rd (inter-layer) dimension.See Mag Mode Filter

Value	Name	Description
0h	NEAREST	Sample the nearest texel
1h	LINEAR	Bilinearly filter the 4 nearest texels
2h	ANISOTROPIC	Perform an "anisotropic" filter on the chosen mip level
4h-5h	Reserved	
6h	Reserved	
7h	Reserved	

13:1 **Texture LOD Bias**

Format:	S4.8

Range: [-16.0, 16.0)

This field specifies the signed bias value added to the calculated texture map LOD prior to min-vs-mag determination and mip-level clamping. Assuming mipmapping is enabled, a positive LOD bias will result in a somewhat blurrier image (using less-detailed mip levels) and possibly higher performance, while a negative bias will result in a somewhat crisper image (using more-detailed mip levels) and may lower performance.

Programming Notes
There is no requirement or need to offset the LOD Bias in order to produce a correct LOD for texture filtering (as was required for correct bilinear and anisotropic filtering in some legacy devices).

0 **LOD algorithm**

Format:	U1

Controls which algorithm is used for LOD calculation. Generally, the EWA approximation algorithm results in higher image quality than the legacy algorithm.

Value	Name	Description
0h	LEGACY	Use the legacy algorithm for anisotropic filtering
1h	EWA Approximation	Use the new EWA approximation algorithm for anisotropic filtering

SAMPLER_STATE

1	31:20	**Min LOD** Format: U4.8 Range: [0.0, 14.0], where the upper limit is also bounded by the Max LOD. This field specifies the minimum value used to clamp the computed LOD after LOD bias is applied. Note that the minification-vs.-magnification status is determined after LOD bias and before this maximum (resolution) mip clamping is applied.The integer bits of this field are used to control the "maximum" (highest resolution) mipmap level that may be accessed (where LOD 0 is the highest resolution map). The fractional bits of this value effectively clamp the inter-level trilinear blend factor when trilinear filtering is in use. **Programming Notes** If Min LOD is greater than Max LOD, Min LOD takes precedence, i.e. the resulting LOD will always be Min LOD.
	19:8	**Max LOD** Format: U4.8 Range: [0.0, 14.0] This field specifies the maximum value used to clamp the computed LOD after LOD bias is applied. Note that the minification-vs.-magnification status is determined after LOD bias and before this minimum (resolution) mip clamping is applied. The integer bits of this field are used to control the "minimum" (lowest resolution) mipmap level that may be accessed. The fractional bits of this value effectively clamp the inter-level trilinear blend factor when trilinear filtering is in use. Force the mip map access to be between the mipmap specified by the integer bits of the Min LOD and the ceiling of the value specified here.
	7	**ChromaKey Enable** Format: Enable **Programming Notes** Supported only on a specific subset of surface formats. See section titled: "Surface Formats" in this volume for supported formats. This field must be disabled if min or mag filter is MAPFILTER_ANISOTROPIC. This field must be disabled if used with a surface of type SURFTYPE_3D. This field must be disable when Mip Mode Filter is no NONE. This bit must not be set if the **Auxiliary Surface Mode** is not AUX_NONE.
	6:5	**ChromaKey Index** Format: U2 Range: [0, 3] This field specifies the index of the ChromaKey Table entry associated with this Sampler. This field is a "don't care" unless **ChromaKey Enable** is ENABLED.

SAMPLER_STATE

4 **ChromaKey Mode**

Format:	U1

This field specifies the behavior of the device in the event of a ChromaKey match. This field is ignored if ChromaKey is disabled.

KEYFILTER_REPLACE_BLACK:In this mode, each texel that matches the chroma key is replaced with (0,0,0,0) (black with alpha=0) prior to filtering. For YCrCb surface formats, the black value is A=0, R(Cr)=0x80, G(Y)=0x10, B(Cb)=0x80. This will tend to darken/fade edges of keyed regions. Note that the pixel pipeline must be programmed to use the resulting filtered texel value to gain the intended effect, e.g., handle the case of a totally keyed-out region (filtered texel alpha==0) through use of alpha test, etc.

Value	Name	Description
0h	KEYFILTER_KILL_ON_ANY_MATCH	In this mode, if any contributing texel matches the chroma key, the corresponding pixel mask bit for that pixel is cleared. The result of this operation is observable only if the Killed Pixel Mask Return flag is set on the input message.
1h	KEYFILTER_REPLACE_BLACK	In this mode, each texel that matches the chroma key is replaced with (0,0,0,0) (black with alpha=0) prior to filtering. For YCrCb surface formats, the black value is A=0, R(Cr)=0x80, G(Y)=0x10, B(Cb)=0x80. This will tend to darken/fade edges of keyed regions. Note that the pixel pipeline must be programmed to use the resulting filtered texel value to gain the intended effect, e.g., handle the case of a totally keyed-out region (filtered texel alpha==0) through use of alpha test, etc.

3:1 **Shadow Function**

Format:	U3

This field is used for shadow mapping support via the sample_c message type, and specifies the specific comparison operation to be used. The comparison is between the texture sample red channel (except for alpha-only formats which use the alpha channel), and the "ref" value provided in the input message.

Value	Name
0h	PREFILTEROP ALWAYS
1h	PREFILTEROP NEVER
2h	PREFILTEROP LESS
3h	PREFILTEROP EQUAL
4h	PREFILTEROP LEQUAL
5h	PREFILTEROP GREATER
6h	PREFILTEROP NOTEQUAL

		7h: PREFILTEROP GEQUAL
	0	**Cube Surface Control Mode** Format: U1 When sampling from a SURFTYPE_CUBE surface, this field controls whether the TC* Address Control Mode fields are interpreted as programmed or overridden to TEXCOORDMODE_CUBE. Value: 0h = PROGRAMMED; 1h = OVERRIDE
2	31:24	**Extended Indirect State Pointer** These 8-bits represent the 8 msb's of the Indirect State Pointer to expand the offset form 16Mbytes to 4GBytes. These 8-bits in conjunction with the **Indirect State Pointer**field are the pointer to SAMPLER_INDIRECT_STATE, which contains the border color. The pointer is relative to the Dynamic State Base Address for Non-Bindless sampler state, and is relative to the Sample State Base Address for Bindless sampler state
	23:6	**Indirect State Pointer** Format: DynamicStateOffset[23:6]SAMPLER_INDIRECT_STATE_BORDER_COLOR This field specifies the pointer to SAMPLER_INDIRECT_STATE, which contains the border color. This pointer is relative to the Dynamic State Base Address for Non-Bindless sampler state, and is relative to the Sample State Base Address for Bindless sampler state If a static sampler state (included in message) is being used (by setting the associated MMIO bit in SAMPLER_MODE register and setting the sampler index to 0xF), then lsb (bit 6) contains a a 0 or 1 to indicate a fixed border color of black or white. The other bits of this field are ignored when static sampler state is being used.
	5	**Force gather4 Behavior** Format: Enable **Description** **Note: This feature should not be enabled. It must remain programmed to 0h.** This field, if enabled, specifies that the sampler should convert all SIMD8*, and SIMD16 sample* messages to behave as if the incoming message is a modified *gather4*, regardless of the actual message delivered. Any parameters included in the incoming message that are not needed by the gather4 operation are ignored by the sampler. The **Gather4 Source Channel Select** in the message header is ignored and set to the RED channel. The channel to sample mapping is modified from the normal *gather4* message as follows: upper left sample = alpha channel; upper right sample = red channel lower left sample = green channel; lower right sample = blue channel

4 **Reserved**

Access:	RO
Format:	MBZ

3 **Return Filter Weight for Border Texels**

This bit, when set, returns the filter_weight in the Alpha channel of all non-border texels. Red, Green, and Blue channels will contain the sample resultwith border texels excluded.
For cases where the surface format contains an Alpha channel, the result returned will be ovewritten to return the filter weight.
For cases where the surface format does not contain Alpha, the result will still be returned in the Alpha Channel.

Value	Name	Description
0h	Disable **[Default]**	When programmed to 0h, normal data will be returned on RGBA channels, including contribution from border color texels.
1h	Enable	When programmed to 1h, RGB channels return filter data contributed from non-border color texels, and A channel returns filter weight of contributing texels.

Programming Notes
If this bit is set then the border color and the Border Color Mode field (in SAMPLER_STATE) are ignored. Certain message types such as sample_c, sample_min/max and gather4_* have restrictions on the use of this mode. See the Messages section of the 3D sampler for more information.

2 **Return Filter Weight for Null Texels**

This bit, when set, causes samples to return filter_weight of all non-NULL texels in the Alpha channel; Red, Green, and Blue channels are contain the filter result with NULL texels excluded; A non-NULL texel is a texel which does not reference a Null Tile.
For cases where Tiled_Resource_Mode is TR_NONE, the result will always be 1.0 since no texels would be NULL.
For cases where the surface format contains an Alpha channel, the result returned will be overridden to return the filter weight.
For cases where the surface format does not contain Alpha, the result will still be returned in the Alpha Channel.

Value	Name	Description
0h	Disable **[Default]**	When programmed to 0h, filter weight will not be returned, and normal data will be returned on the Alpha channel.
1h	Enable	When programmed to 1h, filter weight will be returned on the Alpha channel rather than the normal data expected on the Alpha channel.

Programming Notes
Certain message types such as sample_c, sample_min/max and gather4_* have restrictions on the use of this mode. See the Messages section of the 3D sampler for more information.

1

SRGB DECODE

This bit controls whether the 3D sampler will decode an sRGB formatted surface into RGB prior to any filtering operation.

When set, it does not convert to linear RGB (via a reverse gamma conversion). This bit is ignored for ASTC formats, which are always converted to linear RGB prior to filtering.

Value	Name	Description
0h	DECODE_EXT **[Default]**	When set to 0h, the 3D sampler will convert texels from an sRGB surface to linear RGB prior to filtering and/or returning the value.
1h	SKIP_DECODE_EXT	When set to 1h, the 3D sampler will not convert texels to linear RGB before filtering and returning results.

0

LOD Clamp Magnification Mode

Format:	U1

This field allows the flexibility to control how LOD clamping is handled when in magnification mode.

Value	Name	Description
0h	MIPNONE	When in magnification mode, Sampler will clamp LOD as if the **Mip Mode Filter**is MIPFILTER_NONE. This is how OpenGL defines magnification, and therefore it isexpected that those drivers would not set this bit.
1h	MIPFILTER	When in magnification mode, Sampler will clamp LOD based on the value of **Mip Mode Filter**.

3

31:27

Reserved

Access:	RO
Format:	MBZ

26

low quality filter

Format:	enable

Setting this bit will enable low quality filter to save power.

*Will result in lower precision

* only has an affect if the surface format is unorm8 in the sampler L1

* has no affect if in anisotropic mode.

Value	Name	Description
0h	Disable **[Default]**	When set to 0h, filter quality is high and there is no degradation in precision. Power will be higher for some surface formats.
1h	Enable	When set to 1h, filter quality is lower and precision is reduced. Power will be lower for these surface formats.

Programming Notes
This bit should always be programmed to 1 to ensure low power operation.

25

Reserved

Access:	RO
Format:	MBZ

SAMPLER_STATE

24 **Allow low quality LOD calculation**

Format:	enable

Setting this bit will allow sampler to use the low quality LOD calculation mode for power savings. Note that this will not force low quality and sampler will only do it if the follow conditions are also true. If they are not true it will use the same algorithm as before as selected by the EWA bit
Message type sample/sample_l
Min/Mag/Mip_filter = nearest or linear.
Map type = 2D //No arrays
Indirect offsets must be zero
Coordinates must be normalized
No clamp border or half border
Sampler must not be disabled //Sampler state bit
No chromakey
No posh

23:22 **Reduction Type**

Format:	U2

This field defines the type of reduction that will be performed on the texels in the footprint defined by the **Min/Mag/Mip Filter Mode** fields. This field is ignored if **Reduction Type Enable** is disabled.

Value	Name	Description
0h	STD_FILTER	standard filter
1h	COMPARISON	comparison followed by standard filter
2h	MINIMUM	minimum of footprint
3h	MAXIMUM	maximum of footprint

Programming Notes
The following message types ignore this field: *sample_min, sample_max, sample_unorm*, resinfo, sampleinfo, LOD, ld*, sample_8x8*.
The *sample_c, sample_l_c, sample_d_c, sample_b_c, gather4_c*, and *gather4_po_c* message types, when used with STD_FILTER, MINIMUM, or MAXIMUM settings of this field, perform the operation of the message of the same name without the "_c". The ref parameter is ignored by hardware.
For message types not listed above, when used with COMPARISON setting of this field, perform the operation of the message of the same name with "_c" included. The ref parameter used by the operation (since it is not delivered in the message) is set to zero.
Restrictions applying to the message whose behavior is being performed must be followed. For example, a sample message used with COMPARISON reduction filter must follow all of the restrictions of *sample_c*. An exception to this is the MINIMUM and MAXIMUM reduction types allow SURFTYPE_1D, 2D, 3D, and CUBE, including with **Surface Array** enabled, even though the sample_min/max messages only allow 2D.
Restrictions applying to the message delivered need not be followed. For example, a *sample_c* message used with STD_FILTER reduction filter needs to follow only the restrictions of sample,

not the restrictions of *sample_c*.

21:19 **Maximum Anisotropy**

Format:	U3

This field clamps the maximum value of the anisotropy ratio used by the MAPFILTER_ANISOTROPIC filter (Min or Mag Mode Filter).

Value	Name	Description
0h	RATIO 2:1	At most a 2:1 aspect ratio filter is used
1h	RATIO 4:1	At most a 4:1 aspect ratio filter is used
2h	RATIO 6:1	At most a 6:1 aspect ratio filter is used
3h	RATIO 8:1	At most a 8:1 aspect ratio filter is used
4h	RATIO 10:1	At most a 10:1 aspect ratio filter is used
5h	RATIO 12:1	At most a 12:1 aspect ratio filter is used
6h	RATIO 14:1	At most a 14:1 aspect ratio filter is used
7h	RATIO 16:1	At most a 16:1 aspect ratio filter is used

18 **U Address Mag Filter Rounding Enable**

Format:	Enable

Controls whether the texture address is converted to 16.8 fixed point before being truncated to select to sample. Provides independent control of rounding on one texture address dimension (U/V/R) in either mag or min filter mode.

Programming Notes
Hardware will **not** force rounding enable.
U Address Min Rounding Enable and **U Address Mag Filter Rounding Enable** must be set to the same value if the **Min Mode Filter** and **Mag Mode Filter** are programmed to the same value.

17 **U Address Min Filter Rounding Enable**

Format:	Enable

Controls whether the texture address is converted to 16.8 fixed point before being truncated to select to sample. Provides independent control of rounding on one texture address dimension (U/V/R) in either mag or min filter mode.

Programming Notes
Hardware will **not** force rounding enable.
U Address Min Rounding Enable and **U Address Mag Filter Rounding Enable** must be set to the same value if the **Min Mode Filter** and **Mag Mode Filter** are programmed to the same value.

16 **V Address Mag Filter Rounding Enable**

Format:	Enable

Controls whether the texture address is converted to 16.8 fixed point before being truncated to select to sample. Provides independent control of rounding on one texture address dimension (U/V/R) in either mag or min filter mode.

Programming Notes
Hardware will **not** force rounding enable.
V Address Min Rounding Enable and **V Address Mag Filter Rounding Enable** must be set to the same value if the **Min Mode Filter** and **Mag Mode Filter** are programmed to the same value.

15 **V Address Min Filter Rounding Enable**

Format:	Enable

Controls whether the texture address is converted to 16.8 fixed point before being truncated to select to sample. Provides independent control of rounding on one texture address dimension (U/V/R) in either mag or min filter mode.

Programming Notes
Hardware will **not** force rounding enable.
V Address Min Rounding Enable and **V Address Mag Filter Rounding Enable** must be set to the same value if the **Min Mode Filter** and **Mag Mode Filter** are programmed to the same value.

14 **R Address Mag Filter Rounding Enable**

Format:	Enable

Controls whether the texture address is converted to 16.8 fixed point before being truncated to select to sample. Provides independent control of rounding on one texture address dimension (U/V/R) in either mag or min filter mode.

Programming Notes
Hardware will **not** force rounding enable.
R Address Min Rounding Enable and **R Address Mag Filter Rounding Enable** must be set to the same value if the **Min Mode Filter** and **Mag Mode Filter** are programmed to the same value.

13 **R Address Min Filter Rounding Enable**

Format:	Enable

Controls whether the texture address is converted to 16.8 fixed point before being truncated to select to sample. Provides independent control of rounding on one texture address dimension (U/V/R) in either mag or min filter mode.

Programming Notes
Hardware will **not** force rounding enable.
R Address Min Rounding Enable and **R Address Mag Filter Rounding Enable** must be set to the same value if the **Min Mode Filter** and **Mag Mode Filter** are programmed to the same value.

12:11 **MIP Linear Filter Quality**

This 2-bit field controls the rounding of LOD for MIP Linear Filtering modes (e.g. Trilinear, etc).

Value	Name	Description
0h	FULL QUALITY **[Default]**	No rounding of LOD is done, the full 8-bit quality is used for filtering.

1h	HIGH QUALITY	LOD values which are within 12.5% of an integer LOD value are rounded to that value prior to filtering and filtering effectively becomes the same as MIP Nearest.
2h	MEDIUM QUALITY	LOD values which are within 16.67% of an integer LOD value are rounded to that value prior to filtering and filtering effectively becomes the same as MIP Nearest.
3h	LOW QUALITY	LOD values which are within 25% of an integer LOD value are rounded to that value prior to filtering and filtering effectively becomes the same as MIP Nearest

10 **Non-normalized Coordinate Enable**

Format:	Enable

This field, if enabled, specifies that the input coordinates (U/V/R) are in non-normalized space, where each integer increment is one texel on LOD 0. If disabled, coordinates are normalized, where the range 0 to 1 spans the entire surface.

Programming Notes

The following state must be set as indicated if this field is *enabled*:

- TCX/Y/Z Address Control Mode must be TEXCOORDMODE_CLAMP, TEXCOORDMODE_HALF_BORDER, or TEXCOORDMODE_CLAMP_BORDER.
- Surface Type must be SURFTYPE_2D or SURFTYPE_3D.
- Mag Mode Filter must be MAPFILTER_NEAREST or MAPFILTER_LINEAR.
- Min Mode Filter must be MAPFILTER_NEAREST or MAPFILTER_LINEAR.
- Mip Mode Filter must be MIPFILTER_NONE.
- Min LOD must be 0.
- Max LOD must be 0.
- MIP Count must be 0.
- Surface Min LOD must be 0.
- Texture LOD Bias must be 0.

9 **Reduction Type Enable**

Format:	Enable

This field enables the **Reduction Type** field to modify the behavior of messages based on its setting. If this field is disabled, all messages behave as defined and the **Reduction Type** field is ignored.

8:6 **TCX Address Control Mode**

Format:	**Texture Coordinate Mode**

Controls how the 1st (TCX, aka U) component of input texture coordinates are mapped to texture map addresses - specifically, how coordinates "outside" the texture are handled (wrap/clamp/mirror). The setting of this field is subject to being overridden by the Cube Surface Control Mode field when sampling from a SURFTYPE_CUBE surface.

		Programming Notes When using cube map texture coordinates, each TC component must have the same Address Control Mode. When TEXCOORDMODE_CUBE is not used accessing a cube map, the map's Cube Face Enable field must be programmed to 111111b (all faces enabled). If **Surface Format** is PLANAR*, this field must be set to TEXCOORDMODE_CLAMP.
	5:3	**TCY Address Control Mode** Format: **Texture Coordinate Mode** Controls how the 2nd (TCY, aka V) component of input texture coordinates are mapped to texture map addresses - specifically, how coordinates "outside" the texture are handled (wrap/clamp/mirror). See Address TCX Control Mode above for details **Programming Notes** If this field is set to TEXCOORDMODE_CLAMP_BORDER or TEXCOORDMODE_HALF_BORDER and a 1D surface is sampled, incorrect blending with the border color in the vertical direction may occur.
	2:0	**TCZ Address Control Mode** Format: **Texture Coordinate Mode** Controls how the 3rd (TCZ) component of input texture coordinates are mapped to texture map addresses - specifically, how coordinates "outside" the texture are handled (wrap/clamp/mirror).See Address TCX Control Mode above for details **Programming Notes** TCZ Address Control Mode Cannot use MIRROR_101 mode. MIRROR_101 mode only works for 2D surfaces.

Sampler Message Header

SAMPLER_MSG_HEADER - Sampler Message Header

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

This defines the contents of the Message Header. Message Header the optional first 256-bits of any Sampler Message.

DWord	Bit	Description
0	31:0	**Reserved** Format: MBZ
1	31:0	**Reserved** Format: MBZ
2	31	**Reserved** Format: MBZ
	30	**Reserved** Format: MBZ
	29:24	**Reserved** Format: MBZ
	23	**Pixel Null Mask Enable** Format: ENABLE **Pixel Null Mask Enable** Specifies whether the writeback message includes an extra phase indicating the pixel null mask. Refer to the **Writeback Message** section for details on format. This field must be disabled for ChromaKey and all SIMD32/64 messages. **Programming Notes** If this is set Shader channel select {Red Green Blue} must be set to zero for all corresponding missing color channels. Shader channel select Alpha must be set to one if the alpha channel is missing
	22	**Return Filter Weights for Remaining On-Map Texels** Format: ENABLE When set to 1: Sampler returns filter weights of remaining on-map texels. Filter weights are returned as a 32-bit float in the Red Channel. It can be used in conjunction with Return Filter Weights for Null Texels
	21	**Slot Group Select** **Slot Group Select** This field selects whether slots 7:0 or slots 15:8 are used for bypassed data. Bypassed data only includes the scale factors for CPS LOD Compensation. This field is ignored if **CPS Message LOD Compensation Enable** is disabled. For 8-pixel dispatches, SLOTGRP_0 must be selected on every message. For 16-pixel dispatches,

this field must be set correctly for each SIMD8* message based on which slots are currently being processed. For SIMD16* messages, SLOTGRP_0 must be selected.

Value	Name	Description
0h	SLOTGRP_0 **[Default]**	Choose bypassed data for slots 7:0
1h	SLOTGRP_1	Choose bypassed data for slots 15:8

20 **Reserved**

Access:	RO
Format:	MBZ

19:18 **Reserved**

Format:	MBZ

17:16 **Gather4 Source Channel Select**

Gather4 Source Channel Select: Selects the source channel to be sampled in the gather4* messages. Ignored for other message types.

For gather4*_c messages, this field must be set to 0 (Red channel).

Value	Name
0h	RED **[Default]**
1h	GREEN
2h	BLUE
3h	ALPHA

15 **Alpha Write Channel Mask**

Enables the alpha channel to be written back to the originating thread.

0: Alpha channel is written back.

1: Alpha channel is not written back.

Restrictions for Channel Write Masks:

- A message with all four channels masked is not allowed.
- This field is ignored for the deinterlace message.
- This field must be set to zero for all gather4* messages.
- This field must be set to zero for sample_8x8 in VSA mode.
- For Sample_8x8 messages, Alpha/Blue/Red channels should be always masked (set to 1) and only Green channel is enabled (set to 0).

14 **Blue Write Channel Mask**

Enables the blue channel to be written back to the originating thread.

See Alpha Channel Write Mask for usage restrictions.

13 **Green Write Channel Mask**

Enables the green channel to be written back to the originating thread.

See Alpha Channel Write Mask for usage restrictions.

12 **Red Write Channel Mask**

Enables the red channel to be written back to the originating thread.

See Alpha Channel Write Mask for usage restrictions.

SAMPLER_MSG_HEADER - Sampler Message Header

	11:8	**U Offset** Format: S3 **U Offset:**The u offset from the _aoffimmi modifier on the*sample*or*ld*instruction in DX10. Must be zero if the**Surface Type**is SURFTYPE_CUBE or SURFTYPE_BUFFER. Must be set to zero if _aoffimmi is not specified. Format is S3 2's complement. • This field is ignored for the sample_unorm*, sample_8x8, and deinterlace messages. • This field is ignored if the*offu*parameter is included in the gather4* messages.
	7:4	**V Offset** Format: S3 **V Offset:**The v offset from the _aoffimmi modifier on the*sample*or*ld*instruction in DX10. Must be zero if the**Surface Type**is SURFTYPE_CUBE or SURFTYPE_BUFFER. Must be set to zero if _aoffimmi is not specified. Format is S3 2's complement. • This field is ignored for the sample_unorm*, sample_8x8, and deinterlace messages. • This field is ignored if the*offu*parameter is included in the gather4* messages.
	3:0	**R Offset** Format: S3 **R Offset:**The r offset from the _aoffimmi modifier on the*sample*or*ld*instruction in DX10. Must be zero if the**Surface Type**is SURFTYPE_CUBE or SURFTYPE_BUFFER. Must be set to zero if _aoffimmi is not specified. Format is S3 2's complement. This field is ignored for the sample_unorm*, sample_8x8, and deinterlace messages. Texel offsets can only be applied to messages with floating-point normalized coordinates or integer non-normalized coordinates.
3	31:4	**Sampler State Pointer** **Sampler State Pointer:**Specifies the 16-byte aligned pointer to the sampler state table. This field is ignored for*ld*and*resinfo*message types. This pointer is relative to the**Dynamic State Base Address**or**Bindless Sampler State Base Address**depending on the setting of**Sampler State Base Address Select**field below. Format = StateOffset[31:4] The Sampler State Pointer does not have to be defined by the Message Header (many messages do not require a message header). The Sampler State Pointer may be delivered from the Command Streamer without the need for a Message Header.
	3:1	**Reserved** Format: MBZ
	0	**Sampler State Base Address Select** **Sampler State Base Address Select:**Selects which base address is used for sampler state accesses. The Sampler State Base Address Select does not have to be defined by the Message Header (many messages do not require a message header). The Sampler State Base Address Select may be delivered from the Command Streamer without the need for a Message Header.

SAMPLER_MSG_HEADER - Sampler Message Header

		Value / **Name** / **Description** 0h / SAMP_DYNAMIC **[Default]** / Use Dynamic State Base Address 1h / SAMP_BINDLESS / Use Bindless Sampler State Base Address
4	31:0	**Reserved** Format: MBZ
5	31:0	**Reserved** Format: MBZ
6	31:0	**Reserved**
7	31:0	**Reserved**

SCALER_COEFFICIENT_FORMAT

SCALER_COEFFICIENT_FORMAT

Size (in bits): 16

Default Value: 0x00000000

Scaler coefficients are stored in sign-exponent-mantissa format. Two coefficients are stored in each dword, the table below show the data packing in each dword.

DWord	Bit	Description
0	15	**Sign** Value: 0b — Name: Positive Value: 1b — Name: Negative
	14	**Reserved** Access: RO Format: MBZ
	13:12	**Exponent** All the tap coefficients use 2 bits of exponent. 00b — 2 — x2 or mantissa is b.bbbbbbbb 01b — 1 — x1 or mantissa is 0.bbbbbbb.. 10b — 0.5 — x0.5 or mantissa is 0.0bbbbbbbb.. 11b — 0.25 — x0.25 or mantissa is 0.00bbbbbbb.. Others — Reserved — Reserved
	11:3	**Mantissa** All the tap coefficients use all 9 bits of mantissa.
	2:0	**Reserved** Access: RO Format: MBZ

SCISSOR_RECT

SCISSOR_RECT

Source: RenderCS

Size (in bits): 64

Default Value: 0x00000000, 0x00000000

Restriction
When executed in the POCS command stream, this command programs the scissor state for the SFR stage of the POCS pipeline

DWord	Bit	Description
0	31:16	**Scissor Rectangle Y Min** Format: U16 Specifies Y Min coordinate of (inclusive) Scissor Rectangle used for scissor test. Pixels with (Draw Rectangle-relative) Y coordinates less than Y Min will be clipped out if Scissor Rectangle is enabled. NOTE: If Y Min is set to a value greater than Y Max, all primitives will be discarded for this viewport. Value: [0,16383] — Name:
	15:0	**Scissor Rectangle X Min** Format: U16 Specifies X Min coordinate of (inclusive) Scissor Rectangle used for scissor test. Pixels with (Draw Rectangle-relative) X coordinates less than X Min will be clipped out if Scissor Rectangle is enabled. NOTE: If X Min is set to a value greater than X Max, all primitives will be discarded for this viewport. Value: [0,16383] — Name:
1	31:16	**Scissor Rectangle Y Max** Format: U16 Specifies Y Max coordinate of (inclusive) Scissor Rectangle used for scissor test. Pixels with (Draw Rectangle-relative) Y coordinates greater than Y Max will be clipped out if Scissor Rectangle is enabled. Value: [0,16383] — Name:
	15:0	**Scissor Rectangle X Max** Format: U16 Specifies X Max coordinate of (inclusive) Scissor Rectangle used for scissor test. Pixels with (Draw Rectangle-relative) Y coordinates greater than X Max will be clipped out if Scissor Rectangle is enabled. Value: 0-16383 — Name:

Scratch Hword Block Message Header

MH_A32_HWB - Scratch Hword Block Message Header

Source: EuSubFunctionDataPort0

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0..2	95:0	**Reserved** Access: RO Format: MBZ
3	31:0	**Per Thread Scratch Space** Format: MHC_PTSS Specifies amount of scratch space used by this thread, for Stateless bounds checking.
4	31:0	**Reserved** Access: RO Format: MBZ
5	31:0	**Buffer Base Address** Format: MHC_A32_BBA Specifies the surface address offset page [31:10] for A32 stateless messages.
6..7	63:0	**Reserved** Access: RO Format: MBZ

SF_CLIP_VIEWPORT

SF_CLIP_VIEWPORT

Source: RenderCS

Size (in bits): 512

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

Restriction

When executed in the POCS command stream, this command programs the viewport state for the CLR and SFR stage of the POCS pipeline.

DWord	Bit	Description
0	31:0	**Viewport Matrix Element m00** Format: IEEE_FLOAT
1	31:0	**Viewport Matrix Element m11** Format: IEEE_FLOAT
2	31:0	**Viewport Matrix Element m22** Format: IEEE_FLOAT
3	31:0	**Viewport Matrix Element m30** Format: IEEE_FLOAT
4	31:0	**Viewport Matrix Element m31** Format: IEEE_FLOAT
5	31:0	**Viewport Matrix Element m32** Format: IEEE_FLOAT
6	31:0	**Reserved** Access: RO Format: MBZ
7	31:0	**Reserved** Access: RO Format: MBZ
8	31:0	**X Min Clip Guardband** Format: IEEE_FLOAT . This 32-bit float represents the XMin guardband boundary (normalized to Viewport.XMin == -1.0f). This corresponds to the left boundary of the NDC guardband.

SF_CLIP_VIEWPORT

9	31:0	**X Max Clip Guardband** Format: IEEE_FLOAT This 32-bit float represents the XMax guardband boundary (normalized to Viewport..XMax == 1.0f). This corresponds to the right boundary of the NDC guardband.
10	31:0	**Y Min Clip Guardband** Format: IEEE_FLOAT This 32-bit float represents the YMin guardband boundary (normalized to Viewport.YMin == -1.0f). This corresponds to the bottom boundary of the NDC guardband.
11	31:0	**Y Max Clip Guardband** Format: IEEE_FLOAT This 32-bit float represents the YMax guardband boundary (normalized to Viewport.YMax == 1.0f). This corresponds to the top boundary of the NDC guardband.
12	31:0	**X Min ViewPort** Format: IEEE_FLOAT This 32-bit float represents the Viewport.XMin. This is the X min of the viewport extents as programmed by API, and this value should be programmed in Screen Space coordinate and not as normalized coordinate.
13	31:0	**X Max ViewPort** Format: IEEE_FLOAT This 32-bit float represents the Viewport.XMax. This is the X max of the viewport extents as programmed by API, and this value should be programmed in Screen Space coordinate and not as normalized coordinate.
14	31:0	**Y Min ViewPort** Format: IEEE_FLOAT This 32-bit float represents the Viewport.YMin. This is the Y min of the viewport extents as programmed by API, and this value should be programmed in Screen Space coordinate and not as normalized coordinate.
15	31:0	**Y Max ViewPort** Format: IEEE_FLOAT This 32-bit float represents the Viewport.Ymax. This is the Y max of the viewport extents as programmed by API, and this value should be programmed in Screen Space coordinate and not as normalized coordinate.

SF_OUTPUT_ATTRIBUTE_DETAIL

SF_OUTPUT_ATTRIBUTE_DETAIL

Source: RenderCS
Size (in bits): 16
Default Value: 0x00000000

DWord	Bit	Description
0	15	**Component Override W** Format: Enable If set, the W component of this output Attribute is overridden by the W component of the constant vector specified by ConstantSource.
	14	**Component Override Z** Format: Enable If set, the Z component of this output Attribute is overridden by the Z component of the constant vector specified by ConstantSource.
	13	**Component Override Y** Format: Enable If set, the Y component of output Attribute is overridden by the Y component of the constant vector specified by ConstantSource.
	12	**Component Override X** Format: Enable If set, the X component of output Attribute is overridden by the X component of the constant vector specified by ConstantSource.
	11	**Swizzle Control Mode** Format: U1 When Attribute Swizzle Enable is ENABLED, this bit controls whether attributes 0-15 or 16-31 are subject to the following swizzle controls: • Component Override X/Y/Z/W • Constant Source • Swizzle Select • Source Attribute • WrapShortest Enables Note that the Number of SF Output Attributes field specifies how many attributes are output. Note: This field does not impact any functions which provide separate states for all 32 attributes (e.g., Point sprite, Constant interpolation). Note: This field is only valid for the first indexed attribute (Attribute[0]). For all other indices, it is Reserved and MBZ.

SF_OUTPUT_ATTRIBUTE_DETAIL

10:9 Constant Source

Format:	U2

This state selects a constant vector which can be used to override individual components of this Attribute

Value	Name	Description
0h	CONST_0000	Constant.xyzw = 0.0,0.0,0.0,0.0
1h	CONST_0001_FLOAT	Constant.xyzw = 0.0,0.0,0.0,1.0
2h	CONST_1111_FLOAT	Constant.xyzw = 1.0,1.0,1.0,1.0
3h	PRIM_ID	Constant.xyzw = PrimID (replicated)

8 Reserved

Access:	RO
Format:	MBZ

7:6 Swizzle Select

Format:	U2

This state, along with Source Attribute, specifies the source for this output Attribute.

Value	Name	Description
0h	INPUTATTR	This attribute is sourced from AttrInputReg[SourceAttribute]
1h	INPUTATTR_FACING	If the object is front-facing, this attribute is sourced from AttrInputReg[SourceAttribute]. If the object is back-facing, this attribute is sourced from AttrInputReg[SourceAttribute+1].
2h	INPUTATTR_W	This attribute is sourced from AttrInputReg[SourceAttribute]. The W component is copied to the X component.
3h	INPUTATTR_FACING_W	If the object is front-facing, this attribute is sourced from AttrInputReg[SourceAttribute]. If the object is back-facing, this attribute is sourced from AttrInputReg[SourceAttribute+1]. The W component is copied to the X component.

5 Reserved

Access:	RO
Format:	MBZ

4:0 Source Attribute

Format:	U5

This field selects the source attribute for this Attribute. Source attribute 0 corresponds to the first 128 bits of data indicated by Vertex URB Entry Read Offset

SFC_8x8_AVS_COEFFICIENTS

SFC_8x8_AVS_COEFFICIENTS		
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
ExistsIf = AVS && (Function_mode = 0)		
DWord	**Bit**	**Description**
0	31:24	**ZeroYFilterCoefficient1** Format: S1.6 **Range:** [-2, +2)
	23:16	**ZeroXFilterCoefficient1** Format: S1.6 **Range:** [-2, +2)
	15:8	**ZeroYFilterCoefficient0** Format: S1.6 **Range:** [-2, +2)
	7:0	**ZeroXFilterCoefficient0** Format: S1.6 **Range:** [-2, +2)
1	31:24	**ZeroYFilterCoefficient3** Format: S1.6 **Range:** [-2, +2)
	23:16	**ZeroXFilterCoefficient3** Format: S1.6 **Range:** [-2, +2)
	15:8	**ZeroYFilterCoefficient2** Format: S1.6 **Range:** [-2, +2)
	7:0	**ZeroXFilterCoefficient2** Format: S1.6 **Range:** [-2, +2)

SFC_8x8_AVS_COEFFICIENTS

2	31:24	**ZeroYFilterCoefficient5** Format: S1.6 **Range:** [-2, +2)
	23:16	**ZeroXFilterCoefficient5** Format: S1.6 **Range:** [-2, +2)
	15:8	**ZeroYFilterCoefficient4** Format: S1.6 **Range:** [-2, +2)
	7:0	**ZeroXFilterCoefficient4** Format: S1.6 **Range:** [-2, +2)
3	31:24	**ZeroYFilterCoefficient7** Format: S1.6 **Range:** [-2, +2)
	23:16	**ZeroXFilterCoefficient7** Format: S1.6 **Range:** [-2, +2)
	15:8	**ZeroYFilterCoefficient6** Format: S1.6 **Range:** [-2, +2)
	7:0	**ZeroXFilterCoefficient6** Format: S1.6 **Range:** [-2, +2)
4	31:24	**OneXFilterCoefficient3** Format: S1.6 **Range:** [-2.0, +2.0)
	23:16	**OneXFilterCoefficient2** Format: S1.6 **Range:** [-1.0, +1.0)

SFC_8x8_AVS_COEFFICIENTS		
	15:0	**Reserved** Access: RO Format: MBZ
5	31:16	**Reserved** Access: RO Format: MBZ
	15:8	**OneXFilterCoefficient5** Format: S1.6 **Range:** [-1.0, +1.0)
	7:0	**OneXFilterCoefficient4** Format: S1.6 **Range:** [-2.0, +2.0)
6	31:24	**OneYFilterCoefficient3** Format: S1.6 **Range:** [-2.0, +2.0)
	23:16	**OneYFilterCoefficient2** Format: S1.6 **Range:** [-1.0, +1.0)
	15:0	**Reserved** Access: RO Format: MBZ
7	31:16	**Reserved** Access: RO Format: MBZ
	15:8	**OneYFilterCoefficient5** Format: S1.6 **Range:** [-1.0, +1.0)
	7:0	**OneYFilterCoefficient4** Format: S1.6 **Range:** [-2.0, +2.0)

SFC_AVS_CHROMA_COEFF_TABLE_BODY

SFC_AVS_CHROMA_COEFF_TABLE_BODY		
Size (in bits):	64	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0..1	63:56	**Table 1Y Filter Coefficient[[n],5]** Format: S1.6 **Range:** [-2, +2) Chroma table for Y-direction. **Programming Notes** Filter tap index3 in U/V 4-tap filtering
	55:48	**Table 1X Filter Coefficient[[n],5]** Format: S1.6 **Range:** [-2, +2) Chroma table for X-direction. **Programming Notes** Filter tap index3 in U/V 4-tap filtering
	47:40	**Table 1Y Filter Coefficient[[n],4]** Format: S1.6 **Range:** [-2, +2) Chroma table for Y-direction. **Programming Notes** Filter tap index 2 in U/V 4-tap filtering
	39:32	**Table 1X Filter Coefficient[[n],4]** Format: S1.6 **Range:** [-2, +2) Chroma table for X-direction. **Programming Notes** Filter tap index 2 in U/V 4-tap filtering

SFC_AVS_CHROMA_COEFF_TABLE_BODY

	31:24	**Table 1Y Filter Coefficient[[n],3]** Format: S1.6 **Range:** [-2, +2) Chroma table for Y-direction. **Programming Notes** Filter tap index1 in U/V 4-tap filtering
	23:16	**Table 1X Filter Coefficient[[n],3]** Format: S1.6 **Range:** [-2, +2) Chroma table for X-direction. **Programming Notes** Filter tap index1 in U/V 4-tap filtering
	15:8	**Table 1Y Filter Coefficient[[n],2]** Format: S1.6 **Range:** [-2, +2) Chroma table for Y-direction. **Programming Notes** Filter tap index0 in U/V 4-tap filtering
	7:0	**Table 1X Filter Coefficient[[n],2]** Format: S1.6 **Range:** [-2, +2) Chroma table for X-direction. **Programming Notes** Filter tap index0 in U/V 4-tap filtering

SFC_AVS_LUMA_COEFF_TABLE_BODY

SFC_AVS_LUMA_COEFF_TABLE_BODY		
Size (in bits):	128	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0..3	127:120	**Table 0Y Filter Coefficient[[n],7]** Format: S1.6 **Range:** [-2, +2) Luma table for Y-direction. **Programming Notes** For 5x5 filter mode, table 0, 6 and 7 must be programmed to zero.
	119:112	**Table 0X Filter Coefficient[[n],7]** Format: S1.6 **Range:** [-2, +2) Luma table for X-direction. **Programming Notes** For 5x5 filter mode, table 0, 6 and 7 must be programmed to zero.
	111:104	**Table 0Y Filter Coefficient[[n],6]** Format: S1.6 **Range:** [-2, +2) Luma table for Y-direction. **Programming Notes** For 5x5 filter mode, table 0, 6 and 7 must be programmed to zero.
	103:96	**Table 0X Filter Coefficient[[n],6]** Format: S1.6 **Range:** [-2, +2) Luma table for X-direction. **Programming Notes** For 5x5 filter mode, table 0, 6 and 7 must be programmed to zero.

SFC_AVS_LUMA_COEFF_TABLE_BODY

	95:88	**Table 0Y Filter Coefficient[[n],5]** Format: S1.6 **Range:** [-2, +2) Luma table for Y-direction. **Programming Notes** For 5x5 filter mode, table 0, 6 and 7 must be programmed to zero.
	87:80	**Table 0X Filter Coefficient[[n],5]** Format: S1.6 **Range:** [-2, +2) Luma table for X-direction. **Programming Notes** For 5x5 filter mode, table 0, 6 and 7 must be programmed to zero.
	79:72	**Table 0Y Filter Coefficient[[n],4]** Format: S1.6 **Range:** [-2, +2) Luma table for Y-direction. **Programming Notes** For 5x5 filter mode, table 0, 6 and 7 must be programmed to zero.
	71:64	**Table 0X Filter Coefficient[[n],4]** Format: S1.6 **Range:** [-2, +2) Luma table for X-direction. **Programming Notes** For 5x5 filter mode, table 0, 6 and 7 must be programmed to zero.
	63:56	**Table 0Y Filter Coefficient[[n],3]** Format: S1.6 **Range:** [-2, +2) Luma table for Y-direction.

SFC_AVS_LUMA_COEFF_TABLE_BODY

		Programming Notes For 5x5 filter mode, table 0, 6 and 7 must be programmed to zero.
	55:48	**Table 0X Filter Coefficient[[n],3]** Format: S1.6 **Range:** [-2, +2) Luma table for X-direction. **Programming Notes** For 5x5 filter mode, table 0, 6 and 7 must be programmed to zero.
	47:40	**Table 0Y Filter Coefficient[[n],2]** Format: S1.6 **Range:** [-2, +2) Luma table for Y-direction. **Programming Notes** For 5x5 filter mode, table 0, 6 and 7 must be programmed to zero.
	39:32	**Table 0X Filter Coefficient[[n],2]** Format: S1.6 **Range:** [-2, +2) Luma table for X-direction. **Programming Notes** For 5x5 filter mode, table 0, 6 and 7 must be programmed to zero.
	31:24	**Table 0Y Filter Coefficient[[n],1]** Format: S1.6 **Range:** [-2, +2) Luma table for Y-direction. **Programming Notes** For 5x5 filter mode, table 0, 6 and 7 must be programmed to zero.

SFC_AVS_LUMA_COEFF_TABLE_BODY

	23:16	**Table 0X Filter Coefficient[[n],1]** Format: S1.6 **Range:** [-2, +2) Luma table for X-direction. **Programming Notes** For 5x5 filter mode, table 0, 6 and 7 must be programmed to zero.
	15:8	**Table 0Y Filter Coefficient[[n],0]** Format: S1.6 **Range:** [-2, +2) Luma table for Y-direction. **Programming Notes** For 5x5 filter mode, table 0, 6 and 7 must be programmed to zero.
	7:0	**Table 0X Filter Coefficient[[n],0]** Format: S1.6 **Range:** [-2, +2) Luma table for X-direction. **Programming Notes** For 5x5 filter mode, table 0, 6 and 7 must be programmed to zero.

SFC_AVS_STATE_BODY

SFC_AVS_STATE_BODY

Size (in bits): 96

Default Value: 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:24	**Sharpness Level** Format: U8 When adaptive scaling is off, determines the balance between sharp and smooth scalers. Value / Name / Description: 0 — Contribute 1 from the smooth scalar 255 — Contribute 1 from the sharp scalar
	23:7	**Reserved** Access: RO Format: MBZ
	6:4	**Transition Area with 4 Pixels** Format: U3 Used in adaptive filtering to specify the width of the transition area for the 4 pixel calculation.
	3	**Reserved** Access: RO Format: MBZ
	2:0	**Transition Area with 8 Pixels** Format: U3 Used in adaptive filtering to specify the width of the transition area for the 8 pixel calculation.
1	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**Max Derivative 4 Pixels** Format: U8 Used in adaptive filtering to specify the lower boundary of the smooth 4 pixel area.
	15:8	**Reserved** Access: RO Format: MBZ

SFC_AVS_STATE_BODY

	7:0	**MAX Derivative Point 8** Format: U8 Used in adaptive filtering to specify the lower boundary of the smooth 8 pixel area.
2	31:13	**Reserved** Access: RO Format: MBZ
	12	**Reserved** Access: RO Format: MBZ
	11:8	**Input Horizontal Siting Value - Specifies the horizontal siting of the input**

Value	Name
0000b	0(fraction in integer)
0001b	1/8
0010b	2/8
0011b	3/8
0100b	4/8
0101b	5/8
0110b	6/8
0111b	7/8
1000b	8/8

Programming Notes
For 444 format, horizontal chroma siting should be programmed to zero.

	7:4	**Reserved** Access: RO Format: MBZ
	3:0	**Input Vertical Siting - Specifies the vertical siting of the input**

Value	Name
000b	0
0001b	1/8
0010b	2/8
0011b	3/8
0100b	4/8
0101b	5/8
0110b	6/8

SFC_AVS_STATE_BODY

		0111b	7/8
		1000b	8/8

Programming Notes
For 444 and 422 format, vertical chroma siting should be programmed to zero.

SFC_FRAME_START_BODY

SFC_FRAME_START_BODY		
Size (in bits): 32 Default Value: 0x00000000		
DWord	**Bit**	**Description**
0	31:0	**Reserved** Access: RO Format: MBZ

SFC_HDR_STATE

SFC_HDR_STATE

Size (in bits): 96

Default Value: 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0..1	63:48	**Reserved** Access: RO Format: MBZ
	47:12	**Address** Format: GraphicsAddress[47:12] Specifies the graphics base address used to fetch SFC_EOTF_OETF_STATE surfacetable into SFC.
	11:0	**Reserved** Access: RO Format: MBZ
2	31:15	**Reserved** Access: RO Format: MBZ
	14:13	**Surface Tiled Mode** Format: U2 **For Media Surfaces:** This field specifies the tiled resource mode. 0 TRMODE_NONE — No tiled resources 1 TRMODE_TILEYF — 4KB tiled resources 2 TRMODE_TILEYS — 64KB tiled resources 3 Reserved
	12	**Reserved** Access: RO Format: MBZ
	11	**Scratch Buffer Cache Select** Default Value: 0 Disable Format: U1 **Programming Notes** This must be set to 0

SFC_HDR_STATE

	10	**Compression Type** Default Value: 0 Disable Format: boolean This field is applicable only when Memory compression is enabled.As memory compression is not supported on this surface, it must be 0.
	9	**Memory Compression Enable** Default Value: 0 Disable Format: Enable **Programming Notes** Memory compression is not supported for this surface Must be 0.
	8:7	**Arbitration Priority Control** Format: HEVC_ARBITRATION_PRIORITY
	6:1	**Index to Memory Object Control State (MOCS) Tables** Format: U6 The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**

SFC_HISTOGRAM_SURFACE

SFC_HIST_SURF - SFC_HISTOGRAM_SURFACE	
Size (in bits):	160
Default Value:	0x00000000, 0x00000000,

SFC_HIST_SURF - SFC_HISTOGRAM_SURFACE

0x00000000, 0x00000000, 0x00000000, 0x00000000

SFC would output histogram to a surface with 256-bins when histogram streamout is enabled in SFC_STATE. This represent the write out surface structure.

DWord	Bit	Description
0..3	127:96	**HistogramBinLumaY[3]** Format: U32
	95:64	**HistogramBinLumaY[2]** Format: U32
	63:32	**HistogramBinLumaY[1]** Format: U32
	31:0	**HistogramBinLumaY[0]** Format: U32
4..255	31:0	**HistogramBinLumaY[4..255]**

SFC_IEF_STATE_BODY

SFC_IEF_STATE_BODY		
Size (in bits):	736	
Default Value:	0x0294806C, 0x39CFD1FF, 0x039F0000, 0x9A6E4000, 0x00601180, 0xFFFE2F2E, 0x00000000, 0xD82E0000, 0x8285ECEC, 0x00008282, 0x00000000, 0x02117000, 0xA38FEC96, 0x00008CC8, 0x00000000, 0x00002000, 0x00000000, 0x00000400, 0x00000000, 0x00000400, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0	31:28	**Reserved** Access: RO Format: MBZ
	27:23	**R3c Coefficient** Default Value: 5 Format: U0.5 IEF smoothing coefficient, *see IEF map*.
	22:18	**R3x Coefficient** Default Value: 5 Format: U0.5 IEF smoothing coefficient, *see IEF map*.
	17:12	**Strong Edge Threshold** Default Value: 8 Format: U6 If EM > Strong Edge Threshold the basic VSA detects a strong edge.
	11:6	**Weak Edge Threshold** Default Value: 1 Format: U6 If Strong Edge Threshold > EM > Weak Edge Threshold the basic VSA detects a weak edge.
	5:0	**Gain Factor** Default Value: 44 Format: U6 User control sharpening strength.

SFC_IEF_STATE_BODY

1	31:27	**R5c Coefficient** Default Value: 7 Format: U0.5 IEF smoothing coefficient, *see IEF map*.
	26:22	**R5cx Coefficient** Default Value: 7 Format: U0.5 IEF smoothing coefficient, *see IEF map*.
	21:17	**R5x Coefficient** Default Value: 7 Format: U0.5 IEF smoothing coefficient, *see IEF map*.
	16:14	**Strong Edge Weight** Default Value: 7 Format: U3 Sharpening strength when a STRONG edge is found in basic VSA.
	13:11	**Regular Weight** Default Value: 2 Format: U3 Sharpening strength when a WEAK edge is found in basic VSA.
	10:8	**Non Edge Weight** Default Value: 1 Format: U3 . Sharpening strength when NO EDGE is found in basic VSA.
	7:0	**Global Noise Estimation** Default Value: 255 Format: U8 Global noise estimation of previous frame.
2	31:28	**Reserved** Access: RO Format: MBZ

SFC_IEF_STATE_BODY

	27:22	**Hue_Max** Default Value: 14 Format: U6 Rectangle half width.
	21:16	**Sat_Max** Default Value: 31 Format: U6 Rectangle half length.
	15:8	**STD Cos(alpha)** Format: S0.7 **Default Value =** 79/128
	7:0	**STD Sin(alpha)** Format: S0.7 **Default Value =** 101/128
3	31:24	**V_Mid** Default Value: 154 Format: U8 Rectangle middle-point V coordinate.
	23:16	**U_Mid** Default Value: 110 Format: U8 Rectangle middle-point U coordinate.
	15	**VY_STD_Enable** Format: Enable Enables STD in the VY subspace.
	14:12	**Diamond Margin** Default Value: 4 Format: U3
	11	**Reserved** Access: RO Format: MBZ

SFC_IEF_STATE_BODY

	10:0	**S3U** Format: S2.8 Slope 3 of the upper part of the detection PWLF. **Deafult:** 0/256
4	31	**Skin Detail Factor** Format: U1 This flag bit is in operation only when one of the following conditions exists: • when the control bit **SkinToneTunedIEF_Enable** is on. • When **SkinDetailFactor** is equal to 0, sign(**SkinDetailFactor**) is equal to +1, and the content of the detected skin tone area is detail revealed.When **SkinDetailFactor** is equal to 1, sign(**SkinDetailFactor**) is equal to -1, and the content of the detected skin tone area is not detail revealed. **Value** / **Name** 0 – Detail Revealed **[Default]** 1 – Not Detail Revealed
	30:24	**Diamond_du** Default Value: 0 Format: S6 Rhombus center shift in the sat-direction, relative to the rectangle center.
	23:21	**HS_margin** Default Value: 3 Format: U3 Defines rectangle margin.
	20:13	**Diamond_alpha** Format: U2.6 1 / tan() **Deafult:** 100/64
	12:7	**Diamond_Th** Default Value: 35 Format: U6 Half length of the rhombus axis in the sat-direction.

SFC_IEF_STATE_BODY

	6:0	**Diamond_dv** Default Value: 0 Format: S6 Rhombus center shift in the hue-direction, relative to the rectangle center.
5	31:24	**Y_point_4** Default Value: 255 Format: U8 Fourth point of the Y piecewise linear membership function.
	23:16	**Y_point_3** Default Value: 254 Format: U8 Third point of the Y piecewise linear membership function.
	15:8	**Y_point_2** Default Value: 47 Format: U8 Second point of the Y piecewise linear membership function.
	7:0	**Y_point_1** Default Value: 46 Format: U8 First point of the Y piecewise linear membership function.
6	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**INV_Margin_VYL** Format: U0.16 1 / Margin_VYL **Default:** 3300/65536
7	31:24	**P1L** Default Value: 216 Format: U8 Y Point 1 of the lower part of the detection PWLF.

SFC_IEF_STATE_BODY

	23:16	**P0L** Default Value: 46 Format: U8 Y Point 0 of the lower part of the detection PWLF.
	15:0	**INV_Margin_VYU** Format: U0.16 1 / Margin_VYL **Default:** 1600/65536
8	31:24	**B1L** Default Value: 130 Format: U8 V Bias 1 of the lower part of the detection PWLF.
	23:16	**B0L** Default Value: 133 Format: U8 V Bias 0 of the lower part of the detection PWLF.
	15:8	**P3L** Default Value: 236 Format: U8 Y Point 3 of the lower part of the detection PWLF.
	7:0	**P2L** Default Value: 236 Format: U8 Y Point 2 of the lower part of the detection PWLF.
9	31:27	**Y_Slope_2** Format: U2.3 Slope between points Y3 and Y4. **Default:** 31/8

SFC_IEF_STATE_BODY

	26:16	**S0L** Format: S2.8 Slope 0 of the lower part of the detection PWLF. **Default:** -5/256
	15:8	**B3L** Default Value: 130 Format: U8 V Bias 3 of the lower part of the detection PWLF.
	7:0	**B2L** Default Value: 130 Format: U8 V Bias 2 of the lower part of the detection PWLF.
10	31:22	**Reserved** Access: RO Format: MBZ
	21:11	**S2L** Format: S2.8 **Default:** 0/256 Slope 2 of the lower part of the detection PWLF.
	10:0	**S1L** Format: S2.8 **Default:** 0/256 Slope 1 of the lower part of the detection PWLF.
11	31:27	**Y_Slope1** Format: U2.3 Slope between points Y1 and Y2. **Default:** 31/8
	26:19	**P1U** Default Value: 66 Format: U8 Y Point 1 of the upper part of the detection PWLF.

SFC_IEF_STATE_BODY

	18:11	**P0U** Default Value: 46 Format: U8 Y Point 0 of the upper part of the detection PWLF.
	10:0	**S3L** Format: S2.8 Slope 3 of the lower part of the detection PWLF. **Default:** 0/256
12	31:24	**B1U** Default Value: 163 Format: U8 V Bias 1 of the upper part of the detection PWLF.
	23:16	**B0U** Default Value: 143 Format: U8 V Bias 0 of the upper part of the detection PWLF.
	15:8	**P3U** Default Value: 236 Format: U8 Y Point 3 of the upper part of the detection PWLF.
	7:0	**P2U** Default Value: 150 Format: U8 Y Point 2 of the upper part of the detection PWLF.
13	31:27	**Reserved** Access: RO Format: MBZ
	26:16	**S0U** Format: S2.8 Slope 0 of the upper part of the detection PWLF. **Default:** 256/256

SFC_IEF_STATE_BODY

	15:8	**B3U** Default Value: 140 Format: U8 V Bias 3 of the upper part of the detection PWLF.
	7:0	**B2U** Default Value: 200 Format: U8 V Bias 2 of the upper part of the detection PWLF.
14	31:22	**Reserved** Access: RO Format: MBZ
	21:11	**S2U** Format: S2.8 **Default:** -179/256 Slope 2 of the upper part of the detection PWLF.
	10:0	**S1U** Format: S2.8 **Default:** 113/256 Slope 1 of the upper part of the detection PWLF.
15	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**C1** Default Value: 0 Format: S2.10 Transform coefficient
	15:3	**C0** Default Value: 1024 Format: S2.10 Transform coefficient

SFC_IEF_STATE_BODY

	2	**Reserved** Access: RO Format: MBZ
	1	**YUV Channel Swap**
	0	**Transform Enable**
16	31:26	**Reserved** Access: RO Format: MBZ
	25:13	**C3** Default Value: 0 Format: S2.10 Transform coefficient
	12:0	**C2** Default Value: 0 Format: S2.10 Transform coefficient
17	31:26	**Reserved** Access: RO Format: MBZ
	25:13	**C5** Default Value: 0 Format: S2.10 Transform coefficient
	12:0	**C4** Default Value: 1024 Format: S2.10 Transform coefficient
18	31:26	**Reserved** Access: RO Format: MBZ
	25:13	**C7** Default Value: 0 Format: S2.10 Transform coefficient

SFC_IEF_STATE_BODY

	12:0	**C6** Default Value: 0 Format: S2.10 Transform coefficient
19	31:13	**Reserved** Access: RO Format: MBZ
	12:0	**C8** Default Value: 1024 Format: S2.10 Transform coefficient
20	31:22	**Reserved** Access: RO Format: MBZ
	21:11	**Offset out 1** Default Value: 0 Format: S2.8 Offset out for Y/R.
	10:0	**Offset in 1** Default Value: 0 Format: S2.8 Offset in for Y/R.
21	31:22	**Reserved** Access: RO Format: MBZ
	21:11	**Offset out 2** Default Value: 0 Format: S2.8 Offset out for U/G.
	10:0	**Offset in 2** Default Value: 0 Format: S2.8 Offset in for U/G.

SFC_IEF_STATE_BODY		
22	31:22	**Reserved** Access: RO Format: MBZ
	21:11	**Offset out 3** Default Value: 0 Format: S2.8 Offset out for V/B.
	10:0	**Offset in 3** Default Value: 0 Format: S2.8 Offset in for V/B.

SFC_LOCK_BODY

SFC_LOCK_BODY

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:3	**Reserved** Access: RO Format: MBZ
	2	**Reserved** Access: RO Format: MBZ
	1	**Pre-Scaled Output Surface Output Enable** **VD - Reconstructed Pixel Output Enable** For VD Mode, this field specifies the enabling of writing out the display reconstructed pixel to memory. It could be pre or post- ILDB filter pixel output based on the pre- and post- filter setting in the AVC state command.

Pre-Deblock Flag	Post-Deblock Flag	VD Pixels Output to Memory	VD Pixels Output to SFC
0	0	Invalid for SFC Mode	Invalid for SFC Mode
0	1	Filtered Pixels (allow ON/OFF)	Filter Pixels Sent to SFC for Scaling
1	0	Non-filter (bypass) pixels (allow ON/OFF)	Non-Filter Pixels Sent to SFC for Scaling
1	1	Non-filter (bypass) pixels (allow ON/OFF)Filtered pixels (always OFF)	Filter Pixels Sent to SFC for Scaling.

VE - image enhanced pixel Output Enable For VE Mode, this field indicates if the VEBOX will enable writing out the image enhanced pixels to memory which is streamed to SFC pipeline for scaling.Filtered data is streamed directly from VEBOX to SFC through a dedicated internal interface. The pixel data send from VE to SFC is YUV format in 12-bit precision irrespective of VEBOX input surface type, pixel precision, chroma format, and color format (RGBA/YUVA). The following table shows allowed usage with VE -image enhanced pixel output enable along with SFC being enabled.

VE Output Surface Format	Bits per channel	Can SFC be enabled ?
Y8/ NV12/ AYUV/ YUYV/YVYU/UYVY/VYUY	8bit	Yes
RGBA8	8bit	No
RGBA10	10bit	No
RGBA16	16bit	No
Y16/ P216/P016/ Y216/ Y416	16bit	Yes

SFC_LOCK_BODY		
		When DN is enabled, DN output is always on.When DI is Enabled, VE must send the first DI constructed surface to SFC in case VEBOX state indicate two DI frames output, while stream out the second DI constructed surface to memory.Else the DI output which is enabled will be sent out to SFC. VE output surface Type is programmed in VE_State command.
	0	**VE-SFC Pipe Select**

SFC_STATE_BODY

SFC_STATE_BODY

Size (in bits): 1568

Default Value: 0x00000000, 0x00001000, 0x00000000

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23:18	**Reserved** Access: RO Format: MBZ
	17:14	**Reserved** Access: RO Format: MBZ
	13:12	**SFC Engine Mode** Format: U2 Value / Name / Description: 00b — Single SFC mode. 01b — Left Most SFC in scalability/split-frame mode. 10b — Right Most SFC in scalability/split-frame mode. 11b — Middle SFC in scalability/split-frame mode. **Programming Notes** If SFC Pipe Mode is HCP-to-SFC and VE-to-SFC modes .Programmer need to ensure SFC Engine Mode bits programmed is the same as HCP_PIPE_MODE_SELECT command, Multi-Engine Modebits. For VE-SFC mode, this field has to be programmed according to the position of the split This Field is ignored for other SFC Pipe Mode.

SFC_STATE_BODY

		Restriction In case of scalability, the tile width has to be a minimum of 128.
	11	**Reserved** Access: RO Format: MBZ
	10:8	**VD/VE Input Ordering Mode** Format: U3 • VD mode: (SFC pipe mode set as "0") • VE mode: (pipe mode set as "1 and 4") For values for each mode, please refer to the table below: • HCP mode : SFC Pipe Mode set as "2" For values for each mode, please refer to the table below:

Value	Name	Description	Programming Notes	Exists If
0		16x16 block z-scan order - no shift		//VD Mode
1		16x16 block z-scan order - 4 pixels shift upward		//VD Mode
2		8x8 block jpeg z-scan order		//VD Mode
3		16x16 block jpeg z-scan order		//VD Mode
4		16x16 block VP8 row-scan order - no shift		//VD Mode
5-7		Reserved		//VD Mode
0		16x16 block HEVC Decoderrow-scan order -4 pixel shift upward		//HCP Mode
1		32x32block HEVC Decoderrow-scan order -4 pixel shift upward		//HCP Mode
2		64x64 block HEVC Decoder row-scan order -4 pixel shift upward		//HCP Mode
3		64x64 block VP9 Decoderrow-scan order - 8 pixel shift upward		//HCP Mode
[4-7]		Reserved		//HCP Mode

SFC_STATE_BODY

0		8x4 block column order, 64 pixel column		//VE Mode
1		4x4 block column order, 64 pixel column		//VE Mode
[2-7]		Reserved		//VE Mode

Programming Notes

This field shall be programmed according to video modes used in VDBOX. NOTE: SFC supports progressive input and output only (Interlaced/MBAFF is not supported).

Video Mode	Surface Format	SFC Input Chroma Sub-Sampling	VD/VE Input Ordering Mode
VC1 w/o LF and w/o OS Note: VC1 LF applies for either ILDB	420 (NV12)	1	0
VC1 w/ LF or w/ OS or w/ both Note: VC1 LF applies for either ILDB		INVALID with SFC	INVALID with SFC
AVC w/o LF	Monochrome	0	0
AVC w/o LF	420 (NV12)	1	0
AVC with LF	Monochrome	0	1
AVC/VP8 with LF	420 (NV12)	1	1
VP8 w/o LF	420 (NV12)	1	4
JPEG (YUV Interleaved)	Monochrome	0	2
JPEG (YUV Interleaved)	420	1	3
JPEG (YUV Interleaved)	422H_2Y	2	2
JPEG (YUV Interleaved)	422H_4Y	2	3
JPEG (YUV Interleaved)	444	4	2
JPEG (YUV Interleaved)	411	5	2

This field shall be programmed according to Image enhancement modes used in VEBOX.

VEBOX MODE	VEBOX Single Pipe Enable Bit	SFC Input Surface Format	SFC Input Chroma Sub Sampling	VD/VE Input Ordering Mode
1. DN/HP with RGB input 2. Camera pipe (DM) enabled 3. IECP with FECSC, CCM, FGC filters enabled	1	Monochrome	0	1
	1	420 (NV12)	1	1
	1	422H	2	1
	1	444	4	1
All other modes:	0	Monochrome	0	0

(Legacy DN/DI/IECP features)	0	420 (NV12)	1	0
	0	422H	2	0
	0	444	4	0

This field shall be programmed according to video mode used in HCP. Note: SFC supports progressive input and output only (interlace/mbaff is not supported).

Video Mode	Surface Format	SFC Input Chroma Sub Sampling	VD/VE Input ordering mode
HEVC 16x16 LCU	420/422/444	1 / 2 / 4	0
HEVC 32x32 LCU	420/422/444	1 / 2 / 4	1
HEVC 64x64 LCU	420/422/444	1/ 2 / 4	2
VP9 64x64 LCU	420/444	1 / 4	3 / 4

7:4 **SFC Input Chroma Sub-Sampling**

Value	Name	Description
0	4:0:0	SFC to insert UV channels
1	4:2:0	
2	4:2:2 Horizonatal	VD: 2:1:1
3	Reserved	
4	4:4:4 Progressive/Interleaved	

Programming Notes

This field shall be programmed according to video modes used in VDBOX. NOTE: SFC supports progressive input and output only (Interlaced/MBAFF is not supported).

Video Mode	Surface Format	SFC Input Chroma Sub-Sampling	VD/VE Input Ordering Mode
VC1 w/o LF and w/o OS Note: VC1 LF applies for either ILDB	420 (NV12)	1	0
VC1 w/ LF or w/ OS or w/ both Note: VC1 LF applies for either ILDB		INVALID with SFC	INVALID with SFC
AVC w/o LF	Monochrome	0	0
AVC w/o LF	420 (NV12)	1	0
AVC with LF	Monochrome	0	1
AVC/VP8 with LF	420 (NV12)	1	1
VP8 w/o LF	420 (NV12)	1	4
JPEG (YUV Interleaved)	Monochrome	0	2
JPEG (YUV Interleaved)	420	1	3

SFC_STATE_BODY

JPEG (YUV Interleaved)	422H_2Y	2	2
JPEG (YUV Interleaved)	422H_4Y	2	3
JPEG (YUV Interleaved)	444	4	2

This field shall be programmed according to Image enhancement modes used in VEBOX.

VEBOX MODE	Surface Format	SFC Input Chroma Sub Sampling	VD/VE Input Ordering Mode
Legacy DN/DI/IECP features	Monochrome	0	0
Legacy DN/DI/IECP features	420 (NV12)	1	0
Legacy DN/DI/IECP features	422H	2	0
Legacy DN/DI/IECP features	444	4	0
Capture/Camera pipe enabled(Demosaic)	Monochrome	0	1
Capture/Camera pipe enabled(Demosaic)	420 (NV12)	1	1
Capture/Camera pipe enabled(Demosaic)	422H	2	1
Capture/Camera pipe enabled(Demosaic)	444	4	1

3:0 **SFC Pipe Mode**

Value	Name	Description
0		VD-to-SFC AVS
1		VE-to-SFC AVS + IEF + Rotation
2		HCP-to-SFC AVS
3		Reserved
4		VE-to-SFC Integral Image
5		
[6,15]		Reserved

Programming Notes
Note: for SFC Pipe mode set to VE-to-SFC AVS mode. IECP pipeline mode MUST be enabled. However, each sub-IECP feature can be turned on/off independently.

1 31:30 **Reserved**

Access:	RO
Format:	MBZ

29:16 **Input Frame Resolution Height**

Format:	U14-1

Minus 1 in unit of pixel [13:0]. It is set to the value of the output resolution or number of pixels

SFC_STATE_BODY

		streaming into SFC from VD/HCP or VEBOX. Since the Max value support in 16K pixels,the max value allowed in 16K minus 1. • VDBOX frame height is multiple of 16 for Video source and JPEG formats other than 400, 444 and 422H_2Y. • VDBOX frame height is multiple of 8 for JPEG formats 400, 444 and 422H_2Y. • VEBOX frame height is multiple of 4. • HEVC frame height is multiple of 8 • VP9 frame height is multiple of 8.
		Min Resolution is 32 pixels. Max Resolution is up to 16K pixel eg. for 1920x1080 content, FrameHeightInMBsMinus1 is equal to 1087 (1080 rounded up 16 pixel boundary, minus 1. i.e. effectively specified as 1088 instead).
		Restriction : For Integral Image Mode, this field is Reserved and MBZ.
	15:14	**Reserved** Access: RO Format: MBZ
	13:0	**Input Frame Resolution Width** Format: U14-1 Minus 1 in unit of pixel [13:0]. It is set to the value of the output resolution or number of pixels streaming into SFC from VD/HCP or VEBox. Since the max value support is 16k pixels, the max value allowed is 16K minus 1. • VDBOX frame width is multiple of 16 for Video source and JPEG formats other than 400, 444 and 422H_2Y. • VDBOX frame width is multiple of 8 for JPEG formats 400, 444 and 422H_2Y. • VEBOX frame width is multiple of 16. • HEVC frame width is multiple of 8. • VP9 frame width is multiple of 8. *Min Resolution* is 32 pixels. *Max Resolution* is up to 16K pixels. e.g. for 1920x1080 content, FrameHeightInMBsMinus1 is equal to1087 (1080 rounded up 16 pixel boundary, minus 1. i.e. effectively specified as 1088 instead). **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
2	31:24	**Reserved** Access: RO Format: MBZ

SFC_STATE_BODY

23:17 **Reserved**

Access:	RO
Format:	MBZ

16 **Input Color Space - 0- YUV/1 - RGB**

THis specifies the color space of the input format. RGB is valid only with the VE-SFC mode.

Value	Name
0	YUV Color Space
1	RGB Color Space

15:12 **Output Chroma Downsampling co-siting position Horizontal Direction**

Format:	U4

This field specifies the fractional position of the bilinear filter for chroma downsampling. In the X-axis.

Value	Name	Description
0000b	0/8 (Left full pixel)	0 (fraction_in_integer)
0001b	1/8	1 (fraction_in_integer)
0010b	1/4 (2/8)	2 (fraction_in_integer)
0011b	3/8	3 (fraction_in_integer)
0100b	1/2 (4/8)	4 (fraction_in_integer)
0101b	5/8	5 (fraction_in_integer)
0110b	3/4 (6/8)	6 (fraction_in_integer)
0111b	7/8	7 (fraction_in_integer)
1000b	8/8	

Programming Notes
For 444 format, horizontal chroma-siting should be programmed to zero.

11:8 **Output Chroma Downsampling co-siting position Vertical Direction**

Format:	U4

This field specifies the fractional position of the bilinear filter for chroma downsampling. In the Y-axis.

Value	Name	Description
0000b	0/8 (Left full pixel)	0 (fraction_in_integer)
0001b	1/8	1 (fraction_in_integer)
0010b	1/4 (2/8)	2 (fraction_in_integer)
0011b	3/8	3 (fraction_in_integer)
0100b	1/2 (4/8)	4 (fraction_in_integer)
0101b	5/8	5 (fraction_in_integer)
0110b	3/4 (6/8)	6 (fraction_in_integer)
0111b	7/8	7 (fraction_in_integer)

1000b	8/8	

Programming Notes
For 444 and 422 format, vertical chroma co-siting value should be programmed to zero.

7:6 **Reserved**

Access:	RO
Format:	MBZ

5 **RGBA_Channel_Swap Enable**

Default Value:	0
Format:	Enable

This bit should only be used with RGB output formats and CSC conversion is turned on. When this bit is set,the R and B channels are swapped into the output RGB channels as shown in the following table:

Name	Bits	MSB Color Order	Swapped
RGBA8	8:8:8:8	A:B:G:R	A:R:G:B
RGBA10	2:10:10:10	A:R:G:B	A:B:G:R
RGB 5:6:5	5:6:5	R:G:B	B:G:R

4 **Reserved**

Access:	RO
Format:	MBZ

3:0 **Output Surface Format type**

SFC output surface format type.
Reserved

Value	Name	Description	Exists If
0		AYUV 4:4:4 (8:8:8:8 MSB-A:Y:U:V)	//Tile-Y/ Tile-X/Linear
1		RGBA8 4:4:4:4 (8:8:8:8 MSB-A:B:G:R)	//Tile-Y/ Tile-X/Linear
2		RGBA10 10:10:10:2 (2:10:10:10 MSB-A:R:G:B)	//Tile-Y/ Tile-X/Linear
3		RGB 5:6:5 (5:6:5 MSB-R:G:B)	//Tile-Y/ Tile-X/Linear
4		Planar NV12 4:2:0 8-bit	//Tile-Y
5		Packed YUYV 4:2:2 8-bit	//Tile-Y/ Tile-X/Linear
6		Packed UYVY 4:2:2 8-bit	//Tile-Y/ Tile-X/Linear
7		Packed integral Image 32-bit	//Linear
8		Packed integral Image 64-bit	//Linear
9		P016 format	//Tile-Y
10		Y210 / Y216 Format BitDepth = 0 => Y210 BitDepth = 1 => Y216	//Tile-Y / Tile-X / Linear

SFC_STATE_BODY

		11		Y410 / Y416 Format BitDepth = 0 => Y410 BitDepth = 1 => Y416	//Tile-Y / Tile-X / Linear

Restriction
For Integral Image Mode, output surface format type must be set to 32/64-bit Integral Image Plane.Driver/SW must ensure the max accumulated integral image value does not exceed the programmable output precision.HW will simply generate wrong value once it overflow in wrap around case.

3	31	**Reserved**
	30	**Reserved** Access: RO Format: MBZ
	29:24	**Reserved** Access: RO Format: MBZ
	23	**Histogram Streamout** Format: Boolean
	22	**Tile Type** Format: Boolean 0 : Real HCP Tile Mode 1 : Virtual HCP Tile Mode

Programming Notes
This field is only used when SFC Pipe Mode is HCP-to-SFC. In Real HCP Tile Mode, video streams defines the tile boundary. In Virtual HCP Tile Mode, driverstreams defines the tile boundary.

	21:20	**BitDepth** This field is valid only for output formats P016/Y216/Y416. This field is used to specify how many of the LSB bits have valid data.

Value	Name	Description
0	10BitFormat	Higher 10 bits are valid and lower 6 bits are 0

	19	**CSC Enable** This field is set when YUV to RGB or RGB to YUV conversion is required or the RGB/YUV range conversion is required. CSC conversion matrix need to be programmed accordingly.

Restriction
For Integral Image Mode, this field is Reserved and MBZ.

SFC_STATE_BODY

18 **Color Fill Enable**

Programming Notes
This field could be enabled only if the scaled resolution is smaller than the output/display resolution. If enabled, HW will fill the gap with programmable pixel values. Else, nothing will be filled in the gap region.
Usage: Color fill must be enabled for the first time/pass when a new surface is allocated/ used. Optional for subsequence frames since the gap region is filled with default pixels by prior passes.
In scalability mode ie. (SFC Engine Mode != 00), gray fill should be set only for left mosttile and for other tiles it should be disabled.

17:16 **Rotation Mode**

Format:	U2

Value	Name
00b	0 (degrees)
01b	90 Clockwise
10b	180 Clockwise
11b	270 Clockwise

Programming Notes
SFC rotation (90, 180 and 270) should be set only on VEBox input mode and SFC output set to TileY. Restriction: • For Integral Image Mode, this field is Reserved and MBZ. • For VDBox Mode, this field is Reserved and MBZ. • For linear or TileX SFC output, this field is Reserved and MBZ.

15 **Reserved**

Access:	RO
Format:	MBZ

14 **Mirror mode**

Format:	Boolean

Value	Name
0	Mirror mode Disabled
1	Mirror mode Enabled

Programming Notes
0 : Mirror Mode disabled

SFC_STATE_BODY

	1 : Mirror Mode enabled
13	**Mirror type** Format: Boolean **Value** / **Name** 0 – Horizontal flip 1 – Vertical flip **Programming Notes** 0 : Horizontal flip 1 : Vertical flip **Restriction** When Mirror mode is set to 0, this field should be programmed to 0.
12	**Chroma Upsampling Enable** This field enables the high-quality UV channel upsampler prior to IEF filter process. This field should be disabled when the source pixels and output pixels are kept with the same chroma sub-sample type and IEF is disabled. **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
11	**Reserved** Access: RO Format: MBZ
10	**RGB Adaptive** This should be always set to 0 for YUV input and can be enabled/disabled for RGB input. This should be enabled only if we enable 8-tap adaptive filter for RGB input. 0: Disable the RGB Adaptive equation and use G-Ch directly for adaptive filter 1: Enable the RGB Adaptive filter using the equation (Y=(R+2G+B)»2)
9	**Bypass X Adaptive Filtering** **Value** / **Name** / **Description** 0 – Enable X Adaptive Filtering 1 – Disable X Adaptive Filtering – The X direction will use **Default Sharpness Level** to blend between the smooth and sharp filters rather than the calculated value.
8	**Bypass Y Adaptive Filtering** **Value** / **Name** / **Description** 0 – Enable Y Adaptive Filtering

SFC_STATE_BODY

1	Disable Y Adaptive Filtering	The Y direction will use **Default Sharpness Level** to blend between the smooth and sharp filters rather than the calculated value.

7 **AVS Scaling Enable**

Value	Name	Description
1	Enable	
0	Disable	The scaling factor is ignored and a scaling ratio of 1:1 is assumed.

6 **Adaptive Filter for all Channels**

Value	Name	Description
1	Enable Adaptive Filter on UV/RB Channels	8-tap Adaptive Filter Mode is on
0	Disable Adaptive Filter on UV/RB Channels	

Programming Notes
The field can be enabled if 8-tap Adaptive filter mode is on. Else it should be disabled.

5:4 **AVS Filter Mode**

Value	Name
0	5x5 Poly-phase filter + Bilinear (adaptive)
1	8x8 poly-phase filter + Bilinear (adaptive)
2	Bilinear filter only
3	Reserved

Programming Notes
In VD-to-SFC mode, value of 1 is not allowed.

3 **Enable 8 tap for Chroma channels filtering**

This bit enables 8 tap filtering for Chroma Channels.

Programming Notes
8tap enable should only be enabled when SFC Input Chroma Sub-Sampling = 4 (ie. 444 input format to SFC).

2 **IEF4Smooth_Enable**

Value	Name	Description
0	**[Default]**	IEF is operating as a content adaptive detail filter based on 5x5 region.
1		IEF is operating as a content adaptive smooth filter based on 3x3 region

Restriction
For Integral Image Mode, this field is Reserved and MBZ.

1 **Skin Tone Tuned IEF_Enable**

Exists If:	//IEF Enable = 1

SFC_STATE_BODY

		Restriction For Integral Image Mode, this field is Reserved and MBZ.
	0	**IEF Enable** **Value** / **Name** / **Description** 1 / Enable / IEF Filter is Enabled 0 / Disable / IEF Filter is Disabled **Restriction** For Integral Image Mode and VD Mode, this field is Reserved and MBZ.
4	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**Source Region Height** Format: U14-1 **Source/Crop Region Height Minus 1 of the Input Frame in Unit of Pixel [13:0].** This field specifies the source/crop region of the input frame used for scaling of the graphic view. It defines the out-of-frame boundary used prior to AVS/IEF interpolation operation. The max value should be programmed to be equal or small than the input FrameHeightinMBminus 1 field. e.g. for 1920x1080 content, FrameHeightInMBsMinus1 is equal to1087 (1088 lines); however, the crop region height should be set to 1079(1080 lines). The last 8 lines are assumed to be not usable and should not be used as source pixels for Scaling or IEF operations. Otherwise, the bad pixels will breach and cause artifacts into the scaled output frame. Restriction : For Integral Image Mode, this field is Reserved and MBZ. Restriction : For AVS mode, the restriction is tied to chroma input format type: 420 - multiple of 2. 422/444/400 - no restrictions, except for AVS bypass case (ie. 1:1 scaling) where restriction is tied to chroma output format. **Min Resolution** is 32 pixels. **Max Resolution** is 16K pixels. **Restriction** In VD-to-SFC and HCP-to-SFC modes, this field must be programmed to same value as Input Frame Resolution Height.
	15:14	**Reserved** Access: RO Format: MBZ
	13:0	**Source Region Width** Format: U14-1 **Source/Crop Region Width Minus 1 of the Input Frame in Unit of Pixel [13:0].** This field specifies the source/crop region of the input frame used for scaling of the graphic

SFC_STATE_BODY

		view. It defines the out-of-frame boundary used prior to AVS/IEF interpolation operation. The max value should be programmed to be equal or small than the input FrameWidthinMBminus 1 field. e.g. for 1920x1080 content, FrameWidthInMBsMinus1 is equal to1919 (1920 pixel wide); however, the crop region width should be set to less than 1909(1910 pixel wide). The last 10 pixels of the frame are assumed to be not usable and should not be used as source pixels for Scaling or IEF operations. Otherwise, the bad pixels will breach and cause artifacts into the scaled output frame. Restriction : For Integral Image Mode, this field is Reserved and MBZ. Restriction : For AVS mode, the restriction is tied to chroma input format type: 420 - multiple of 2. 422 - multiple of 2. 444/400 - no restrictions, except for AVS bypass case (ie. 1:1 scaling) where restriction is tied to chroma output format. **Min Resolution** is 32 pixels. **Max Resolution** is 16K pixels. **Restriction** In VD-to-SFC and HCP-to-SFC modes, this field must be programmed to same value as Input Frame Resolution Width.
5	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**Source Region Vertical Offset** Format: U14 **Vertical Offset Of The SRC Region Relative To The Starting Position Of The Input Frame In Unit Of Pixel [13:0]** This field specifies the vertical offset of the starting position of the scaled region relatives to the starting position (pixel 0,0) of the output frame. It defines the out-of-frame boundary used prior to AVS/IEF interpolation operation. This value should be set to zero if the starting corner of the crop region is same as the input frame region. The sum of this value and the src/crop region size heightminus1 must be programmed to be equal or small than the input FrameHeightinMBminus 1 field. Restriction : For Integral Image Mode, this field is Reserved and MBZ. Restriction : For AVS mode, the restriction is tied to chroma input format type: 420 - multiple of 2. 422/444/400 - no restrictions. **Restriction** In VD-to-SFC and HCP-to-SFC modes, this field is Reserved and MBZ..
	15:14	**Reserved** Access: RO Format: MBZ

SFC_STATE_BODY

	13:0	**Source Region Horizontal Offset** Format: U14 **Horizontal Offset Of The SRC Region Relative To The Starting Position Of The Input Frame In Unit Of Pixel [13:0]** This field specifies the horizontal offset of the starting position of the scaled region relatives to the starting position (pixel 0,0) of the output frame. It defines the out-of-frame boundary used prior to AVS/IEF interpolation operation. This value should be set to zero if the starting corner of the crop region is same as the input frame region. The sum of this value and the src/crop region size widthminus1 must be programmed to be equal or small than the input FrameWidthinMBminus 1 field. Restriction : For Integral Image Mode, this field is Reserved and MBZ. Restriction : For AVS mode, the restriction is tied to chroma input format type: 420 - multiple of 2. 422 - multiple of 2. 444/400 - no restrictions. **Restriction** In VD-to-SFC and HCP-to-SFC modes, this field is Reserved and MBZ..
6	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**Output Frame Height** Format: U14-1 It is set to the value of the final output resolution of the graphic view. Since the max value support is 16k pixels, the max value allowed is 16K minus 1. Restriction : For Integral Image Mode, this field is Reserved and MBZ. Restriction : For AVS mode, the restriction is tied to chroma output format type: 420 - multiple of 2. 422/444/400 - no restrictions. **Min Resolution** is 32 pixels. **Max Resolution** is 16K pixels.
	15:14	**Reserved** Access: RO Format: MBZ
	13:0	**Output Frame Width** Format: U14-1 It is set to the value of the final output resolution of the graphic view. Since the max value support is 16k pixels, the max value allowed is 16K minus 1. Restriction : For Integral Image Mode, this field is Reserved and MBZ. Restriction : For AVS mode, the restriction is tied to chroma output format type: 420 - multiple of 2. 422 - multiple of 2. 444/400 - no restrictions. **Min Resolution** is 32 pixels. **Max Resolution** is 16K pixels.

SFC_STATE_BODY

7	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**Scaled Region Size Height** Format: U14-1 It is set to the height of the scaled region over the output frame of the graphic view. Restriction : For AVS mode, if rotation_mode = 0/180, the restriction is tied to chroma output format type: 420 - multiple of 2. 422/444/400 - no restrictions. For AVS mode, if rotation_mode = 90/270, the restriction is tied to chroma output format type: 420/422 - multiple of 2. 444/400 - no restrictions. **Min Resolution** is 32 pixels. **Max Resolution** is 16K pixels. **Programming Notes** The Max Value =< [The Output Frame Height Minus1]. **Restriction** In VD-to-SFC, HCP-to-SFC and AV1-to-SFCmodes, this field should be programmed to a value of Output Frame Height Minus1
	15:14	**Reserved** Access: RO Format: MBZ
	13:0	**Scaled Region Size Width** Format: U14-1 It is set to the Width of the scaled region over the output frame of the graphic view. Restriction : For AVS mode, the restriction is tied to chroma output format type: 420 - multiple of 2. 422 - multiple of 2. 444/400 - no restrictions. **Min Resolution** is 32 pixels. **Max Resolution** is 16K pixels **Programming Notes** The Max Value =< [The Output Frame Width Minus1]. **Restriction** In VD-to-SFC, HCP-to-SFC and AV1-to-SFCmodes, this field should be programmed to a value of Output Frame Width Minus1
8	31	**Reserved** Access: RO Format: MBZ

SFC_STATE_BODY

	30:16	**Scaled Region Vertical Offset** Format: S14 **Vertical Offset (in pixels) Of The Scaled Region Relatives to The Starting Position Of The Output Frame In Unit Of Pixel [13:0]** This field specifies the vertical offset of the starting position of the scaled region relatives to the starting position (pixel 0,0) of the output frame. The gap between the scaled and output frame shall be filled by hardware with a set of programmed YUV/RGB values (Grey Bar). This value should be set to zero if the starting corner of the scaled region is same as the output frame region. The sum of this value and the scaled region size Heightminus1 must be programmed to be equal or small than the output FrameHeightinMBminus 1 field plus 16. **Restriction** For Integral Mode, this field is reserved and MBZ For AVS mode, the restriction is tied to chroma output format type: 420 - multiple of 2. 422/444/400 - no restrictions. In VD-to-SFC, HCP-to-SFC and AV1-to-SFC modes, this field is Reserved and MBZ..
	15	**Reserved** Access: RO Format: MBZ
	14:0	**Scaled Region Horizontal Offset** Format: S14 **Horizontal Offset (in pixels) Of The Scaled Region Relatives to The Starting Position Of The Output Frame In Unit Of Pixel [13:0]** This field specifies the horizontal offset of the starting position of the scaled region relatives to the starting position (pixel 0,0) of the output frame. The gap between the scaled and output frame shall be filled by hardware with a set of programmed YUV/RGB values (Grey Bar). This value should be set to zero if the starting corner of the scaled region is same as the output frame region. The sum of this value and the scaled region size Widthminus1 must be programmed to be equal or small than the output FrameWidthinMBminus 1 field plus 16. Restriction : For Integral Image Mode, this field is Reserved and MBZ. Restriction : For AVS mode, the restriction is tied to chroma output format type: 420 - multiple of 2. 422 - multiple of 2. 444/400 - no restrictions. **Restriction** In VD-to-SFC, HCP-to-SFC and AV1-to-SFC modes, this field is Reserved and MBZ..
9	31:26	**Reserved** Access: RO Format: MBZ
	25:16	**Gray Bar Pixel - Y/R** Format: U1.9

SFC_STATE_BODY

		Range:[0.0, +1.0] This is the default value used to fill in the area between the scaled region and the output frame size (aka Gray Bar) in Y or R channel on the AYUV or RGBA domain respectively. **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
	15:10	**Reserved** Access: RO Format: MBZ
	9:0	**Gray Bar Pixel - U/G** Format: U1.9 **Range:**[0.0, +1.0] This is the default value used to fill in the area between the scaled region and the output frame size (aka Gray Bar) in U or G channel on the AYUV or RGBA domain respectively. **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
10	31:26	**Reserved** Access: RO Format: MBZ
	25:16	**Gray Bar Pixel - V/B** Format: U1.9 **Range:**[0.0, +1.0] This is the default value used to fill in the area between the scaled region and the output frame size (aka Gray Bar) in V or B channel on the AYUV or RGBA domain respectively. **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
	15:10	**Reserved** Access: RO Format: MBZ
	9:0	**Gray Bar Pixel - A** Format: U1.9 **Range:**[0.0, +1.0] This is the default value used to fill in the area between the scaled region and the output frame

SFC_STATE_BODY

		size (aka Gray Bar) in A channel on the AYUV or RGBA domain respectively. **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
11	31:26	**Reserved** Access: RO Format: MBZ
	25:16	**UV Default value for V channel (For Mono Input Support)** Exists If: //Input NOT originated by VEBOX. Format: U1.9 **Range:**[0.0, +1.0] This field specifies the UV default value fill in to the UV output channels when input from VDBOX is set to Monochrome. **Restriction** Not used when input is originated by VEBOX (Including Integral Image Mode).
	15:10	**Reserved** Access: RO Format: MBZ
	9:0	**UV Default value for U channel (For Mono Input Support)** Exists If: //Input NOT originated by VEBOX. Format: U1.9 **Range:**[0.0, +1.0] This field specifies the UV default value fill in to the UV output channels when input from VDBOX is set to Monochrome. **Restriction** Not used when input is originated by VEBOX (Including Integral Image Mode).
12	31:10	**Reserved** Access: RO Format: MBZ
	9:0	**Alpha Default Value** Format: U1.9 **Range:**[0.0, +1.0] This field specifies the Alpha default value fill into the alpha output channel when output

SFC_STATE_BODY

		format type is set to RGBA8/10. **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
13	31:28	**Reserved** Access: RO Format: MBZ
	27:5	**Scaling Factor Height** Format: U4.19 This field specifies the scaling ratio of the vertical sizes between the crop/source region and the scaled region. The destination pixel coordinate, y-axis, is multiplied with this scaling factor to mapping back to the source input pixel coordinate. The field specifies the ratio of crop height resolution/ scaled height resolution. This implies $1/sf_v$ in the equation.
	4:0	**Reserved** Access: RO Format: MBZ
14	31:28	**Reserved** Access: RO Format: MBZ
	27:5	**Scale Factor Width** Format: U4.19 This field specifies the scaling ratio of the horizontal sizes between the crop/source region and the scaled region. The destination pixel coordinate, x-axis, is multiplied with this scaling factor to mapping back to the source input pixel coordinate. The field specifies the ratio of crop width resolution/ scaled width resolution. This implies $1/sf_u$ in the equations above.
	4:0	**Reserved** Access: RO Format: MBZ
15	31:22	**Reserved** Access: RO Format: MBZ
	21:0	**Reserved** Access: RO Format: MBZ

SFC_STATE_BODY

16 | 31:12 | Output Frame Surface Base Address

Specifies the 4K byte aligned frame buffer address for outputting the scaled up/down image. Data is stored in Tile-Y format.

For Integral Image mode, the accumulated integral image values will be packed linear in this surface.

Programming Notes
This field is ignored if I-frame only mode is set to 0 (Disable).

11:0 | Reserved

Access:	RO
Format:	MBZ

17 | 31:16 | Reserved

Access:	RO
Format:	MBZ

15:0 | Output Frame Surface Base Address High

This field is for the upper range [47:32] of Output Frame Surface Base Address.

For Integral Image mode, the accumulated integral image values will be packed linear in this surface.

18 | 31:15 | Reserved

Access:	RO
Format:	MBZ

14:13 | Output Surface Tiled Mode

Format:	U2

For Media Surfaces: This field specifies the tiled resource mode.

Value	Name	Description
0h	TRMODE_NONE	No tiled resource
1h	TRMODE_TILEYF	Reserved
2h	TRMODE_TILEYS	reserved
3h	Reserved	

12 | Output Frame Surface Base Address - Row Store Scratch Buffer Cache Select

Format:	MBZ

Value	Name	Description
0	Disable **[Default]**	This field must be programmed to 0

Programming Notes
This must be set to 0

SFC_STATE_BODY

	11	**Reserved** Access: RO Format: MBZ
	10	**Compression Type** Format: U1 This field is applicable only when Memory compression is enabled. 0 – Media Compression Enabled **[Default]** 1 – Render Compression Enabled
	9	**Output Frame Surface Base Address - Memory Compression Enable** Format: Enable Memory compression will be attempted for this surface.
	8:7	**Output Frame Surface Base Address - Arbitration Priority Control** Format: HEVC_ARBITRATION_PRIORITY
	6:1	**Output Frame Surface Base Address - Index to Memory Object Control State (MOCS) Tables** Format: U6 The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**
19	31:12	**AVS Line Buffer Surface Base Address** Specifies the 4K byte aligned frame buffer address for scratch space used for row/column store. This surface is used only if the internal buffer inside the SFC HW is not large enough to contain all row/column memory accesses. The AVS line buffer needs to be a valid address even for 1:1 scaling if SFC is used. **Programming Notes** This field needs to be programmed separately and exclusively for each pipe when SFC engine mode is not programmed 2'b00(Scalability workloads).
	11:0	**Reserved** Access: RO Format: MBZ
20	31:16	**Reserved** Access: RO Format: MBZ

SFC_STATE_BODY

	15:0	**AVS Line Buffer Surface Base Address High** This field is for the upper range [47:32] of AVS Line Buffer Surface Base Address. AVS Line buffer address needs to be valid even for 1:1 scaling if SFC is used. **Programming Notes** This field needs to be programmed separately and exclusively for each pipe when SFC engine mode is not programmed 2'b00.
21	31:15	**Reserved** Access: RO Format: MBZ
	14:13	**AVS Line Buffer Tiled Mode** Format: U2 **For Media Surfaces:** This field specifies the tiled resource mode. 0h – TRMODE_NONE – No tiled resource 1h – TRMODE_TILEYF – 4KB tiled resources 2h – TRMODE_TILEYS – 64KB tiled resources 3h – Reserved
	12	**AVS Line Buffer Base Address - Row Store Scratch Buffer Cache Select** Format: U1 This field controls if the Row Store is going to store inside Media Cache (rowstore cache) or to LLC. 0 – LLC – Buffer going to LLC **Programming Notes** This suface does not support to put in Row Store Scratch Buffer. Must be set to 0
	11	**Reserved** Access: RO Format: MBZ
	10	**AVS Line Buffer Base Address - Memory Compression Mode** Default Value: 0 Horizontal Compression Mode Format: U1 Distinguishes vertical from horizontal compression. Please refer to vol1a? **Memory Data Formats chapter - section** ?media Memory Compression for more details. **Programming Notes** Memory compression is not supported. ?This bit is not used. Default to 0

SFC_STATE_BODY

	9	**AVS Line Buffer Base Address - Memory Compression Enable** Default Value: 0 Disable Format: Enable This bit control memory compression for this surface **Programming Notes** This bit must be set to 0 (Memory compression is not supported in this surface)
	8:7	**AVS Line Buffer Base Address - Arbitration Priority Control** Format: HEVC_ARBITRATION_PRIORITY
	6:1	**AVS Line Buffer Base Address - Index to Memory Object Control State (MOCS) Tables** Format: U6 The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**
22	31:12	**IEF Line Buffer Surface Base Address** Specifies the 4K byte aligned frame buffer address for the scratch space used for row/column store. This surface is used only if the internal buffer inside SFC HW is not large enough to contain all row/column memory accesses. **Programming Notes** This field needs to be programmed separately and exclusively for each pipe when SFC engine mode is not programmed 2'b00. **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
	11:0	**Reserved** Access: RO Format: MBZ
23	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**IEF Line Buffer Surface Base Address High** This field is for the upper range [47:32] of IEF Line Buffer Surface Base Address. **Programming Notes** This field needs to be programmed separately and exclusively for each pipe when SFC engine mode is not programmed 2'b00.

SFC_STATE_BODY

		Restriction For Integral Image Mode, this field is Reserved and MBZ.
24	31:15	**Reserved** Access: RO Format: MBZ
	14:13	**IEF Line Buffer Tiled Mode** Format: U2 **For Media Surfaces:**This field specifies the tiled resource mode. **Value** / **Name** / **Description** 0h / TRMODE_NONE / No tiled resource 1h / TRMODE_TILEYF / 4KB tiled resources 2h / TRMODE_TILEYS / 64KB tiled resources 3h / Reserved /
	12	**IEF Line Buffer Base Address - Row Store Scratch Buffer Cache Select** Format: U1 This field controls if the Row Store is going to store inside Media Cache (rowstore cache) or to LLC. **Value** / **Name** / **Description** 0 / LLC / Buffer going to LLC **Programming Notes** This surface does not support Rowstore Scratch Buffer Cache. Must be programmed to 0
	11	**Reserved** Access: RO Format: MBZ
	10	**IEF Line Buffer Base Address - Memory Compression Mode** Default Value: 0 Format: U1 Distinguishes vertical from horizontal compression. **Programming Notes** Must be zero; memory compression is not supported for this surface. Default to0
	9	**IEF Line Buffer Base Address - Memory Compression Enable** Default Value: 0 Disable Format: Enable **Programming Notes** Memory compression is not supported for this surface

SFC_STATE_BODY

		Must be 0.
	8:7	**IEF Line Buffer Base Address - Arbitration Priority Control** Format: HEVC_ARBITRATION_PRIORITY
	6:1	**IEF Line Buffer Base Address - Index to Memory Object Control State (MOCS) Tables** Format: U6 The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**
25	31:12	**SFD Line Buffer Surface Base Address** Specifies the 4K byte aligned frame buffer address for the scratch space used for row/column store. This surface is used only if the internal buffer inside SFC HW is not large enough to contain all row/column memory accesses. **Programming Notes** This field needs to be programmed separately and exclusively for each pipe when SFC engine mode is not programmed 2'b00. **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
	11:0	**Reserved** Access: RO Format: MBZ
26	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**SFD Line Buffer Surface Base Address High** This field is for the upper range [47:32] of SFD Line Buffer Surface Base Address. **Programming Notes** This field needs to be programmed separately and exclusively for each pipe when SFC engine mode is not programmed 2'b00. **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
27	31:15	**Reserved** Access: RO Format: MBZ

SFC_STATE_BODY

14:13 **SFD Line Buffer Tiled Mode**

Format:	U2

For Media Surfaces: This field specifies the tiled resource mode.

Value	Name	Description
0h	TRMODE_NONE	No tiled resource
1h	TRMODE_TILEYF	4KB tiled resources
2h	TRMODE_TILEYS	64KB tiled resources
3h	Reserved	

12 **SFD Line Buffer Base Address - Row Store Scratch Buffer Cache Select**

Format:	U1

This field controls if the Row Store is going to store inside Media Cache (rowstore cache) or to LLC.

Value	Name	Description
0	LLC	Buffer going to LLC
1	Media Storage **[Default]**	Data will first cache in Media Storage

Programming Notes
This surface does not support Rowstore Scratch Buffer Cache. Must be programmed to 0

11 **Reserved**

Access:	RO
Format:	MBZ

10 **SFD Line Buffer Base Address - Memory Compression Mode**

Default Value:	0
Format:	U1

Distinguishes vertical from horizontal compression. Please refer to vol1a **Memory Data Formats chapter - section** media Memory Compression for more details.

Programming Notes
Must be zero; memory compression is not supported for this surface. Default to0

9 **SFD Line Buffer Base Address - Memory Compression Enable**

Default Value:	0 Disable
Format:	Enable

Programming Notes
Memory compression is not supported for this surface Must be 0.

8:7 **SFD Line Buffer Base Address - Arbitration Priority Control**

Format:	HEVC_ARBITRATION_PRIORITY

SFC_STATE_BODY

	6:1	**SFD Line Buffer Base Address - Index to Memory Object Control State (MOCS) Tables** Format: U6 The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**
28	31:28	**Output Surface Format**
	27	**Output Surface Interleave Chroma Enable**
	26:22	**Reserved** Access: RO Format: MBZ
	21:3	**Output Surface Pitch** Format: U19-1 This field specifies the surface pitch.

Value	Name	Description
[0,2047]	SURFTYPE_BUFFER Surfaces	[1B, 2048B]
[0, 524287]	Other Linear Surfaces	[64B, 512KB] = [1 CL, 8K CLs]
[511, 524287]	X-tiled Surface	[512B, 256KB] = [1tile, 512 tiles]
[127, 524287]	Y-tiled surfaces	[128B,256KB] = [1 tile, 2048 tiles]

Programming Notes

- For tiled surfaces, the pitch must be a multiple of the tile width
- For Linear surfaces, the pitch must be a multiple of CL (64B) width
- If **Half Pitch for Chroma** is set, this field must be a multiple of two tile widths for tiled surfaces, or a multiple of 2 bytes for linear surfaces.

If Media Memory Compression is enabled, the following max pitch size restriction must be honored. For larger resolution, Media Memory compression Must be disabled.

Tiling Mode	Pixel Format	Max Frame Width (bytes)	Max Frame Width (pixels)	Max Pitch (bytes)
Legacy 4K	8bpp	16k	16k	16k + 127
	16bpp	16k	8k	16k + 127
	32bpp	16k	4k	16k + 127
	64bpp	16k	2k	16k + 127
	128bpp	16k	1k	16k + 127
TileYF	8bpp	8k	8k	8k + 63
	16bpp	16k	8k	16k + 127

SFC_STATE_BODY

	32bpp	16k	4k	16k + 127
	64bpp	16k	2k	16k + 255
	128bpp	16k	1k	16k + 255
TileYS	8bpp	16k	16k	16k + 255
	16bpp	16k	8k	16k + 511
	32bpp	16k	4k	16k + 511
	64bpp	16k	2k	16k + 1023
	128bpp	16k	1k	16k + 1023

2 **Output Surface Half Pitch For Chroma**

Exists If:	//PLANAR Surface Formats Only
Format:	Enable

This field indicates that the chroma plane(s) will use a pitch equal to half the value specified in the Surface Pitch field.

1 **Output Surface Tiled**

Format:	Boolean

This field specifies whether the surface is tiled.

Value	Name	Description
1	True	Tiled
0	FALSE	Linear

Programming Notes

- Linear surfaces can be mapped to Main Memory (uncached) or System Memory (cacheable, snooped). Tiled surfaces can only be mapped to Main Memory.
- The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit.

0 **Output Surface Tile Walk**

Format:	Boolean

This field specifies the type of memory tiling (XMajor or YMajor) employed to tile this surface. See *Memory Interface Functions* for details on memory tiling and restrictions.

Value	Name
0	TILEWALK_XMAJOR
1	TILEWALK_YMAJOR

Programming Notes

- The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit.

SFC_STATE_BODY

		This field is ignored when the surface is linear.
29	31:16	**Output Surface X Offset For U** Exists If: //PLANAR Surface Formats Only Format: U16 This field specifies the horizontal offset in pixels from the **Surface Base Address** to the start (origin) of the U(Cb) plane or the interleaved UV plane if **Interleave Chroma** is enabled. **Programming Notes** For PLANAR_420 and PLANAR_422 surface formats, this field must be zero.
	15:0	**Output Surface Y Offset For U** Exists If: //PLANAR Surface Formats Only Format: U16 This field specifies the vertical offset in rows from the **Surface Base Address** to the start (origin) of the U(Cb) plane or the interleaved UV plane if **Interleave Chroma** is enabled. **Programming Notes** For PLANAR_420 and PLANAR_422 surface formats, this field must be multiple of 16 pixels - i.e. multiple MBs.
30	31:16	**Output Surface X Offset For V** Exists If: //PLANAR Surface Formats with Interleaved Chroma Disable Format: U16 This field specifies the horizontal offset in pixels from the **Surface Base Address** to the start (origin) of the V(Cr) plane. **Programming Notes** For PLANAR_420 and PLANAR_422 surface formats, this field must indicate an even number of pixels.
	15:0	**Output Surface Y Offset For V** Exists If: //PLANAR Surface Formats with Interleaved Chroma Disable Format: U16 This field specifies the vertical offset in rows from the Surface Base Address to the start (origin) of the V(Cr) plane. **Programming Notes** For PLANAR_420 and PLANAR_422 surface formats, this field must indicate an even number of pixels.
31	31:0	**Reserved** Access: RO Format: MBZ
32	31:0	**Reserved** Access: RO Format: MBZ

SFC_STATE_BODY

33	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**SourceEndX** Format: U14 Indicates the X-direction end location in the original input frame to SFC. For 420/422 this field should be in multiple of 2. **Programming Notes** This field is only programmed when SFC Pipe Mode is HCP-to-SFC This should be in sync with tile widthsize programmed inHCP_TILE_CODING command
	15:14	**Reserved** Access: RO Format: MBZ
	13:0	**SourceStartX** Format: U14 Indicates the X-direction start location in the original input frame to SFC. For 420/422 this field should be in multiple of 2. **Programming Notes** This field is only programmed when SFC Pipe Mode is HCP-to-SFC This should be in sync with tile width size programmed in HCP_TILE_CODING command
34	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**DestinationEndX** Format: U14 Indicates the X-direction end location in the output frame of SFC. **Programming Notes** This field is valid only in Scalability Mode. Please refer to SFC Programming Model to program this field.
	15:14	**Reserved** Access: RO Format: MBZ
	13:0	**DestinationStartX** Format: U14 Indicate the X-direction start location in the output frame of SFC. **Programming Notes** This field is valid only in Scalability Mode. Please refer to SFC Programming Model to program this field.

SFC_STATE_BODY

35	31:29	**Reserved** Access: RO Format: MBZ
	28:5	**Xphaseshift** Format: s4.19 Xphaseshift would be programmed to do output centering in x-direction. **Programming Notes** This field allows user to program the horizontal address/coordinate of the center of scaling. For the valid programming where the scaling center is within the original image, the numerical/floatingvalue for the Xphaseshift would be c_x*(1/sf_hor - 1). The sf_hor in the above equation is the numerical/floating value of the horizontal scaling factor while c_x corresponds to the normalized horizontal coordinate of the scaling center (i.e., 0 <= c_x <= 0.5). For example, if (c_x, c_y) = (0, 0), the scaling center would be the legacy top-left mode while (c_x, c_y) = (0.5, 0.5) would be the center mode which corresponds to the default of many other display solutions.
	4:0	**Reserved** Access: RO Format: MBZ
36	31:29	**Reserved** Access: RO Format: MBZ
	28:5	**Yphaseshift** Format: s4.19 Yphaseshift would be programmed to do output centering in y-direction. **Programming Notes** This field allows user to program the verticaladdress/coordinate of the center of scaling. For the valid programming where the scaling center is within the original image, the numerical/floatingvalue for the Yphaseshift would be c_y*(1/sf_ver- 1). The sf_verin the above equation is the numerical/floating value of the verticalscaling factor while c_ycorresponds to the normalized verticalcoordinate of the scaling center (i.e., 0 <= c_y<= 0.5). For example, if (c_x, c_y) = (0, 0), the scaling center would be the legacy top-left mode while (c_x, c_y) = (0.5, 0.5) would be the center mode which corresponds to the default of many other display solutions.
	4:0	**Reserved** Access: RO Format: MBZ

SFC_STATE_BODY

37	31:12	**AVS Line Tile Buffer Surface Base Address** Specifies the 4K byte aligned frame buffer address for the scratch space used for row/column store. This surface is used only if the internal buffer inside SFC HW is not large enough to contain all row/column memory accesses. **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
	11:0	**Reserved** Access: RO Format: MBZ
38	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**AVS Line Tile Buffer Surface Base Address High** This field is for the upper range [47:32] of AVS Line Tile Buffer Surface Base Address. **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
39	31:15	**Reserved** Access: RO Format: MBZ
	14:13	**AVS Line Tile Buffer Tiled Mode** Format: U2 **For Media Surfaces:** This field specifies the tiled resource mode. **Value** / **Name** / **Description** 0h / TRMODE_NONE / No tiled resource 1h / TRMODE_TILEYF / 4KB tiled resources 2h / TRMODE_TILEYS / 64KB tiled resources 3h / Reserved /
	12	**AVS Line Tile Buffer Base Address - Row Store Scratch Buffer Cache Select** Format: U1 This field controls if the Row Store is going to store inside Media Cache (rowstore cache) or to LLC. **Value** / **Name** / **Description** 0 / LLC **[Default]** / Buffer going to LLC **Programming Notes** This surface does not support Rowstore Scratch Buffer Cache. Must be programmed to 0

SFC_STATE_BODY

	11	**Reserved** Access: RO Format: MBZ
	10	**AVS Line Tile Buffer Base Address - Memory Compression Mode** Default Value: 0 Format: U1 Distinguishes vertical from horizontal compression. Please refer to vol1a **Memory Data Formats chapter - section** media Memory Compression for more details. **Programming Notes** Must be zero; memory compression is not supported for this surface. Default to0
	9	**AVS Line Tile Buffer Base Address - Memory Compression Enable** Default Value: 0 Disable Format: Enable **Programming Notes** Memory compression is not supported for this surface Must be 0.
	8:7	**AVS Line Tile Buffer Base Address - Arbitration Priority Control** Format: HEVC_ARBITRATION_PRIORITY
	6:1	**AVS Line Tile Buffer Base Address - Index to Memory Object Control State (MOCS) Tables** Format: U6 The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**
40	31:12	**IEF Line Tile Buffer Surface Base Address** Specifies the 4K byte aligned frame buffer address for the scratch space used for row/column store. This surface is used only if the internal buffer inside SFC HW is not large enough to contain all row/column memory accesses. **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
	11:0	**Reserved** Access: RO Format: MBZ

SFC_STATE_BODY

41 — 31:16 — Reserved

Access:	RO
Format:	MBZ

41 — 15:0 — IEF Line Tile Buffer Surface Base Address High

This field is for the upper range [47:32] of IEF Line Tile Buffer Surface Base Address.

Restriction
For Integral Image Mode, this field is Reserved and MBZ.

42 — 31:15 — Reserved

Access:	RO
Format:	MBZ

42 — 14:13 — IEF Line Tile Buffer Tiled Mode

Format:	U2

For Media Surfaces: This field specifies the tiled resource mode.

Value	Name	Description
0h	TRMODE_NONE	No tiled resource
1h	TRMODE_TILEYF	4KB tiled resources
2h	TRMODE_TILEYS	64KB tiled resources
3h	Reserved	

42 — 12 — IEF Line Tile Buffer Base Address - Row Store Scratch Buffer Cache Select

Format:	U1

This field controls if the Row Store is going to store inside Media Cache (rowstore cache) or to LLC.

Value	Name	Description
0	LLC **[Default]**	Buffer going to LLC

Programming Notes
This surface does not support Rowstore Scratch Buffer Cache. Must be programmed to 0

42 — 11 — Reserved

Access:	RO
Format:	MBZ

42 — 10 — IEF Line Tile Buffer Base Address - Memory Compression Mode

Default Value:	0
Format:	U1

Distinguishes vertical from horizontal compression. Please refer to vol1a **Memory Data Formats chapter - section** media Memory Compression for more details.

Programming Notes
Must be zero; memory compression is not supported for this surface. Default to0

SFC_STATE_BODY

	9	**IEF Line Tile Buffer Base Address - Memory Compression Enable** Default Value: 0 Disable Format: Enable **Programming Notes** Memory compression is not supported for this surface Must be 0.
	8:7	**IEF Line Tile Buffer Base Address - Arbitration Priority Control** Format: HEVC_ARBITRATION_PRIORITY
	6:1	**IEF Line Tile Buffer Base Address - Index to Memory Object Control State (MOCS) Tables** Format: U6 The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**
43	31:12	**SFD Line Tile Buffer Surface Base Address** Specifies the 4K byte aligned frame buffer address for the scratch space used for row/column store. This surface is used only if the internal buffer inside SFC HW is not large enough to contain all row/column memory accesses. **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
	11:0	**Reserved** Access: RO Format: MBZ
44	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**SFD Line Tile Buffer Surface Base Address High** This field is for the upper range [47:32] of SFD Line Tile Buffer Surface Base Address. **Restriction** For Integral Image Mode, this field is Reserved and MBZ.
45	31:15	**Reserved** Access: RO Format: MBZ

SFC_STATE_BODY

14:13 SFD Line Tile Buffer Tiled Mode

Format:	U2

For Media Surfaces: This field specifies the tiled resource mode.

Value	Name	Description
0h	TRMODE_NONE	No tiled resource
1h	TRMODE_TILEYF	4KB tiled resources
2h	TRMODE_TILEYS	64KB tiled resources
3h	Reserved	

12 SFD Line Tile Buffer Base Address - Row Store Scratch Buffer Cache Select

Format:	U1

This field controls if the Row Store is going to store inside Media Cache (rowstore cache) or to LLC.

Value	Name	Description
0	LLC **[Default]**	Buffer going to LLC

Programming Notes
This surface does not support Rowstore Scratch Buffer Cache. Must be programmed to 0

11 Reserved

Access:	RO
Format:	MBZ

10 SFD Line Tile Buffer Base Address - Memory Compression Mode

Default Value:	0
Format:	U1

Distinguishes vertical from horizontal compression. Please refer to vol1a **Memory Data Formats chapter - section** media Memory Compression for more details.

Programming Notes
Must be zero; memory compression is not supported for this surface. Default to0

9 SFD Line Tile Buffer Base Address - Memory Compression Enable

Default Value:	0 Disable
Format:	Enable

Programming Notes
Memory compression is not supported for this surface Must be 0.

8:7 SFD Line Tile Buffer Base Address - Arbitration Priority Control

Format:	HEVC_ARBITRATION_PRIORITY

SFC_STATE_BODY

	6:1	**SFD Line Tile Buffer Base Address - Index to Memory Object Control State (MOCS) Tables** Format: U6 The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**
46	31:12	**Histogram Surface Base Address** Specifies the CL aligned frame buffer address for Y histogram. The SFC_HISTOGRAM_SURFACEspecifies how histogram data would be places in surface.
	11:0	**Reserved** Access: RO Format: MBZ
47	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Histogram Surface Base Address High** This field is for the upper range [47:32] of Histogram Surface Base Address.
48	31:15	**Reserved** Access: RO Format: MBZ
	14:13	**Histogram Tiled Mode** Format: U2 **For Media Surfaces:**This field specifies the tiled resource mode. **Value** / **Name** / **Description** 0h / TRMODE_NONE / No tiled resource 1h / TRMODE_TILEYF / 4KB tiled resources 2h / TRMODE_TILEYS / 64KB tiled resources 3h / Reserved /
	12	**Histogram Base Address - Cache Select** Format: U1 This field controls if the Histogram need to be cached in LLC or not. **Value** / **Name** / **Description** 0 / LLC **[Default]** / Buffer going to LLC **Programming Notes** Must be programmed to 0

SFC_STATE_BODY

	11	**Reserved** Access: RO Format: MBZ
	10	**Histogram Base Address - Memory Compression Type** Default Value: 0 Format: U1 Distinguishes Media or 3D compression. **Memory Data Formats chapter - section** media Memory Compression for more details. **Programming Notes** Must be zero; memory compression is not supported for this surface. Default to 0
	9	**Histogram Base Address - Memory Compression Enable** Default Value: 0 Disable Format: Enable **Programming Notes** Memory compression is not supported for this surface Must be 0.
	8:7	**Histogram Base Address - Arbitration Priority Control** Format: HEVC_ARBITRATION_PRIORITY
	6:1	**Hisgotram Base Address - Index to Memory Object Control State (MOCS) Tables** Format: U6 The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**

SIMD1 Untyped BUFFER Surface 64-Bit Address Payload

MAP64B_USU_SIMD1 - SIMD1 Untyped BUFFER Surface 64-Bit Address Payload		
Size (in bits):	64	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.1	63:0	**U0** Format: U64 Specifies the U channel for slot [0]

SIMD8 Dual Source Render Target Data Payload

MDP_RTW_8DS - SIMD8 Dual Source Render Target Data Payload

Size (in bits): 2048

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Src0 Red** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src0 Red
1.0-1.7	255:0	**Src0 Green** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src0 Green
2.0-2.7	255:0	**Src0 Blue** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src0 Blue
3.0-3.7	255:0	**Src0 Alpha** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src0 Alpha
4.0-4.7	255:0	**Src1 Red** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src1 Red
5.0-5.7	255:0	**Src1 Green** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src1 Green

<table>
<tr><th colspan="3">MDP_RTW_8DS - SIMD8 Dual Source Render Target Data Payload</th></tr>
<tr><td>6.0-6.7</td><td>255:0</td><td>Src1 Blue<table><tr><td>Format:</td><td>MDP_DW_SIMD8</td></tr></table>Slots[7:0] or [15:8] of Src1 Blue</td></tr>
<tr><td>7.0-7.7</td><td>255:0</td><td>Src1 Alpha<table><tr><td>Format:</td><td>MDP_DW_SIMD8</td></tr></table>Slots[7:0] or [15:8] of Src1 Alpha</td></tr>
</table>

SIMD8 LOD Message Address Payload Control

MACR_LOD_SIMD8 - SIMD8 LOD Message Address Payload Control

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0.0	31:0	**Slot0 LOD** Format: MACD_LOD Specifies the LOD for slot 0
0.1	31:0	**Slot1 LOD** Format: MACD_LOD Specifies the LOD for slot 1
0.2	31:0	**Slot2 LOD** Format: MACD_LOD Specifies the LOD for slot 2
0.3	31:0	**Slot3 LOD** Format: MACD_LOD Specifies the LOD for slot 3
0.4	31:0	**Slot4 LOD** Format: MACD_LOD Specifies the LOD for slot 4
0.5	31:0	**Slot5 LOD** Format: MACD_LOD Specifies the LOD for slot 5
0.6	31:0	**Slot6 LOD** Format: MACD_LOD Specifies the LOD for slot 6
0.7	31:0	**Slot7 LOD** Format: MACD_LOD Specifies the LOD for slot 7

SIMD8 Render Target Data Payload

MDP_RTW_8 - SIMD8 Render Target Data Payload		
Size (in bits):	1024	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Red** Format: MDP_DW_SIMD8 Slots [7:0] Red
1.0-1.7	255:0	**Green** Format: MDP_DW_SIMD8 Slots [7:0] Green
2.0-2.7	255:0	**Blue** Format: MDP_DW_SIMD8 Slots [7:0] Blue
3.0-3.7	255:0	**Alpha** Format: MDP_DW_SIMD8 Slots [7:0] Alpha

SIMD8 Typed Surface 32-Bit Address Payload

MAP32B_TS_SIMD8 - SIMD8 Typed Surface 32-Bit Address Payload

Size (in bits): 1024

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**U** Format: MACR_32b Specifies the U channel for slots [7:0]
1.0-1.7	255:0	**V** Format: MACR_32b Specifies the V channel for slots [7:0]
2.0-2.7	255:0	**R** Format: MACR_32b Specifies the R channel for slots [7:0]
3.0-3.7	255:0	**LOD** Format: MACR_LOD_SIMD8 Specifies the LOD for slots [7:0]

SIMD8 Untyped BUFFER Surface 32-Bit Address Payload

MAP32B_USU_SIMD8 - SIMD8 Untyped BUFFER Surface 32-Bit Address Payload

Size (in bits):	256
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**U** Format: **MACR_32b** Specifies the U channel for slots [7:0]

SIMD8 Untyped BUFFER Surface 64-Bit Address Payload

MAP64B_USU_SIMD8 - SIMD8 Untyped BUFFER Surface 64-Bit Address Payload

Size (in bits):	512
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**U3_U0** Format: **MACR_64b** Specifies the U channel for slots [3:0]
1.0-1.7	255:0	**U7_U4** Format: **MACR_64b** Specifies the U channel for slots [7:4]

SIMD8 Untyped SCRATCH Surface 32-Bit Address Payload

MAP32B_USUV_SIMD8 - SIMD8 Untyped SCRATCH Surface 32-Bit Address Payload

Size (in bits):	512
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**U** Format: **MACR_32b** Specifies the U channel for slots [7:0]
1.0-1.7	255:0	**V** Format: **MACR_32b** Specifies the V channel for slots [7:0]

SIMD8 URB Channel Mask Message Address Payload

MAPU_CMASK_SIMD8 - SIMD8 URB Channel Mask Message Address Payload

Size (in bits):	256
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Per Slot Channel Mask** Format: MACD_URB_CMASK[8] Each slot's mask field is combined with the execution mask to determine which Dwords are written to the URB.

SIMD8 URB Offset Message Address Payload

MAPU_SIMD8 - SIMD8 URB Offset Message Address Payload

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Slot Offset** Format: U32[8] Each slot's offset field is added to the **Global Offset**(specified in the message descriptor) and the slot's URB Handle (specified in the message header)to generate the URB address for this access. This offset and the Global Offset are specified as Oword units (128 bits). Value: [0-2047] Name:

SIMD16 Render Target Data Payload

MDP_RTW_16 - SIMD16 Render Target Data Payload

Size (in bits): 2048

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Red[7:0]** Format: **MDP_DW_SIMD8** Slots [7:0] Red
1.0-1.7	255:0	**Red[15:8]** Format: **MDP_DW_SIMD8** Slots [15:8] Red
2.0-2.7	255:0	**Green[7:0]** Format: **MDP_DW_SIMD8** Slots [7:0] Green
3.0-3.7	255:0	**Green[15:8]** Format: **MDP_DW_SIMD8** Slots [15:8] Green
4.0-4.7	255:0	**Blue[7:0]** Format: **MDP_DW_SIMD8** Slots [7:0] Blue
5.0-5.7	255:0	**Blue[15:8]** Format: **MDP_DW_SIMD8** Slots [15:8] Blue
6.0-6.7	255:0	**Alpha[7:0]**

MDP_RTW_16 - SIMD16 Render Target Data Payload		
		Format: MDP_DW_SIMD8 Slots [7:0] Alpha
7.0-7.7	255:0	**Alpha[15:7]** Format: MDP_DW_SIMD8 Slots [15:7] Alpha

SIMD16 Typed Surface 16-Bit Address Payload

MAP16B_TS_SIMD16 - SIMD16 Typed Surface 16-Bit Address Payload

Size (in bits): 1024

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**U** Format: U16[16] Specifies the U channel for slots [15:0]
1.0-1.7	255:0	**V** Format: U16[16] Specifies the V channel for slots [15:0]
2.0-2.7	255:0	**R** Format: U16[16] Specifies the R for slots [15:0]
3.0-3.7	255:0	**LOD** Format: U16[16] Specifies the LOD for slots [15:0] Value / Name / Description: [0,14] / / representing LOD

SIMD16 Untyped BUFFER Surface 16-Bit Address Payload

MAP16B_USU_SIMD16 - SIMD16 Untyped BUFFER Surface 16-Bit Address Payload

Size (in bits):	256
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**U** Format: U16[16] Specifies the U channel for slots [15:0]

SIMD16 Untyped BUFFER Surface 32-Bit Address Payload

MAP32B_USU_SIMD16 - SIMD16 Untyped BUFFER Surface 32-Bit Address Payload

Size (in bits):	512
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0.0-1.7	511:0	**U** Format: U32[16] Specifies the U channel for slots [15:0]

SIMD16 Untyped BUFFER Surface 64-Bit Address Payload

MAP64B_USU_SIMD16 - SIMD16 Untyped BUFFER Surface 64-Bit Address Payload

Size (in bits): 1024

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**U3_U0** Format: **MACR_64b** Specifies the U channel for slots [3:0]
1.0-1.7	255:0	**U7_U4** Format: **MACR_64b** Specifies the U channel for slots [7:4]
2.0-2.7	255:0	**U11_U8** Format: **MACR_64b** Specifies the U channel for slots [11:8]
3.0-3.7	255:0	**U15_U12** Format: **MACR_64b** Specifies the U channel for slots [15:12]

SIMD16 Untyped SCRATCH Surface 16-Bit Address Payload

MAP16B_USUV_SIMD16 - SIMD16 Untyped SCRATCH Surface 16-Bit Address Payload

Size (in bits):	512
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**U** Format: U16[16] Specifies the U channel for slots [15:0]
1.0-1.7	255:0	**V** Format: U16[16] Specifies the V channel for slots [15:0]

SIMD16 Untyped SCRATCH Surface 32-Bit Address Payload

MAP32B_USUV_SIMD16 - SIMD16 Untyped SCRATCH Surface 32-Bit Address Payload

Size (in bits):	1024
Default Value:	0x00000000, 0x00000000

DWord	Bit	Description
0.0-1.7	511:0	**U** Format: U32[16] Specifies the U channel for slots [15:0]
2.0-3.7	511:0	**V** Format: U32[16] Specifies the V channel for slots [15:0]

SIMD 32-Bit Address Payload Control

MACR_32B - SIMD 32-Bit Address Payload Control

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0.0	31:0	**Offset0** Format: U32 Specifies the address offset for slot 0 in this payload register.
0.1	31:0	**Offset1** Format: U32 Specifies the address offset for slot 1 in this payload register.
0.2	31:0	**Offset2** Format: U32 Specifies the address offset for slot 2 in this payload register.
0.3	31:0	**Offset3** Format: U32 Specifies the address offset for slot 3 in this payload register.
0.4	31:0	**Offset4** Format: U32 Specifies the address offset for slot 4 in this payload register.
0.5	31:0	**Offset5** Format: U32 Specifies the address offset for slot 5 in this payload register.
0.6	31:0	**Offset6** Format: U32 Specifies the address offset for slot 6 in this payload register.
0.7	31:0	**Offset7** Format: U32 Specifies the address offset for slot 7 in this payload register.

SIMD 64-Bit Address Payload Control

MACR_64B - SIMD 64-Bit Address Payload Control

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.1	63:0	**Offset0** Format: U64 Specifies the address offset for slot 0 in this payload register.
0.2-0.3	63:0	**Offset1** Format: U64 Specifies the address offset for slot 1 in this payload register.
0.4-0.5	63:0	**Offset2** Format: U64 Specifies the address offset for slot 2 in this payload register.
0.6-0.7	63:0	**Offset3** Format: U64 Specifies the address offset for slot 3 in this payload register.

SIMD8 32-Bit Address Payload

MAP32B_SIMD8 - SIMD8 32-Bit Address Payload		
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Offset[7:0]** Format: **MACR_32b** Specifies the address offset for Slots [7:0].

SIMD8 64-Bit Address Payload

MAP64B_SIMD8 - SIMD8 64-Bit Address Payload		
Size (in bits):	512	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Offset[3:0]** Format: **MACR_64b** Specifies the address offset for slots [3:0].
1.0-1.7	255:0	**Offset[7:4]** Format: **MACR_64b** Specifies the address offset for slots [7:4].

SIMD16 16-Bit Address Payload

MAP16B_SIMD16 - SIMD16 16-Bit Address Payload		
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Offset** Format: U16[16] Specifies the address offset for slots [15:0].

SIMD16 32-Bit Address Payload

MAP32B_SIMD16 - SIMD16 32-Bit Address Payload		
Size (in bits):	512	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-1.7	511:0	**Offset** Format: U32[16] Specifies the address offset for slots [15:0].

SIMD16 64-Bit Address Payload

MAP64B_SIMD16 - SIMD16 64-Bit Address Payload		
Size (in bits):	1024	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Offset[3:0]** Format: **MACR_64b** Specifies the address offsets for slots [3:0].
1.0-1.7	255:0	**Offset[7:4]** Format: **MACR_64b** Specifies the address offsets for slots [7:4].
2.0-2.7	255:0	**Offset[11:8]** Format: **MACR_64b** Specifies the address offsets for slots [11:8].
3.0-3.7	255:0	**Offset[15:12]** Format: **MACR_64b** Specifies the address offsets for slots [15:12].

SIMD Mode 2 Message Descriptor Control Field

MDC_SM2 - SIMD Mode 2 Message Descriptor Control Field

Size (in bits): 1

Default Value: 0x00000000

DWord	Bit	Description
0	0	**SIMD Mode** Specifies the SIMD mode of the message (number of slots processed)

Value	Name	Description
00h	SIMD8	SIMD8
01h	SIMD16	SIMD16

SIMD Mode 3 Message Descriptor Control Field

MDC_SM3 - SIMD Mode 3 Message Descriptor Control Field

Size (in bits): 2

Default Value: 0x00000000

DWord	Bit	Description
0	1:0	**SIMD Mode** Specifies the SIMD mode of the message (number of slots processed)

Value	Name	Description
00h	Reserved	Ignored
01h	SIMD16	SIMD16
02h	SIMD8	SIMD8
03h	Reserved	Ignored

SLICE_HASH_TABLE

SLICE_HASH_TABLE - SLICE_HASH_TABLE

Size (in bits): 1024

Default Value: 0x00000000, 0x00000000

Description

The slice hash table state is stored as an array tables (2 slices-8 slices), each of which contains the 32 DWords described here. 16x16 lookup table for slice indexed by lower bits of pixel block address. Each entry in the table indicates the **physical** slice_id to map that XY. If a slice is disabled, then it must not be present in the table. Entries in the table that point to disabled slice will be mapped to lowest enabled slice_id.

DWord	Bit	Description
0..31	1023:0	**Slice Hashing Table Entries** Format: SLICE_HASHING_TABLE_ENTRY[16] Each entry has the Enabled Physical Iz PipeID to map that [Y][X] pixel block address.

SLICE_HASHING_TABLE_ENTRY

SLICE_HASHING_TABLE_ENTRY		
Size (in bits): 64 Default Value: 0x00000000, 0x00000000		
DWord	**Bit**	**Description**
0	31:28	**Entry7** Format: U4
	27:24	**Entry6** Format: U4
	23:20	**Entry5** Format: U4
	19:16	**Entry4** Format: U4
	15:12	**Entry3** Format: U4
	11:8	**Entry2** Format: U4
	7:4	**Entry1** Format: U4
	3:0	**Entry0** Format: U4
1	31:28	**Entry15**
	27:24	**Entry14** Format: U4
	23:20	**Entry13** Format: U4
	19:16	**Entry12** Format: U4
	15:12	**Entry11** Format: U4
	11:8	**Entry10** Format: U4
	7:4	**Entry9** Format: U4
	3:0	**Entry8** Format: U4

SLM Block Message Header

MH_SLM_GO - SLM Block Message Header

Source: EuSubFunctionDataPort0

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0..1	63:0	**Reserved** Access: RO Format: MBZ
2	31:0	**Global Offset** Format: U32 Specifies the global element index into the buffer, in units of Hwords, Owords, Dwords, or Bytes (depending on the message). **Programming Notes** The Global Offset for Aligned Block operations is specified as a Dword-aligned byte offset (offset bits [1:0] = 0), or Oword-aligned byte offset (offset bits [3:0]=0), or Hword-aligned byte offset (offset bits [4:0]=0).
3..7	159:0	**Reserved** Access: RO Format: MBZ

Slot Group 2 Message Descriptor Control Field

MDC_SG2 - Slot Group 2 Message Descriptor Control Field

Size (in bits): 1

Default Value: 0x00000000

DWord	Bit	Description
0	0	**SIMD Mode** Controls which 8 bits of Pixel/Sample Mask in the message header are ANDed with the execution mask to determine which slots are accessed. This field is ignored if the header is not present.

Value	Name	Description
00h	SG8L	Use low 8 slots
01h	SG8U	Use high 8 slots

Slot Group 3 Message Descriptor Control Field

MDC_SG3 - Slot Group 3 Message Descriptor Control Field

Size (in bits): 2

Default Value: 0x00000000

DWord	Bit	Description
0	1:0	**SIMD Mode** Controls which 8 bits of Pixel/Sample Mask in the message header are ANDed with the execution mask to determine which slots are accessed. This field is ignored if the header is not present.

Value	Name	Description
00h	Reserved	Ignored
01h	SG8L	[] Use low 8 slots
02h	SG8U	Use high 8 slots
03h	Reserved	Ignored

Slot Group Select Render Cache Message Descriptor Control Field

MDC_RT_SGS - Slot Group Select Render Cache Message Descriptor Control Field

Size (in bits): 1

Default Value: 0x00000000

DWord	Bit	Description
0	0	**Slot Group Select** This field selects whether slots 15:0 or slots 31:16 are used for bypassed data. Bypassed data includes the antialias alpha, multisample coverage mask, and if the header is not present also includes the X/Y addresses and pixel enables. For 8- and 16-pixel dispatches, SLOTGRP_LO must be selected on every message. For 32-pixel dispatches, this field must be set correctly for each message based on which slots are currently being processed.

Value	Name	Description
00h	SLOTGRP_LO	Choose bypassed data for slots 15:0
01h	SLOTGRP_HI	Choose bypassed data for slots 31:16

SO_DECL

SO_DECL		
Source:	RenderCS	
Size (in bits):	16	
Default Value:	0x00000000	

A list of SO_DECL structures are passed in the 3DSTATE_SO_DECL_LIST command. Each structure specifies either (a) the source and destination of an up-to-4-DWord appending write into an SO buffer, or (b) how many DWords to skip over in the destination SO buffer (i.e., a "hole" where the previous buffer contents are maintained).

DWord	Bit	Description
0	15:14	**Reserved** Access: RO Format: MBZ
	13:12	**Output Buffer Slot** Format: U2 This field selects the destination output buffer slot.
	11	**Hole Flag** Format: Enable If set, the Component Mask field indirectly specifies a number of 32-bit locations to skip over (leave unmodified in memory) in the selected output buffer. The Register Index field is ignored. The only permitted Component Mask values are as follows: 0x0 No Dwords are skipped over (SO_DECL performs no operation) 0x1 (X) Skip 1 DWord 0x3 (XY) Skip 2 DWords 0x7 (XYZ) Skip 3 DWords 0xF (XYZW) Skip 4 DWords
	10	**Reserved** Access: RO Format: MBZ
	9:4	**Register Index** Format: U6 If Hole Flag is clear, this field specifies the 128-bit offset into the source vertex data which supplies the source data to be written to the destination buffer, where the individual 32-component destination locations are selected by Component Mask. e.g., Register Index 0 corresponds with the first 128 bits of the data read from the vertex URB entry (as per corresponding Vertex Read Offset state) There is only enough internal storage for the 128-bit vertex header and 32 128-bit vertex

attributes.

Value	Name
[0,32]	
0h	**[Default]**

Programming Notes
It is the responsibility of software to map any API-visible source data specifications (e.g., vertex register number) into 128-bit granular URB read offsets.

3:0 **Component Mask**

Format:	U4

This field is a 4-bit bitmask that selects which contiguous 32-bit component(s) are either written or skipped-over in the destination buffer.If this field is zero the SO_DECL operation is effectively a no-op. No data will be appended to the destination and the destination buffer's write pointer will not be advanced.If the **Hole Flag** is set, this field (if non-zero) indirectly specifies how much the destination buffer's write pointer should be advanced.See **Hole Flag** description above for restrictions on this field.If the **Hole Flag** is clear, this field (if non-zero) selects which source components are to be written to the destination buffer. The components must be contiguous, e.g. YZW is legal, but XZW is not. The selected source components are written to the destination buffer starting at the current write pointer, and then the write pointer is advanced past the written data. E.g., if YZW is specified, the three (YZW) components of the source register will be written to the destination buffer at the current write pointer, and the write pointer will be advanced by 3 DWords.

Value	Name
0h	SO_DECL_COMPMASK_NONE **[Default]**
xxx1b	SO_DECL_COMPMASK_X
xx1xb	SO_DECL_COMPMASK_Y
x1xxb	SO_DECL_COMPMASK_Z
1xxxb	SO_DECL_COMPMASK_W

SO_DECL_ENTRY

SO_DECL_ENTRY		
Source:	RenderCS	
Size (in bits):	64	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0..1	63:48	**Stream 3 Decl** Format: SO_DECL This field contains Stream 3 SO_DECL [n]
	47:32	**Stream 2 Decl** Format: SO_DECL This field contains Stream 2 SO_DECL [n]
	31:16	**Stream 1 Decl** Format: SO_DECL This field contains Stream 1 SO_DECL [n]
	15:0	**Stream 0 Decl** Format: SO_DECL This field contains Stream 0 SO_DECL [n]

Split_coding_unit_flags

Split_coding_unit_flags		
Source:	VideoCS	
Size (in bits):	21	
Default Value:	0x00000000	
Contains the split level flags, level 0 through 2.		
DWord	**Bit**	**Description**
0	20	**Split_flag_level0** Format: U1
	19:16	**Split_flag_level1** Format: U4 [19:16] is in raster order. Bit16 is for partition0 in raster order.
	15:12	**Split_flag_level2 level1part3** Format: U4 Split flags for bit19 partition. [15:12] is in raster order. Bit12 is for partition0 in raster order.
	11:8	**Split_flag_level2 level1part2** Format: U4 Split flags for bit18 partition. [11:8] is in raster order. Bit8 is for partition0 in raster order.
	7:4	**Split_flag_level2 level1part1** Format: U4 Split flags for bit17 partition. [7:4] is in raster order. Bit4 is for partition0 in raster order.
	3:0	**Split_flag_level2 level1part0** Format: U4 Split flags for bit16 partition. [3:0] is in raster order. Bit0 is for partition0 in raster order.

SplitBaseAddress4KByteAligned

SplitBaseAddress4KByteAligned		
Size (in bits):	64	
Default Value:	0x00000000, 0x00000000	
Specifies a 64-bit (48-bit canonical) 4K-byte aligned memory base address. GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and MBZ.		
Programming Notes		
Bits 63:48 must be zero.		
DWord	**Bit**	**Description**
0..1	63:12	**Base Address** Format: VIRTUAL_ADDR[63:12]
	11:0	**Reserved** Access: RO Format: MBZ

SplitBaseAddress64ByteAligned

SplitBaseAddress64ByteAligned		
Size (in bits):	64	
Default Value:	0x00000000, 0x00000000	
Specifies a 64-bit (48-bit canonical) 64-byte aligned memory base address.		
Programming Notes		
Bits 63:48 must be zero.		

DWord	Bit	Description
0..1	63:6	**Base Address** Format: VIRTUAL_ADDR[63:6]
	5:0	**Reserved** Access: RO Format: MBZ

Stateless Binding Table Index Message Descriptor Control Field

MDC_STATELESS - Stateless Binding Table Index Message Descriptor Control Field

Size (in bits): 8

Default Value: 0x00000000

DWord	Bit	Description
0	7:0	**Binding Table Index** Specifies the message is Stateless

Value	Name	Description
0FDh	A32_A64_NC	Specifies a A32 or A64 Stateless access that is non-coherent (coherent within a thread).
Others	Reserved	Ignored

Stateless Block Message Header

MH_A32_GO - Stateless Block Message Header

Source: EuSubFunctionDataPort0

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0..1	63:0	**Reserved** Access: RO Format: MBZ
2	31:0	**Global Offset** Format: U32 Specifies the global element index into the buffer, in units of Owords, Dwords, or Bytes (depending on the message). **Programming Notes** If the address offset calculated with the Buffer Base Address and Global Offset is greater than the PTSS size or the GeneralStateBufferSize, then the access is Out-of-Bounds.
3	31:0	**Per Thread Scratch Space** Format: MHC_PTSS Specifies amount of scratch space used by this thread, for Stateless bounds checking.
4	31:0	**Reserved** Access: RO Format: MBZ
5	31:0	**Buffer Base Address** Format: MHC_A32_BBA Specifies the surface address offset page [31:10] for A32 stateless messages. Restriction : When using stateless A32 Data Port messages, General State Base Address[47:12] + Buffer Base Address[31:10] must be less than 2^48. It is illegal for this to be greater or equal than 2^48.
6..7	63:0	**Reserved** Access: RO Format: MBZ

Stateless Surface Message Header

MH1_A32 - Stateless Surface Message Header

Source: EuSubFunctionDataPort1

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0..4	159:0	**Reserved** Access: RO Format: MBZ
5	31:0	**Buffer Base Address** Format: MHC_A32_BBA Specifies the surface address offset page [31:10] for A32 stateless messages.
6..7	63:0	**Reserved** Access: RO Format: MBZ

Stateless Surface Pixel Mask Message Header

MH1_A32_PSM - Stateless Surface Pixel Mask Message Header

Source:	EuSubFunctionDataPort1
Size (in bits):	256
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0..4	159:0	**Reserved** Access: RO Format: MBZ
5	31:0	**Buffer Base Address** Format: MHC_A32_BBA Specifies the surface address offset page [31:10] for A32 stateless messages.
6	31:0	**Reserved** Access: RO Format: MBZ
7	31:0	**Reserved** Access: RO Format: MBZ

Stencil Message Data Payload Register

MDPR_STENCIL - Stencil Message Data Payload Register

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:24	**Stencil3** Format: U8 Stencil for Slot 3.
	23:16	**Stencil2** Format: U8 Stencil for Slot 2.
	15:8	**Stencil1** Format: U8 Stencil for Slot 1.
	7:0	**Stencil0** Format: U8 Stencil for Slot 0.
1	31:24	**Stencil7** Format: U8 Stencil for Slot 7.
	23:16	**Stencil6** Format: U8 Stencil for Slot 6.
	15:8	**Stencil5** Format: U8 Stencil for Slot 5.
	7:0	**Stencil4** Format: U8 Stencil for Slot 4.
2..7	191:0	**Reserved** Access: RO Format: MBZ

Subset Atomic Integer Trinary Operation Message Descriptor Control Field

MDC_AOP3S - Subset Atomic Integer Trinary Operation Message Descriptor Control Field

Size (in bits): 4

Default Value: 0x0000000E

DWord	Bit	Description
0	3:0	**Atomic Integer Operation Type** Specifies the atomic integer trinary operation to be performed

Value	Name	Description
0Eh	AOP_CMPWR **[Default]**	new_dst = (src0 == old_dst) ? src1 : old_dst
Others	Reserved	Ignored

Programming Notes
When Return Data Control is set, old_dst is returned.

Subset Reversed SIMD Mode 2 Message Descriptor Control Field

MDC_SM2RS - Subset Reversed SIMD Mode 2 Message Descriptor Control Field

Size (in bits): 1

Default Value: 0x00000001

DWord	Bit	Description
0	0	**SIMD Mode** Specifies the SIMD mode of the message (number of slots processed)

Value	Name	Description
0h	Reserved	Not used
01h	SIMD8 **[Default]**	SIMD8

Subset SIMD Mode 2 Message Descriptor Control Field

MDC_SM2S - Subset SIMD Mode 2 Message Descriptor Control Field

Size (in bits): 1

Default Value: 0x00000000

DWord	Bit	Description
0	0	**SIMD Mode** Specifies the SIMD mode of the message (number of slots processed)

Value	Name	Description
00h	SIMD8	SIMD8
01h	Reserved	Reserved.

Subset SIMD Mode 3 Message Descriptor Control Field

MDC_SM3S - Subset SIMD Mode 3 Message Descriptor Control Field

Size (in bits): 2

Default Value: 0x00000000

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>1:0</td><td>SIMD Mode
Specifies the SIMD mode of the message (number of slots processed)
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>00h</td><td>Reserved</td><td>Ignored</td></tr>
<tr><td>01h</td><td>Reserved</td><td>Ignored</td></tr>
<tr><td>02h</td><td>SIMD8</td><td>SIMD8</td></tr>
<tr><td>03h</td><td>Reserved</td><td>Ignored</td></tr>
</table></td></tr>
</table>

Subspan Render Target Message Header Control

MHC_RT_SUBSPAN - Subspan Render Target Message Header Control

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:16	**Y** Format: U16 Y coordinate for upper-left pixel of this subspan
	15:0	**X** Format: U16 X coordinate for upper-left pixel of this subspan

Surface Binding Table Index Message Descriptor Control Field

MDC_BTS - Surface Binding Table Index Message Descriptor Control Field

Size (in bits): 8

Default Value: 0x00000000

DWord	Bit	Description
0	7:0	**Binding Table Index** Specifies the Binding Table index for the message, which must be a Surface State Model.

Value	Name	Description
00h-0EFh	BTS	Index of Binding Table State Surfaces
0F0h-0FAh	Reserved	Reserved for future use
0FCh	SSO_BINDLESS	Specifies a Surface State Offset into the Bindless Surface State heap, supplied by the extended message descriptor.
0FBh	Reserved	
Others	Reserved	Ignored

Programming Notes
For Render Target Views, the Binding Table index need to be confined to the 00h to 0Fh range if Binding Table is not in the 256B alignment (18:8) mode. In the 256B alignment mode, the Binding Table Index need to be confined to the 00h to 3Fh range if slice common register 3. state cache perf fix disabled is set to 1, the entire range of BTI is supported.

Surface or Stateless Binding Table Index Message Descriptor Control Field

MDC_BTS_A32 - Surface or Stateless Binding Table Index Message Descriptor Control Field

Size (in bits): 8

Default Value: 0x00000000

DWord	Bit	Description
0	7:0	**Binding Table Index** Specifies the surface for the message, either Surface State Model or Stateless.

Value	Name	Description
00h-0EFh	BTS	Index of Binding Table State Surfaces
0F0h-0FAh	Reserved	Reserved for future use
0FCh	SSO_BINDLESS	Specifies a Surface State Offset into the Bindless Surface State heap, supplied by the extended message descriptor.
0FBh	Reserved	
0FFh	A32_A64	Specifies a A32 or A64 Stateless access that is locally coherent (coherent within a thread group)
0FDh	A32_A64_NC	Specifies a A32 or A64 Stateless access that is non-coherent (coherent within a thread).
Others	Reserved	Ignored

Restriction
When using A32_A64_NC, SW must ensure that 2 threads do not both access the same cache line (64B)

Surface Pixel Mask Message Header

MH1_BTS_PSM - Surface Pixel Mask Message Header		
Source:	EuSubFunctionDataPort1	
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0..6	223:0	**Reserved** Access: RO Format: MBZ
7	31:0	**Reserved** Access: RO Format: MBZ

SW Generated BINDING_TABLE_STATE

SW Generated BINDING_TABLE_STATE

Size (in bits): 32

Default Value: 0x00000000

Description
The binding table binds surfaces to logical resource indices used by shaders and other compute engine kernels. It is stored as an array of up to 256 elements, each of which contains one dword as defined here. The start of each element is spaced one dword apart.
The first element of the binding table is aligned to a 64-byte boundary.
Binding table indexes beyond 256 will automatically be mapped to entry 0 by the HW, w/ the exception of any messages which support the special indexes 240 through 255, inclusive.

DWord	Bit	Description
0	31:6	**Surface State Pointer** Format: SurfaceStateOffset[31:6] This 64-byte aligned address points to a surface state block. This pointer is relative to the **Surface State Base Address**
	5:0	**Reserved** Access: RO Format: MBZ

SWSB

SWSB

Size (in bits): 8

Default Value: 0x00000000

DWord	Bit	Description
0	7	**Mode** This field specifies the way SWSB information is specified. SingleInfo: (Type, Value) or DualInfo: (Value, Value). **Value** / **Name** / **Description** 0 / SingleInfo **[Default]** / SingleInfo. Specifies the SWSB information as (Type, Value) pair. Where Value can be RegDist or SBID information. 1 / DualInfo / DualInfo. Specifies the SWSB information as (RegDistValue, SBIDValue) pair.
	6:4	**Type** Exists If: ([Mode]=='SingleInfo') This field specifies the type of SWSB information. **Value** / **Name** 000b / RegDist **[Default]** 001b / Reserved 010b / SBID.dst 011b / SBID.src 100b / SBID.set [101b-111b] / Reserved
	6:4	**RegDistInfo** Exists If: ([Mode]=='DualInfo') **Value** / **Name** 0 / Reserved [1-7] / RegDistInfo
	3	**Reserved** Exists If: ([Mode]=='SingleInfo') AND ([Type]=='RegDist') Format: MBZ
	3:0	**SBIDValue** Exists If: ([Mode]=='SingleInfo') AND (([Type]=='SBID.dst') OR ([Type]=='SBID.src') OR ([Type]=='SBID.set'))

SWSB

		Value / **Name** [0-15] / SBIDInfo
	3:0	**SBIDValue** Exists If: ([Mode]=='DualInfo') **Value** / **Name** [0-15] / SBIDInfo
	2:0	**RegDistValue** Exists If: ([Mode]=='SingleInfo') AND ([Type]=='RegDist') **Value** / **Name** 0 / No Dependency [1-7] / RegDistInfo

SZ OM S0A SIMD8 Render Target Data Payload

MDP_RTW_ZMA8 - SZ OM S0A SIMD8 Render Target Data Payload

Size (in bits): 1792

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Source 0 Alpha** Format: **MDP_DW_SIMD8** Slots [7:0] Source 0 Alpha
1.0-1.7	255:0	**oMask** Format: **MDPR_OMASK** Slots [7:0] oMask. Upper half ignored.
2.0-2.7	255:0	**Red** Format: **MDP_DW_SIMD8** Slots [7:0] Red
3.0-3.7	255:0	**Green** Format: **MDP_DW_SIMD8** Slots [7:0] Green
4.0-4.7	255:0	**Blue** Format: **MDP_DW_SIMD8** Slots [7:0] Blue
5.0-5.7	255:0	**Alpha** Format: **MDP_DW_SIMD8** Slots [7:0] Alpha
6.0-6.7	255:0	**Source Depth** Format: **MDP_DW_SIMD8** Slots [7:0] Source Depth

SZ OM S0A SIMD16 Render Target Data Payload

MDP_RTW_ZMA16 - SZ OM S0A SIMD16 Render Target Data Payload

Size (in bits):	3328
Default Value:	0x00000000, 0x00000000

DWord	Bit	Description
0.0-1.7	511:0	**Source 0 Alpha** Format: **MDP_DW_SIMD16** Slots [15:0] Source 0 Alpha
2.0-2.7	255:0	**oMask** Format: **MDPR_OMASK** Slots [15:0] oMask
3.0-4.7	511:0	**Red** Format: **MDP_DW_SIMD16** Slots [15:0] Red
5.0-6.7	511:0	**Green** Format: **MDP_DW_SIMD16** Slots [15:0] Green

MDP_RTW_ZMA16 - SZ OM S0A SIMD16 Render Target Data Payload		
7.0-8.7	511:0	**Blue** Format: MDP_DW_SIMD16 Slots [15:0] Blue
9.0-10.7	511:0	**Alpha** Format: MDP_DW_SIMD16 Slots [15:0] Alpha
11.0-12.7	511:0	**Source Depth** Format: MDP_DW_SIMD16 Slots [15:0] Source Depth

SZ OM SIMD8 Dual Source Render Target Data Payload

MDP_RTW_ZM8DS - SZ OM SIMD8 Dual Source Render Target Data Payload

Size (in bits): 2560

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**oMask** Format: **MDPR_OMASK** oMask for slots [7:0] and [15:8]. Operation selects upper or lower half.
1.0-1.7	255:0	**Src0 Red** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src0 Red
2.0-2.7	255:0	**Src0 Green** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src0 Green
3.0-3.7	255:0	**Src0 Blue** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src0 Blue
4.0-4.7	255:0	**Src0 Alpha** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src0 Alpha

MDP_RTW_ZM8DS - SZ OM SIMD8 Dual Source Render Target Data Payload		
5.0-5.7	255:0	**Src1 Red** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src1 Red
6.0-6.7	255:0	**Src1 Green** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src1 Green
7.0-7.7	255:0	**Src1 Blue** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src1 Blue
8.0-8.7	255:0	**Src1 Alpha** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src1 Alpha
9.0-9.7	255:0	**Source Depth** Format: **MDP_DW_SIMD8** Slots [7:0] or [15:8] of Source Depth

SZ OM SIMD8 Render Target Data Payload

MDP_RTW_ZM8 - SZ OM SIMD8 Render Target Data Payload

Size (in bits): 1536

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**oMask** Format: **MDPR_OMASK** Slots [7:0] oMask. Upper half ignored.
1.0-1.7	255:0	**Red** Format: **MDP_DW_SIMD8** Slots [7:0] Red
2.0-2.7	255:0	**Green** Format: **MDP_DW_SIMD8** Slots [7:0] Green
3.0-3.7	255:0	**Blue** Format: **MDP_DW_SIMD8** Slots [7:0] Blue
4.0-4.7	255:0	**Alpha** Format: **MDP_DW_SIMD8** Slots [7:0] Alpha
5.0-5.7	255:0	**Source Depth** Format: **MDP_DW_SIMD8** Slots [7:0] Source Depth

SZ OM SIMD16 Render Target Data Payload

MDP_RTW_ZM16 - SZ OM SIMD16 Render Target Data Payload		
Size (in bits):	2816	
Default Value:	0x00000000, 0x00000000	

DWord	Bit	Description
0.0-0.7	255:0	**oMask** Format: **MDPR_OMASK** Slots [15:0] oMask
1.0-1.7	255:0	**Red[7:0]** Format: **MDP_DW_SIMD8** Slots [7:0] Red
2.0-2.7	255:0	**Red[15:8]** Format: **MDP_DW_SIMD8** Slots [15:8] Red
3.0-3.7	255:0	**Green[7:0]** Format: **MDP_DW_SIMD8** Slots [7:0] Green
4.0-4.7	255:0	**Green[15:7]** Format: **MDP_DW_SIMD8** Slots [15:8] Green

MDP_RTW_ZM16 - SZ OM SIMD16 Render Target Data Payload		
5.0-5.7	255:0	**Blue[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Blue
6.0-6.7	255:0	**Blue[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Blue
7.0-7.7	255:0	**Alpha[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Alpha
8.0-8.7	255:0	**Alpha[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Alpha
9.0-9.7	255:0	**Source Depth[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Source Depth
10.0-10.7	255:0	**Source Depth[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Source Depth

SZ S0A SIMD8 Render Target Data Payload

MDP_RTW_ZA8 - SZ S0A SIMD8 Render Target Data Payload		
Size (in bits):	1536	
Default Value:	0x00000000, 0x00000000	

DWord	Bit	Description
0.0-0.7	255:0	**Source 0 Alpha** Format: **MDP_DW_SIMD8** Slots [7:0] Source 0 Alpha
1.0-1.7	255:0	**Red** Format: **MDP_DW_SIMD8** Slots [7:0] Red
2.0-2.7	255:0	**Green** Format: **MDP_DW_SIMD8** Slots [7:0] Green
3.0-3.7	255:0	**Blue** Format: **MDP_DW_SIMD8** Slots [7:0] Blue
4.0-4.7	255:0	**Alpha** Format: **MDP_DW_SIMD8** Slots [7:0] Alpha
5.0-5.7	255:0	**Source Depth** Format: **MDP_DW_SIMD8** Slots [7:0] Source Depth

SZ S0A SIMD16 Render Target Data Payload

MDP_RTW_ZA16 - SZ S0A SIMD16 Render Target Data Payload

Size (in bits): 3072

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Source 0 Alpha[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Source 0 Alpha
1.0-1.7	255:0	**Source 0 Alpha[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Source 0 Alpha
2.0-2.7	255:0	**Red[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Red
3.0-3.7	255:0	**Red[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Red
4.0-4.7	255:0	**Green[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Green

MDP_RTW_ZA16 - SZ S0A SIMD16 Render Target Data Payload		
5.0-5.7	255:0	**Green[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Green
6.0-6.7	255:0	**Blue[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Blue
7.0-7.7	255:0	**Blue[15:7]** Format: MDP_DW_SIMD8 Slots [15:8] Blue
8.0-8.7	255:0	**Alpha[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Alpha
9.0-9.7	255:0	**Alpha[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Alpha
10.0-10.7	255:0	**Source Depth[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Source Depth
11.0-11.7	255:0	**Source Depth[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Source Depth

SZ SIMD8 Dual Source Render Target Data Payload

MDP_RTW_Z8DS - SZ SIMD8 Dual Source Render Target Data Payload		
Size (in bits):	2304	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Src0 Red** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src0 Red
1.0-1.7	255:0	**Src0 Green** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src0 Green
2.0-2.7	255:0	**Src0 Blue** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src0 Blue
3.0-3.7	255:0	**Src0 Alpha** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src0 Alpha
4.0-4.7	255:0	**Src1 Red** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src1 Red
5.0-5.7	255:0	**Src1 Green** Format: MDP_DW_SIMD8 Slots[7:0] or [15:8] of Src1 Green

MDP_RTW_Z8DS - SZ SIMD8 Dual Source Render Target Data Payload		
6.0-6.7	255:0	**Src1 Blue** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src1 Blue
7.0-7.7	255:0	**Src1 Alpha** Format: **MDP_DW_SIMD8** Slots[7:0] or [15:8] of Src1 Alpha
8.0-8.7	255:0	**Source Depth** Format: **MDP_DW_SIMD8** Slots [7:0] or [15:8] of Source Depth

SZ SIMD8 Render Target Data Payload

MDP_RTW_Z8 - SZ SIMD8 Render Target Data Payload		
Size (in bits):	1280	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Red** Format: MDP_DW_SIMD8 Slots [7:0] Red
1.0-1.7	255:0	**Green** Format: MDP_DW_SIMD8 Slots [7:0] Green
2.0-2.7	255:0	**Blue** Format: MDP_DW_SIMD8 Slots [7:0] Blue
3.0-3.7	255:0	**Alpha** Format: MDP_DW_SIMD8 Slots [7:0] Alpha
4.0-4.7	255:0	**Source Depth** Format: MDP_DW_SIMD8 Slots [7:0] Source Depth

SZ SIMD16 Render Target Data Payload

MDP_RTW_Z16 - SZ SIMD16 Render Target Data Payload

Size (in bits):	2560
Default Value:	0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Red[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Red
1.0-1.7	255:0	**Red[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Red
2.0-2.7	255:0	**Green[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Green
3.0-3.7	255:0	**Green[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Green
4.0-4.7	255:0	**Blue[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Blue
5.0-5.7	255:0	**Blue[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Blue

MDP_RTW_Z16 - SZ SIMD16 Render Target Data Payload		
6.0-6.7	255:0	**Alpha[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Alpha
7.0-7.7	255:0	**Alpha[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Alpha
8.0-8.7	255:0	**Source Depth[7:0]** Format: MDP_DW_SIMD8 Slots [7:0] Source Depth
9.0-9.7	255:0	**Source Depth[15:8]** Format: MDP_DW_SIMD8 Slots [15:8] Source Depth

Thread EOT Message Descriptor

TS_EOT - Thread EOT Message Descriptor

Source:	RenderCS
Size (in bits):	32
Default Value:	0x02000000

End of Thread message is sent to SFID_TS (07h) to end GPGPU and Media threads. The EU send instruction must also set the EOT control (bit 5) of the extended message descriptor.
This message is sent with single register message payload, which is a copy of the R0 thread payload sent with the thread dispatch.

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Default Value: 1h One GRF Format: U4
	24:20	**Response Length** Default Value: 0h Zero GRF Format: U5
	19	**Header Present** Format: MBZ
	18:1	**Reserved** Access: RO Format: MBZ
	0	**Message Type** Default Value: 0h End Thread Format: Opcode End of Thread message opcode

TILE_RECT

TILE_RECT

Source: RenderCS, PositionCS

Size (in bits): 64

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0	31:16	**Tile Rectangle Y Min** Format: U16 Specifies Y Min coordinate of (inclusive) Tile Region used for tile rendering test. Value: [0,16383] — Name:
	15:0	**Tile Rectangle X Min** Format: U16 Specifies X Min coordinate of (inclusive) Tile Region used for tile rendering test. Value: [0,16383] — Name:
1	31:16	**Tile Rectangle Y Max** Format: U16 Specifies Y Max coordinate of (inclusive) Tile Region used for tile rendering test. Value: [0,16383] — Name:
	15:0	**Tile Rectangle X Max** Format: U16 Specifies X Max coordinate of (inclusive) Tile Region used for tile rendering test. Value: [0,16383] — Name:

Timeout Data Payload

MDP_TIMEOUT - Timeout Data Payload

Source:	EuSubFunctionGateway
Size (in bits):	256
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:10	**Reserved** Access: RO Format: MBZ
	9:0	**Timeout Value** Format: U10 **Description** The amount of time GW should wait before sending a writeback message. This value is in terms of 1024 clocks. Thus, with a 1Ghz clock it would be approximately in terms of uS. 0 and 1 are illegal values since the actual timeout time can be short by up to 1 increment of the timeout value.
1..7	223:0	**Reserved** Access: RO Format: MBZ

U24_X8

U24_X8 - U24_X8		
Size (in bits):	32	
Default Value:	0x00000000	
32bit packed 24Unorm and 8bit of unused format (named X8)		
DWord	**Bit**	**Description**
0	31:8	**MSB_U24** Format: U24
	7:0	**LSB_X8** Format: U8

Untyped Write Channel Mask Message Descriptor Control Field

MDC_UW_CMASK - Untyped Write Channel Mask Message Descriptor Control Field

Size (in bits): 4

Default Value: 0x00000000

DWord	Bit	Description
0	3:0	**Mask** For untyped surface write messages, indicates which channels are included in the message payload and written to the surface.

Value	Name	Description
00h	RGBA **[Default]**	Red, Green, Blue, and Alpha are included
08h	RGB	Red, Green, and Blue are included
0Ch	RG	Red and Green are included
0Eh	R	Red is included
Others	Reserved	Ignored

Upper Oword Block Data Payload

MDP_OW1U - Upper Oword Block Data Payload		
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.3	127:0	**Reserved** Access: RO Format: MBZ
0.4-0.7	127:0	**Oword** Format: U128 Specifies the upper Oword data element

URB Channel Mask Payload Control

MACD_URB_CMASK - URB Channel Mask Payload Control

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**Channel Mask** Format: Enable[8] For each channel present in the message data payload, the corresponding channel mask bit is ANDed with the slot's execution mask to determine the final channel enable. When final channel enable is 1 it indicates that Dword data will be written to the surface.
	15:0	**Reserved** Access: RO Format: MBZ

URB Handle Message Header

MH_URB_HANDLE - URB Handle Message Header		
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0 This is the URB handle where slot 0 results are written or read.	31:0	**Handle 0** Format: MHC_URB_HANDLE
0.1 This is the URB handle where slot 1 results are written or read.	31:0	**Handle 1** Format: MHC_URB_HANDLE
0.2 This is the URB handle where slot 2 results are written or read.	31:0	**Handle 2** Format: MHC_URB_HANDLE
0.3 This is the URB handle where slot 3 results are written or read.	31:0	**Handle 3** Format: MHC_URB_HANDLE
0.4 This is the URB handle where slot 4 results are written or read.	31:0	**Handle 4** Format: MHC_URB_HANDLE
0.5 This is the URB handle where slot 5 results are written or read.	31:0	**Handle 5** Format: MHC_URB_HANDLE
0.6 This is the URB handle where slot 6 results are written or read.	31:0	**Handle 6** Format: MHC_URB_HANDLE
0.7 This is the URB handle where slot 7 results are written or read.	31:0	**Handle 7** Format: MHC_URB_HANDLE

URB Handle Message Header Control

MHC_URB_HANDLE - URB Handle Message Header Control		
Size (in bits): 32		
Default Value: 0x00000000		
DWord	**Bit**	**Description**
0	31:25	**Reserved** Access: RO Format: MBZ
	24:0	**Reserved** Access: RO Format: MBZ

VC1

VC1		
Source:	VideoCS	
Size (in bits):	16	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	15:8	**Reserved** Access: RO Format: MBZ
	7	**Syncmarker Error** This flag indicates missing sync marker SEs coded in the bit-stream.
	6	**Mbmode SE Error** This flag indicates inconsistent Macroblock SEs coded in the bit-stream.
	5	**Transformtype SE Error** This flag indicates inconsistent transform type SEs coded in the bit-stream.
	4	**Coefficient Error** This flag indicates inconsistent Coefficient SEs coded in the bit-stream.
	3	**Motion Vector SE Error** This flag indicates inconsistent Motion Vector SEs coded in the bit-stream.
	2	**Coded Block Pattern CY SE Error** This flag indicates inconsistent CBPCY SEs coded in the bit-stream.
	1	**Mquant Error** This flag indicates inconsistent MQUANT SEs coded in the bit-stream.
	0	**MB Concealment Flag** . Each pulse from this flag indicates one MB is concealed by hardware.

VCS Hardware-Detected Error Bit Definitions

VCS Hardware-Detected Error Bit Definitions

Source: VideoCS

Size (in bits): 16

Default Value: 0x00000000

DWord	Bit	Description
0	15:12	**Reserved** Access: RO Format: MBZ
	11	**Reserved** Access: RO Format: MBZ
	10:3	**Reserved** Access: RO Format: MBZ
	2	**Command Privilege Violation Error** This bit is set if a command classified as privileged is parsed in a non-privileged batch buffer. The command will be converted to a NOOP and parsing will continue.
	1	**Reserved** Access: RO Format: MBZ
	0	**Instruction Error** This bit is set when the Renderer Instruction Parser detects an error while parsing an instruction.Instruction errors include: • Client ID value (Bits 31:29 of the Header) is not supported (only MI, 2D and 3D are supported). • Defeatured MI Instruction Opcodes: Value: 1 — Name: — Description: Instruction Error detected **Programming Notes** This error indications cannot be cleared except by reset (i.e., it is a fatal error).

VD_CONTROL_STATE_BODY

VD_CONTROL_STATE_BODY		
Size (in bits):	64	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0	31:30	**Reserved** Access: RO Format: MBZ
	29	**Reserved**
	28	**Reserved**
	27:1	**Reserved** Access: RO Format: MBZ
	0	**Pipeline Initialization** **Description** This bit, when set, clears internal states for HCP Pipe if Media Instruction Opcode is set for HCP Pipe.. This bit, when set, clears internal states for AVP Pipe if Media Instruction Opcode is set for AVP Pipe.
1	31:3	**Reserved** Access: RO Format: MBZ
	2	**Memory Implicit Flush** This is used to initiate an implicit flush to memory to make sure all the memory request goes to memory. This should be programmed at the end of each frame after frame completion and before MI_FLUSH.
	1	**Scalable Mode Pipe Unlock** This is used for decoder/encoder pipe to unlock all the pipes for scalable mode. It should be programmed at the end of frame.
	0	**Scalable Mode Pipe Lock** This is used for decoder/encoder pipe to lock all the pipes for scalable mode. It should be programmed at the start of frame.

VDENC_64B_Aligned_Lower_Address

VDENC_64B_Aligned_Lower_Address		
Source:	VideoCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:6	**Address** Format: GraphicsAddress[31:6] [31:6] This field is for the 26 bits of the lower address.
	5:0	**Reserved** Access: RO Format: MBZ

VDENC_64B_Aligned_Upper_Address

VDENC_64B_Aligned_Upper_Address		
Source:	VideoCS	
Size (in bits):	32	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Address Upper DWord** Bits [47:32] of the Address. GraphicsAddress is 64-bit value [63:0], but only a portion of it is used by hardware. The uppermost bits [63:48] are ignored and MBZ.

VDENC_Block_8x8_4

VDENC_Block_8x8_4		
Source:	VideoCS	
Size (in bits):	16	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	15:12	**Block8x8[3]** Format: U4
	11:8	**Block8x8[2]** Format: U4
	7:4	**Block8x8[1]** Format: U4
	3:0	**Block8x8[0]** Format: U4

VDENC_Colocated_MV_Picture

VDENC_Colocated_MV_Picture		
Source:	VideoCS	
Size (in bits):	96	
Default Value:	0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0	31:0	**Lower Address** Format: VDENC_64B_Aligned_Lower_Address 64 byte aligned buffer. This field is used to write the DMV data by VDEnc. VDEnc only supports spatial direct prediction andnot temporal direct. Hence the HW precomputes the ColZeroFlag per 8x8 block and writes 8-bits per macroblock.HW accumulates a CL worth of data before writing it out. This is a linear buffer, can be considered tobe a frame level row-store. There is no read/write happening to the surface for any given frame. HW only writes to this surface for P-Frames. HW only reads from this surface for B-Frames. Size = 8-bits/ MB linear buffer.
1	31:0	**Upper Address** Format: VDENC_64B_Aligned_Upper_Address
2	31:0	**Picture Fields** Format: VDENC_Surface_Control_Bits

VDENC_Delta_MV_XY

VDENC_Delta_MV_XY

Source: VideoCS

Size (in bits): 128

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000

16x16	16x08		08x16		08x08	
0	1		3	4	5	6
	2				7	8

Calculates the difference between the actual MV for the Sub Macroblock and the predicted MV based onthe availability of the neighbors.

Programming Notes

This is calculated and populated for Inter frames only. In case of an Intra MB in Inter frames, thisvalue should be 0.

DWord	Bit	Description
0..3	127:112	**Y3** Default Value: 0 Format: U16 Mb_type: 8x8, PartID: 8
	111:96	**X3** Default Value: 0 Format: U16 Mb_type: 8x8, PartID: 8
	95:80	**Y2** Default Value: 0 Format: U16 Mb_type: 8x8, PartID: 7
	79:64	**X2** Default Value: 0 Format: U16

VDENC_Delta_MV_XY

		Mb_type / **PartID**: 8x8 / 7
	63:48	**Y1** Default Value: 0 Format: U16 **Mb_type** / **PartID**: 16x8 / 2; 8x16 / 4; 8x8 / 6
	47:32	**X1** Default Value: 0 Format: U16 **Mb_type** / **PartID**: 16x8 / 2; 8x16 / 4; 8x8 / 6
	31:16	**Y0** Default Value: 0 Format: U16 **Mb_type** / **PartID**: 16x16 / 0; 16x8 / 1; 8x16 / 3; 8x8 / 5
	15:0	**X0** Default Value: 0 Format: U16 **Mb_type** / **PartID**: 16x16 / 0; 16x8 / 1; 8x16 / 3; 8x8 / 5

VDENC_Down_Scaled_Reference_Picture

VDENC_Down_Scaled_Reference_Picture		
Source:	VideoCS	
Size (in bits):	96	
Default Value:	0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0	31:0	**Lower Address** Format: VDENC_64B_Aligned_Lower_Address Specifies the 64 byte aligned DownScaled reference frame buffer address.VDEnc supports upto 3 down-scaled reference pictures for HME search. (2 fwd and 1 bwd).
1	31:0	**Upper Address** Format: VDENC_64B_Aligned_Upper_Address
2	31:0	**Picture Fields** Format: VDENC_Surface_Control_Bits

VDENC_Original_Uncompressed_Picture

VDENC_Original_Uncompressed_Picture		
Source:	VideoCS	
Size (in bits):	96	
Default Value:	0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0	31:0	**Lower Address** Format: VDENC_64B_Aligned_Lower_Address Specifies the 64 byte aligned frame buffer address for fetching YUV pixel data from the originaluncompressed input picture for encoding. This field is only valid in encoding mode.
1	31:0	**Upper Address** Format: VDENC_64B_Aligned_Upper_Address
2	31:0	**Picture Fields** Format: VDENC_Surface_Control_Bits

VDENC_Reference_Picture

VDENC_Reference_Picture		
Source:	VideoCS	
Size (in bits):	96	
Default Value:	0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0	31:0	**Lower Address** Format: VDENC_64B_Aligned_Lower_Address Specifies the 64 byte aligned reference frame buffer addresses corresponding to fwd reference index = 0in the bitstream. VDEnc supports upto 4 reference pictures for IME search. (3 fwd and 1 bwd).
1	31:0	**Upper Address** Format: VDENC_64B_Aligned_Upper_Address
2	31:0	**Picture Fields** Format: VDENC_Surface_Control_Bits

VDENC_Row_Store_Scratch_Buffer_Picture

VDENC_Row_Store_Scratch_Buffer_Picture		
Source:	VideoCS	
Size (in bits):	96	
Default Value:	0x00000000, 0x00000000, 0x00000000	
Structure_VDENC_Row_Store_Scratch_Buffer_Picture: true		
DWord	**Bit**	**Description**
0	31:0	**Lower Address** Format: **VDENC_64B_Aligned_Lower_Address** This field provides the base address of the scratch buffer (read/write) used by VDEnc to store MBinformation of the previous row for processing of each macroblock in the current row.The Row Store buffer must be 64-byte cacheline aligned. Hardware uses the horizontal address of thecurrent macroblock to address the Row Store. **AVC:** Size = 256 bits/MB. 4K wide picture needs 128 CLs.
1	31:0	**Upper Address** Format: **VDENC_64B_Aligned_Upper_Address**
2	31:0	**Buffer Picture Fields** Format: **VDENC_Surface_Control_Bits**

VDENC_Sub_Mb_Pred_Mode

VDENC_Sub_Mb_Pred_Mode		
Source:	VideoCS	
Size (in bits):	8	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	7:6	**SubMbPredMode[3]** Format: U2
	5:4	**SubMbPredMode[2]** Format: U2
	3:2	**SubMbPredMode[1]** Format: U2
	1:0	**SubMbPredMode[0]** Format: U2

VDENC_Surface_Control_Bits

VDENC_Surface_Control_Bits

Source: VideoCS
Size (in bits): 32
Default Value: 0x00000000

DWord	Bit	Description
0	31:21	**Reserved** Format: MBZ
	20:16	**Reserved** Format: MBZ
	15	**Reserved** Format: MBZ
	14:13	**Tiled Resource Mode** Format: U2 **For Media Surfaces**: This field specifies the tiled resource mode. Only supports 4K tiling for all surfaces.

Value	Name	Description
0h	TRMODE_NONE	No tiled resource.
1h	TRMODE_TILEYF	4KB tiled resources
2h	TRMODE_TILEYS	64KB tiled resources
3h	Reserved	

Programming Notes
This field should be programmed the same for all these VDEnc surfaces listed below. **DS FWD REF0, DS FWD REF1, DS BWD REF0. FWD REF0, FWD REF1, FWD REF2, BWD REF0.**

12	**Cache Select** Exists If: (Property[Structure_VDENC_Row_Store_Scratch_Buffer_Picture] == 'true') Format: U1

Description
This field controls if the Row Store is going to store inside Media Cache (rowstore cache) or to LLC.
When this is programmed to "1" (going to Media Cache), the corresponding base address will be programmed with the starting position in the media cache. The programming table is in "Buffer Size Requirement Page" in HEVC section

VDENC_Surface_Control_Bits

Value	Name	Description
0		Buffer going to LLC.
1		Buffer going to Internal Media Storage.

11 **Reserved**

Access:	RO
Format:	MBZ

10 **Compression Type**

Value	Name
0	Media Compression Enabled **[Default]**
1	Render Compression Enabled

9 **Memory Compression Enable**

Format:	Enable

Memory compression will be attempted for this surface.

Value	Name
0h	Disable
1h	Enable

Programming Notes
This field should be programmed the same for all these VDEnc surfaces listed below. **DS FWD REF0, DS FWD REF1, DS BWD REF0. FWD REF0, FWD REF1, FWD REF2, BWD REF0.**

8:7 **Arbitration Priority Control**

Format:	U2

This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.

Value	Name	Description
00b	Highest priority	
01b	Second highest priority	
10b	Third highest priority	
11b	Lowest priority	

6:1 **Index to Memory Object Control State (MOCS) Tables:**

Format:	U6

The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers.
The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.

0 **Reserved**

VDENC_Surface_State_Fields

VDENC_Surface_State_Fields

Source: VideoCS

Size (in bits): 128

Default Value: 0x00000000, 0x00000003, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:18	**Height** Format: U14-1 This field specifies the height of the Picture in units of pixels. For PLANAR surface formats,this field indicates the height of the Y (luma) plane. Value: [0,16383] — Name: — Description: Representing heights [1,16384] **Programming Notes** AVC specific Note: • When surface tiling is TileY, the Frame Height needs to be programmed as 16-pixel aligned. • When surface tiling is Linear (supported only for Source surface), the Frame Height can be programmed 2-pixel aligned.
	17:4	**Width** Format: U14-1 This field specifies the width of the Picture in units of pixels/residuals. For PLANAR surface formats,this field indicates the width of the Y (luma) plane. Value: [0,16383] — Name: — Description: Representing widths [1,16384] **Programming Notes** • The Width specified by this field multiplied by the pixel size in bytes must be less than or equal to the surface pitch (specified in bytes via the Surface Pitch field). • Width (field value + 1) must be a multiple of 2 for PLANAR_420, • VDEnc HW does not use this field, the picture width is read from IMG State instead, because this field may not equal to the actual picture width. This field is used by the KMD to allocate surface in GTT.
	3	**Color space selection** Format: U1 0 — Use BT.601 Space conversion 1 — Use BT.709 Space conversion

VDENC_Surface_State_Fields

2 — Reserved

Access:	RO
Format:	MBZ

1:0 — Cr(V)/Cb(U) Pixel Offset V Direction

Format:	U0.2

Exactly as shown in the original spec.

Specifies the distance to the U/V values with respect to the even numbered Y channels in the V direction.

Programming Notes

This field is ignored for all formats except PLANAR_420_8.

1 — 31:27 — Reserved

Access:	RO
Format:	MBZ

26:23 — Reserved

Access:	RO
Format:	MBZ

22:20 — Chroma Downsample Filter Control

Format:	U3

Value	Left Side Tap	Center Tap	Right Side Tap
0	0	64	0
1	16	32	16
2	15	34	15
3	14	36	14
4	13	38	13
5	12	40	12
6	11	42	11
7	0	32	32

Programming Notes

For Tile Y 444 -> 420, Filter settings on 0 and 7 are valid. All other combinations are invalid. This is true for 10 bit and 8 bit.

19:3 — Surface Pitch

Format:	U17

-1 pitch in Bytes

VDENC_Surface_State_Fields

This field specifies the surface pitch in (#Bytes).

Programming Notes
For tiled surfaces, the pitch must be a multiple of the tile width (i.e.128 bytes aligned). If Half Pitch for Chroma is set, this field must be a multiple of two tile widths for tiled surfaces, or a multiple of 2 bytes for linear surfaces. For Y-tiled surfaces: Range = [127,131071] to [128B,128KB] = [1 tile, 1028 tiles].
For TileYF and TileYS surfaces, the range is dependent on the Cu parameter (refer to Memory Data Formats section for the definition of the Cu parameter depending on the case). The range in bytes is [2^{Cu}-1, 131071] -> [(2^{Cu})B, 128KB] = [1 tile, 128KB/($2^{Cu)\ tiles}$] The field specifies the surface pitch in (#Bytes - 1)

2 **Half Pitch for Chroma**

Format:	Enable

(This field must be set to Disable.) This field indicates that the chroma plane(s) will use a pitch equal to half the value specified in the Surface Pitch field. This field is only used for PLANAR surface formats. This field is ignored by VDEnc (unless we support YV12).

Value	Name	Description
0	Disable **[Default]**	
1	Enable	

1 **Tiled Surface**

Format:	Boolean

(This field must be set to TRUE: Tiled.) This field specifies whether the surface is tiled. This field is ignored by VDEnc usage.

Value	Name	Description
0	False	Linear
1	True **[Default]**	Tiled

Programming Notes
Linear surfaces can be mapped to Main Memory (uncached) or System Memory (cacheable, snooped).Tiled surfaces can only be mapped to Main Memory. The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit.

0 **Tile Walk**

Format:	Boolean

(This field must be set to 1: TILEWALK_YMAJOR.) This field specifies the type of memory tiling(XMajor or YMajor) employed to tile this surface. See Memory Interface Functions for detailson memory tiling and restrictions.This field is ignored when the surface is linear. InternallyH/W always treats this as set to 1 for all VDEnc usage.

Value	Name	Description
0h	XMAJOR	TILEWALK_XMAJOR

		1h	YMAJOR **[Default]**	TILEWALK_YMAJOR

Programming Notes
The corresponding cache(s) must be invalidated before a previously accessed surface is accessed againwith an altered state of this bit.

2	31	**Reserved**		
		Access:	RO	
		Format:	MBZ	
	30:16	**X Offset for U(Cb)**		
		Format:	U15	

Pixel Offset
This field specifies the horizontal offset in pixels from the Surface Base Address to the start(origin) of the U(Cb) plane or the interleaved UV plane if Interleave Chroma is enabled. This fieldis only used for PLANAR surface formats. This field must be set to zero. X Offset for U(Cb) in pixel.(This field must be zero for NV12 and IMC 1 and 3).

Programming Notes
For PLANAR_420 and PLANAR_422 surface formats, this field must be zero.

	15	**Reserved**		
		Access:	RO	
		Format:	MBZ	
	14:0	**Y Offset for U(Cb)**		
		Format:	U15	

Pixel Row Offset
This field specifies the vertical offset in rows from the Surface Base Address to the start(origin) of the U(Cb) plane or the interleaved UV plane if Interleave Chroma is enabled. This fieldis only used for PLANAR surface formats.

Programming Notes
For PLANAR_420 and PLANAR_422 surface formats, this field must be multiple of 8 pixels - i.e. multiple MBs.

3	31:29	**Reserved**		
		Access:	RO	
		Format:	MBZ	
	28:16	**X Offset for V(Cr)**		
		Format:	U13	

VDENC_Surface_State_Fields

		Offset in Pixels This field must be zero for NV12 and IMC 1 and 3. This field specifies the horizontal offset in pixels from the Surface Base Address to the start (origin)of the V(Cr) plane. This field is only used for PLANAR surface formats with Interleave Chroma disabled. **Programming Notes** For PLANAR_420 and PLANAR_422 surface formats, this field must indicate an even number of pixels.
	15:0	**Y Offset for V(Cr)** Format: U16 Row Offset in Pixels This field specifies the vertical offset in rows from the Surface Base Address to the start (origin)of the V(Cr) plane. This field is only used for PLANAR surface formats with Interleave Chroma disabled.This field is ignored by all video codec, only used by JPEG. **Programming Notes** For PLANAR_420 surface formats, this field must be multiple of 16 pixels - i.e. multiple MBs.

VEBOX_ACE_LACE_STATE

VEBOX_ACE_LACE_STATE

Source: VideoEnhancementCS

Size (in bits): 992

Default Value: 0x00000068, 0x4C382410, 0x9C887460, 0xEBD8C4B0, 0x604C3824, 0xB09C8874, 0x0000D8C4, 0x04000400, 0x04000400, 0x04000400, 0x04000400, 0x04000400, 0x00000400, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

This state structure contains the IECP State Table Contents for ACE state.

DWord	Bit	Description
0	31:16	**Min_ACE_luma** Format: U16
	15:14	**LACE Single Histogram Set** This bit tells LACE which frames will be included in the histogram when the Deinterlacer is enabled. Value / Name / Description: 00b / Current / The histogram includes only the current frame. 01b / Previous / The histogram includes only the previous frame. 10b / Current + Previous / The histogram includes pixels from both the current and previous frame. 11b / Previous + Current / The histogram includes the previous frame followed by the current frame. **Programming Notes** When the Deinterlacer is disabled, this field must be 00b.If **DI Output Frames** is set to only output a single field then the histogram can not be collected on the disabled field.This Field must be set to 00b when DN/DI First Frame is set to 1
	13	**LACE Histogram Size** Value / Name: 0 / 128-bin histogram 1 / 256-bin histogram
	12	**LACE Histogram Enable** Default Value: 0 This bit enables the collection of LACE histogram data. If this bit is 0 then only the ACE histogram will be collected.

VEBOX_ACE_LACE_STATE

	11:7	**Reserved** Access: RO Format: MBZ
	6:2	**Skin Threshold** Format: U5 Used for Y analysis (min/max) for pixels which are higher than skin threshold. Value: [1,31] Value: 26 **[Default]**
	1	**Reserved** Access: RO Format: MBZ
	0	**ACE Enable** Format: Enable
1	31:24	**Y3** Default Value: 76 Format: U8 The value of the y_pixel for point 3 in PWL.
	23:16	**Y2** Default Value: 56 Format: U8 The value of the y_pixel for point 2 in PWL.
	15:8	**Y1** Default Value: 36 Format: U8 The value of the y_pixel for point 1 in PWL.
	7:0	**Ymin** Default Value: 16 Format: U8 The value of the y_pixel for point 0 in PWL.
2	31:24	**Y7** Default Value: 156 Format: U8 The value of the y_pixel for point 7 in PWL.

VEBOX_ACE_LACE_STATE

	23:16	**Y6** Default Value: 136 Format: U8 The value of the y_pixel for point 6 in PWL.
	15:8	**Y5** Default Value: 116 Format: U8 The value of the y_pixel for point 5 in PWL.
	7:0	**Y4** Default Value: 96 Format: U8 The value of the y_pixel for point 4 in PWL.
3	31:24	**Ymax** Default Value: 235 Format: U8 The value of the y_pixel for point 11 in PWL.
	23:16	**Y10** Default Value: 216 Format: U8 The value of the y_pixel for point 10 in PWL.
	15:8	**Y9** Default Value: 196 Format: U8 The value of the y_pixel for point 9 in PWL.
	7:0	**Y8** Default Value: 176 Format: U8 The value of the y_pixel for point 8 in PWL.
4	31:24	**B4** Default Value: 96 Format: U8 The value of the bias for point 4 in PWL.

VEBOX_ACE_LACE_STATE

	23:16	**B3** Default Value: 76 Format: U8 The value of the bias for point 3 in PWL.
	15:8	**B2** Default Value: 56 Format: U8 The value of the bias for point 2 in PWL.
	7:0	**B1** Default Value: 36 Format: U8 The value of the bias for point 1 in PWL.
5	31:24	**B8** Default Value: 176 Format: U8 The value of the bias for point 8 in PWL.
	23:16	**B7** Default Value: 156 Format: U8 The value of the bias for point 7 in PWL.
	15:8	**B6** Default Value: 136 Format: U8 The value of the bias for point 6 in PWL.
	7:0	**B5** Default Value: 116 Format: U8 The value of the bias for point 5 in PWL.
6	31:16	**Reserved** Access: RO Format: MBZ

VEBOX_ACE_LACE_STATE

	15:8	**B10** Default Value: 216 Format: U8 The value of the bias for point 10 in PWL.
	7:0	**B9** Default Value: 196 Format: U8 The value of the bias for point 9 in PWL.
7	31:27	**Reserved** Access: RO Format: MBZ
	26:16	**S1** Default Value: 1024 Format: U1.10 The value of the slope for point 1 in PWL The default is 1024/1024
	15:11	**Reserved** Access: RO Format: MBZ
	10:0	**S0** Default Value: 1024 Format: U1.10 The value of the slope for point 0 in PWL The default is 1024/1024
8	31:27	**Reserved** Access: RO Format: MBZ
	26:16	**S3** Default Value: 1024 Format: U1.10 The value of the slope for point 3 in PWL The default is 1024/1024

VEBOX_ACE_LACE_STATE

	15:11	**Reserved** Access: RO Format: MBZ
	10:0	**S2** Default Value: 1024 Format: U1.10 The value of the slope for point 2 in PWL The default is 1024/1024
9	31:27	**Reserved** Access: RO Format: MBZ
	26:16	**S5** Default Value: 1024 Format: U1.10 The value of the slope for point 5 in PWL The default is 1024/1024
	15:11	**Reserved** Access: RO Format: MBZ
	10:0	**S4** Default Value: 1024 Format: U1.10 The value of the slope for point 4 in PWL The default is 1024/1024
10	31:27	**Reserved** Access: RO Format: MBZ
	26:16	**S7** Default Value: 1024 Format: U1.10 The value of the slope for point 7 in PWL The default is 1024/1024

VEBOX_ACE_LACE_STATE

	15:11	**Reserved** Access: RO Format: MBZ
	10:0	**S6** Default Value: 1024 Format: U1.10 The default is 1024/1024
11	31:27	**Reserved** Access: RO Format: MBZ
	26:16	**S9** Default Value: 1024 Format: U1.10 The value of the slope for point 9 in PWL The default is 1024/1024
	15:11	**Reserved** Access: RO Format: MBZ
	10:0	**S8** Default Value: 1024 Format: U1.10 The value of the slope for point 8 in PWL The default is 1024/1024
12	31:16	**Max_ACE_luma** Format: U16 The maximum luma for which ACE correction will be used.
	15:11	**Reserved** Access: RO Format: MBZ
	10:0	**S10** Default Value: 1024 Format: U1.10 The value of the slope for point 10 in PWL.

VEBOX_ACE_LACE_STATE

13	31	**LACE Color Correction Enable** Enables LACE Color Correction
	30:16	**Reserved** Access: RO Format: MBZ
	15:0	**LACE Y Offset** Format: U16 Y offset for LACE Color Correction
14	31:16	**LACE V Offset** Format: U16 V offset for LACE Color Correction
	15:0	**LACE U Offset** Format: U16 U offset for LACE Color Correction
15	31:21	**LACE gamma curve slope 0** Format: U1.10 Slope value for segment 0
	20:8	**LACE gamma curve bias 0** Format: U5.8 Bias value for segment 0
	7:0	**LACE gamma curve point 0** Format: U5.3 Point value for segment 0 **Programming Notes** This value must be 0
16	31:21	**LACE gamma curve slope 1** Format: U1.10 Slope value for segment 1
	20:8	**LACE gamma curve bias 1** Format: U5.8 Bias value for segment 1
	7:0	**LACE gamma curve point 1** Format: U5.3 Point value for segment 1

VEBOX_ACE_LACE_STATE		
17	31:21	**LACE gamma curve slope 2** Format: U1.10 Slope value for segment 2
	20:8	**LACE gamma curve bias 2** Format: U5.8 Bias value for segment 2
	7:0	**LACE gamma curve point 2** Format: U5.3 Point value for segment 2
18	31:21	**LACE gamma curve slope 3** Format: U1.10 Slope value for segment 3
	20:8	**LACE gamma curve bias 3** Format: U5.8 Bias value for segment 3
	7:0	**LACE gamma curve point 3** Format: U5.3 Point value for segment 3
19	31:21	**LACE gamma curve slope 4** Format: U1.10 Slope value for segment 4
	20:8	**LACE gamma curve bias 4** Format: U5.8 Bias value for segment 4
	7:0	**LACE gamma curve point 4** Format: U5.3 Point value for segment 4
20	31:21	**LACE gamma curve slope 5** Format: U1.10 Slope value for segment 5

VEBOX_ACE_LACE_STATE		
	20:8	**LACE gamma curve bias 5** Format: U5.8 Bias value for segment 5
	7:0	**LACE gamma curve point 5** Format: U5.3 Point value for segment 5
21	31:21	**LACE gamma curve slope 6** Format: U1.10 Slope value for segment 6
	20:8	**LACE gamma curve bias 6** Format: U5.8 Bias value for segment 6
	7:0	**LACE gamma curve point 6** Format: U5.3 Point value for segment 6
22	31:21	**LACE gamma curve slope 7** Format: U1.10 Slope value for segment 7
	20:8	**LACE gamma curve bias 7** Format: U5.8 Bias value for segment 7
	7:0	**LACE gamma curve point 7** Format: U5.3 Point value for segment 7
23	31:21	**LACE gamma curve slope 8** Format: U1.10 Slope value for segment 8
	20:8	**LACE gamma curve bias 8** Format: U5.8 Bias value for segment 8

VEBOX_ACE_LACE_STATE		
	7:0	**LACE gamma curve point 8** Format: U5.3 Point value for segment 8
24	31:21	**LACE gamma curve slope 9** Format: U1.10 Slope value for segment 9
	20:8	**LACE gamma curve bias 9** Format: U5.8 Bias value for segment 9
	7:0	**LACE gamma curve point 9** Format: U5.3 Point value for segment 9
25	31:21	**LACE gamma curve slope 10** Format: U1.10 Slope value for segment 10
	20:8	**LACE gamma curve bias 10** Format: U5.8 Bias value for segment 10
	7:0	**LACE gamma curve point 10** Format: U5.3 Point value for segment 10
26	31:21	**LACE gamma curve slope 11** Format: U1.10 Slope value for segment 11
	20:8	**LACE gamma curve bias 11** Format: U5.8 Bias value for segment 11
	7:0	**LACE gamma curve point 11** Format: U5.3 Point value for segment 11

VEBOX_ACE_LACE_STATE		
27	31:21	**LACE gamma curve slope 12** Format: U1.10 Slope value for segment 12
	20:8	**LACE gamma curve bias 12** Format: U5.8 Bias value for segment 12
	7:0	**LACE gamma curve point 12** Format: U5.3 Point value for segment 12
28	31:21	**LACE gamma curve slope 13** Format: U1.10 Slope value for segment 13
	20:8	**LACE gamma curve bias 13** Format: U5.8 Bias value for segment 13
	7:0	**LACE gamma curve point 13** Format: U5.3 Point value for segment 13
29	31:21	**LACE gamma curve slope 14** Format: U1.10 Slope value for segment 14
	20:8	**LACE gamma curve bias 14** Format: U5.8 Bias value for segment 14
	7:0	**LACE gamma curve point 14** Format: U5.3 Point value for segment 14
30	31:21	**LACE gamma curve slope 15** Format: U1.10 Slope value for segment 15

VEBOX_ACE_LACE_STATE		
	20:8	**LACE gamma curve bias 15** Format: U5.8 Bias value for segment 15
	7:0	**LACE gamma curve point 15** Format: U5.3 Point value for segment 15 **Programming Notes** This value must be 0xff

VEBOX_ALPHA_AOI_STATE

VEBOX_ALPHA_AOI_STATE

Source: VideoEnhancementCS
Size (in bits): 96
Default Value: 0x00000000, 0x00000000, 0x00000000

This state structure contains the IECP State Table Contents for Fixed Alpha State and Area of Interest State.

DWord	Bit	Description
0	31:18	**Reserved** Access: RO Format: MBZ
	17	**Full Image Histogram** Default Value: 0 Format: Enable Used to ignore the area of interest for a histogram across the full image. This applies to all statistics that are affected by AOI (Area of Interest).
	16	**Alpha from State Select** Format: U1 Value / Name: 0 – alpha is taken from message 1 – alpha is taken from state **Programming Notes** If the input format does not have alpha available and the output format provides alpha, this bit should be set to 1. This should be 0 when Alpha Plane Enable is 1.
	15:0	**Color Pipe Alpha** Format: U16 **Programming Notes** The 8 MSB of this field will be used for output formats that have 8-bits of alpha.
1	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**AOI Max X** Default Value: 0 Format: U14

VEBOX_ALPHA_AOI_STATE

		Area of Interest Minimum X - The ACE histogram and Skin Tone Detection statistic gathering will occur within the MinX/MinY to MaxX/MaxY area (inclusive). AOI must intersect the frame such that at least 1 pixel is in the AOI. The Area of Interest applies to the RGB Histogram and the White/Gray point sums as well. **Programming Notes** **This value must be a multiple of 4 minus 1.**
	15:14	**Reserved** Access: RO Format: MBZ
	13:0	**AOI Min X** Default Value: 0 Format: U14 **Programming Notes** **This value must be a multiple of 4.**
2	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**AOI Max Y** Default Value: 0 Format: U14 **Programming Notes** **This value must be a multiple of 4 minus 1.**
	15:14	**Reserved** Access: RO Format: MBZ
	13:0	**AOI Min Y** Default Value: 0 Format: U14 **Programming Notes** **This value must be a multiple of 4.**

VEBOX_CAPTURE_PIPE_STATE

VEBOX_CAPTURE_PIPE_STATE

Source:	VideoEnhancementCS
Size (in bits):	224
Default Value:	0x8511FF23, 0xAA64AFAA, 0xE6FD4000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

This command contains variables for controlling Demosaic and the White Balance Statistics.

DWord	Bit	Description
0	31:30	**DirMap_Scale** Default Value: 2 Format: U2
	29:24	**Good Pixel Threshold** Default Value: 5h Format: U6 The difference threshold between adjacent pixels for a pixel to be considered "good".
	23	**Reserved** Access: RO Format: MBZ
	22:20	**Shift Min Cost** Default Value: 1h Format: U3 The amount to shift the H2/V2 versions of min_cost.
	19:16	**Green Imbalance Threshold** Default Value: 1h Format: U4
	15:8	**Average Color Threshold** Default Value: FFh Format: U8 The threshold between two colors in a pixel for the Avg interpolation to be considered. **Programming Notes** Must be set to 255.
	7:6	**Reserved** Access: RO Format: MBZ

	5:0	**Good Pixel Neighbor Threshold** Default Value: 23h Format: U6 Number of comparisons with neighbor pixels which pass before a pixel is considered good.
1	31:28	**Scale For Min Cost** Default Value: Ah The amount to scale the min_cost difference during the confidence check.
	27:24	**Good Intesity Threshold** Default Value: Ah Format: U4
	23:16	**Bad Color Threshold 1** Default Value: 64h Format: U8 Color value threshold used during the bad pixel check.
	15:8	**Bad Color Threshold 2** Default Value: AFh Format: U8 Color value threshold used during the bad pixel check.
	7:4	**Number Big Pixel Threshold** Default Value: Ah Format: U4 Number of comparisons with neighbor pixels which pass before a pixel is considered good.
	3:0	**Bad Color Threshold 3** Default Value: Ah Format: U4 Color value threshold used during the bad pixel check.
2	31:24	**Y Bright Value** Default Value: E6h The whitepoint threshold percentile in the Y histogram. Any pixel with Y value above this could be a whitepoint. This is the larger of the calculated Ybright value and the Ythreshold value, which is the minimum Y required to be considered a white point. **Programming Notes** "00000000" is appended to the LSBs before comparing with Y.

	23:16	**Y Outlier Value** Default Value: FDh The outlier threshold percentile in the Y histogram. Any pixel with Y value above this either clipped or an outlier in the image. These points will not be included in the white patch calculation. **Programming Notes** "00000000" is appended to the LSBs before comparing with Y.
	15:8	**UV Threshold Value** The value denotes the maximum threshold of the ratio between U+V to Y can have to be considered a gray point. Value / Name / Description: [255,0] / / Encode a value from 255/256 to 0/256 64 / **[Default]** / 0.25 * 255 = 64
	7	**Black Point Offset Red MSB**
	6	**Black Point Offset Green Top MSB**
	5	**Black Point Offset Blue MSB**
	4	**Black Point Offset Green Bottom MSB**
	3	**RGB Histogram Enable** Enables the collection of RGB Histograms for Auto-white balance correction and other uses. **Programming Notes** This bit can be set without White Balance enable being set.
	2	**Vignette Correction Format** Defines what shift should be assumed for the **Vignette** Correction input values: Value / Name: 0 / U8.8 1 / U4.12
	1	**Black Point Correction Enable** Format: Enable
	0	**White Balance Correction Enable** Format: Enable **Programming Notes** RGB Histogram enable must be set if this bit is set.
3	31:16	**Black Point Offset Red** Default Value: 0 Format: U16 Value subtracted from Red pixels of Bayer pattern - combined with MSB to form a 2's complement signed number.

VEBOX_CAPTURE_PIPE_STATE

	15:0	**Black Point Offset Green Top** Default Value: 0 Format: U16 Value subtracted from the top Green pixels of Bayer pattern (X=1, Y=0 for Bayer Pattern #1) - combined with MSB to form a 2's complement signed number.
4	31:16	**Black Point Offset Blue** Default Value: 0 Format: U16 Value subtracted from Blue pixels of Bayer pattern - Combine with MSB to form a 2's complement signed number.
	15:0	**Black Point Offset Green Bottom** Default Value: 0 Format: U16 Value subtracted from the bottom Green pixels of Bayer pattern (X=0, Y=1 for Bayer Pattern #1) - combined with MSB to form a 2's complement signed number.
5	31:16	**White Balance Red Correction** Format: U4.12 The correction factor multiplied by the Red pixels of the Bayer pattern.
	15:0	**White Balance Green Top Correction** Format: U4.12 The correction factor multiplied by the top Green pixels of the Bayer pattern(X=1, Y=0 for Bayer Pattern #1).
6	31:16	**White Balance Blue Correction** Format: U4.12 The correction factor multiplied by the Blue pixels of the Bayer pattern.
	15:0	**White Balance Green Bottom Correction** Format: U4.12 The correction factor multiplied by the bottom Green pixels of the Bayer pattern (X=0, Y=1 for Bayer Pattern #1)

VEBOX_CCM_STATE

VEBOX_CCM_STATE

Source:	VideoEnhancementCS
Size (in bits):	480
Default Value:	0x00004750, 0x0000AE80, 0x00000470, 0x00000220, 0x01FFFCC0, 0x0000D230, 0x00000A80, 0x01FFFF40, 0x0000D6A0, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

This state structure contains the IECP State Table Contents for the Color Correction Matrix State.

DWord	Bit	Description
0	31	**Color Correction Matrix Enable** Format: Enable This bit enables the Color Correction Matrix. **Programming Notes** Single Pipe IECP Enable must also be set if this bit is enabled.
	30:27	**Reserved** Access: RO Format: MBZ
	26:0	**C1** Default Value: 0004750h = 18256/65536 Format: S4.22 Coefficient of 3x3 Transform matrix. In HDR mode, bits [26:22] form the signed integer portion and bits [21:0] form the fraction portion of the coefficient. In non-HDR mode, bits [26:22] form the signed integer portion and bits [21:6] form the fraction portion of the coefficient.
1	31:27	**Reserved** Access: RO Format: MBZ
	26:0	**C0** Default Value: 000AE80h = 44672/65536 Format: S4.22 Coefficient of 3x3 Transform matrix. In HDR mode, bits [26:22] form the signed integer portion and bits [21:0] form the fraction portion of the coefficient. In non-HDR mode, bits [26:22] form the signed integer portion and bits [21:6] form the fraction portion of the coefficient.
2	31:27	**Reserved** Access: RO Format: MBZ

VEBOX_CCM_STATE

	26:0	**C3** Default Value: 0000470h = 1136/65536 Format: S4.22 Coefficient of 3x3 Transform matrix. In HDR mode, bits [26:22] form the signed integer portion and bits [21:0] form the fraction portion of the coefficient. In non-HDR mode, bits [26:22] form the signed integer portion and bits [21:6] form the fraction portion of the coefficient.
3	31:27	**Reserved** Access: RO Format: MBZ
	26:0	**C2** Default Value: 0000220h = 544/65536 Format: S4.22 Coefficient of 3x3 Transform matrix. In HDR mode, bits [26:22] form the signed integer portion and bits [21:0] form the fraction portion of the coefficient. In non-HDR mode, bits [26:22] form the signed integer portion and bits [21:6] form the fraction portion of the coefficient.
4	31:27	**Reserved** Access: RO Format: MBZ
	26:0	**C5** Default Value: 1FFFCC0h = -832/65536 Format: S4.22 Coefficient of 3x3 Transform matrix. In HDR mode, bits [26:22] form the signed integer portion and bits [21:0] form the fraction portion of the coefficient. In non-HDR mode, bits [26:22] form the signed integer portion and bits [21:6] form the fraction portion of the coefficient.
5	31:27	**Reserved** Access: RO Format: MBZ
	26:0	**C4** Default Value: 000D230h = 53808/65536 Format: S4.22 Coefficient of 3x3 Transform matrix. In HDR mode, bits [26:22] form the signed integer portion and bits [21:0] form the fraction portion of the coefficient. In non-HDR mode, bits [26:22] form the signed integer portion and bits [21:6] form the fraction portion of the coefficient.
6	31:27	**Reserved** Access: RO Format: MBZ

VEBOX_CCM_STATE

	26:0	**C7** Default Value: 0000A80h = 2688/65536 Format: S4.22 Coefficient of 3x3 Transform matrix. In HDR mode, bits [26:22] form the signed integer portion and bits [21:0] form the fraction portion of the coefficient. In non-HDR mode, bits [26:22] form the signed integer portion and bits [21:6] form the fraction portion of the coefficient.
7	31:27	**Reserved** Access: RO Format: MBZ
	26:0	**C6** Default Value: 1FFFF40h = -192/65536 Format: S4.22 Coefficient of 3x3 Transform matrix. In HDR mode, bits [26:22] form the signed integer portion and bits [21:0] form the fraction portion of the coefficient. In non-HDR mode, bits [26:22] form the signed integer portion and bits [21:6] form the fraction portion of the coefficient.
8	31:27	**Reserved** Access: RO Format: MBZ
	26:0	**C8** Default Value: 000D6A0h = 54944/65536 Format: S4.22 Coefficient of 3x3 Transform matrix. In HDR mode, bits [26:22] form the signed integer portion and bits [21:0] form the fraction portion of the coefficient. In non-HDR mode, bits [26:22] form the signed integer portion and bits [21:6] form the fraction portion of the coefficient.
9	31:0	**Offset_in_R** Default Value: 0 Format: S31 The input offset for red component.In HDR mode, the range of the value is $-2^{31}$to $2^{31}-1$. In non-HDR mode,the range of the value is $-2^{16}$to $2^{16}-1$
10	31:0	**Offset_in_G** Default Value: 0 Format: S31 The input offset for green component.In HDR mode, the range of the value is $-2^{31}$to $2^{31}-1$. In non-HDR mode,the range of the value is $-2^{16}$to $2^{16}-1$

<table>
<tr><th colspan="3">VEBOX_CCM_STATE</th></tr>
<tr><td>11</td><td>31:0</td><td>Offset_in_B<table><tr><td>Default Value:</td><td>0</td></tr><tr><td>Format:</td><td>S31</td></tr></table>The input offset for blue component.In HDR mode, the range of the value is -$2^{31}$to 2^{31}-1. In non-HDR mode,the range of the value is -$2^{16}$to 2^{16}-1</td></tr>
<tr><td>12</td><td>31:0</td><td>Offset_out_R<table><tr><td>Default Value:</td><td>0</td></tr><tr><td>Format:</td><td>S31</td></tr></table>The output offset for red component.In HDR mode, the range of the value is -$2^{31}$to 2^{31}-1. In non-HDR mode,the range of the value is -$2^{16}$to 2^{16}-1</td></tr>
<tr><td>13</td><td>31:0</td><td>Offset_out_G<table><tr><td>Default Value:</td><td>0</td></tr><tr><td>Format:</td><td>S31</td></tr></table>The output offset for green component.In HDR mode, the range of the value is -$2^{31}$to 2^{31}-1. In non-HDR mode,the range of the value is -$2^{16}$to 2^{16}-1</td></tr>
<tr><td>14</td><td>31:0</td><td>Offset_out_B<table><tr><td>Default Value:</td><td>0</td></tr><tr><td>Format:</td><td>S31</td></tr></table>The output offset for blue component.In HDR mode, the range of the value is -$2^{31}$to 2^{31}-1. In non-HDR mode,the range of the value is -$2^{16}$to 2^{16}-1</td></tr>
</table>

VEBOX_Ch_Dir_Filter_Coefficient

VEBOX_Ch_Dir_Filter_Coefficient		
Size (in bits):	64	
Default Value:	0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0..1	63:56	**Filter Coefficient[7]** Format: S1.6 **Range:** [-2, +2)
	55:48	**Filter Coefficient[6]** Format: S1.6 **Range:** [-2, +2)
	47:40	**Filter Coefficient[5]** Format: S1.6 **Range:** [-2, +2)
	39:32	**Filter Coefficient[4]** Format: S1.6 **Range:** [-2, +2)
	31:24	**Filter Coefficient[3]** Format: S1.6 **Range:** [-2, +2)
	23:16	**Filter Coefficient[2]** Format: S1.6 **Range:** [-2, +2)
	15:8	**Filter Coefficient[1]** Format: S1.6 **Range:** [-2, +2)
	7:0	**Filter Coefficient[0]** Format: S1.6 **Range:** [-2, +2)

VEBOX_CSC_STATE

VEBOX_CSC_STATE		
Source:	VideoEnhancementCS	
Size (in bits):	384	
Default Value:	0x00010000, 0x00000000, 0x00000000, 0x00000000, 0x00010000, 0x00000000, 0x00000000, 0x00000000, 0x00010000, 0x00000000, 0x00000000, 0x00000000	
This state structure contains the IECP State Table Contents for CSC state.		
DWord	**Bit**	**Description**
0	31	**Transform Enable** Format: Enable
	30:19	**Reserved** Access: RO Format: MBZ
	18:0	**C0** Default Value: 10000h or 1.0 Format: S2.16 Transform coefficient.
1	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**C1** Default Value: 0 Format: S2.16 Transform coefficient.
2	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**C2** Default Value: 0 Format: S2.16 Transform coefficient.
3	31:19	**Reserved** Access: RO Format: MBZ

VEBOX_CSC_STATE

	18:0	**C3** Default Value: 0 Format: S2.16 Transform coefficient.
4	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**C4** Default Value: 10000h or 1.0 Format: S2.16 Transform coefficient.
5	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**C5** Default Value: 0 Format: S2.16 Transform coefficient.
6	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**C6** Default Value: 0 Format: S2.16 Transform coefficient.
7	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**C7** Default Value: 0 Format: S2.16 Transform coefficient.

<table>
<tr><th colspan="3">VEBOX_CSC_STATE</th></tr>
<tr><td rowspan="2">8</td><td>31:19</td><td>Reserved<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>18:0</td><td>C8<table><tr><td>Default Value:</td><td>10000h or 1.0</td></tr><tr><td>Format:</td><td>S2.16</td></tr></table>Transform coefficient. The offset value is multiplied by 2 before being added to the output.</td></tr>
<tr><td rowspan="2">9</td><td>31:16</td><td>Offset Out 1<table><tr><td>Default Value:</td><td>0</td></tr><tr><td>Format:</td><td>S15</td></tr></table>Offset in for Y/R. The offset value is multiplied by 2 before being added to the output.</td></tr>
<tr><td>15:0</td><td>Offset in 1<table><tr><td>Default Value:</td><td>0</td></tr><tr><td>Format:</td><td>S15</td></tr></table>Offset in for Y/R. The offset value is multiplied by 2 before being added to the output.</td></tr>
<tr><td rowspan="2">10</td><td>31:16</td><td>Offset Out 2<table><tr><td>Default Value:</td><td>0</td></tr><tr><td>Format:</td><td>S15</td></tr></table>Offset out for U/G. The offset value is multiplied by 2 before being added to the output.</td></tr>
<tr><td>15:0</td><td>Offset in 2<table><tr><td>Default Value:</td><td>0</td></tr><tr><td>Format:</td><td>S15</td></tr></table>Offset out for U/G. The offset value is multiplied by 2 before being added to the output.</td></tr>
<tr><td rowspan="2">11</td><td>31:16</td><td>Offset Out 3<table><tr><td>Default Value:</td><td>0</td></tr><tr><td>Format:</td><td>S15</td></tr></table>Offset out for V/B. The offset value is multiplied by 2 before being added to the output.</td></tr>
<tr><td>15:0</td><td>Offset in 3<table><tr><td>Default Value:</td><td>0</td></tr><tr><td>Format:</td><td>S15</td></tr></table>Offset out for V/B. The offset value is multiplied by 2 before being added to the output.</td></tr>
</table>

VEBOX_DNDI_STATE

VEBOX_DNDI_STATE

Source: VideoEnhancementCS

Size (in bits): 1568

Default Value: 0x00000000, 0x80000000, 0x00000400, 0x00000000, 0x00000000, 0x00000000, 0x000000A0, 0x00000000, 0x0800F000, 0x0F025800, 0x00007800, 0x2D000000, 0x00000254, 0x00000000, 0x00000400, 0x05FF05FF, 0x00000000, 0x005064A5, 0x00000000, 0x00000000, 0x1314640F, 0x00000000, 0xFF804020, 0x5DC48F64, 0x190F8DA4, 0x03FFFFFF, 0x00043CA0

This state table is used by the *Denoise and Deinterlacer functions*

DWord	Bit	Description
0	31:17	**Denoise STAD Threshold** Format: U15 Threshold for denoise sum of temporal absolute differences.
	16:8	**Reserved** Access: RO Format: MBZ
	7:0	**Denoise Maximum History** Format: U8 Maximum allowed value for denoise history. Value: [128,240] — Name:
1	31:28	**Denoise History increase** Amount that denoise_history is increased by. MAX:15 8h — **[Default]** 15 — Maximum Allowed
	27:23	**Denoise Moving Pixel Threshold** Format: U5 Value: [0,16] — Name:

VEBOX_DNDI_STATE

	22:12	**Reserved** Access: RO Format: MBZ
	11:0	**Denoise ASD Threshold** Format: U12 Threshold for denoise absolute sum of differences. Value: [0,1023] — Name:
2	31:20	**Temporal Difference Threshold** Format: U12 **Programming Notes** 0 < (Temporal Difference Threshold - Low Temporal Difference Threshold) <= 256 except when both thresholds are set to 0.
	19:11	**Reserved** Access: RO Format: MBZ
	10:5	**Initial Denoise History** Default Value: 32 Format: U6 **Programming Notes** Initial value for Denoise history for both Luma and Chroma
	4:0	**Reserved** Access: RO Format: MBZ
3	31:20	**Low Temporal Difference Threshold** Format: U12 0 < (Temporal Difference Threshold - Low Temporal Difference Threshold) <= 256 except when both thresholds are set to 0.
	19:12	**Reserved** Access: RO Format: MBZ
	11	**Reserved** Access: RO Format: MBZ

VEBOX_DNDI_STATE

	10	**Progressive DN** Format: Enable Indicates that the denoise algorithm should assume progressive input when filtering neighboring pixels. **This bit must be set if the input to Denoise is RGB.** Value / Name / Description: 0 — DN assumes interlaced video and filters alternate lines together 1 — DN assumes progressive video and filters neighboring lines together **Programming Notes** **DI Enable** must be disabled when this field is enabled.
	9:2	**Hot Pixel Count Luma** Format: U8 Number of neighboring pixels different more than **Hot Pixel Threshold** before a pixel is considered hot. Value / Name: [0,8] **Programming Notes** 0 will cause all pixels to be considered hot and will perform a median filter on the entire image.
	1:0	**Reserved** Access: RO Format: MBZ
4	31:20	**Denoise Threshold for Sum of Complexity Measure Luma** Format: U12
	19:12	**Hot Pixel Threshold Luma** Format: U8 Threshold for a difference from the value of a neighboring pixel. Is shifted up to 16-bits before compare.
	11:0	**Block Noise Estimate Noise Threshold** Format: U12 Threshold for noise maximum/minimum. Value / Name: [0,4095]
5	31:17	**Chroma Denoise STAD Threshold** Format: U15 Threshold for denoise sum of temporal absolute differences.

VEBOX_DNDI_STATE

	16	**Reserved** Access: RO Format: MBZ
	15:8	**Hot Pixel Threshold Chroma U** Format: U8 Threshold for a difference from the value of a neighboring pixel. Is shifted up to 16-bits before compare.
	7:0	**Hot Pixel Count Chroma U** Format: U8 Number of neighboring pixels different more than **Hot Pixel Threshold** before a pixel is considered hot
6	31:20	**Chroma Temporal Difference Threshold** Format: U12 0 < (Chroma Temporal Difference Threshold - Chroma Low Temporal Difference Threshold) <=256 except when both thresholds are set to 0
	19:12	**Reserved** Access: RO Format: MBZ
	11:1	**Block Noise Estimate Edge Threshold** Default Value: 80 Threshold for detecting an edge in block noise estimate.
	0	**Chroma Denoise Enable** Format: Enable **Value** / **Name** / **Description** 0 / / The U and V channels will be passed to the next stage after DN unchanged. 1 / / The U and V chroma channels will be denoise filtered.
7	31:20	**Chroma Low Temporal Difference Threshold** Format: U12 0 < (Chroma Temporal Difference Threshold - Chroma Low Temporal Difference Threshold) <= 256 except when both thresholds are set to 0
	19:16	**Reserved** Access: RO Format: MBZ

VEBOX_DNDI_STATE

	15:8	**Hot Pixel Threshold Chroma V** Format: U8 Threshold for a difference from the value of a neighboring pixel. Is shifted up to 16-bits before compare.
	7:0	**Hot Pixel Count Chroma V** Format: U8 Number of neighboring pixels different more than **Hot Pixel Threshold** before a pixel is considered hot
8	31:29	**Reserved** Access: RO Format: MBZ
	28:24	**Chroma Denoise Moving Pixel Threshold** Format: U5
	23:12	**Chroma Denoise ASD Threshold** Format: U12 Threshold for denoise absolute sum of differences.
	11:0	**Chroma Denoise Threshold for Sum of Complexity Measure**
9	31:30	**Reserved** Access: RO Format: MBZ
	29:25	**DnY_Wr5[4:0]** Weight to be applied when: th4 <= (difference in luma, Bayer or RGB value)
	24:20	**DnY_Wr4[4:0]** Weight to be applied when: th3 <= (difference in luma, Bayer or RGB value) < th4
	19:15	**DnY_Wr3[4:0]** Weight to be applied when: th2 <= (difference in luma, Bayer or RGB value) < th3
	14:10	**DnY_Wr2[4:0]** Weight to be applied when: th1 <= (difference in luma, Bayer or RGBvalue) < th2
	9:5	**DnY_Wr1[4:0]** Weight to be applied when: th0 <= (difference in luma, Bayer or RGB value) < th1
	4:0	**DnY_Wr0[4:0]** Weight to be applied when: (difference in luma, Bayer or RGB value) < th0
10	31:29	**Reserved** Access: RO Format: MBZ

VEBOX_DNDI_STATE		
	28:16	**DnY_thmax[12:0]** Maximum threshold value for luma, Bayer or RGB
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**DnY_thmin[12:0]** Minimum threshold value
11	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**DnY_prt5[12:0]**
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**DnY_dyn_thmin[12:0]** Minimum Dynamic threshold value
12	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**DnY_prt4[12:0]** Multiplied by thrscale and then used as the threshold for comparing the luma or RGB differences.
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**DnY_prt3[12:0]**
13	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**DnY_prt2[12:0]**
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**DnY_prt1[12:0]**
14	31:29	**Reserved** Access: RO Format: MBZ

VEBOX_DNDI_STATE

	28:16	**DnY_prt0[12:0]**
	15	**Reserved** Access: RO Format: MBZ
	14:10	**DnY_wd22[4:0]** Weight to be applied to the 4 luma, Bayer or RGB pixels that are at X2 and Y2
	9:5	**DnY_wd21[4:0]** Weight to be applied to the 4 luma, Bayer or RGB pixels that are at X1 and Y2
	4:0	**DnY_wd20[4:0]** Weight to be applied to the 2 luma, Bayer or RGB pixels that are at X and Y2
15	31:30	**Reserved** Access: RO Format: MBZ
	29:25	**DnY_wd12[4:0]** Weight to be applied to the 4 luma, Bayer or RGB pixels that are at X2 and Y1
	24:20	**DnY_wd11[4:0]** Weight to be applied to the 4 luma, Bayer or RGB pixels that are at X1 and Y1
	19:15	**DnY_wd10[4:0]** Weight to be applied to the 2 luma, Bayer or RGB pixels that are at X and Y1
	14:10	**DnY_wd02[4:0]** Weight to be applied to the 2 luma, Bayer or RGB pixels that are at X2 and Y
	9:5	**DnY_wd01[4:0]** Weight to be applied to the 2 luma, Bayer or RGB pixels that are at X1 and Y
	4:0	**DnY_wd00[4:0]** Weight to be applied to the 1 luma, Bayer or RGB pixels that are at X and Y
16	31:30	**Reserved** Access: RO Format: MBZ
	29:25	**DnU_Wr5[4:0]** Weight to be applied when: th4 <= (difference in chroma U value)
	24:20	**DnU_Wr4[4:0]** Weight to be applied when: th3 <= (difference in chroma U value) < th4
	19:15	**DnU_Wr3[4:0]** Weight to be applied when: th2 <= (difference in chroma U value) < th3
	14:10	**DnU_Wr2[4:0]** Weight to be applied when: th1 <= (difference in chroma Uvalue) < th2
	9:5	**DnU_Wr1[4:0]** Weight to be applied when: th0 <= (difference in chroma U value) < th1
	4:0	**DnU_Wr0[4:0]** Weight to be applied when: (difference in chroma U value) < th0

VEBOX_DNDI_STATE		
17	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**DnU_thmax[12:0]** Maximum threshold value for chroma U
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**DnU_thmin[12:0]** Minimum threshold value
18	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**DnU_prt5[12:0]**
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**DnU_dyn_thmin[12:0]** Minimum Dynamic threshold value.
19	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**DnU_prt4[12:0]** Multiplied by thrscale and then used as the threshold for comparing chroma U differences.
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**DnU_prt3[12:0]**
20	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**DnU_prt2[12:0]**
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**DnU_prt1[12:0]**

VEBOX_DNDI_STATE		
21	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**DnU_prt0[12:0]**
	15	**Reserved** Access: RO Format: MBZ
	14:10	**DnU_wd22[4:0]** Weight to be applied to the 4 chroma U pixels that are at X2 and Y2
	9:5	**DnU_wd21[4:0]** Weight to be applied to the 4 chroma U pixels that are at X1 and Y2
	4:0	**DnU_wd20[4:0]** Weight to be applied to the 2 chroma U pixels that are at X and Y2
22	31:30	**Reserved** Access: RO Format: MBZ
	29:25	**DnU_wd12[4:0]** Weight to be applied to the 4 chroma U pixels that are at X2 and Y1
	24:20	**DnU_wd11[4:0]** Weight to be applied to the 4 chroma U pixels that are at X1 and Y1
	19:15	**DnU_wd10[4:0]** Weight to be applied to the 2 chroma U pixels that are at X and Y1
	14:10	**DnU_wd02[4:0]** Weight to be applied to the 2 chroma U pixels that are at X2 and Y
	9:5	**DnU_wd01[4:0]** Weight to be applied to the 2 chroma U pixels that are at X1 and Y
	4:0	**DnU_wd00[4:0]** Weight to be applied to the 1 chroma U pixels that are at X and Y
23	31:30	**Reserved** Access: RO Format: MBZ
	29:25	**DnV_Wr5[4:0]** Weight to be applied when: th4 <= (difference in chroma V value)
	24:20	**DnV_Wr4[4:0]** Weight to be applied when: th3 <= (difference in chroma V value) < th4
	19:15	**DnV_Wr3[4:0]** Weight to be applied when: th2 <= (difference in chroma V value) < th3
	14:10	**DnV_Wr2[4:0]** Weight to be applied when: th1 <= (difference in chroma V value) < th2

VEBOX_DNDI_STATE

	9:5	**DnV_Wr51[4:0]** Weight to be applied when: th0 <= (difference in chroma V value) < th1
	4:0	**DnV_Wr0[4:0]** Weight to be applied when: (difference in chroma V value) < th0
24	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**DnV_thmax[12:0]** Maximum threshold value for chroma V
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**DnV_thmin[12:0]** Minimum threshold value
25	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**DnV_prt5[12:0]**
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**DnV_dyn_thmin[12:0]** Minimum Dynamic threshold value.
26	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**DnV_prt4[12:0]** Multiplied by thrscale and then used as the threshold for comparing chroma V differences.
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**DnV_prt3[12:0]**
27	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**DnV_prt2[12:0]**
	15:13	**Reserved**

VEBOX_DNDI_STATE

		Access: RO Format: MBZ
	12:0	**DnV_prt1[12:0]**
28	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**DnV_prt0[12:0]**
	15	**Reserved** Access: RO Format: MBZ
	14:10	**DnV_wd22[4:0]** Weight to be applied to the 4 chroma V pixels that are at X2 and Y2
	9:5	**DnV_wd21[4:0]** Weight to be applied to the 4 chroma V pixels that are at X1 and Y2
	4:0	**DnV_wd20[4:0]** Weight to be applied to the 2 chroma V pixels that are at X and Y2
29	31:30	**Reserved** Access: RO Format: MBZ
	29:25	**DnV_wd12[4:0]** Weight to be applied to the 4 chroma V pixels that are at X2 and Y1
	24:20	**DnV_wd11[4:0]** Weight to be applied to the 4 chroma V pixels that are at X1 and Y1
	19:15	**DnV_wd10[4:0]** Weight to be applied to the 2 chroma V pixels that are at X and Y1
	14:10	**DnV_wd02[4:0]** Weight to be applied to the 2 chroma V pixels that are at X2 and Y
	9:5	**DnV_wd01[4:0]** Weight to be applied to the 2 chroma V pixels that are at X1 and Y
	4:0	**DnV_wd00[4:0]** Weight to be applied to the 1 chroma V pixels that are at X and Y
30	31:17	**Eight Direction Edge Threshold** Default Value: 1024 Format: U15 Threshold to determine an edge in eight directional edge detector
	16:7	**Valid Pixel Threshold** Default Value: 480 Format: U10

VEBOX_DNDI_STATE

	6:0	**Reserved** Access: RO Format: MBZ
31	31:19	**Small Sobel Threshold** Default Value: 480 Format: U13 Threshold for weak Sobel response
	18:6	**Large Sobel Threshold** Default Value: 2400 Format: U13 Threshold for strong Sobel response **Programming Notes** Large Sobel Threshold> Small Sobel Threshold
	5:0	**Small Sobel Count Threshold** Format: U6 Threshold for number of pixels in a block that have weak Sobel response (Default: 6)
32	31:26	**Median Sobel Count Threshold** Format: U6 Threshold for number of pixels in a block that have regular Sobel response (Default: 40)
	25:20	**Large Sobel Count Threshold** Format: U6 Threshold for number of pixels in a block that have strong Sobel response (Default: 6)
	19:6	**Block Sigma Diff Threshold** Default Value: 480 Format: U14 Threshold for the difference between maximum and minimum sigma within a block
	5:0	**Reserved** Access: RO Format: MBZ
33	31:19	**Max Sobel Threshold** Default Value: 1440 Format: U13
	18:0	**Reserved**

		Access: RO Format: MBZ
34	31:16	**Reserved** Access: RO Format: MBZ
	15:13	**Reserved** Access: RO Format: MBZ
	12:10	**STMM C2** Format: U3 Bias for divisor in STMM equation. Value / Name / Description: [0,7] / / Representing values [1,8]
	9:6	**Content Adaptive Threshold Slope** Format: U4 Determines the slope of the Content Adaptive Threshold. Value / Name / Description: 9 / **[Default]** / CAT_slope value = 10 Programming Notes: +1 added internally to get CAT_slope.
	5:2	**SAD Tight Threshold** Default Value: 5 Format: U4
	1:0	**Smooth MV Threshold** Format: U2
35	31	**STMM Blending Constant Select** Format: U1 Value / Name / Description: 0 / / Use the blending constant for small values of STMM for stmm_md_th 1 / / Use the blending constant for large values of STMM for stmm_md_th
	30:24	**Blending constant across time for large values of STMM** Format: U7
	23:16	**Blending constant across time for small values of STMM** Format: U8
	15:14	**Reserved**

		Access: RO Format: MBZ
	13:8	**Multiplier for VECM** Format: U6 Determines the strength of the vertical edge complexity measure.
	7:0	**Maximum STMM** Format: U8 Largest allowed STMM in blending equations.
36	31:24	**Minimum STMM** Format: U8 Smallest allowed STMM in blending equations
	23:22	**STMM Shift Down** Format: U2 Amount to shift STMM down (quantize to fewer bits) Value / Name: 0 – Shift by 4 1 – Shift by 5 2 – Shift by 6 3 – Reserved
	21:20	**STMM Shift Up** Format: U2 Amount to shift STMM up (set range). Value / Name: 0 – Shift by 6 1 – Shift by 7 2 – Shift by 8 3 – Reserved
	19:16	**STMM Output Shift** Format: U4 Value / Name: [0,15] **Programming Notes** The value of this field must satisfy the following equation: stmm_max - stmm_min = 2 ^

VEBOX_DNDI_STATE

		stmm_output_shift
	15:12	**ChromaTDM_WT** Default Value: 0 Format: U2.2
	11:8	**LumaTDM_WT** Default Value: 4 Format: U2.2
	7:0	**FMD Temporal Difference Threshold** Format: U8
37	31:28	**Reserved** Access: RO Format: MBZ
	27:24	**Deltabit value for SHCM** Format: U4 Value: 5 — **[Default]** Value: [0,8] — Range
	23:16	**Coring Threshold for SHCM** Default Value: 255 Format: U8
	15:12	**Reserved** Access: RO Format: MBZ
	11:8	**Deltabit value for SVCM** Format: U4 Value: 5 — **[Default]** Value: [0,8] — Range
	7:0	**Coring Threshold for SVCM** Default Value: 255 Format: U8
38	31:24	**FMD #1 Vertical Difference Threshold** Format: U8
	23:16	**FMD #2 Vertical Difference Threshold**

		Format: U8
	15:14	**CAT Threshold** Default Value: 0 Format: U2
	13:8	**FMD Tear Threshold** Format: U6
	7	**MCDI Enable** Use Motion Compensated Deinterlace algorithm. **Programming Notes** This bit is Ignored if DI Enable is off.
	6:4	**Reserved** Access: RO Format: MBZ
	3	**DN/DI Top First** Format: Enable Indicates the top field is first in sequence, otherwise bottom is first. Value 0: Bottom field occurs first in sequence Value 1: Top field occurs first in sequence
	2:0	**Reserved** Access: RO Format: MBZ
39	31:26	**Reserved** Access: RO Format: MBZ
	25	**FasterCovergence** Default Value: 0 Format: U1 For synthetic content faster convergence to current STMM value is preferred.
	24	**Luma Smaller Window for TDM** Format: U1
	23	**Chroma Smaller Window for TDM** Format: U1
	22:19	**Neighbor Pixel Threshold** Default Value: 10 Format: U4

VEBOX_DNDI_STATE

	18	**Reserved** Access: RO Format: MBZ
	17:16	**Progressive Cadence Reconstruction For 2nd Field Of Previous Frame** Format: U2 Value / Name / Description: 0 – Deinterlace 1 – Put together with previous field in sequence – 1st field of previous frame 2 – Put together with next field in sequence – 1st field of current frame **Programming Notes** Deflicker can be enabled only in De-interlace mode and not in Cadence construction mode.
	15:10	**MC Pixel Consistency Threshold** Default Value: 25 Format: U6
	9:8	**Progressive Cadence Reconstruction for 1st Field of Current Frame** Format: U2 Value / Name / Description: 0 – Deinterlace 1 – Put together with previous field in sequence – 2nd field of previous frame 2 – Put together with next field in sequence – 2nd field of current frame **Programming Notes** Deflicker can be enabled only in De-interlace mode and not if either fields are in Cadence construction mode.
	7:4	**SAD THB** Default Value: 10 Format: U4
	3:0	**SAD THA** Default Value: 5 Format: U4
40	31:24	**SAD_WT[3]** Format: U8 Value / Name: 192 – Default for Natural

		38 — Default for Synthetic
	23:16	**SAD_WT[2]** Format: U8 179 — Default for Natural 25 — Default for Synthetic
	15:8	**SAD_WT[1]** Format: U8 166 — Default for Natural 12 — Default for Synthetic
	7:0	**SAD_WT[0]** Format: U8 0 — Default for Natural and Synthetic
41	31:24	**Coring Threshold for Chroma SAD calculation** Default Value: 0 Format: U8
	23:16	**Coring Threshold for Luma SAD calculation** Default Value: 0 Format: U8
	15:8	**SAD_WT[6]** Format: U8 217 — Default for Natural 64 — Default for Synthetic
	7:0	**SAD_WT[4]** Format: U8 218 — Default for Natural 90 — Default for Synthetic
42	31	**Reserved**

VEBOX_DNDI_STATE

		Access: RO Format: MBZ
	30	**Bypass Deflicker** Format: U1
	29	**PAR_UseSyntheticContentMedian** Default Value: 0 Format: U1
	28	**PAR_LocalCheck** Default Value: 1 Format: U1
	27	**PAR_SyntheticContentCheck** Default Value: 0 Format: U1
	26:24	**PAR_DirectionCheckTh** Default Value: 3 Format: U3
	23:16	**PAR_TearingLowThreshold** Default Value: 20 Format: U8
	15:8	**PAR_TearingHighThreshold** Default Value: 100 Format: U8
	7:0	**PAR_DiffCheckSlackThreshold** Default Value: 15 Format: U8
43	31:24	**LPFWtLUT[3]** Default Value: 0 Format: U8
	23:16	**LPFWtLUT[2]** Default Value: 0 Format: U8
	15:8	**LPFWtLUT[1]** Default Value: 0 Format: U8
	7:0	**LPFWtLUT[0]** Default Value: 0

VEBOX_DNDI_STATE

		Format: U8
44	31:24	**LPFWtLUT[7]** Default Value: 255 Format: U8
	23:16	**LPFWtLUT[6]** Default Value: 128 Format: U8
	15:8	**LPFWtLUT[5]** Default Value: 64 Format: U8
	7:0	**LPFWtLUT[4]** Default Value: 32 Format: U8
45	31:20	**Synthetic Content Threshold** Default Value: 1500 Format: U12
	19	**Synthtic frame** Default Value: 0 Format: enable
	18:16	**TDM Harmonic Factor Natural** Default Value: 4 Format: U3
	15:13	**TDM Harmonic Factor Synthetic** Default Value: 4 Format: U3
	12:7	**HV UV threshold** Default Value: 30 Format: U6
	6:0	**TDM UV Threshold** Default Value: 100 Format: U7
46	31:29	**Reserved** Access: RO Format: MBZ
	28:21	**Natural Content Threshold** Default Value: 200

VEBOX_DNDI_STATE

		Format: U8
	20:16	**Max Harmonic Counter Threshold** Default Value: 15 Format: U5 **Restriction** The maximum value that can be programmed is15.
	15:12	**Harmonic Counter Threshold** Default Value: 8 Format: U4
	11:9	**SHCM Harmonic Factor Natural** Default Value: 6 Format: U3
	8:6	**SVCM Harmonic Factor Natural** Default Value: 6 Format: U3
	5:3	**SHCM Harmonic Factor Synthetic** Default Value: 4 Format: U3
	2:0	**SVCM Harmonic Factor Synthetic** Default Value: 4 Format: U3
47	31:0	**Maximum Value** Default Value: 0x3FFFFFF Format: U32
48	31:20	**Reserved** Access: RO Format: MBZ
	19:16	**Num Inliner Denominator Threshold** Default Value: 4 Format: U4
	15:12	**Num Inliner Numerator Threshold** Default Value: 3 Format: U4
	11:6	**HV_Y Threshold** Default Value: 50

VEBOX_DNDI_STATE			
		Format:	U6
	5:0	**Shifting value**	
		Default Value:	32
		Format:	U6

VEBOX_Filter_Coefficient

VEBOX_Filter_Coefficient		
Size (in bits):	8	
Default Value:	0x00000000	
DWord	**Bit**	**Description**
0	7:0	**2's Complement Filter Coefficient** Format: S1.6 **Range:** [-2, +2)

VEBOX_FORWARD_GAMMA_CORRECTION_STATE

VEBOX_FORWARD_GAMMA_CORRECTION_STATE	
Source:	VideoEnhancementCS
Size (in bits):	64
Default Value:	0x00000000, 0x00000000,

VEBOX_FORWARD_GAMMA_CORRECTION_STATE
0x00000000, 0x00000000,

VEBOX_FORWARD_GAMMA_CORRECTION_STATE
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,

VEBOX_FORWARD_GAMMA_CORRECTION_STATE
0x00000000, 0x00000000,

VEBOX_FORWARD_GAMMA_CORRECTION_STATE
0x00000000, 0x00000000,

intel

VEBOX_FORWARD_GAMMA_CORRECTION_STATE
0x00000000, 0x00000000,

VEBOX_FORWARD_GAMMA_CORRECTION_STATE
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,

VEBOX_FORWARD_GAMMA_CORRECTION_STATE

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000

DWord	Bit	Description
0..2047	63:0	**PRGB Corrected Value**

Format:	VEBOX_RGB_TO_GAMMA_CORRECTION

Programming Notes

Order in which the values are stored:

Interleaves									
0	256	1	257	2	258	...	...	255	511
512	768	513	769	514	770	...	...	767	1023

Point 0-255, 256-511 are interleaved first followed by interleaving the next set of 512points,interleaving between points 512-767, 768-1023.

VEBOX_FRONT_END_CSC_STATE

VEBOX_FRONT_END_CSC_STATE

Source: VideoEnhancementCS

Size (in bits): 384

Default Value: 0x00010000, 0x00000000, 0x00000000, 0x00000000, 0x00010000, 0x00000000, 0x00000000, 0x00000000, 0x00010000, 0x00000000, 0x00000000, 0x00000000

This state structure contains the IECP State Table Contents for Front-end CSC state.

DWord	Bit	Description
0	31	**Front End CSC Transform Enable** Format: Enable **Programming Notes** **Single Pipe IECP Enable** must also be set if this is enabled.
	30:19	**Reserved** Access: RO Format: MBZ
	18:0	**FECSC C0: Transform coefficient** Default Value: 10000h or 1.0 Format: S2.16
1	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**FECSC C1: Transform coefficient** Default Value: 0 or 0.0 Format: S2.16
2	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**FECSC C2: Transform coefficient** Default Value: 0 or 0.0 Format: S2.16
3	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**FECSC C3: Transform coefficient** Default Value: 0 or 0.0

VEBOX_FRONT_END_CSC_STATE

		Format: S2.16
4	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**FECSC C4: Transform coefficient** Default Value: 10000h or 1.0 Format: S2.16
5	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**FECSC C5: Transform coefficient** Default Value: 0 or 0.0 Format: S2.16
6	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**FECSC C6: Transform coefficient** Default Value: 0 or 0.0 Format: S2.16
7	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**FECSC C7: Transform coefficient** Default Value: 0 or 0.0 Format: S2.16
8	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**FECSC C8: Transform coefficient** Default Value: 10000h or 1.0 Format: S2.16
9	31:16	**FEC SC Offset out 1: Offset out for Y/R** Default Value: 0 Format: S15 The offset value is multiplied by 2 before being added to the output.

<table>
<tr><th colspan="3">VEBOX_FRONT_END_CSC_STATE</th></tr>
<tr><td></td><td>15:0</td><td>FEC SC Offset in 1: Offset in for Y/R<table><tr><td>Default Value:</td><td>0</td></tr><tr><td>Format:</td><td>S15</td></tr></table>The offset value is multiplied by 2 before being added to the output.</td></tr>
<tr><td rowspan="2">10</td><td>31:16</td><td>FEC SC Offset out 2: Offset out for U/G<table><tr><td>Default Value:</td><td>0</td></tr><tr><td>Format:</td><td>S15</td></tr></table>The offset value is multiplied by 2 before being added to the output.</td></tr>
<tr><td>15:0</td><td>FEC SC Offset in 2: Offset out for U/G<table><tr><td>Default Value:</td><td>0</td></tr><tr><td>Format:</td><td>S15</td></tr></table>The offset value is multiplied by 2 before being added to the output.</td></tr>
<tr><td rowspan="2">11</td><td>31:16</td><td>FEC SC Offset out 3: Offset out for V/B<table><tr><td>Default Value:</td><td>0</td></tr><tr><td>Format:</td><td>S15</td></tr></table>The offset value is multiplied by 2 before being added to the output.</td></tr>
<tr><td>15:0</td><td>FEC SC Offset in 3: Offset out for V/B<table><tr><td>Default Value:</td><td>0</td></tr><tr><td>Format:</td><td>S15</td></tr></table>The offset value is multiplied by 2 before being added to the output.</td></tr>
</table>

VEBOX_GAMUT_CONTROL_STATE

VEBOX_GAMUT_CONTROL_STATE

Source: VideoEnhancementCS

Size (in bits): 576

Default Value: 0xDA004750, 0x0000AE80, 0x00000470, 0x00000220, 0x001FFCC0, 0x0000D230, 0x00000A80, 0x001FFF40, 0x0000D6A0, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x0CD2911F, 0xB0000334, 0x00000000

DWord	Bit	Description
0	31:23	**A(r)** Default Value: 436 Format: U9 Gain_factor_R (default: 436, preferred range: 256-511).
	22	**Global Mode Enable** Format: U1 The gain factor derived from state CM(w). Value: 0 — Advance Mode Value: 1 — Basic Mode
	21	**Reserved** Access: RO Format: MBZ
	20:0	**C1** Default Value: 0004750h = 18256/65536 Format: S4.16 Coefficient of 3x3 Transform matrix.
1	31:22	**CM(w)** Format: U10 WeightingFactorForGain_factor (only enabled when the GlobalModeEnable is on).
	21	**Reserved** Access: RO Format: MBZ
	20:0	**C0** Default Value: 000AE80h = 44672/65536 Format: S4.16 Coefficient of 3x3 Transform matrix.

VEBOX_GAMUT_CONTROL_STATE

2	31:22	**CM(s)** Format: U2.8 AccurateColorComponentScaling (default: 640/256, preferred range: [512-1023]/256).
	21	**Reserved** Access: RO Format: MBZ
	20:0	**C3** Default Value: 0000470h = 1136/65536 Format: S4.16 Coefficient of 3x3 Transform matrix.
3	31:25	**A(g)** Format: U7 Gain_factor_G (default: 26/256, preferred range: [26-127]/256).
	24:21	**Reserved** Access: RO Format: MBZ
	20:0	**C2** Default Value: 0000220h = 544/65536 Format: S4.16 Coefficient of 3x3 Transform matrix.
4	31:25	**A(b)** Format: U7 Gain_factor_B (default: 26/256, preferred range: [26-127]/256).
	24:21	**Reserved** Access: RO Format: MBZ
	20:0	**C5** Default Value: 1FFCC0h = -832/65536 Format: S4.16 Coefficient of 3x3 Transform matrix.
5	31:22	**R(s)** Format: U2.8

VEBOX_GAMUT_CONTROL_STATE

		RedScaling (default: 768/256, preferred range: [512-1023]/256).
	21	**Reserved** Access: RO Format: MBZ
	20:0	**C4** Default Value: 000D230h = 53808/65536 Format: S4.16 Coefficient of 3x3 Transform matrix.
6	31:24	**CM(i)** Format: U0.8 AccurateColorComponentOffset (default: 192/256, preferred range: [0-192]/256).
	23:21	**Reserved** Access: RO Format: MBZ
	20:0	**C7** Default Value: 0000A80h = 2688/65536 Format: S4.16 Coefficient of 3x3 Transform matrix.
7	31:24	**R(i)** Format: U0.8 RedOffset (default: 128/256, preferred range: [0-128]/256).
	23:21	**Reserved** Access: RO Format: MBZ
	20:0	**C6** Default Value: 1FFF40h = -192/65536 Format: S4.16 Coefficient of 3x3 Transform matrix.
8	31:21	**Reserved** Access: RO Format: MBZ
	20:0	**C8**

VEBOX_GAMUT_CONTROL_STATE

		Default Value: 00D6A0h = 54944/65536 Format: S4.16 Coefficient of 3x3 Transform matrix.
9	31:17	**Reserved** Access: RO Format: MBZ
	16:0	**Offset_in_R** Default Value: 0 Format: S16 The input offset for red component.
10	31:17	**Reserved** Access: RO Format: MBZ
	16:0	**Offset_in_G** Default Value: 0 Format: S16 The input offset for green component.
11	31:17	**Reserved** Access: RO Format: MBZ
	16:0	**Offset_in_B** Default Value: 0 Format: S16 The input offset for blue component.
12	31:17	**Reserved** Access: RO Format: MBZ
	16:0	**Offset_out_R** Default Value: 0 Format: S16 The output offset for red component.
13	31:17	**Reserved** Access: RO

		Format: MBZ
	16:0	**Offset_out_G** Default Value: 0 Format: S16 The output offset for green component.
14	31:17	**Reserved** Access: RO Format: MBZ
	16:0	**Offset_out_B** Default Value: 0 Format: S16 The output offset for blue component.
15	31	**Reserved** Access: RO Format: MBZ
	30	**FullRangeMappingEnable** Format: U1 **Value** / **Name** 0 – Basic Mode **[Default]** 1 – Advance Mode
	29:20	**d(in,default)** Default Value: 205 Format: U10 InnerTriangleMappingLength.
	19:10	**d(out, default)** Default Value: 164 Format: U10 OuterTriangleMappingLength.
	9:0	**d1(out)** Default Value: 287 Format: U10 OuterTriangleMappingLengthBelow.

VEBOX_GAMUT_CONTROL_STATE

16 / 31 — xvYccDecEncEnable

Format:	U1

This bit is valid only when ColorGamutCompressionnEnable is on.

Value	Name
1	Both xvYcc decode and xvYcc encode are enabled **[Default]**
0	To disable both xvYcc decode and xvYcc encode

16 / 30:28 — CompressionLineShift

Format:	U3

.

Value	Name
3	**[Default]**
0,4	

16 / 27:10 — Reserved

Access:	RO
Format:	MBZ

16 / 9:0 — d1(in)

Default Value:	820
Format:	U10

InnerTriangleMappingLengthBelow.

17 / 31:30 — GCC BasicModeSelection

Format:	U2

Value	Name	Description
00b	Default **[Default]**	
01b	Scaling Factor	Used along with Dword66 Bits 28:11
10b	Single Axis Gamma Correction	Used along with Dword67 Bit 29
11b	Scaling factor with fixed luma	Used along with Dword37 Bits 28:11

17 / 29 — LumaChormaOnlyCorrection

Format:	U1

Value	Name
0	Luma Only Correction **[Default]**
1	Chorma Only Correction

17 / 28:25 — Reserved

Access:	RO
Format:	MBZ

VEBOX_GAMUT_CONTROL_STATE

	24:11	**BasicModeScalingFactor** Format: U2.12 Used when FullRangeMappingEnable is in basic mode and base mode selection bit is set to scaling factor.
	10:1	**Reserved** Access: RO Format: MBZ
	0	**Cpi Override** Format: U1 **Value** / **Name** 0 **[Default]** 1 Override Cpi calculation

VEBOX_PROCAMP_STATE

VEBOX_PROCAMP_STATE

Source: VideoEnhancementCS
Size (in bits): 64
Default Value: 0x01000001, 0x01000000

This state structure contains the IECP State Table Contents for ProcAmp state.

DWord	Bit	Description
0	31:28	**Reserved** Access: RO Format: MBZ
	27:17	**Contrast** Default Value: 80h = 1.0 in fixed point U4.7 Format: U4.7 Contrast magnitude.
	16:13	**Reserved** Access: RO Format: MBZ
	12:1	**Brightness** Default Value: 0 or 0.0 Format: S7.4 Brightness magnitude.
	0	**PROCAMP Enable** Default Value: 1 Format: Enable
1	31:16	**Cos_c_s** Default Value: 256 Format: S7.8 UV multiplication cosine factor.
	15:0	**Sin_c_s** Default Value: 0 Format: S7.8 UV multiplication sine factor.

VEBOX_RGB_TO_GAMMA_CORRECTION

VEBOX_RGB_TO_GAMMA_CORRECTION		
Source:	VideoEnhancementCS	
Size (in bits):	64	
Default Value:	0x00000000, 0x00000000	
Color depth is 16 bits.		
DWord	**Bit**	**Description**
0..1	63:48	**B-ch Corrected Value** Default Value: 0h Format: U16
	47:32	**G-ch Corrected Value** Default Value: 0h Format: U16
	31:16	**R-ch Corrected Value** Default Value: 0h Format: U16
	15:0	**Pixel Value** Default Value: 0h Format: U16 **Programming Notes** N indicates the index into the table. Pixel value 0 and Pixel Value 1023 should be always programmed to 0 and 0xFFFF respectively.

VEBOX_STD_STE_STATE

VEBOX_STD_STE_STATE

Source:	VideoEnhancementCS
Size (in bits):	928
Default Value:	0x9A6E39F0, 0x400D3C65, 0x000C9180, 0xFE2F2E00, 0x0003FFFF, 0x00140000, 0xD82E0640, 0x8285ECEC, 0x07FB8282, 0x00000000, 0x02117000, 0xA38FEC96, 0x0100C8C8, 0x003A6871, 0x01478000, 0x0007C300, 0x1291F008, 0x00094855, 0x1C1BD100, 0x03802008, 0x0002A980, 0x00080180, 0x0007CFFB, 0x18D1F07C, 0x000800BD, 0x1C080100, 0x03800000, 0x0008012B, 0x0008012B

This state structure contains the state used by the STD/STE function.

DWord	Bit	Description
0	31:24	**V_Mid** Default Value: 154 Format: U8 Rectangle middle-point V coordinate.
	23:16	**U_Mid** Default Value: 110 Format: U8 Rectangle middle-point U coordinate.
	15:10	**Hue_Max** Default Value: 14 Format: U6 Rectangle half width.
	9:4	**Sat_Max** Default Value: 31 Format: U6 Rectangle half length.
	3	**Reserved** Access: RO Format: MBZ
	2	**Output Control** Value 0: Output Pixels Value 1: Output STD Decisions

VEBOX_STD_STE_STATE

	1	**STE Enable** Format: Enable
	0	**STD Enable** Format: Enable **Programming Notes** This needs to be enabled if 'STD Score Output' is enabled.
1	31	**STD Score Output** Format: Enable
	30:28	**Diamond Margin** Default Value: 4 Format: U3
	27:21	**Diamond_du** Default Value: 0 Format: S6 Rhombus center shift in the sat-direction, relative to the rectangle center.
	20:18	**HS_margin** Default Value: 3 Format: U3 Defines rectangle margin.
	17:10	**Cos(Î±)** Default Value: 79 Format: S0.7 The default is 79/128
	9:8	**Reserved** Access: RO Format: MBZ
	7:0	**Sin(Î±)** Default Value: 101 Format: S0.7 The default is 101/128
2	31:21	**Reserved** Access: RO Format: MBZ

VEBOX_STD_STE_STATE

	20:13	**Diamond_alpha** Default Value: 100 Format: U2.6 1/tan()The default is 100/64
	12:7	**Diamond_Th** Default Value: 35 Format: U6 Half length of the rhombus axis in the sat-direction.
	6:0	**Diamond_dv** Default Value: 0 Format: S6 Rhombus center shift in the hue-direction, relative to the rectangle center.
3	31:24	**Y_point_3** Default Value: 254 Format: U8 Third point of the Y piecewise linear membership function.
	23:16	**Y_point_2** Default Value: 47 Format: U8 Second point of the Y piecewise linear membership function.
	15:8	**Y_point_1** Default Value: 46 Format: U8 First point of the Y piecewise linear membership function.
	7	**VY_STD_Enable** Format: Enable Enables STD in the VY subspace.
	6:0	**Reserved** Access: RO Format: MBZ
4	31:18	**Reserved** Access: RO

VEBOX_STD_STE_STATE

		Format: MBZ
	17:13	**Y_Slope_2** Default Value: 31 Format: U2.3 Slope between points Y3 and Y4. The default is 31/8
	12:8	**Y_Slope_1** Default Value: 31 Format: U2.3 Slope between points Y1 and Y2. The default is 31/8
	7:0	**Y_point_4** Default Value: 255 Format: U8 Fourth point of the Y piecewise linear membership function.
5	31:16	**INV_Skin_types_margin** Format: U0.16 1/(2* Skin_types_margin) Value: 20 — Name: Skin_Type_margin **[Default]** Value: 1638 — Name:
	15:0	**INV_Margin_VYL** Format: U0.16 1 / Margin_VYL1/ Margin_VYL = 3300/65536
6	31:24	**P1L** Default Value: 216 Format: U8 Y Point 1 of the lower part of the detection PWLF.
	23:16	**P0L** Default Value: 46 Format: U8 Y Point 0 of the lower part of the detection PWLF.

VEBOX_STD_STE_STATE

	15:0	**INV_Margin_VYU** Default Value: 1600 Format: U0.16 1 / Margin_VYU = 1600/65536
7	31:24	**B1L** Default Value: 130 Format: U8 V Bias 1 of the lower part of the detection PWLF.
	23:16	**B0L** Default Value: 133 Format: U8 V Bias 0 of the lower part of the detection PWLF.
	15:8	**P3L** Default Value: 236 Format: U8 Y Point 3 of the lower part of the detection PWLF.
	7:0	**P2L** Default Value: 236 Format: U8 Y Point 2 of the lower part of the detection PWLF.
8	31:27	**Reserved** Access: RO Format: MBZ
	26:16	**S0L** Default Value: 7FBh Format: S2.8 Slope 0 of the lower part of the detection PWLF. The default is -5/256
	15:8	**B3L** Default Value: 130 Format: U8 V Bias 3 of the lower part of the detection PWLF.

VEBOX_STD_STE_STATE

	7:0	**B2L** Default Value: 130 Format: U8 V Bias 2 of the lower part of the detection PWLF.
9	31:22	**Reserved** Access: RO Format: MBZ
	21:11	**S2L** Default Value: 0 Format: S2.8 The default is 0/256
	10:0	**S1L** Default Value: 0 Format: S2.8 Slope 1 of the lower part of the detection PWLF. The default is 0/256
10	31:27	**Reserved** Access: RO Format: MBZ
	26:19	**P1U** Default Value: 66 Format: U8 Y Point 1 of the upper part of the detection PWLF.
	18:11	**P0U** Default Value: 46 Format: U8 Y Point 0 of the upper part of the detection PWLF.
	10:0	**S3L** Default Value: 0 Format: S2.8 Slope 3 of the lower part of the detection PWLF. The default is 0/256

VEBOX_STD_STE_STATE

11	31:24	**B1U** Default Value: 163 Format: U8 V Bias 1 of the upper part of the detection PWLF.
	23:16	**B0U** Default Value: 143 Format: U8 V Bias 0 of the upper part of the detection PWLF.
	15:8	**P3U** Default Value: 236 Format: U8 Y Point 3 of the upper part of the detection PWLF.
	7:0	**P2U** Default Value: 150 Format: U8 Y Point 2 of the upper part of the detection PWLF.
12	31:27	**Reserved** Access: RO Format: MBZ
	26:16	**S0U** Default Value: 256 Format: S2.8 Slope 0 of the upper part of the detection PWLF. The default is 256/256
	15:8	**B3U** Format: U8 V Bias 3 of the upper part of the detection PWLF. Value: 200 **[Default]**; 140
	7:0	**B2U** Default Value: 200 Format: U8 V Bias 2 of the upper part of the detection PWLF.

VEBOX_STD_STE_STATE

13	31:22	**Reserved** Access: RO Format: MBZ
	21:11	**S2U** Default Value: 74Dh Format: S2.8 Slope 2 of the upper part of the detection PWLF. The default is -179/256
	10:0	**S1U** Default Value: 113 Format: S2.8 Slope 1 of the upper part of the detection PWLF. The default is 113/256
14	31:28	**Reserved** Access: RO Format: MBZ
	27:20	**Skin_types_margin** Default Value: 20 Format: U8 Skin types Y marginRestrict Skin_types_thresh >= Skin_types_margin > 0Restrict (Skin_types_thresh + Skin_types_margin) <= 255
	19:12	**Skin_types_thresh** Default Value: 120 Format: U8 Skin types Y marginRestrict Skin_types_thresh >= Skin_types_margin > 0Restrict (Skin_types_thresh + Skin_types_margin) <= 255
	11	**Skin_Types_Enable** Default Value: 0 Disable Format: Enable Treat differently bright and dark skin types
	10:0	**S3U** Default Value: 0

VEBOX_STD_STE_STATE

		Format: S2.8 Slope 3 of the upper part of the detection PWLF. The default is 0/256
15	31	**Reserved** Access: RO Format: MBZ
	30:21	**SATB1** Format: S7.2 First bias for the saturation PWLF (bright skin). The default numerical value is -8/4 Value: 3F8h; Name:
	20:14	**SATP3** Default Value: 31 Format: S6 Third point for the saturation PWLF (bright skin).
	13:7	**SATP2** Default Value: 6 Format: S6 Second point for the saturation PWLF (bright skin).
	6:0	**SATP1** Format: S6 First point for the saturation PWLF (bright skin). The default numerical valueis -6/64. Value: 7Ah; Name:
16	31	**Reserved** Access: RO Format: MBZ
	30:20	**SATS0** Default Value: 297 Format: U3.8

		Zeroth slope for the saturation PWLF (bright skin) The default is 297/256
	19:10	**SATB3** Default Value: 124 Format: S7.2 Third bias for the saturation PWLF (bright skin) The default is 124/4
	9:0	**SATB2** Default Value: 8 Format: S7.2 Second bias for the saturation PWLF (bright skin) The default is 8/4
17	31:22	**Reserved** Access: RO Format: MBZ
	21:11	**SATS2** Default Value: 297 Format: U3.8 Second slope for the saturation PWLF (bright skin) The default is 297/256
	10:0	**SATS1** Default Value: 85 Format: U3.8 First slope for the saturation PWLF (bright skin) The default is 85/256
18	31:25	**HUEP3** Default Value: 14 Format: S6 Third point for the hue PWLF (bright skin)
	24:18	**HUEP2** Default Value: 6 Format: S6 Second point for the hue PWLF (bright skin)

VEBOX_STD_STE_STATE

	17:11	**HUEP1** Default Value: 7Ah -6 Format: S6 First point for the hue PWLF (bright skin)
	10:0	**SATS3** Default Value: 256 Format: U3.8 Third slope for the saturation PWLF (bright skin) The default is 256/256
19	31:30	**Reserved** Access: RO Format: MBZ
	29:20	**HUEB3** Default Value: 56 Format: S7.2 Third bias for the hue PWLF (bright skin) The default is 56/4
	19:10	**HUEB2** Default Value: 8 Format: S7.2 Second bias for the hue PWLF (bright skin) The default is 8/4
	9:0	**HUEB1** Format: S7.2 First bias for the hue PWLF (bright skin) The default is 8/4 **Value** / **Name** 8 / **[Default]** 0xf8 /
20	31:22	**Reserved** Access: RO

VEBOX_STD_STE_STATE

		Format: MBZ
	21:11	**HUES1** Default Value: 85 Format: U3.8 First slope for the hue PWLF (bright skin) The default is 85/256
	10:0	**HUES0** Default Value: 384 Format: U3.8 Zeroth slope for the hue PWLF (bright skin) The default is 384/256
21	31:22	**Reserved** Access: RO Format: MBZ
	21:11	**HUES3** Default Value: 256 Format: U3.8 Third slope for the hue PWLF (bright skin) The default is 256/256
	10:0	**HUES2** Default Value: 384 Format: U3.8 Second slope for the hue PWLF (bright skin) The default is 384/256
22	31	**Reserved** Access: RO Format: MBZ
	30:21	**SATB1_DARK** Default Value: 0 Format: S7.2 First bias for the saturation PWLF (dark skin) The default is 0/4

VEBOX_STD_STE_STATE

	20:14	**SATP3_DARK** Default Value: 31 Format: S6 Third point for the saturation PWLF (dark skin)
	13:7	**SATP2_DARK** Default Value: 31 Format: S6 Second point for the saturation PWLF (dark skin)
	6:0	**SATP1_DARK** Default Value: 7Bh Format: S6 First point for the saturation PWLF (dark skin) Default Value: -5
23	31	**Reserved** Access: RO Format: MBZ
	30:20	**SATS0_DARK** Default Value: 397 Format: U3.8 Zeroth slope for the saturation PWLF (dark skin) The default is 397/256
	19:10	**SATB3_DARK** Default Value: 124 Format: S7.2 Third bias for the saturation PWLF (dark skin) The default is 124/4
	9:0	**SATB2_DARK** Default Value: 124 Format: S7.2 Second bias for the saturation PWLF (dark skin) The default is 124/4
24	31:22	**Reserved** Access: RO

VEBOX_STD_STE_STATE

		Format: MBZ
	21:11	**SATS2_DARK** Default Value: 256 Format: U3.8 Second slope for the saturation PWLF (dark skin) The default is 256/256
	10:0	**SATS1_DARK** Default Value: 189 Format: U3.8 First slope for the saturation PWLF (dark skin) The default is 189/256
25	31:25	**HUEP3_DARK** Default Value: 14 Format: S6 Third point for the hue PWLF (dark skin).
	24:18	**HUEP2_DARK** Default Value: 2 Format: S6 Second point for the hue PWLF (dark skin).
	17:11	**HUEP1_DARK** Default Value: 0 Format: S6 First point for the hue PWLF (dark skin).
	10:0	**SATS3_DARK** Default Value: 256 Format: U3.8 Third slope for the saturation PWLF (dark skin) The default is 256/256
26	31:30	**Reserved** Access: RO Format: MBZ
	29:20	**HUEB3_DARK**

VEBOX_STD_STE_STATE

		Default Value: 56 Format: S7.2 Third bias for the hue PWLF (dark skin). The default is 56/4
	19:10	**HUEB2_DARK** Default Value: 0 Format: S7.2 Second bias for the hue PWLF (dark skin). The default is 0/4
	9:0	**HUEB1_DARK** Default Value: 0 Format: S7.2 First bias for the hue PWLF (dark skin). The default is 0/4
27	31:22	**Reserved** Access: RO Format: MBZ
	21:11	**HUES1_DARK** Default Value: 256 Format: U3.8 First slope for the hue PWLF (dark skin). The default is 256/256
	10:0	**HUES0_DARK** Format: U3.8 Zeroth slope for the hue PWLF (dark skin). The default is 299/256 **Value** / **Name** 299 / **[Default]** 256 /
28	31:22	**Reserved** Access: RO Format: MBZ

VEBOX_STD_STE_STATE

	21:11	**HUES3_DARK** Default Value: 256 Format: U3.8 Third slope for the hue PWLF (dark skin). The default is 256/256
	10:0	**HUES2_DARK** Default Value: 299 Format: U3.8 Second slope for the hue PWLF (dark skin). The default is 299/256

VEBOX_TCC_STATE

VEBOX_TCC_STATE

Source:	VideoEnhancementCS
Size (in bits):	352
Default Value:	0xDCDCDC00, 0xDCDCDC00, 0x1E34CC91, 0x3E3CCE91, 0x02E80195, 0x0197046B, 0x01790174, 0x00096000, 0x00000000, 0x03030000, 0x009201C0

This state structure contains the IECP State Table Contents for TCC state.

DWord	Bit	Description
0	31:24	**SatFactor3** Format: U1.7 The saturation factor for yellow. The default is 220/128 220 **[Default]** 160
	23:16	**SatFactor2** Format: U1.7 The saturation factor for red. The default is 220/128 220 **[Default]** 160
	15:8	**SatFactor1** Format: U1.7 The saturation factor for magenta. The default is 220/128 220 **[Default]** 160
	7	**TCC Enable** Format: Enable
	6:0	**Reserved**

VEBOX_TCC_STATE

		Access: RO Format: MBZ
1	31:24	**SatFactor6** Format: U1.7 The saturation factor for blue. The default is 220/128 **Value** / **Name** 220 / **[Default]** 160 /
	23:16	**SatFactor5** Format: U1.7 The saturation factor for cyan. The default is 220/128 **Value** / **Name** 220 / **[Default]** 160 /
	15:8	**SatFactor4** Format: U1.7 The saturation factor for green. The default is 220/128 **Value** / **Name** 220 / **[Default]** 160 /
	7:0	**Reserved** Access: RO Format: MBZ
2	31:30	**Reserved** Access: RO Format: MBZ
	29:20	**BaseColor3** Default Value: 483 Format: U10

VEBOX_TCC_STATE

		Base Color 3 - this value must be greater than BaseColor2
	19:10	**BaseColor2** Default Value: 307 Format: U10 Base Color 2 - this value must be greater than BaseColor1
	9:0	**BaseColor1** Default Value: 145 Format: U10 Base Color 1
3	31:30	**Reserved** Access: RO Format: MBZ
	29:20	**BaseColor6** Default Value: 995 Format: U10 Base Color 6 - this value must be greater than BaseColor5
	19:10	**BaseColor5** Default Value: 819 Format: U10 Base Color 5 - this value must be greater than BaseColor4
	9:0	**BaseColo4** Default Value: 657 Format: U10 Base Color 4 - this value must be greater than BaseColor3
4	31:16	**ColorTransitSlope23** Default Value: 744 Format: U0.16 The calculation result of 1 / (BC3 - BC2) [1/62]
	15:0	**ColorTransitSlope2** Default Value: 405 Format: U0.16 The calculation result of 1 / (BC2 - BC1) [1/57]

VEBOX_TCC_STATE

5	31:16	**ColorTransitSlope45** Default Value: 407 Format: U0.16 The calculation result of 1 / (BC5 - BC4) [1/57]
	15:0	**ColorTransitSlope34** Default Value: 1131 Format: U0.16 The calculation result of 1 / (BC4 - BC3) [1/61]
6	31:16	**ColorTransitSlope61** Default Value: 377 Format: U0.16 The calculation result of 1 / (BC1 - BC6) [1/62]
	15:0	**ColorTransitSlope56** Default Value: 372 Format: U0.16 The calculation result of 1 / (BC6 - BC5) [1/62]
7	31:22	**ColorBias3** Default Value: 0 Format: U2.8 Color bias for BaseColor3.
	21:12	**ColorBias2** Default Value: 150 Format: U2.8 Color bias for BaseColor2. The default is 150/256
	11:2	**ColorBias1** Default Value: 0 Format: U2.8 Color bias for BaseColor1.
	1:0	**Reserved** Access: RO

		Format: MBZ
8	31:22	**ColorBias6** Default Value: 0 Format: U2.8 Color bias for BaseColor6.
	21:12	**ColorBias5** Default Value: 0 Format: U2.8 Color bias for BaseColor5.
	11:2	**ColorBias4** Default Value: 0 Format: U2.8 Color bias for BaseColor4.
	1:0	**Reserved** Access: RO Format: MBZ
9	31	**Reserved** Access: RO Format: MBZ
	30:24	**UV Threshold** Default Value: 3 Format: U7 Low UV threshold.
	23:19	**Reserved** Access: RO Format: MBZ
	18:16	**UV Threshold Bits** Default Value: 3 Format: U3 Low UV transition width bits.
	15:13	**Reserved** Access: RO Format: MBZ

<table>
<tr><th colspan="3">VEBOX_TCC_STATE</th></tr>
<tr><td></td><td>12:8</td><td>STE Threshold<table><tr><td>Default Value:</td><td>0</td></tr><tr><td>Format:</td><td>U5</td></tr></table>Skin tone pixels enhancement threshold.</td></tr>
<tr><td></td><td>7:3</td><td>Reserved<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>2:0</td><td>STE Slope Bits<table><tr><td>Default Value:</td><td>0</td></tr><tr><td>Format:</td><td>U3</td></tr></table>Skin tone pixels enhancement slope bits.</td></tr>
<tr><td>10</td><td>31:16</td><td>Inv_UVMaxColor<table><tr><td>Default Value:</td><td>146</td></tr><tr><td>Format:</td><td>U16</td></tr></table>1 / UVMaxColor. Used for the SFs2 calculation.</td></tr>
<tr><td></td><td>15:9</td><td>Reserved<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>8:0</td><td>UVMaxColor<table><tr><td>Default Value:</td><td>448</td></tr><tr><td>Format:</td><td>U9</td></tr></table>The maximum absolute value of the legal UV pixels. Used for the SFs2 calculation.</td></tr>
</table>

VEBOX_VERTEX_TABLE_ENTRY

VEBOX_VERTEX_TABLE_ENTRY

Source: VideoEnhancementCS

Size (in bits): 32

Default Value: 0x00000000

DWord	Bit	Description
0	31:28	**Reserved** Access: RO Format: MBZ
	27:16	**Vertex table entry 0 - Lv (12 bits)** Value: 100h-ED6h; Name: ; Description: Range for Vertices BT601 and BT709
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Vertex table entry 0 - Cv (12 bits)** Value: 400h-A00h; Name: ; Description: Range for Vertices BT601 and BT709

VECS Hardware-Detected Error Bit Definitions

VECS Hardware-Detected Error Bit Definitions

Source: VideoEnhancementCS
Size (in bits): 16
Default Value: 0x00000000

DWord	Bit	Description
0	15:12	**Reserved** Access: RO Format: MBZ
	11	**Reserved** Access: RO Format: MBZ
	10:3	**Reserved** Access: RO Format: MBZ
	2	**Command Privilege Violation Error** This bit is set if a command classified as privileged is parsed in a non-privileged batch buffer. The command will be converted to a NOOP and parsing will continue.
	1	**Reserved** Access: RO Format: MBZ
	0	**Instruction Error** This bit is set when the Renderer Instruction Parser detects an error while parsing an instruction.Instruction errors include: • Client ID value (Bits 31:29 of the Header) is not supported (only MI, 2D and 3D are supported). • Defeatured MI Instruction Opcodes: Value: 1 — Name: — Description: Instruction Error detected **Programming Notes** This error indications cannot be cleared except by reset (i.e., it is a fatal error).

VERTEX_BUFFER_STATE

VERTEX_BUFFER_STATE

Source: RenderCS

Size (in bits): 128

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000

This structure is used in 3DSTATE_VERTEX_BUFFERS to set the state associated with a VB. The VF function will use this state to determine how/where to extract vertex element data for all vertex elements associated with the VB.

DWord	Bit	Description
0	31:26	**Vertex Buffer Index** Format: U6 This field contains an index value which selects the VB state being defined. Value: [0,32] — Name:
	25	**L3 Bypass Disable** Format: Disable **Description** When set, vertex data fetches from this vertex buffer are routed through the L3 and therefore that vertex data may be coherent with the L3 cache, as controlled by the Memory Object Control State (MOCS) value. I.e., if portions of the vertex buffer already reside in the L3 (e.g., were written or read by another L3 agent), reads from VF may hit in the L3 with the cached data returned to VF. If reads from VF miss in the L3 cache, the reads are directed to the next higher in the memory hierarchy, but the data returned in not placed in the L3 cache. The MOCS value must not be set to cache the data in L3. When clear, vertex data reads from this vertex buffer bypass the L3 logic, therefore precluding the coherency of that data in the L3 cache. If the vertex buffer data can becached in L3, the L3 cache must first be flushed to maintain vertex buffer data coherency.
	24:23	**Reserved** Access: RO Format: MBZ
	22:16	**Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this vertex buffer.
	15	**Reserved** Access: RO Format: MBZ
	14	**Address Modify Enable**

VERTEX_BUFFER_STATE

If set, the Buffer Starting Address field is used to update the state of this buffer. If clear, that field is ignored and the previously-programmed value is maintained.

13 Null Vertex Buffer

Format:	Enable

This field enabled causes any fetch for vertex data to return 0.

Programming Notes

VERTEX_BUFFER_STATE.Null Vertex Buffer must be set when the VERTEX_BUFFER_STATE.Buffer Size is 0x0.

12 Reserved

Access:	RO
Format:	MBZ

11:0 Buffer Pitch

Format:	U12

This field specifies the pitch in bytes of the structures accessed within the VB. This information is required in order to access elements in the VB via a structure index.

Value	Name	Description
[0,4095]		Bytes

Programming Notes

- Different VERTEX_BUFFER_STATE structures can refer to the same memory region using different Buffer Pitch values.
- See note on 64-bit float alignment in Buffer Starting Address.

1..2 63:0 Buffer Starting Address

Format:	GraphicsAddress[63:0]

Description

This field contains the byte-aligned Graphics Address of the first element of interest within the VB. Software must program this value with the combination (sum) of the base address of the memory resource and the byte offset from the base address to the starting structure within the buffer. If the Address ModifyEnable bit is clear, this field is ignored and the previous value of Buffer Starting Address for this buffer is maintained.

GraphicsAddress [63:48] are ignored by the HW.

Programming Notes

- 64-bit floating point values must be 64-bit aligned in memory, or UNPREDICTABLE data will be fetched. When accessing an element containing 64-bit floating point values, the Buffer Starting Address and Source Element Offset values must add to a 64-bit aligned address, and BufferPitch must be a multiple of 64-bits.
- VBs can only be allocated in linear (not tiled) graphics memory.

VERTEX_BUFFER_STATE

		• As computed index values are, by definition, interpreted as unsigned values, there is no issue with accesses to locations before (lower address value) the start of the buffer. However, these wrapped indices are subject to Max Index checking (see below).
3	31:0	**Buffer Size** Format: U32 This field specifies the size of the buffer in bytes. Vertex element accesses which straddle or go past the end of the buffer will return 0's for all elements. Note that BufferSize=0 indicates that there is no valid data in the buffer. **Value** / **Name** [0, FFFFFFFFh]

VERTEX_ELEMENT_STATE

VERTEX_ELEMENT_STATE

Source:	RenderCS
Size (in bits):	64
Default Value:	0x00000000, 0x00000000

This structure is used in 3DSTATE_VERTEX_ELEMENTS to set the state associated with a vertex element. A vertex element is defined as an entity supplying from one tofour DWord vertex components, to be stored in the vertex URB entry.

The number of supported vertex elements is 34.

The VF function will use this state, and possibly the state of the associated vertex buffer, to fetch/generate the source vertex element data, perform any required format conversions, padding with zeros, and store the resulting destination vertex element data into the vertex URB entry.

Programming Notes

- The (new) 3DSTATE_VF_SGVS command is used to specify optional insertion of VertexID and/or InstanceID into the input vertex data, logically following the processing of the VERTEX_ELEMENT_STATE structures. The VFCOMP_STORE_VID/IID encodings are no longer available in VERTEX_ELEMENT_STATE.
- When SourceElementFormat is set to one of the *64*_PASSTHRU formats, 64-bit components are stored in the URB without any conversion. In this case, vertex elements must be written as 128 or 256 bits, with VFCOMP_STORE_0 being used to pad the output as required. E.g., if R64_PASSTHRU is used to copy a 64-bit Red component into the URB, Component 1 must be specified as VFCOMP_STORE_0 (with Components 2,3 set to VFCOMP_NOSTORE) in order to output a 128-bit vertex element, or Components 1-3 must be specified as VFCOMP_STORE_0 in order to output a 256-bit vertex element. Likewise, use of R64G64B64_PASSTHRU requires Component 3 to be specified as VFCOMP_STORE_0 in order to output a 256-bit vertex element.
- When SourceElementFormat is set to one of the *64*_PASSTHRU formats then VFCOMP_STORE_SRC must be used for every valid component.
- Any SourceElementFormat of *64*_PASSTHRU cannot be used with an element which has edge flag enabled.

The SourceElementFormat needs to be a single-component format with an element which has edge flag enabled.

Software shall not attempt to disable any components (via 3DSTATE_VF_COMPONENT_PACKING) for elements associated with 256-bit SURFACE_FORMATs.

DWord	Bit	Description
0	31:26	**Vertex Buffer Index** Format: U6 This field specifies which vertex buffer the element is sourced from. [0,32] Up to 33 VBs are supported

VERTEX_ELEMENT_STATE

Programming Notes

It is possible for a vertex element to include only internally-generated data (VertexID, etc.), in which case the associated vertex buffer state is ignored.

25 **Valid**

Format:	Boolean

Value	Name	Description
1h	TRUE	this vertex element is used in vertex assembly
0h	FALSE	this vertex element is not used.

24:16 **Source Element Format**

Format:	SURFACE_FORMAT

Range: Valid formats are found in the 3D Primitive Processing FormatConversion portion of the vertex fetch chapter.

Format:

```
 The encoding of this field is identical the Surface Format field of the
SURFACE_STATE structure, as described in the Sampler chapter.
```

This field specifies the format in which the memory-resident source data for this particular vertex element is stored in the memory buffer. This only applies to elements stored with VFCOMP_STORE_SRC component control. (All other component types have an explicit format).

15 **Edge Flag Enable**

Format:	Enable

When ENABLED, the source element is interpreted as an EdgeFlag for the vertex. If the source element is zero, the EdgeFlag will be set to FALSE. If the source element is non-zero, the EdgeFlag will be set to TRUE. The EdgeFlag bit will travel down the fixed function pipeline along with the vertex handle, etc. and not be stored in the vertex data like the other vertex elements. Refer to the fixed function descriptions for how this EdgeFlag affects rendering.
Edge flags are supported for the following primitive topology types only, otherwise EdgeFlagEnable must not be ENABLED.

- 3DPRIM_TRILIST*
- 3DPRIM_POLYGON
- 3DPRIM_QUADLIST

If this bit is DISABLED for all valid VERTEX_ELEMENTs, the vertex will be assigned a default EdgeFlag of TRUE.

Programming Notes

- This bit must only be ENABLED on the last valid VERTEX_ELEMENT structure.
- When set, Component 0 Control must be set to VFCOMP_STORE_SRC, and Component 1-3 Control must be set to VFCOMP_NOSTORE.

14:12 **Reserved**

VERTEX_ELEMENT_STATE

		Access: RO Format: MBZ
	11:0	**Source Element Offset** Format: U12 Byte offset of the source vertex element data in the structures comprising the vertex buffer. Value: [0,2047] — Name: **Programming Notes** See note on 64-bit float alignment in Buffer Starting Address.
1	31	**Reserved** Access: RO Format: MBZ
	30:28	**Component 0 Control** Format: 3D_Vertex_Component_Control Refer to the 3D_Vertex_Component_Control table below
	27	**Reserved** Access: RO Format: MBZ
	26:24	**Component 1 Control** Format: 3D_Vertex_Component_Control Refer to the 3D_Vertex_Component_Control table below
	23	**Reserved** Access: RO Format: MBZ
	22:20	**Component 2 Control** Format: 3D_Vertex_Component_Control Refer to the 3D_Vertex_Component_Control table below
	19	**Reserved** Access: RO Format: MBZ
	18:16	**Component 3 Control** Format: 3D_Vertex_Component_Control Refer to the 3D_Vertex_Component_Control table below

VERTEX_ELEMENT_STATE		
	15:0	**Reserved** Access: RO Format: MBZ

Vertical Line Stride Override Message Descriptor Control Field

MDC_VLSO - Vertical Line Stride Override Message Descriptor Control Field

Size (in bits): 3

Default Value: 0x00000000

DWord	Bit	Description
0	2	**Vertical Line Stride Override** Format: Enable If set, override the Vertical Line Stride and Vertical Line Stride Offset fields in the surface state with the fields below.
	1	**Vertical Line Stride** Format: U1 Specifies number of lines (0 or 1) to skip between logically adjacent lines - provides support of interleaved (field) surfaces as textures.
	0	**Vertical Line Stride Offset** Format: U1 Specifies the offset of the initial line from the beginning of the buffer. Ignored when Override VerticalLine Stride is 0.

VideoDecoder Interrupt Vector

VIDEODECODER_INTR_VEC - VideoDecoder Interrupt Vector

Size (in bits): 16

Default Value: 0x00000000

DWord	Bit	Description
0	15	**Catastrophic Error** This interrupt signals that a unrecoverable error (for e.g encountered fault when accessing a page mapped in Global GTT) during the engine processing. When Memory interface signals this error, the Command Streamer will stop parsing any more instructions. Scheduler is expected to reset the engine to evict the context
	14:12	**Reserved** Access: RO Format: MBZ
	11	**VCS Wait On Semaphore** Exec-List Scheduling: Set when MI_SEMAPHORE_WAIT command is un-successful and when "Inhibit Synchronous Context Switch" is set. Scheduler can use this interrupt to preempt the context waiting on semaphore wait.Ring Buffer Scheduling: Set when MI_SEMAPHORE_WAIT command is un-successful.
	10	**Reserved** Access: RO Format: MBZ
	9	**CS TR Invalid Tile Detection**
	8	**VCS Context Switch Interrupt** Set when a context switch has just occurred. **Execlist Enable bit** needs to be set for this interrupt to occur.
	7	**Legacy Context Per Process Page Fault Interrupt** Fault interrupt is generated by GA fabric, not by the CS This interrupt is for handling Legacy context PP GTT Page Fault.
	6	**VCS Watchdog Counter Expired** Set when the VCS timeout counter has reached the timeout thresh-hold value.
	5	**Reserved**
	4	**VCS MI Flush DW Notify** The Pipe Control packet (Fences) specified in 3D pipeline document may optionally generate an Interrupt. The Store QW associated with a fence is completed ahead of the interrupt.
	3	**VCS Error Interrupt** When this status bit is set, it indicates that the hardware has detected an error. It is set by the device upon an error condition and cleared by a CPU write of a one to the appropriate bit contained in the Error ID register followed by a write of a one to this bit in the IIR. Further information on the source of the error comes from the "Error Status Register" which along with

VIDEODECODER_INTR_VEC - VideoDecoder Interrupt Vector

		the "Error Mask Register" determine which error conditions will cause the error status bit to be set and the interrupt to occur. **Page Table Error:** Indicates a page table error. **Instruction Parser Error:** The Blitter Instruction Parser encounters an error while parsing an instruction.
	2:1	**Reserved** Access: RO Format: MBZ
	0	**VCS1 MI User Interrupt** This status bit is set when an MI_USER_INTERRUPT instruction is executed on the Video Command Parser. Note that instruction execution is not halted and proceeds normally. A mechanism such as an MI_STORE_DATA instruction is required to associate a particular meaning to a user interrupt.

VideoEnhancement Interrupt Vector

VIDEOENHANCE_INTR_VEC - VideoEnhancement Interrupt Vector

Size (in bits): 16

Default Value: 0x00000000

DWord	Bit	Description
0	15	**Catastrophic Error** This interrupt signals that a unrecoverable error (for e.g encountered fault when accessing a page mapped in Global GTT) during the engine processing. When Memory interface signals this error, the Command Streamer will stop parsing any more instructions. Scheduler is expected to reset the engine to evict the context
	14:12	**Reserved** Access: RO Format: MBZ
	11	**VECS Wait On Semaphore**
	10	**Reserved** Access: RO Format: MBZ
	9	**CS TR Invalid Tile Detection**
	8	**VECS Context Switch Interrupt**
	7	**Legacy Context Per Process Page Fault Interrupt** Fault interrupt is generated by GA fabric, not by the CS This interrupt is for handling Legacy context PPTGTT Page Fault.
	6	**VECS Watchdog Counter Expired**
	5	**Reserved** Access: RO Format: MBZ
	4	**VECS MI Flush DW Notify**
	3	**VECS Error Interrupt**
	2:1	**Reserved** Access: RO Format: MBZ
	0	**VECS MI User Interrupt**

VP8 Encoder StreamOut Format

VP8 Encoder StreamOut Format		
Source:	VideoCS	
Size (in bits):	128	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0	31:24	**MbY** Format: U8
	23:16	**MbX** Format: U8
	15:8	**MbClock16** Format: U8
	7:3	**Reserved** Access: RO Format: MBZ
	2	**MbRcFlag** Format: U1
	1	**MBLevelInterMBConformanceFlag** Format: U1
	0	**MBLevelIntraMBConformanceFlag** Format: U1
1	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**MB_Residual_BitCount** Format: U13
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**MB_Total_BitCount** Format: U13
2	31:25	**Reserved** Access: RO Format: MBZ
	24:0	**Cbp**

VP8 Encoder StreamOut Format

		Format: U25
3	31	**Reserved** Access: RO Format: MBZ
	30	**LastMbFlag** Format: U1
	29	**IntraMBFlag** Format: U1
	28:24	**MBType5Bits** Format: U5
	23:19	**Reserved** Access: RO Format: MBZ
	18	**QindexClampHigh** Format: U1
	17	**QindexClampLow** Format: U1
	16	**CoeffClampStatus** Format: U1
	15:0	**Reserved** Access: RO Format: MBZ

WDBoxOAInterrupt Vector

WDOA_INTR_VEC - WDBoxOAInterrupt Vector

Size (in bits): 16

Default Value: 0x00000000

DWord	Bit	Description
0	15:13	**Reserved** Access: RO Format: MBZ
	12	**Performance Monitoring Buffer Half-Full Interrupt** For internal trigger (timer event based) reporting, this interrupt is generated if the report buffer crosses the half full limit.
	11:8	**Reserved** Access: RO Format: MBZ
	7:6	**Reserved** Access: RO Format: MBZ
	5	**WDBox 2 Status Interrupt**
	4	**WDBox 2 End of Frame Interrupt**
	3:2	**Reserved** Access: RO Format: MBZ
	1	**WDBox 1 Status Interrupt**
	0	**WDBox 1 End of Frame Interrupt**

WD Interrupt Bit Definition

WD Interrupt Bit Definition		
Size (in bits):	16	
Default Value:	0x00000000	
The WD Interrupt Registers all share the same bit definitions from this table.		
DWord	**Bit**	**Description**
0	15:8	**Reserved** Access: RO Format: MBZ
	7	**WD_Frame_Complete** This event occurs when WD capture fully completes a frame.
	6	**WD_GTT_Fault** This event occurs when a GTT fault is detected.
	5	**WD_Vblank** This event occurs at the start of the WD internal vertical blank.
	4	**Unused_Int_4** These interrupts are currently unused.
	3	**WD_Capturing** This event occurs when WD capture starts to capture pixels.
	2	**WD_Writes_Complete** This event occurs when WD capture data writes complete for the current frame, before the data has been flushed to memory. WD Frame Complete should be used to find when the captured data can be accessed.
	1	**Unused_Int_1** These interrupts are currently unused.
	0	**Unused_Int_0** These interrupts are currently unused.

Word Data Payload Register

MDCR_W - Word Data Payload Register

Size (in bits): 256

Default Value: 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000

DWord	Bit	Description
0	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Word0** Format: U16 Specifies the slot 0 data in this payload register
1	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Word1** Format: U16 Specifies the slot 1 data in this payload register
2	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Word2** Format: U16 Specifies the slot 2 data in this payload register
3	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Word3** Format: U16 Specifies the slot 3 data in this payload register
4	31:16	**Reserved** Access: RO Format: MBZ

MDCR_W - Word Data Payload Register

	15:0	**Word4** Format: U16 Specifies the slot 4 data in this payload register
5	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Word5** Format: U16 Specifies the slot 5 data in this payload register
6	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Word6** Format: U16 Specifies the slot 6 data in this payload register
7	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Word7** Format: U16 Specifies the slot 7 data in this payload register

Word SIMD8 Atomic Operation CMPWR Message Data Payload

MDP_AOP8_W2 - Word SIMD8 Atomic Operation CMPWR Message Data Payload		
Size (in bits):	512	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Src0** Format: MDCR_W Specifies the Slot [7:0] Source 0 data
1.0-1.7	255:0	**Src1** Format: MDCR_W Specifies the Slot [7:0] Source 1 data

Word SIMD8 Data Payload

MDP_W_SIMD8 - Word SIMD8 Data Payload		
Size (in bits):	256	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Data[7:0]** Format: MDCR_W Specifies the Slot [7:0] data

Word SIMD16 Atomic Operation CMPWR Message Data Payload

MDP_AOP16_W2 - Word SIMD16 Atomic Operation CMPWR Message Data Payload

Size (in bits): 1024

Default Value: 0x00000000, 0x00000000

DWord	Bit	Description
0.0-0.7	255:0	**Src0[7:0]** Format: MDCR_W Specifies the Source 0 data for Slot [7:0]
1.0-1.7	255:0	**Src0[15:8]** Format: MDCR_W Specifies the Source 0 data for Slot [15:8]
2.0-2.7	255:0	**Src1[7:0]** Format: MDCR_W Specifies the Source 1 data for Slot [7:0]
3.0-3.7	255:0	**Src1[15:8]** Format: MDCR_W Specifies the Source 1 data for Slot [15:8]

Word SIMD16 Data Payload

MDP_W_SIMD16 - Word SIMD16 Data Payload		
Size (in bits):	512	
Default Value:	0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000	
DWord	**Bit**	**Description**
0.0-0.7	255:0	**Data[7:0]** Format: **MDCR_W** Specifies the Slot [7:0] data
1.0-1.7	255:0	**Data[15:8]** Format: **MDCR_W** Specifies the Slot [15:8] data

Intel® Iris® Xe and UHD Graphics Open Source

Programmer's Reference Manual

For the 2020-2021 11th Generation Intel Xeon®, Core™, Celeron®, Pentium® Gold Processors based on the "Tiger Lake" Platform

Volume 3: GPU Overview

December 2021, Revision 1.0

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Table of Contents

Introduction **1**

Graphics Processing Unit (GPU) 1

Rendering Engine Overview 2

Command Stream (CS) Unit 3

3D Pipeline 3

Media Pipeline 3

Thread Dispatching 3

Execution Units (EUs) 3

Shared Functions 4

Video Codec Engine 4

Introduction

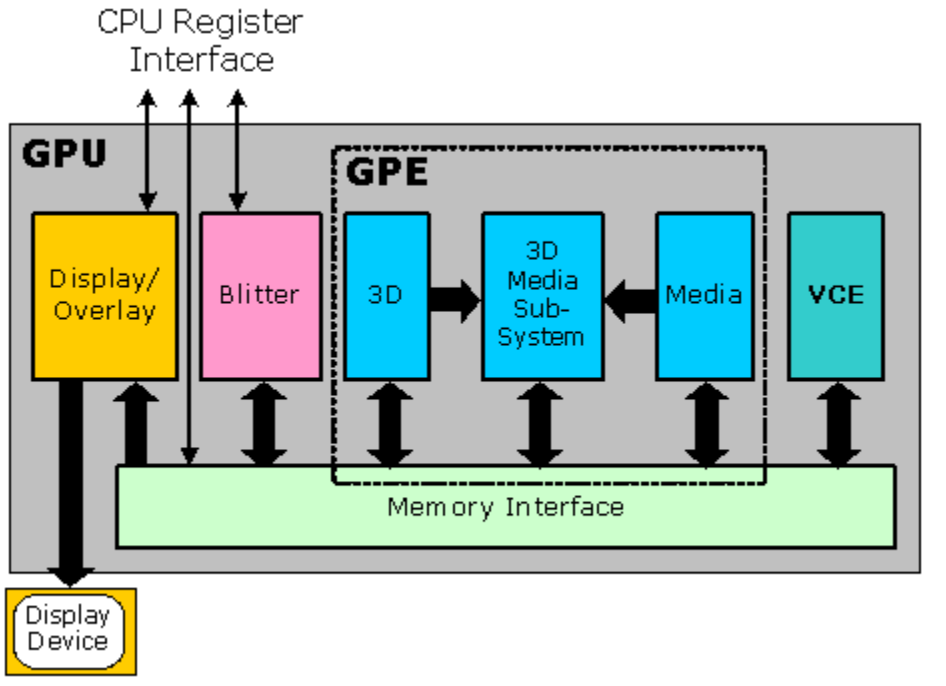
The Graphics Processing Unit, or GPU, is the general term for a rendering and media accelerator device. It consists of a combination of several independent processing engines: 'render engine' for accelerated 2D/3D rendering and processing of high-throughput parallel compute tasks, a 'media engine' for the encode/decode and related processing of video streams, and a 'display engine' for processing of displayable buffers along with hardware interfaces to display panels. Intel's GPUs may be instantiated with a CPU and other support logic within a single package as a portion of a System on Chip (SOC) solution, utilizing a shared memory subsystem, in which case the GPU is classified as 'integrated graphics'. Alternatively, the GPU may be instantiated as a stand-alone chip(s) in its own package with its own dedicated memory subsystem, and thus classified as 'discrete graphics'.

Graphics Processing Unit (GPU)

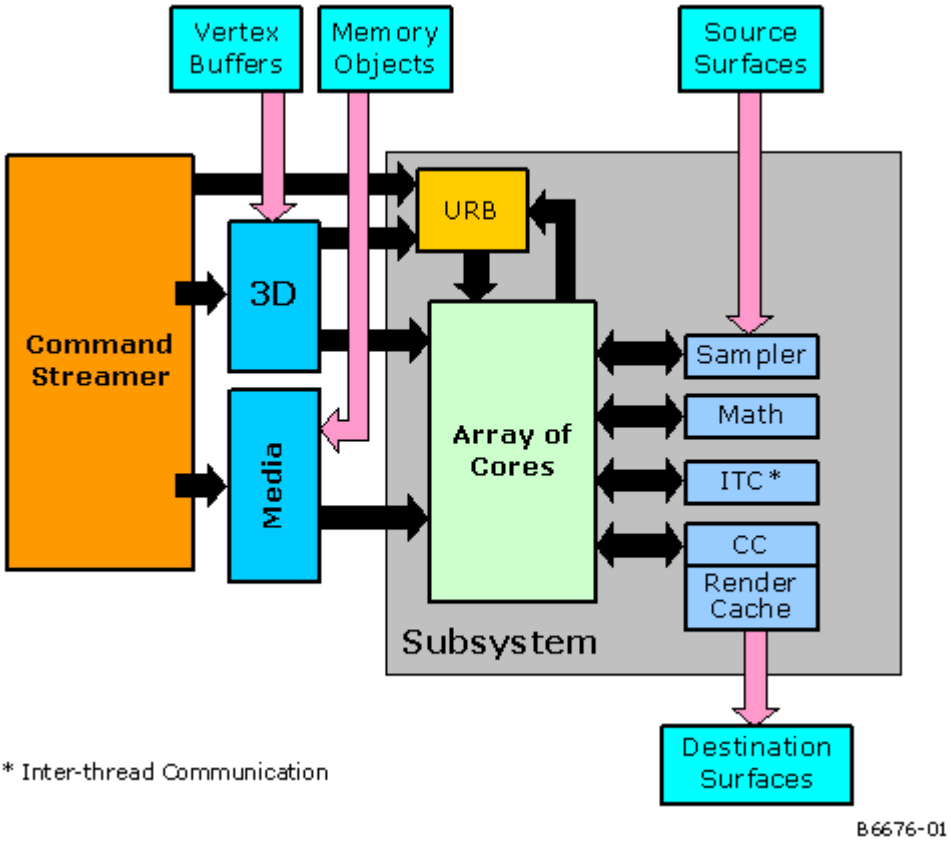
The Graphics Processing Unit is controlled by the CPU through a direct interface of memory-mapped IO registers, and indirectly by parsing commands that the CPU has placed in memory. The Display interface and Blitter (**bl**ock **i**mage **t**ransferr**er**) are controlled primarily by direct CPU register addresses, while the 3D and Media pipelines, part of Graphics Processing Engines (GPE) and the parallel Video Codec Engine (VCE) are controlled primarily through instruction lists in memory.

The subsystem contains an array of execution units, with a number of "shared functions", which receive and process messages at the request of programs. The shared functions perform critical tasks, such as sampling textures and updating the render target (usually the frame buffer). They are described by an instruction set architecture, or ISA.

Block Diagram of the GPU



Rendering Engine Overview



The subsystem consists of an array of *execution units* (*EUs)*, along with a set of *shared functions* outside the EUs that the EUs leverage for I/O and for complex computations. Programmers access the subsystem via the 3D or Media pipelines.

EUs support a rich instruction set that has been optimized to support various 3D API shader languages as well as media functions (primarily video) processing.

Shared functions are hardware units which serve to provide specialized supplemental functionality for the EUs. A shared function is implemented where the demand for a given specialized function is insufficient to justify the costs on a per-EU basis. Instead, a single instantiation of that specialized function is implemented as a stand-alone entity outside the EUs and shared among the EUs.

Invocation of the shared functionality is performed via a communication mechanism called a *message*. A message is a small self-contained packet of information created by a kernel and directed to a specific shared function. A message is defined by a range of GRF registers that hold message operands, a destination shared function ID, a function-specific encoding of the desired operation, and a destination GRF register to which any writeback response is to be directed. Messages are dispatched to the shared function under software control via the send instruction. This instruction identifies the contents of the message and the GRF register locations to direct any response.

The message construction and delivery mechanisms are general in their definition and capable of supporting a wide variety of shared functions.

Command Stream (CS) Unit

The Command Stream (CS) unit manages the use of the 3D and Media pipelines; it performs switching between pipelines and forwarding command streams to the currently active pipeline. It manages allocation of the URB and helps support the Constant URB Entry (CURBE) function.

3D Pipeline

The 3D Pipeline provides specialized 3D primitive processing functions. These functions are provided by a pipeline of "fixed function" stages (units) and threads spawned by these units. See *3D Pipeline Overview*.

Media Pipeline

The Media pipeline provides both specialized media-related processing functions and the ability to perform more general ("generic") functionality. These Media-specific functions are provided by a Video Front End (VFE) unit. A Thread Spawner (TS) unit is utilized to spawn threads requested by the VFE unit, or as required when the pipeline is used for general processing. See *Media Pipeline Overview*.

Thread Dispatching

When the 3D and Media pipelines send requests for thread initiation to the Subsystem, the thread Dispatcher receives the requests. The dispatcher performs such tasks as arbitrating between concurrent requests, assigning requested threads to hardware threads on EUs, allocating register space in each EU among multiple threads, and initializing a thread's registers with data from the fixed functions and from the URB. This operation is largely transparent to software.

Execution Units (EUs)

The Execution Units (EUs) are the programmable shader units of the Architecture. Each is a stand-alone programmable computational unit used for execution of 3D shaders and media/gpgpu kernels. Internally each is capable of multi-issue SIMD execution, and their hardware multi-threaded operation provides a very high-efficiency execution environment in the face of long data latencies typically associated with memory accesses. Each hardware thread within an EU has a dedicated large-capacity high-bandwidth register file (GRF) and associated independent thread-state. Execution is multi-issue per clock to pipelines capable of integer, single and double precision floating point operations, SIMD branch capability, logical operations, transcendental operations, and other miscellaneous operations. Communication to support units (shared functions) for operations such as texture sampling or scatter/gather load/stores is via 'messages' programmatically constructed and 'sent' to those functions, with dependency hardware causing the issuing thread to sleep until the requested data has been returned.

EU instance count varies by product generation, as well as by SKU within a given generation, and their capabilities have evolved over the many generation of the Architecture. Please see "Device Attributes" in the "Configuration" chapter for specific rates and capacities associated with Execution Units.

Shared Functions

Shared functions are hardware units that provide specialized supplemental functionality for the EUs. A shared function is implemented where the demand for a given specialized function is insufficient to justify the costs on a per-EU basis. Instead, a single instantiation of that specialized function is implemented as a stand-alone entity outside the EUs and shared among the EUs.

Invocation of the shared functionality is performed via a communication mechanism called a message. A message is a small self-contained packet of information created by a kernel and directed to a specific shared function.

Programming Note	
Context:	Communication mechanism in shared functions
The message is defined by a range of Message Register File (MRF) registers that hold message operands, a destination shared function ID, a function-specific encoding of the desired operation, and a destination General Register File (GRF) register to which any writeback response is directed.	

Messages are dispatched to the shared function under software control via the *send* instruction. This instruction identifies the contents of the message and the GRF register locations to direct any response.

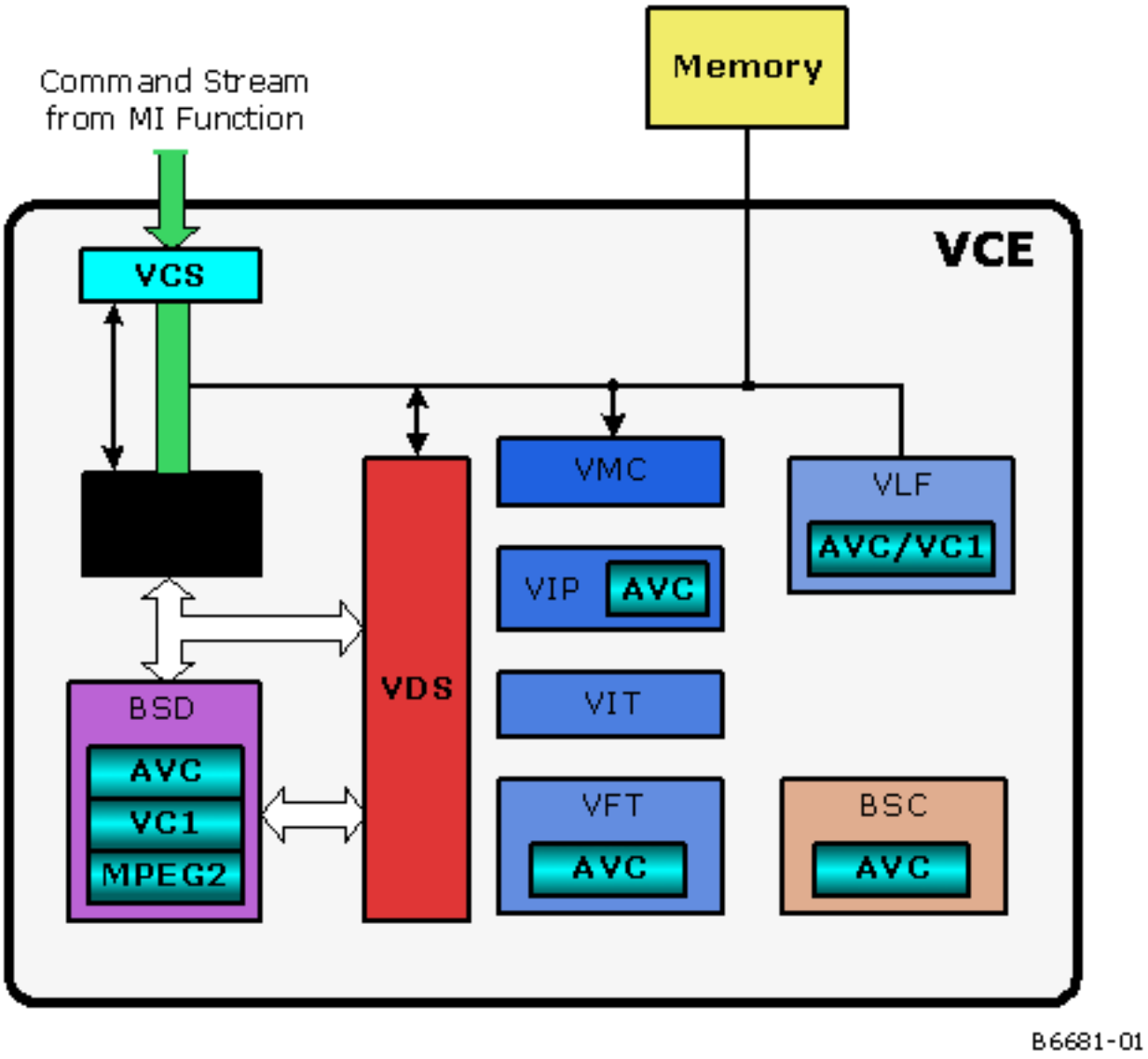
The message construction and delivery mechanisms are general in their definition and capable of supporting a wide variety of shared functions.

Video Codec Engine

The parallel Video Codec Engine (VCE) is a fixed function video decoder and encoder engine. It is also referred to as the multi-format codec (MFX) engine, as a unified fixed function pipeline is implemented to support multiple video coding standards such as MPEG2, VC1, and AVC:

- VCS - VCE Command Streamer unit
- BSD - Bitstream Decoder unit
- VDS - Video Dispatcher unit
- VMC - Video Motion Compensation unit
- VIP - Video Intra Prediction unit
- VIT - Video Inverse Transform unit
- VLF - Video Loop Filter unit
- VFT - Video Forward Transform unit (encoder only)
- BSC - Bitstream Encoder unit (encoder only)

VCE Diagram



Device	AVC BSD	VC1 BSD	AVC Dec	VC1 Dec	MPEG2 Dec	AVC Enc
	No	No	Yes	Yes	Yes	Yes

Intel® Iris® Xe and UHD Graphics Open Source

Programmer's Reference Manual

For the 2020-2021 11th Generation Intel Xeon®, Core™, Celeron®, Pentium® Gold Processors based on the "Tiger Lake" Platform

Volume 4: Configurations

December 2021, Revision 1.0

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Table of Contents

Configurations **1**

Product Mapping Table 1

Top Level Block Diagrams 2

Device Attributes 3

Device Attributes Xe_M 7

Device Attributes Display 8

Steppings and Device IDs 9

Configurations

This chapter contains configurations details as described in the following sections:

- Top Level Block Diagrams
- Device Attributes
- Steppings and Device IDs

Product Mapping Table

Product Configuration Attribute Table			
Product Family	**TGL**		
SKU Name	**UP4**	**UP3**	**H81**
Status	**POR**	**POR**	**POR**
Global Attributes			
Render Engine	1x6x16	1x6x16	1x2x16
Media Engine	Xe_M	Xe_M	Xe_M
Display Engine	Xe_D	Xe_D	Xe_D
Chassis			
LLC Size	12MB	12MB	24MB
In-Package Memory	N/A	N/A	N/A

Top Level Block Diagrams

Overview

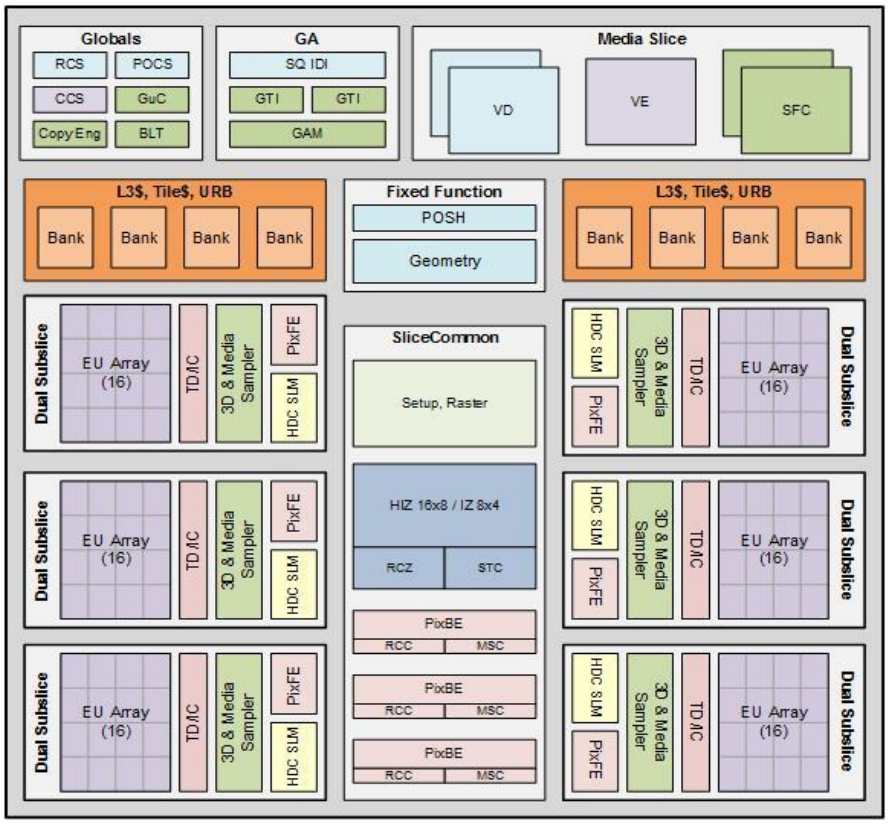
The slice is constructed of 6 dual subslices, each containing 16eus and a double sampler each capable of 8 tex/clk. Slice common will be constructed of a single Z pipe, capable of 16x8 HIZ and 8x4 IZ. 3 Color pipes, each of 8f8b; sharing the compression block for area efficiency. Total L3 cache size is defined to be 4MB, built from 8 banks of 480kB each. Initial sizing includes 50% growth of shared L3 data cache and URB, from 256kB->384kB per bank and maintains 128kB of tile cache per bank.

The machine will consist of a single geometry fixed function pipeline and maintain the position only shading (POSH) pipeline introduced in previous IP.

It will include support for a second IDI port to sustain the ~100GB/s memory bandwidth requirement to support the 96eu machine. Memory subsystem in this timeframe expected to be of similar bandwidth. As such, a single IDI is no longer sufficient as peak BW is limited to 64GB/s per direction at 1GHz. A single GAM will be designed to support both IDIs, offering a combined 128r+128w bytes per clock.

Media fixed function blocks are as follows: 2 VDBoxes, 1 VEBox, and 2 SFC. These assets are deemed sufficient to meet usage and throughput requirements for a majority of client SKUs. Within the slice, there will be a total of 6 VME and 6 AVS; built 1 per dual subslice. While there are inversions on VME assets vs previous generations; native support for HEVC 422 encode and random access will enable VDEnc to serve these usages at a lower power than previously offered with VME.

1x6x16



Device Attributes

The following table lists detailed GT device attributes for proposed SKUs.

NOTE: This information is preliminary, and subject to change.

Product Family	TGL			
Architectural Name	1x2x16	1x3x16	1x5x16	1x6x16
SKU Name	GT1	GT1.xf	GT1.xf	GT2
	Global Attributes			
Slice Count	1	1	1	1
Dual-Subslice (DSS) Count	2	3	5	6
EU/DSS	16	16	16	16
EU Count (Total)	32	48	80	96
Threads / EU	7	7	7	7
Thread Count (Total)	224	336	560	672
FLOPs/Clk - Half Precision, MAD (peak)	1024	1536	2560	3072
FLOPs/Clk - Single Precision, MAD (peak)	512	768	1280	1536
FLOPs/Clk - Double Precision, MAD (peak)	N/A	N/A	N/A	N/A
Unslice clocking (coupled/decoupled from Cr slice)	Coupled	Coupled	Coupled	Coupled
GTI/Ring Interfaces	2	2	2	2
GTI bandwidth (bytes/unslice-clk)	r: 128 w: 128	r: 128 w: 128	r: 128 w: 128	r: 128 w: 128
eDRAM Support	N/A	N/A	N/A	N/A
Graphics Virtual Address Range	48 bit	48 bit	48 bit	48 bit
Graphics Physical Address Range	39 bit	39 bit	39 bit	39 bit
	Cacches & Dedicated Memories			
L3 Cache, total size (bytes)[1]	1920k	2880k	3840k	3840k
L3 Cache, bank count[1]	4	6	8	8
L3 Cache, bandwidth (bytes/clk)	4x 64 R|W	6x 64 R|W	8x 64 R|W	8x 64 R|W
L3 Cache, URB bandwidth (bytes/clk)[(1)]	4x 64 R|W	6x 64 R|W	8x 64 R|W	8x 64 R|W
L3 Cache, D$ Size (Kbytes)[(2)]	704K	1056K	1408K	1408K
L3 Cache, Tile cache size (Kbytes)[(2)]	1024K	1536K	2048K	2048K
L3 Cache, Command buffer cache size (Kbytes)[(2)]	64K	96K	128K	128K
URB Size (kbytes)[(2)]	256K	384K	512K	512K
SLM Size (kbytes)	256k	384k	640k	768k
Instruction Cache (instances, bytes ea.)	2x 48k	3x 48k	5x 48k	6x 48k

Color Cache (RCC, bytes)	1x 32k	2x 32k	3x 32k	3x 32k
MSC Cache (MSC, bytes)	1x 16k	2x 16k	3x 16k	3x 16k
HiZ Cache (HZC, bytes)	1x 12k	1x 12k	1x 12k	1x 12k
Z Cache (RCZ, bytes)	1x 32k	1x 32k	1x 32k	1x 32k
Stencil Cache (STC, bytes)	1x 8k	1x 8k	1x 8k	1x 8k
	Instruction Issue Rates			
FMAD, SP (ops/EU/clk)	8	8	8	8
FMUL, SP (ops/EU/clk)	8	8	8	8
FADD, SP (ops/EU/clk)	8	8	8	8
MIN,MAX, SP (ops/EU/clk)	8	8	8	8
CMP, SP (ops/EU/clk)	8	8	8	8
INV, SP (ops/EU/clk)	2	2	2	2
SQRT, SP (ops/EU/clk)	2	2	2	2
RSQRT, SP (ops/EU/clk)	2	2	2	2
LOG, SP (ops/EU/clk)	2	2	2	2
EXP, SP (ops/EU/clk)	2	2	2	2
IDIV, SP (ops/EU/clk)	1-6	1-6	1-6	1-6
TRIG, SP (ops/EU/clk)	2	2	2	2
	Load Store			
Data Ports (HDC)	2	3	5	6
L3 Load/Store - same addresses within msg (Bytes/clk)	128	192	320	384
L3 Load/Store - unique addresses within msg (Bytes/clk)	128	192	320	384
SLM Load//Store - same addresses within msg (Bytes/clk)	256	384	640	768
SLM Load//Store - unique addresses within msg (Bytes/clk)	256	384	640	768
Atomic, Local 32b - same addresses within msg (dwords/clk)	2	3	5	6
Atomic, Global 32b - unique addresses within msg (dwords/clk)	32	48	80	96
	3D Attributes			
Geometry pipes	1	1	1	1
Samplers (3D)	2	3	5	6
2D Texel Rate, point, 32b (tex/clk)	16	24	40	48
2D Texel Rate, point, 64b (tex/clk)	16	24	40	48
2D Texel Rate, point, 128b (tex/clk)	16	24	40	48
2D Texel Rate, bilinear, 32b (tex/clk)	16	24	40	48
2D Texel Rate, bilinear, 64b (tex/clk)	16	24	40	48
2D Texel Rate, bilinear, 128b (tex/clk)	4	6	10	12

2D Texel Rate, trilinear, 32b (tex/clk)	8	12	20	24
2D Texel Rate, trilinear, 64b (tex/clk)	8	12	20	24
2D Texel Rate, trilinear, 128b (tex/clk)	2	3	5	6
2D Texel Sample Rate, aniso 2x (MIP nearest), 32b (tex/clk)	16	24	40	48
2D Texel Sample Rate, aniso 4x (MIP nearest), 32b (tex/clk)	8	12	20	24
2D Texel Sample Rate, aniso 8x (MIP nearest), 32b (tex/clk)	4	6	10	12
2D Texel Sample Rate, aniso 16x (MIP nearest), 32b (tex/clk)	2	3	5	6
3D Texel Sample Rate, point, 32b (tex/clk)	16	24	40	48
3D Texel Sample Rate, point, 64b (tex/clk)	16	24	40	48
3D Texel Sample Rate, point, 128b (tex/clk)	8	12	20	24
3D Texel Sample Rate, bilinear, 32b (tex/clk)	8	12	20	24
3D Texel Sample Rate, bilinear, 64b (tex/clk)	8	12	20	24
3D Texel Sample Rate, bilinear, 128b (tex/clk)	2	3	5	6
HiZ Rate, (ppc)	1x128	1x128	1x128	1x128
IZ Rate, (ppc)	1x32	1x32	1x32	1x32
Stencil Rate (ppc)	1x128	1x128	1x128	1x128
(500 MHz, DDR-4267; Range depends on dynamic compression ratio)				
Pixel Rate, fill, 32bpp (pix/clk, RCC hit)	8	12	20	24
Pixel Rate, fill, 32bpp (pix/clk, LLC hit @ 1.0x unslice clk)	8	12	20	24
Pixel Rate, fill, 32bpp (pix/clk, LLC hit, @ 1.5x unslice clk)	N/A	N/A	N/A	N/A
Pixel Rate, fill, 32bpp (pix/clk, memory, @ 1.0x unslice clk)	8	12	20	24
Pixel Rate, fill, 32bpp (pix/clk, memory, @ 1.5x unslice clk)	N/A	N/A	N/A	N/A
(500 MHz, DDR-4267 or eDRAM; Range depends on dynamic compression ratio)				
Pixel Rate, blend, 32bpp (p/clk, RCC hit)	8	12	20	24
Pixel Rate, blend, 32bpp (p/clk, RCC miss, @ 1.0x unslice clk)	8	12	20	24
Pixel Rate, blend, 32bpp (p/clk, RCC miss, @ 1.5x unslice clk)	N/A	N/A	N/A	N/A
Pixel Rate, blend, 32bpp (pix/clk, memory, @ 1.0x unslice clk)	8	12	17-20	17-24
Pixel Rate, blend, 32bpp (pix/clk, memory, @ 1.5x unslice clk)	N/A	N/A	N/A	N/A

Notes:

* Architectural Name = Slice Count x Subslice Count x EUs per Subslice

(1) L3 cache and URB share the write bandwidth. Read bandwidths of 64B/clk can be achieved independently

(2) URB/Data cache/Tile cache/Command buffer cache sizes are programmable. Indicative values presented in this table.

Device Attributes Xe_M

NOTE: This information is preliminary, and subject to change.

	Product Configuration Attribute Table	
Product Family	**TGL**	
Architectural Name	**1x2x16**	**1x6x16**
SKU Name	**GT1**	**GT2**
	Media Attributes	
Samplers (VME)	2	6
Samplers (AVS)	2	6
VDBox Instances (See)	2	2
VEBox Instances	1	1
SFC Instances	2	2

Device Attributes Display

Refer to display overview

Steppings and Device IDs

SKUs and Device IDs

The following table details all currently planned SKUs. This information is subject to change at any time based on roadmap plans or new steppings.

Segment	SKU	Config	EUs	VDBoxes	TDP	Gfx Name	CPU Stepping	GT Version	Display Version	Dev2 ID	RevID
Mobile	UP3 / H35	1x6x16	96,80	2	15W / 35W	Intel(R) Iris(R) Xe Graphics	B2	GT2 B0	C0	0x9A49	0x1
Mobile	UP3	1x3x16	48	1	15W	Intel(R) UHD Graphics	B2	GT2 B0	C0	0x9A78	0x1
Mobile	UP3 / H35	1x6x16	96,80	2	15W / 35W	Intel(R) Iris(R) Xe Graphics	C0	GT2 C0	D0	0x9A49	0x3
Mobile	UP3	1x3x16	48	1	15W	Intel(R) UHD Graphics	C0	GT2 C0	D0	0x9A78	0x3
Mobile	UP4	1x6x16	96,80	2	5.2W	Intel(R) Iris(R) Xe Graphics	B2	GT2 B0	C0	0x9A40	0x1
Mobile	UP4	1x3x16	48	1	5.2W	Intel(R) UHD Graphics	B2	GT2 B0	C0	0x9A78	0x1
Mobile	UP4	1x6x16	96,80	2	5.2W	Intel(R) Iris(R) Xe Graphics	C0	GT2 C0	D0	0x9A40	0x3
Mobile	UP4	1x3x16	48	1	5.2W	Intel(R) UHD Graphics	C0	GT2 C0	D0	0x9A78	0x3
Perf Mobile	H81	1x2x16	32	2	45W	Intel(R) UHD Graphics	R0	GT1 B0	D0	0x9A60	0x1
Perf Mobile Low Graphics	H81	1x1x16	16	1	45W	Intel(R) UHD Graphics	R0	GT1 B0	D0	0x9A68	0x1
Mobile WS	H81	1x2x16	32	2	45W	Intel(R) UHD Graphics	R0	GT1 B0	D0	0x9A70	0x1

Notes:

The Intel UHD Graphics Device ID SKUs are unified for both UP3 and UP4, eg: there is no unique device ID between UP3 and UP4 for 48 EU devices.

The UP3 and H35 SKUs share the same DevIDs and are equivalent other than being focused on different TDPs.

Intel® Iris® Xe and UHD Graphics Open Source

Programmer's Reference Manual

For the 2020-2021 11th Generation Intel Xeon®, Core™, Celeron®, Pentium® Gold Processors based on the "Tiger Lake" Platform

Volume 5: Memory Data Formats

December 2021, Revision 1.0

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Table of Contents

Memory Data Formats 1

Unsigned Normalized (UNORM) 1

Gamma Conversion (SRGB) 1

Signed Normalized (SNORM) 1

Unsigned Integer (UINT/USCALED) 2

Signed Integer (SINT/SSCALED) 2

Floating Point (FLOAT) 2

64-bit Floating Point 2

32-bit Floating Point 3

16-bit Floating Point 3

11-bit Floating Point 4

10-bit Floating Point 5

Shared Exponent 6

Common Surface Formats 6

Non-Video Surface Formats 6

Compressed Surface Formats 8

Video Pixel/Texel Formats 55

Additional Video Formats 60

Raw Format 69

Surface Memory Organizations 69

Display, Overlay, Cursor Surfaces 69

2D Render Surfaces 69

2D Monochrome Source 69

2D Color Pattern 69

3D Color Buffer (Destination) Surfaces 70

3D Depth Buffer Surfaces 70

3D Separate Stencil Buffer Surfaces 71

Surface Layout and Tiling 71

Maximum Surface Size in Bytes 72

Surface Padding Requirements 102

Address Tiling Function Introduction 105

Linear vs Tiled Storage 105

Auxiliary Surfaces For Sampled Tiled Resources 108
Tile Formats 109
Tiling Algorithm 113
Tiling Support 123
Per-Stream Tile Format Support 127
Memory Compression 128
CCS Surface Encodings 128
Media Memory Compression 130
Memory Object Overview 130

Memory Data Formats

This chapter describes the attributes associated with the memory-resident data objects operated on by the graphics pipeline. This includes object types, pixel formats, memory layouts, and rules/restrictions placed on the dimensions, physical memory location, pitch, alignment, etc. with respect to the specific operations performed on the objects.

Unsigned Normalized (UNORM)

An unsigned normalized value with *n* bits is interpreted as a value between 0.0 and 1.0. The minimum value (all 0's) is interpreted as 0.0, the maximum value (all 1's) is interpreted as 1.0. Values in between are equally spaced. For example, a 2-bit UNORM value would have the four values 0, 1/3, 2/3, and 1.

If the incoming value is interpreted as an n-bit integer, the interpreted value can be calculated by dividing the integer by 2^n-1.

Gamma Conversion (SRGB)

Gamma conversion is only supported on UNORM formats. If this flag is included in the surface format name, it indicates that a reverse gamma conversion is to be done after the source surface is read, and a forward gamma conversion is to be done before the destination surface is written.

Signed Normalized (SNORM)

Programming Note	
Context:	Signed normalized value in memory data formats.
A signed normalized value with *n* bits is interpreted as a value between -1 and +1.0. If the incoming value is interpreted as a 2's-complement n-bit integer, the interpreted value can be calculated by dividing the integer by 2^{n-1}-1. The most negative value of -2^{n-1} will result in a value slightly smaller than -1.0. This value is clamped to -1.0; thus, there are two representations of -1.0 in SNORM format.	

Unsigned Integer (UINT/USCALED)

The UINT and USCALED formats interpret the source as an unsigned integer value with *n* bits with a range of 0 to 2^n-1.

The UINT formats copy the source value to the destination (zero-extending if required), keeping the value as an integer.

The USCALED formats convert the integer into the corresponding floating-point value (e.g., 0x03 --> 3.0f). For 32-bit sources, the value is rounded to nearest even.

Signed Integer (SINT/SSCALED)

A signed integer value with *n* bits is interpreted as a 2's complement integer with a range of -2^{n-1} to $+2^{n-1}$-1.

The SINT formats copy the source value to the destination (sign-extending if required), keeping the value as an integer.

The SSCALED formats convert the integer into the corresponding floating-point value (e.g., 0xFFFD --> -3.0f). For 32-bit sources, the value is rounded to nearest even.

Floating Point (FLOAT)

Refer to IEEE Standard 754 for Binary Floating-Point Arithmetic. The IA-32 Intel (R) Architecture Software Developer's Manual also describes floating point data types.

64-bit Floating Point

Bit	Description
63	**Sign (s)**
62:52	**Exponent (e)** Biased Exponent
51:0	**Fraction (f)** Does not include "hidden one"

The value of this data type is derived as:

- if e == b'11..11' and f != 0, then v is NaN regardless of s
- if e == b'11..11' and f == 0, then v = $(-1)^s$*infinity (signed infinity)
- if 0 < e < b'11..11', then v = $(-1)^s*2^{(e-1023)}*(1.f)$
- if e == 0 and f != 0, then v = $(-1)^s*2^{(e-1022)}*(0.f)$ (denormalized numbers)
- if e == 0 and f == 0, then v = $(-1)^s$*0 (signed zero)

32-bit Floating Point

Bit	Description
31	**Sign (s)**
30:23	**Exponent (e)** Biased Exponent
22:0	**Fraction (f)** Does not include "hidden one"

The value of this data type is derived as:

- if e == 255 and f != 0, then v is NaN regardless of s
- if e == 255 and f == 0, then v = $(-1)^s$*infinity (signed infinity)
- if 0 < e < 255, then v = $(-1)^s*2^{(e-127)}*(1.f)$
- if e == 0 and f != 0, then v = $(-1)^s*2^{(e-126)}*(0.f)$ (denormalized numbers)
- if e == 0 and f == 0, then v = $(-1)^s*0$ (signed zero)

16-bit Floating Point

Bit	Description
15	**Sign (s)**
14:10	**Exponent (e)** Biased Exponent
9:0	**Fraction (f)** Does not include "hidden one"

The value of this data type is derived as:

- if e == 31 and f != 0, then v is NaN regardless of s
- if e == 31 and f == 0, then v = $(-1)^s$*infinity (signed infinity)
- if 0 < e < 31, then v = $(-1)^s*2^{(e-15)}*(1.f)$
- if e == 0 and f != 0, then v = $(-1)^s*2^{(e-14)}*(0.f)$ (denormalized numbers)
- if e == 0 and f == 0, then v = $(-1)^s*0$ (signed zero)

The following table represents relationship between 32 bit and 16 bit floating point ranges:

flt32 exponent	Unbiased exponent	Normalization	flt16 exponent	flt16 fraction
255				
254	127			
...				
127+16	16	Infinity	31	1.1111111111
127+15	15	Max exponent	30	1.xxxxxxxxxx
127	0		15	1.xxxxxxxxxx
113	-14	Min exponent	1	1.xxxxxxxxxx
112		Denormalized	0	0.1xxxxxxxxx
111		Denormalized	0	0.01xxxxxxxx
110		Denormalized	0	0.001xxxxxxx

flt32 exponent	Unbiased exponent	Normalization	flt16 exponent	flt16 fraction
109		Denormalized	0	0.0001xxxxxx
108		Denormalized	0	0.00001xxxxx
107		Denormalized	0	0.000001xxxx
106		Denormalized	0	0.0000001xxx
115		Denormalized	0	0.00000001xx
114		Denormalized	0	0.000000001x
113		Denormalized	0	0.0000000001
112		Denormalized	0	0.0
...				
0			0	0.0

Conversion from the 32-bit floating point format to the 16-bit format should be done with round to nearest even.

11-bit Floating Point

Bits	Description
10:6	**Exponent (e):** Biased exponent (the bias depends on e)
5:0	**Fraction (f):** Fraction bits to the right of the binary point

The value v of an 11-bit floating-point number is calculated from e and f as:

- if e == 31 and f != 0 then v = NaN
- if e == 31 and f == 0 then v = +infinity
- if 0 < e < 31, then v = $2^{(e-15)}$*(1.f)
- if e == 0 and f != 0, then v = $2^{(e-14)}$*(0.f) (denormalized numbers)
- if e == 0 and f == 0, then v = 0 (zero)

There is no sign bit and negative values are not represented.

The 11-bit floating-point format has one more bit of fractional precision than the 10-bit floating-point format.

The maximum representable finite value is 1.111111b * 2^{15} = FE00h = 65024.

10-bit Floating Point

Bits	Description
9:5	**Exponent (e):** Biased exponent (the bias depends on e)
4:0	**Fraction (f):** Fraction bits to the right of the binary point

The value v of a 10-bit floating-point number is calculated from e and f as:

- if e == 31 and f != 0 then v = NaN
- if e == 31 and f == 0 then v = +infinity
- if 0 < e < 31, then v = $2^{(e-15)}$*(1.f)
- if e == 0 and f != 0, then v = $2^{(e-14)}$*(0.f) (denormalized numbers)
- if e == 0 and f == 0, then v = 0 (zero)

There is no sign bit and negative values are not represented.

The maximum representable finite value is 1.11111b * 2^{15} = FC00h = 64512.

The R10G10B10_FLOAT_A2_UNORM format has a 4-bit exponent and a 6-bit fraction.

Bits	Description
9:6	**Exponent (e):** Biased exponent (the bias depends on e)
5:0	**Fraction (f):** Fraction bits to the right of the binary point

The value v of a 10-bit floating-point number is calculated from e and f as:

- if e == 15 and f != 0 then v = NaN
- if e == 15 and f == 0 then v = +infinity
- if 0 < e < 15, then v = $2^{(e-7)}$*(1.f)
- if e == 0 and f != 0, then v = 2^{-6}*(0.f) (denormalized numbers)
- if e == 0 and f == 0, then v = 0 (zero)

There is no sign bit and negative values are not represented.

The maximum representable finite value is 1.111111b * 2^{7} = FC00h = 254.

Shared Exponent

The R9G9B9E5_SHAREDEXP format contains three channels that share an exponent. The three fractions assume an impled "0" rather than an implied "1" as in the other floating-point formats. This format does not support infinity and NaN values. There are no sign bits, only positive numbers and zero can be represented. The value of each channel is determined as follows, where "f" is the fraction of the corresponding channel, and "e" is the shared exponent.

$v = (0.f)*2^{(e-15)}$

Bit	Description
31:27	**Exponent (e)** Biased Exponent
26:18	**Blue Fraction**
17:9	**Green Fraction**
8:0	**Red Fraction**

Common Surface Formats

This section documents surfaces and how they are stored in memory, including 3D and video surfaces, including the details of compressed texture formats. Also covered are the surface layouts based on tiling mode and surface type.

Non-Video Surface Formats

This section describes the lowest-level organization of a surfaces containing discrete "pixel" oriented data (e.g., discrete pixel (RGB,YUV) colors, subsampled video data, 3D depth/stencil buffer pixel formats, bump map values etc. Many of these pixel formats are common to the various pixel-oriented memory object types.

Surface Format Naming

Unless indicated otherwise, all pixels are **stored** in "**little endian**" byte order. i.e., pixel bits 7:0 are stored in byte *n*, pixel bits 15:8 are stored in byte *n*+1, and so on. The format labels include color components in little endian order (e.g., R8G8B8A8 format is physically stored as R, G, B, A).

The name of most of the surface formats specifies its format. Channels are listed in little endian order (LSB channel on the left, MSB channel on the right), with the channel format specified following the channels with that format. For example, R5G5_SNORM_B6_UNORM contains, from LSB to MSB, 5 bits of red in SNORM format, 5 bits of green in SNORM format, and 6 bits of blue in UNORM format.

Intensity Formats

All surface formats containing "I" include an intensity value. When used as a source surface for the sampling engine, the intensity value is replicated to all four channels (R,G,B,A) before being filtered. Intensity surfaces are not supported as destinations.

Luminance Formats

All surface formats containing "L" include a luminance value. When used as a source surface for the sampling engine, the luminance value is replicated to the three color channels (R,G,B) before being filtered. The alpha channel is provided either from another field or receives a default value. Luminance surfaces are not supported as destinations.

R1_UNORM

When used as a texel format, the R1_UNORM format contains 8 1-bit Intensity (I) values that are replicated to all color channels. Note that T0 of byte 0 of a R1_UNORM-formatted texture corresponds to Texel[0,0]. This is different from the format used for monochrome sources in the BLT engine.

7	6	5	4	3	2	1	0
T7	T6	T5	T4	T3	T2	T1	T0

Bit	Description
T0	**Texel 0** On texture reads, this (unsigned) 1-bit value is replicated to all color channels. Format: U1
...	**...**
T7	**Texel 7** On texture reads, this (unsigned) 1-bit value is replicated to all color channels. Format: U1

Compressed Surface Formats

This section contains information on the internal organization of compressed surface formats.

ETC1_RGB8

This format compresses UNORM RGB data using an 8-byte compression block representing a 4x4 block of texels. The texels are labeled as texel[row][column] where both row and column range from 0 to 3. Texel[0][0] is the upper left texel.

The 8-byte compression block is laid out as follows.

High 24 bits if "diff" is zero (individual mode):

Bits	Description
7:4	R0[3:0]
3:0	R1[3:0]
15:12	G0[3:0]
11:8	G1[3:0]
23:20	B0[3:0]
19:16	B1[3:0]

High 24 bits if "diff" is one (differential mode):

Bits	Description
7:3	R0[4:0]
2:0	dR1[2:0]
15:11	G0[4:0]
10:8	dG1[2:0]
23:19	B0[4:0]
18:16	dB1[2:0]

Low 40 bits:

Bits	Description
31:29	lum table index for sub-block 0
28:26	lum table index for sub-block 1
25	diff
24	flip
39	texel[3][3] index MSB
38	texel[2][3] index MSB
37	texel[1][3] index MSB
36	texel[0][3] index MSB
35	texel[3][2] index MSB

Bits	Description
34	texel[2][2] index MSB
33	texel[1][2] index MSB
32	texel[0][2] index MSB
47	texel[3][1] index MSB
46	texel[2][1] index MSB
45	texel[1][1] index MSB
44	texel[0][1] index MSB
43	texel[3][0] index MSB
42	texel[2][0] index MSB
41	texel[1][0] index MSB
40	texel[0][0] index MSB
55	texel[3][3] index LSB
54	texel[2][3] index LSB
53	texel[1][3] index LSB
52	texel[0][3] index LSB
51	texel[3][2] index LSB
50	texel[2][2] index LSB
49	texel[1][2] index LSB
48	texel[0][2] index LSB
63	texel[3][1] index LSB
62	texel[2][1] index LSB
61	texel[1][1] index LSB
60	texel[0][1] index LSB
59	texel[3][0] index LSB
58	texel[2][0] index LSB
57	texel[1][0] index LSB
56	texel[0][0] index LSB

The 4x4 is divided into two 8-pixel sub-blocks, either two 2x4 sub-blocks or two 4x2 sub-blocks controlled by the "flip" bit. If flip=0, sub-block 0 is the 2x4 on the left and sub-block 1 is the 2x4 on the right. If flip=1, sub-block 0 is the 4x2 on the top and sub-block 1 is the 4x2 on the bottom.

The "diff" bit controls whether the red/green/blue values (R0/G0/B0/R1/G1/B1) are stored as one 444 value per sub-block ("individual" mode with diff = 0), or a single 555 value for the first sub-block (R0/G0/B0) and a 333 delta value (dR1/dG1/dB1) for the second sub-block ("differential" mode with diff = 1). The delta values are 3-bit two's-complement values that hold values in the range [-4,3]. These values are added to the 5-bit values for sub-block 0 to obtain the 5-bit values for sub-block 1 (if the value is outside of the range [0,31], the result of the decompression is undefined). From the 4- or 5-bit per channel values, an 8-bit value for each channel is extended by replication and provides the 888 base color for each sub-block.

For each sub-block one of 8 different luminance columns is selected based on the 3-bit lum table index. Then each texel selects one of the 4 rows of the selected column with a 2-bit per-texel index. The chosen value in the table is added to the 8-bit base color for the sub-block (obtained in the previous step) to obtain the texel's color. Values in the table are given in decimal, representing an 8-bit UNORM as an 8-bit signed integer.

Luminance Table

	0	1	2	3	4	5	6	7
0	2	5	9	13	18	24	33	47
1	8	17	29	42	60	80	106	183
2	-2	-5	-9	-13	-18	-24	-33	-47
3	-8	-17	-29	-42	-60	-80	-106	-183

ETC2_RGB8 and ETC2_SRGB8

The ETC2_RGB8 format builds on top of ETC1_RGB8, using a set of invalid bit sequences to enable three new modes. The two modes of ETC1_RGB8 are also supported with ETC2_RGB8, and will not be documented in this section as they are covered in the ETC1_RGB8 section.

The detection of the three new modes is based on RGB and diff bits in locations as defined for ETC1 differential mode. The mode is determined as follows (x indicates don't care):

diff	Rt	Gt	Bt	mode
0	x	x	x	individual
1	0	x	x	T
1	1	0	x	H
1	1	1	0	planar
1	1	1	1	differential

The inputs in the above table are defined as follows:

```
Rt = (R0 + dR1) in [0,31]
Gt = (G0 + dG1) in [0,31]
Bt = (G0 + dB1) in [0,31]
```

8-byte compression block for mode determination

Bits	Description
7:3	R0[4:0]
2:0	dR1[2:0]
15:11	G0[4:0]

Bits	Description
10:8	dG1[2:0]
23:19	B0[4:0]
18:16	dB1[2:0]
31:26	ignored
25	diff
24	ignored
63:32	ignored

The fields in the table above are used *only* for mode determination. Some of the bits in this table are overloaded with other values within each mode. The algorithm is defined such that there is no ambiguity in modes when this is done.

T mode

The "T" mode has the following bit definition:

8-byte compression block for "T" mode

Bits	Description
7:5	ignored
4:3	R0[3:2]
2	ignored
1:0	R0[1:0]
15:12	G0[3:0]
11:8	B0[3:0]
23:20	R1[3:0]
19:16	G1[3:0]
31:28	B1[3:0]
27:26	di[2:1]
25	diff = 1
24	di[0]
39	texel[3][3] index MSB
38	texel[2][3] index MSB
37	texel[1][3] index MSB
36	texel[0][3] index MSB
35	texel[3][2] index MSB
34	texel[2][2] index MSB
33	texel[1][2] index MSB
32	texel[0][2] index MSB
47	texel[3][1] index MSB

Bits	Description
46	texel[2][1] index MSB
45	texel[1][1] index MSB
44	texel[0][1] index MSB
43	texel[3][0] index MSB
42	texel[2][0] index MSB
41	texel[1][0] index MSB
40	texel[0][0] index MSB
55	texel[0][0] index LSB
54	texel[2][3] index LSB
53	texel[1][3] index LSB
52	texel[0][3] index LSB
51	texel[3][2] index LSB
50	texel[2][2] index LSB
49	texel[1][2] index LSB
48	texel[0][2] index LSB
63	texel[3][1] index LSB
62	texel[2][1] index LSB
61	texel[1][1] index LSB
60	texel[0][1] index LSB
59	texel[3][0] index LSB
58	texel[2][0] index LSB
57	texel[1][0] index LSB
56	texel[0][0] index LSB

The "T" mode has two base colors stored as 4 bits per channel, R0/G0/B0 and R1/G1/B1, as in the individual mode, however the bit positions for these are different. For each channel, the 4 bits are extended to 8 bits by bit replication.

A 3-bit distance index "di" is also defined in the compression block. This value is used to look up the distance in the following table:

distance index "di"	distance "d"
0	3
1	6
2	11
3	16
4	23
5	32
6	41

distance index "di"	distance "d"
7	64

Four colors are possible on each texel. These colors are defined as the following:

```
P0 = (R0, G0, B0)
P1 = (R1, G1, B1) + (d, d, d)
P2 = (R1, G1, B1)
P3 = (R1, G1, B1) - (d, d, d)
```

All resulting channels are clamped to the range [0,255]. One of the four colors is then assigned to each texel in the block based on the 2-bit texel index.

H mode

The "H" mode has the following bit definition:

8-byte compression block for "H" mode

Bits	Description
7	ignored
6:3	R0[3:0]
2:0	G0[3:1]
15:13	ignored
12	G0[0]
11	B0[3]
10	ignored
9:8	B0[2:1]
23	B0[0]
22:19	R1[3:0]
18:16	G1[3:1]
31	G1[0]
30:27	B1[3:0]
26	di[2]
25	diff = 1
24	di[1]
39	texel[3][3] index MSB
38	texel[2][3] index MSB
37	texel[1][3] index MSB
36	texel[0][3] index MSB
35	texel[3][2] index MSB
34	texel[2][2] index MSB

Bits	Description
33	texel[1][2] index MSB
32	texel[0][2] index MSB
47	texel[3][1] index MSB
46	texel[2][1] index MSB
45	texel[1][1] index MSB
44	texel[0][1] index MSB
43	texel[3][0] index MSB
42	texel[2][0] index MSB
41	texel[1][0] index MSB
40	texel[0][0] index MSB
55	texel[3][3] index LSB
54	texel[2][3] index LSB
53	texel[1][3] index LSB
52	texel[0][3] index LSB
51	texel[3][2] index LSB
50	texel[2][2] index LSB
49	texel[1][2] index LSB
48	texel[0][2] index LSB
63	texel[3][1] index LSB
62	texel[2][1] index LSB
61	texel[1][1] index LSB
60	texel[0][1] index LSB
59	texel[3][0] index LSB
58	texel[2][0] index LSB
57	texel[1][0] index LSB
56	texel[0][0] index LSB

The "H" mode has two base colors stored as 4 bits per channel, R0/G0/B0 and R1/G1/B1, as in the individual and T modes, however the bit positions for these are different. For each channel, the 4 bits are extended to 8 bits by bit replication.

A 3-bit distance index "di" is defined by 2 MSBs in the compression block and the LSB computed by the following equation, where R/G/B values are the 8-bit values from the first step:

```
di[0] = ((R0 « 16) | (G0 « 8) | B0) >= ((R1 « 16) | (G1 « 8) | B1)
```

The distance "d" is then looked up in the same table used for T mode. The four colors for H mode are computed as follows:

```
P0 = (R0, G0, B0) + (d, d, d)
P1 = (R0, G0, B0) - (d, d, d)
P2 = (R1, G1, B1) + (d, d, d)
```

```
P3 = (R1, G1, B1) - (d, d, d)
```

All resulting channels are clamped to the range [0,255]. One of the four colors is then assigned to each texel in the block based on the 2-bit texel index as in T mode.

Planar mode

The "planar" mode has the following bit definition:

8-byte compression block for "planar" mode

Bits	Description
7	ignored
6:1	R0[5:0]
0	G0[6]
15	ignored
14:9	G0[5:0]
8	B[5]
23:21	ignored
20:19	B[4:3]
18	ignored
17:16	B0[2:1]
31	B0[0]
30:26	RH[5:1]
25	diff = 1
24	RH[0]
39:33	GH[6:0]
32	BH[5]
47:43	BH[4:0]
42:40	RV[5:3]
55:53	RV[2:0]
52:48	GV[6:2]
63:62	GV[1:0]
61:56	BV[5:0]

The "planar" mode has three base colors stored as RGB 676, with red & blue having 6 bits and green having 7 bits. These three base colors are each extended to RGB 888 with bit replication.

The color of each texel is then computed using the following equations, with x and y representing the texel position within the compression block:

```
texel[y][x].R = x(RH-R0)/4 + y(RV-R0)/4 + R0
texel[y][x].G = x(GH-G0)/4 + y(GV-G0)/4 + G0
texel[y][x].B = x(BH-B0)/4 + y(BV-B0)/4 + B0
```

All resulting channels are clamped to the range [0,255].

The ETC2_SRGB8 format is decompressed as if it is ETC2_RGB8, then a conversion from the resulting RGB values to SRGB space is performed.

EAC_R11 and EAC_SIGNED_R11

These formats compress UNORM/SNORM single-channel data using an 8-byte compression block representing a 4x4 block of texels. The texels are labeled as texel[row][column] where both row and column range from 0 to 3. Texel[0][0] is the upper left texel.

The 8-byte compression block is laid out as follows.

EAC_R11 compression block layout

Bits	Description
7:0	R0[7:0]
15:12	m[3:0]
11:8	ti[3:0]
23:21	texel[0][0] index
20:18	texel[1][0] index
17:16,31	texel[2][0] index
30:28	texel[3][0] index
27:25	texel[0][1] index
24,39:38	texel[1][1] index
37:35	texel[2][1] index
34:32	texel[3][1] index
47:45	texel[0][2] index
44:42	texel[1][2] index
41:40,55	texel[2][2] index
54:52	texel[3][2] index
51:49	texel[0][3] index
48,63:62	texel[1][3] index
61:59	texel[2][3] index
58:56	texel[3][3] index

The "ti" (table index) value from the compression block is used to select one of the columns in the table below.

Intensity modifier (im) table

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	-3	-3	-2	-2	-3	-3	-4	-3	-2	-2	-2	-2	-3	-1	-4	-3
1	-6	-7	-5	-4	-6	-7	-7	-5	-6	-5	-4	-5	-4	-2	-6	-5
2	-9	-10	-8	-6	-8	-9	-8	-8	-8	-8	-8	-7	-7	-3	-8	-7
3	-15	-13	-13	-13	-12	-11	-11	-11	-10	-10	-10	-10	-10	-10	-9	-9
4	2	2	1	1	2	2	3	2	1	1	1	1	2	0	3	2
5	5	6	4	3	5	6	6	4	5	4	3	4	3	1	5	4
6	8	9	7	5	7	8	7	7	7	7	7	6	6	2	7	6
7	14	12	12	12	11	10	10	10	9	9	9	9	9	9	8	8

The eight possible color values R_i are then computed from the 8 values in the column labeled im_i, where i ranges from 0 to 7:

For EAC_R11:

$$\texttt{if (m == 0)}\ R_i = R0*8 + 4 + im_i\ \texttt{else}\ R_i = R0*8 + 4 + (im_i * m * 8)$$

Each value is clamped to the range [0,2047].

For EAC_SIGNED_R11:

$$\texttt{if (m == 0)}\ R_i = R0*8 + im_i\ \texttt{else}\ R_i = R0*8 + (im_i * m * 8)$$

Each value is clamped to the range [-1023,1023].

Note that in the signed case, the R0 value is a signed, 2's complement value in the range [-127, 127]. Before being used in the above equations, an R0 value of -128 must be clamped to -127.

Finally, each texel red value is selected from the 8 possible values R_i using the 3-bit index for that texel. The green, blue, and alpha values are set to their default values.

The final value represents an 11-bit UNORM or SNORM as an unsigned/signed integer.

ETC2_RGB8_PTA and ETC2_SRGB8_PTA

The ETC2_RGB8_PTA format is similar to ETC2_RGB8 but eliminates the "individual" mode in favor of allowing a punch-through alpha. The "diff" bit from ETC2_RGB8 is renamed to "opaque" in this format, and the mode selection behaves as if the "diff" bit is always 1, making the "individual" mode inaccessible for these formats.

An alpha value of either 0 or 255 (representing 0.0 or 1.0) is possible with this format. If alpha is determined to be zero, the three other channels are also forced to zero, regardless of what value the normal decompression algorithm would have produced.

Differential Mode

In differential mode, if the opaque bit is set, the luminance table for ETC2_RGB8 is used. If the opaque bit is not set, the following luminance table is used (note that rows 0 and 2 have been zeroed out, otherwise the table is the same):

Luminance Table for opaque bit not set

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	8	17	29	42	60	80	106	183
2	0	0	0	0	0	0	0	0
3	-8	-17	-29	-42	-60	-80	-106	-183

For each texel, if the opaque bit is zero and the corresponding texel index is equal to 2, the alpha value is set to zero (and therefore RGB for that texel will also end up at zero). Otherwise, alpha is set to 255 and RGB is the result of the normal decompression calculations.

T and H Modes

In both of these modes, if the opaque bit is zero and the texel index is equal to 2, the alpha value is set to zero (and therefore RGB will also end up at zero). Otherwise, alpha is set to 255.

Planar Mode

In planar mode, the opaque bit is ignored, and alpha is set to 255.

The ETC2_SRGB8_PTA format is decompressed as if it is ETC2_RGB8_PTA, then a conversion from the resulting RGB values to SRGB space is performed, with alpha remaining unchanged.

ETC2_EAC_RGBA8 and ETC2_EAC_SRGB8_A8

The ETC2_EAC_RGBA8 format is a combination of ETC2_RGB8 and EAC_R8. A 16-byte compression block represents each 4x4. The low-order 8 bytes are used to compute alpha (instead of red) using the EAC_R8 algorithm. The high-order 8 bytes are used to compute RGB using the ETC2_RGB8 algorithm. The EAC_R8 format differs from EAC_R11 as described below.

The ETC2_EAC_SRGB8_A8 format is decompressed as if it is ETC2_EAC_RGBA8, then a conversion from the resulting RGB values to SRGB space is performed, with alpha remaining unchanged.

EAC_R8 Format:

The EAC_R8 format used within these surface formats is identical to EAC_R11 described in an earlier section, except the procedure for computing the eight possible color values Ri is performed as follows:

Ri = R0 + (imi * m)

Each value is clamped to the range [0,255].

EAC_RG11 and EAC_SIGNED_RG11

These formats compress UNORM/SNORM double-channel data using a 16-byte compression block representing a 4x4 block of texels. The texels are labeled as texel[row][column] where both row and column range from 0 to 3. Texel[0][0] is the upper left texel.

The 16-byte compression block is laid out as follows.

EAC_RG11 compression block layout

Bits	Description
63:56	G0[7:0]
55:52	Gm[3:0]
51:48	Gti[3:0]
47:45	texel[0][0] G index
44:42	texel[1][0] G index
41:39	texel[2][0] G index
38:36	texel[3][0] G index
35:33	texel[0][1] G index
32:30	texel[1][1] G index
29:27	texel[2][1] G index
26:24	texel[3][1] G index
23:21	texel[0][2] G index
20:18	texel[1][2] G index
17:15	texel[2][2] G index
14:12	texel[3][2] G index
11:9	texel[0][3] G index
8:6	texel[1][3] G index
5:3	texel[2][3] G index
66:64	texel[3][3] G index
63:56	R0[7:0]
55:52	Rm[3:0]
51:48	Rti[3:0]
47:45	texel[0][0] R index
44:42	texel[1][0] R index

Bits	Description
41:39	texel[2][0] R index
38:36	texel[3][0] R index
35:33	texel[0][1] R index
32:30	texel[1][1] R index
29:27	texel[2][1] R index
26:24	texel[3][1] R index
23:21	texel[0][2] R index
20:18	texel[1][2] R index
17:15	texel[2][2] R index
14:12	texel[3][2] R index
11:9	texel[0][3] R index
8:6	texel[1][3] R index
5:3	texel[2][3] R index
2:0	texel[3][3] R index

These compression formats are identical to the EAC_R11 and EAC_SIGNED_R11 formats, except that they supply two channels of output data, both red and green, from two independent 8-byte portions of the compression block. The low half of the compression block contains the red information, and the high half contains the green information. Blue and alpha channels are set to their default values.

Refer to the EAC_R11 and EAC_SIGNED_R11 specification for details on how the red and green channels are generated using the data in the compression block.

DXT/BC1-3 Texture Formats

Note that non-power-of-2 dimensioned maps may require the surface to be padded out to the next multiple of four texels - here the pad texels are not referenced by the device.

An 8-byte (QWord) block encoding can be used if the source texture contains no transparency (is opaque) or if the transparency can be specified by a one-bit alpha. A 16-byte (DQWord) block encoding can be used to support source textures that require more than one-bit alpha: here the 1st QWord is used to encode the texel alpha values, and the 2nd QWord is used to encode the texel color values.

These three types of format are discussed in the following sections:

- Opaque and One-bit Alpha Textures (DXT1)
- Opaque Textures (DXT1_RGB)
- Textures with Alpha Channels (DXT2-5)

DXT2 and DXT3 are equivalent compression formats from the perspective of the hardware. The only difference between the two is the use of pre-multiplied alpha encoding, which does not affect hardware.

Likewise, DXT4 and DXT5 are the same compression formats with the only difference being the use of pre-multiplied alpha encoding.

Note that the surface formats DXT1-5 are referred to in the DirectX Specification as BC1-3. The mapping between formats is shown below:

- DXT1 => BC1
- DXT2/DXT3 => BC2
- DXT4/DXT5 => BC3

Programming Note	
Context:	DXT Texture Formats
• Any single texture must specify that its data is stored as 64 or 128 bits per group of 16 texels. If 64-bit blocks--that is, format DXT1--are used for the texture, it is possible to mix the opaque and one-bit alpha formats on a per-block basis within the same texture. In other words, the comparison of the unsigned integer magnitude of color_0 and color_1 is performed uniquely for each block of 16 texels. • When 128-bit blocks are used, then the alpha channel must be specified in either explicit (format DXT2 or DXT3) or interpolated mode (format DXT4 or DXT5) for the entire texture. Note that as with color, once interpolated mode is selected then either 8 interpolated alphas or 6 interpolated alphas mode can be used on a block-by-block basis. Again, the magnitude comparison of alpha_0 and alpha_1 is done uniquely on a block-by-block basis.	

Opaque and One-bit Alpha Textures (DXT1/BC1)

Texture format DXT1 is for textures that are opaque or have a single transparent color. For each opaque or one-bit alpha block, two 16-bit R5G6B5 values and a 4x4 bitmap with 2-bits-per-pixel are stored. This totals 64 bits (1 QWord) for 16 texels, or 4-bits-per-texel.

In the block bitmap, there are two bits per texel to select between the four colors, two of which are stored in the encoded data. The other two colors are derived from these stored colors by linear interpolation.

The one-bit alpha format is distinguished from the opaque format by comparing the two 16-bit color values stored in the block. They are treated as unsigned integers. If the first color is greater than the second, it implies that only opaque texels are defined. This means four colors will be used to represent the texels. In four-color encoding, there are two derived colors and all four colors are equally distributed in RGB color space. This format is analogous to R5G6B5 format. Otherwise, for one-bit alpha transparency, three colors are used and the fourth is reserved to represent transparent texels. Note that the color blocks in DXT2-5 formats strictly use four colors, as the alpha values are obtained from the alpha block

In three-color encoding, there is one derived color and the fourth two-bit code is reserved to indicate a transparent texel (alpha information). This format is analogous to A1R5G5B5, where the final bit is used for encoding the alpha mask.

The following piece of pseudo-code illustrates the algorithm for deciding whether three- or four-color encoding is selected:

```
if (color_0 > color_1)
{
  // Four-color block: derive the other two colors.
  // 00 = color_0, 01 = color_1, 10 = color_2, 11 = color_3
  // These two bit codes correspond to the 2-bit fields
  // stored in the 64-bit block.
   color_2 = (2 * color_0 + color_1) / 3;
   color_3 = (color 0 + 2 * color_1) / 3;
}
else
{
  // Three-color block: derive the other color.
  // 00 = color_0, 01 = color_1, 10 = color_2,
  // 11 = transparent.
  // These two bit codes correspond to the 2-bit fields
  // stored in the 64-bit block.
   color_2 = (color_0 + color_1) / 2;
   color_3 = transparent;
}
```

The following tables show the memory layout for the 8-byte block. It is assumed that the first index corresponds to the y-coordinate and the second corresponds to the x-coordinate. For example, Texel[1][2] refers to the texture map pixel at (x,y) = (2,1).

Here is the memory layout for the 8-byte (64-bit) block:

Word Address	16-bit Word
0	Color_0
1	Color_1
2	Bitmap Word_0
3	Bitmap Word_1

Color_0 and Color_1 (colors at the two extremes) are laid out as follows:

Bits	Color
15:11	Red color component
10:5	Green color component
4:0	Blue color component

Bits	Texel
1:0 (LSB)	Texel[0][0]
3:2	Texel[0][1]
5:4	Texel[0][2]
7:6	Texel[0][3]
9:8	Texel[1][0]
11:10	Texel[1][1]
13:12	Texel[1][2]

Bits	Texel
15:14	Texel[1][3]

Bitmap Word_1 is laid out as follows:

Bits	Texel
1:0 (LSB)	Texel[2][0]
3:2	Texel[2][1]
5:4	Texel[2][2]
7:6	Texel[2][3]
9:8	Texel[3][0]
11:10	Texel[3][1]
13:12	Texel[3][2]
15:14 (MSB)	Texel[3][3]

Example of Opaque Color Encoding

As an example of opaque encoding, we will assume that the colors red and black are at the extremes. We will call red color_0 and black color_1. There will be four interpolated colors that form the uniformly distributed gradient between them. To determine the values for the 4x4 bitmap, the following calculations are used:

```
00 ? color_0
01 ? color_1
10 ? 2/3 color_0 + 1/3 color_1
11 ? 1/3 color_0 + 2/3 color_1
```

Example of One-bit Alpha Encoding

This format is selected when the unsigned 16-bit integer, color_0, is less than the unsigned 16-bit integer, color_1. An example of where this format could be used is leaves on a tree to be shown against a blue sky. Some texels could be marked as transparent while three shades of green are still available for the leaves. Two of these colors fix the extremes, and the third color is an interpolated color.

The bitmap encoding for the colors and the transparency is determined using the following calculations:

```
00 ? color_0
01 ? color_1
10 ? 1/2 color_0 + 1/2 color_1
11 ? Transparent
```

Opaque Textures (DXT1_RGB)

Texture format DXT1_RGB is identical to DXT1, with the exception that the One-bit Alpha encoding is removed. Color 0 and Color 1 are not compared, and the resulting texel color is derived strictly from the Opaque Color Encoding. The alpha channel defaults to 1.0.

Programming Note	
Context:	Opaque Textures (DXT1_RGB)
The behavior of this format is not compliant with the OGL spec.	

Compressed Textures with Alpha Channels (DXT2-5 / BC2-3)

There are two ways to encode texture maps that exhibit more complex transparency. In each case, a block that describes the transparency precedes the 64-bit block already described for DXT1. The transparency is either represented as a 4x4 bitmap with four bits per pixel (explicit encoding), or with fewer bits and linear interpolation analogous to what is used for color encoding.

The transparency block and the color block are laid out as follows:

Word Address	64-bit Block
3:0	Transparency block
7:4	Previously described 64-bit block

Explicit Texture Encoding

For explicit texture encoding (DXT2 and DXT3 formats), the alpha components of the texels that describe transparency are encoded in a 4x4 bitmap with 4 bits per texel. These 4 bits can be achieved through a variety of means such as dithering or by simply using the 4 most significant bits of the alpha data. However they are produced, they are used just as they are, without any form of interpolation.

Note: DirectDraw's compression method uses the 4 most significant bits.

The following tables illustrate how the alpha information is laid out in memory, for each 16-bit word.

This is the layout for Word 0:

Bits	Alpha
3:0 (LSB)	[0][0]
7:4	[0][1]
11:8	[0][2]
15:12 (MSB)	[0][3]

This is the layout for Word 1:

Bits	Alpha
3:0 (LSB)	[1][0]
7:4	[1][1]
11:8	[1][2]
15:12 (MSB)	[1][3]

This is the layout for Word 2:

Bits	Alpha
3:0 (LSB)	[2][0]
7:4	[2][1]
11:8	[2][2]
15:12 (MSB)	[2][3]

This is the layout for Word 3:

Bits	Alpha
3:0 (LSB)	[3][0]
7:4	[3][1]
11:8	[3][2]
15:12 (MSB)	[3][3]

Three-Bit Linear Alpha Interpolation

The encoding of transparency for the DXT4 and DXT5 formats is based on a concept similar to the linear encoding used for color. Two 8-bit alpha values and a 4x4 bitmap with three bits per pixel are stored in the first eight bytes of the block. The representative alpha values are used to interpolate intermediate alpha values. Additional information is available in the way the two alpha values are stored. If alpha_0 is greater than alpha_1, then six intermediate alpha values are created by the interpolation. Otherwise, four intermediate alpha values are interpolated between the specified alpha extremes. The two additional implicit alpha values are 0 (fully transparent) and 255 (fully opaque).

The following pseudo-code illustrates this algorithm:

```
// 8-alpha or 6-alpha block?
if (alpha_0 > alpha_1) {
   // 8-alpha block: derive the other 6 alphas.
   // 000 = alpha_0, 001 = alpha_1, others are interpolated
  alpha_2 = (6 * alpha_0 + alpha_1) / 7;     // Bit code 010
  alpha_3 = (5 * alpha_0 + 2 * alpha_1) / 7; // Bit code 011
  alpha_4 = (4 * alpha_0 + 3 * alpha_1) / 7; // Bit code 100
  alpha_5 = (3 * alpha_0 + 4 * alpha_1) / 7; // Bit code 101
  alpha_6 = (2 * alpha_0 + 5 * alpha_1) / 7; // Bit code 110
  alpha_7 = (alpha_0 + 6 * alpha_1) / 7;     // Bit code 111
 }
else {
   // 6-alpha block: derive the other alphas.
   // 000 = alpha_0, 001 = alpha_1, others are interpolated
  alpha_2 = (4 * alpha_0 + alpha_1) / 5;     // Bit code 010
  alpha_3 = (3 * alpha_0 + 2 * alpha_1) / 5; // Bit code 011
```

```
    alpha_4 = (2 * alpha_0 + 3 * alpha_1) / 5; // Bit code 100
    alpha_5 = (alpha_0 + 4 * alpha_1) / 5;     // Bit code 101
    alpha_6 = 0;                               // Bit code 110
    alpha_7 = 255;                             // Bit code 111
}
```

The memory layout of the alpha block is as follows:

Byte	Alpha
0	Alpha_0
1	Alpha_1
2	[0][2] (2 LSBs), [0][1], [0][0]
3	[1][1] (1 LSB), [1][0], [0][3], [0][2] (1 MSB)
4	[1][3], [1][2], [1][1] (2 MSBs)
5	[2][2] (2 LSBs), [2][1], [2][0]
6	[3][1] (1 LSB), [3][0], [2][3], [2][2] (1 MSB)
7	[3][3], [3][2], [3][1] (2 MSBs)

BC4

BC4

These formats (BC4_UNORM and BC4_SNORM) compresses single-component UNORM or SNORM data. An 8-byte compression block represents a 4x4 block of texels. The texels are labeled as texel[row][column] where both row and column range from 0 to 3. Texel[0][0] is the upper left texel.

The 8-byte compression block is laid out as follows:

Bit	Description
7:0	red_0
15:8	red_1
18:16	texel[0][0] bit code
21:19	texel[0][1] bit code
24:22	texel[0][2] bit code
27:25	texel[0][3] bit code
30:28	texel[1][0] bit code
33:31	texel[1][1] bit code
36:34	texel[1][2] bit code
39:37	texel[1][3] bit code
42:40	texel[2][0] bit code
45:43	texel[2][1] bit code
48:46	texel[2][2] bit code
51:49	texel[2][3] bit code
54:52	texel[3][0] bit code

Bit	Description
57:55	texel[3][1] bit code
60:58	texel[3][2] bit code
63:61	texel[3][3] bit code

There are two interpolation modes, chosen based on which reference color is larger. The first mode has the two reference colors plus six equal-spaced interpolated colors between the reference colors, chosen based on the three-bit code for that texel. The second mode has the two reference colors plus four interpolated colors, chosen by six of the three-bit codes. The remaining two codes select min and max values for the colors. The values of red_0 through red_7 are computed as follows:

```
red_0 = red_0;                              // bit code 000
red_1 = red_1;                              // bit code 001
if (red_0 > red_1) {
    red_2 = (6 * red_0 + 1 * red_1) / 7; // bit code 010
    red_3 = (5 * red_0 + 2 * red_1) / 7; // bit code 011
    red_4 = (4 * red_0 + 3 * red_1) / 7; // bit code 100
    red_5 = (3 * red_0 + 4 * red_1) / 7; // bit code 101
    red_6 = (2 * red_0 + 5 * red_1) / 7; // bit code 110
    red_7 = (1 * red_0 + 6 * red_1) / 7; // bit code 111
}
else {
    red_2 = (4 * red_0 + 1 * red_1) / 5; // bit code 010
    red_3 = (3 * red_0 + 2 * red_1) / 5; // bit code 011
    red_4 = (2 * red_0 + 3 * red_1) / 5; // bit code 100
    red_5 = (1 * red_0 + 4 * red_1) / 5; // bit code 101
    red_6 = UNORM ? 0.0 : -1.0;             // bit code 110 (0 for UNORM, -1 for SNORM)
    red_7 = 1.0;                            // bit code 111
}
```

BC5

BC5

These formats (BC5_UNORM and BC5_SNORM) compresses dual-component UNORM or SNORM data. A 16-byte compression block represents a 4x4 block of texels. The texels are labeled as texel[row][column] where both row and column range from 0 to 3. Texel[0][0] is the upper left texel.

The 16-byte compression block is laid out as follows:

Bit	Description
7:0	red_0
15:8	red_1
18:16	texel[0][0] red bit code
21:19	texel[0][1] red bit code
24:22	texel[0][2] red bit code
27:25	texel[0][3] red bit code
30:28	texel[1][0] red bit code
33:31	texel[1][1] red bit code
36:34	texel[1][2] red bit code

Bit	Description
39:37	texel[1][3] red bit code
42:40	texel[2][0] red bit code
45:43	texel[2][1] red bit code
48:46	texel[2][2] red bit code
51:49	texel[2][3] red bit code
54:52	texel[3][0] red bit code
57:55	texel[3][1] red bit code
60:58	texel[3][2] red bit code
63:61	texel[3][3] red bit code
71:64	green_0
79:72	green_1
82:80	texel[0][0] green bit code
85:83	texel[0][1] green bit code
88:86	texel[0][2] green bit code
91:89	texel[0][3] green bit code
94:92	texel[1][0] green bit code
97:95	texel[1][1] green bit code
100:98	texel[1][2] green bit code
103:101	texel[1][3] green bit code
106:104	texel[2][0] green bit code
109:107	texel[2][1] green bit code
112:110	texel[2][2] green bit code
115:113	texel[2][3] green bit code
118:116	texel[3][0] green bit code
121:119	texel[3][1] green bit code
124:122	texel[3][2] green bit code
127:125	texel[3][3] green bit code

There are two interpolation modes, chosen based on which reference color is larger. The first mode has the two reference colors plus six equal-spaced interpolated colors between the reference colors, chosen based on the three-bit code for that texel. The second mode has the two reference colors plus four interpolated colors, chosen by six of the three-bit codes. The remaining two codes select min and max values for the colors. The values of red_0 through red_7 are computed as follows:

```
red_0 = red_0;                          // bit code 000
red_1 = red_1;                          // bit code 001
if (red_0 > red_1) {
    red_2 = (6 * red_0 + 1 * red_1) / 7; // bit code 010
    red_3 = (5 * red_0 + 2 * red_1) / 7; // bit code 011
    red_4 = (4 * red_0 + 3 * red_1) / 7; // bit code 100
    red_5 = (3 * red_0 + 4 * red_1) / 7; // bit code 101
    red_6 = (2 * red_0 + 5 * red_1) / 7; // bit code 110
    red_7 = (1 * red_0 + 6 * red_1) / 7; // bit code 111
}
else {
    red_2 = (4 * red_0 + 1 * red_1) / 5; // bit code 010
    red_3 = (3 * red_0 + 2 * red_1) / 5; // bit code 011
    red_4 = (2 * red_0 + 3 * red_1) / 5; // bit code 100
    red_5 = (1 * red_0 + 4 * red_1) / 5; // bit code 101
    red_6 = UNORM ? 0.0 : -1.0;          // bit code 110 (0 for UNORM, -1 for SNORM)
    red_7 = 1.0;                         // bit code 111
}
```

The same calculations are done for green, using the corresponding reference colors and bit codes.

BC6H

These formats (BC6H_UF16 and BC6H_SF16) compresses 3-channel images with high dynamic range (> 8 bits per channel). BC6H supports floating point denorms but there is no support for INF and NaN, other than with BC6H_SF16 -INF is supported. The alpha channel is not included, thus alpha is returned at its default value.

The BC6H block is 16 bytes and represents a 4x4 block of texels. The texels are labeled as texel[row][column] where both row and column range from 0 to 3. Texel[0][0] is the upper left texel. BC6H has 14 different modes, the mode that the block is in is contained in the least significant bits (either 2 or 5 bits).

The basic scheme consists of interpolating colors along either one or two lines, with per-texel indices indicating which color along the line is chosen for each texel. If a two-line mode is selected, one of 32 partition sets is indicated which selects which of the two lines each texel is assigned to.

Field Definition

There are 14 possible modes for a BC6H block, the format of each is indicated in the 14 tables below. The mode is selected by the unique mode bits specified in each table. The first 10 modes use two lines ("TWO"), and the last 4 use one line ("ONE"). The difference between the various two-line and one-line modes is with the precision of the first endpoint and the number of bits used to store delta values for the remaining endpoints. Two modes (9 and 10) specify each endpoint as an original value rather than using the deltas (these are indicated as having no delta values).

The endpoints values and deltas are indicated in the tables using a two-letter name. The first letter is "r", "g", or "b" indicating the color channel. The second letter is "w", "x", "y", or "z" indicating which of the four endpoints. The first line has endpoints "w" and "x", with "w" being the endpoint that is fully specified (i.e. not as a delta). The second line has endpoints "y" and "z". Modes using ONE mode do not have endpoints "y" and "z" as they have only one line.

In addition to the mode and endpoint data, TWO blocks contain a 5-bit "partition" which selects one of the partition sets, and a 46-bit set of indices. ONE blocks contain a 63-bit set of indices. These are described in more detail below.

Mode 0: (TWO) Red, Green, Blue: 10-bit endpoint, 5-bit deltas

Bit	Description
1:0	mode = 00
2	gy[4]
3	by[4]
4	bz[4]
14:5	rw[9:0]
24:15	gw[9:0]
34:25	bw[9:0]
39:35	rx[4:0]
40	gz[4]
44:41	gy[3:0]
49:45	gx[4:0]
50	bz[0]
54:51	gz[3:0]
59:55	bx[4:0]
60	bz[1]
64:61	by[3:0]
69:65	ry[4:0]
70	bz[2]
75:71	rz[4:0]
76	bz[3]
81:77	partition
127:82	indices

Mode 1: (TWO) Red, Green, Blue: 7-bit endpoint, 6-bit deltas

Bit	Description
1:0	mode = 01
2	gy[5]
3	gz[4]
4	gz[5]
11:5	rw[6:0]
12	bz[0]
13	bz[1]
14	by[4]

Bit	Description
21:15	gw[6:0]
22	by[5]
23	bz[2]
24	gy[4]
31:25	bw[6:0]
32	bz[3]
33	bz[5]
34	bz[4]
40:35	rx[5:0]
44:41	gy[3:0]
50:45	gx[5:0]
54:51	gz[3:0]
60:55	bx[5:0]
64:61	by[3:0]
70:65	ry[5:0]
76:71	rz[5:0]
81:77	partition
127:82	indices

Mode 2: (TWO) Red: 11-bit endpoint, 5-bit deltas

Green, Blue: 11-bit endpoint, 4-bit deltas

Bit	Description
4:0	mode = 00010
14:5	rw[9:0]
24:15	gw[9:0]
34:25	bw[9:0]
39:35	rx[4:0]
40	rw[10]
44:41	gy[3:0]
48:45	gx[3:0]
49	gw[10]
50	bz[0]
54:51	gz[3:0]
58:55	bx[3:0]
59	bw[10]
60	bz[1]
64:61	by[3:0]

Bit	Description
69:65	ry[4:0]
70	bz[2]
75:71	rz[4:0]
76	bz[3]
81:77	partition
127:82	indices

Mode 3: (TWO) Red, Blue: 11-bit endpoint, 4-bit deltas

Green: 11-bit endpoint, 5-bit deltas

Bit	Description
4:0	mode = 00110
14:5	rw[9:0]
24:15	gw[9:0]
34:25	bw[9:0]
38:35	rx[3:0]
39	rw[10]
40	gz[4]
44:41	gy[3:0]
49:45	gx[4:0]
50	gw[10]
54:51	gz[3:0]
58:55	bx[3:0]
59	bw[10]
60	bz[1]
64:61	by[3:0]
68:65	ry[3:0]
69	bz[0]
70	bz[2]
74:71	rz[3:0]
75	gy[4]
76	bz[3]
81:77	partition
127:82	indices

Mode 4: (TWO) Red, Green: 11-bit endpoint, 4-bit deltas

Blue: 11-bit endpoint, 5-bit deltas

Bit	Description
4:0	mode = 01010
14:5	rw[9:0]
24:15	gw[9:0]
34:25	bw[9:0]
38:35	rx[3:0]
39	rw[10]
40	by[4]
44:41	gy[3:0]
48:45	gx[3:0]
49	gw[10]
50	bz[0]
54:51	gz[3:0]
59:55	bx[4:0]
60	bw[10]
64:61	by[3:0]
68:65	ry[3:0]
69	bz[1]
70	bz[2]
74:71	rz[3:0]
75	bz[4]
76	bz[3]
81:77	partition
127:82	indices

Mode 5: (TWO) Red, Green, Blue: 9-bit endpoint, 5-bit deltas

Bit	Description
4:0	mode = 01110
13:5	rw[8:0]
14	by[4]
23:15	gw[8:0]
24	gy[4]
33:25	bw[8:0]
34	bz[4]
39:35	rx[4:0]
40	gz[4]

Bit	Description
44:41	gy[3:0]
49:45	gx[4:0]
50	bz[0]
54:51	gz[3:0]
59:55	bx[4:0]
60	bz[1]
64:61	by[3:0]
69:65	ry[4:0]
70	bz[2]
75:71	rz[4:0]
76	bz[3]
81:77	partition
127:82	indices

Mode 6: (TWO) Red: 8-bit endpoint, 6-bit deltas

Green, Blue: 8-bit endpoint, 5-bit deltas

Bit	Description
4:0	mode = 10010
12:5	rw[7:0]
13	gz[4]
14	by[4]
22:15	gw[7:0]
23	bz[2]
24	gy[4]
32:25	bw[7:0]
33	bz[3]
34	bz[4]
40:35	rx[5:0]
44:41	gy[3:0]
49:45	gx[4:0]
50	bz[0]
54:51	gz[3:0]
59:55	bx[4:0]
60	gz[1]
64:61	by[3:0]
70:65	ry[5:0]
76:71	rz[5:0]

Bit	Description
81:77	partition
127:82	indices

Mode 7: (TWO) Red, Blue: 8-bit endpoint, 5-bit deltas

Green: 8-bit endpoint, 6-bit deltas

Bit	Description
4:0	mode = 10110
12:5	rw[7:0]
13	bz[0]
14	by[4]
22:15	gw[7:0]
23	gy[5]
24	gy[4]
32:25	bw[7:0]
33	gz[5]
34	bz[4]
39:35	rx[4:0]
40	gz[4]
44:41	gy[3:0]
50:45	gx[5:0]
54:51	gz[3:0]
59:55	bx[4:0]
60	bz[1]
64:61	by[3:0]
69:65	ry[4:0]
70	bz[2]
75:71	rz[4:0]
76	bz[3]
81:77	partition
127:82	indices

Mode 8: (TWO) Red, Green: 8-bit endpoint, 5-bit deltas

Blue: 8-bit endpoint, 6-bit deltas

Bit	Description
4:0	mode = 11010
12:5	rw[7:0]
13	bz[1]

Bit	Description
14	by[4]
22:15	gw[7:0]
23	by[5]
24	gy[4]
32:25	bw[7:0]
33	bz[5]
34	bz[4]
39:35	rx[4:0]
40	gz[4]
44:41	gy[3:0]
49:45	gx[4:0]
50	bz[0]
54:51	gz[3:0]
60:55	bx[5:0]
64:61	by[3:0]
69:65	ry[4:0]
70	bz[2]
75:71	rz[4:0]
76	bz[3]
81:77	partition
127:82	indices

Mode 9: (TWO) Red, Green, Blue: 6-bit endpoints for all four, no deltas

Bit	Description
4:0	mode = 11110
10:5	rw[5:0]
11	gz[4]
12	bz[0]
13	bz[1]
14	by[4]
20:15	gw[5:0]
21	gy[5]
22	by[5]
23	bz[2]
24	gy[4]
30:25	bw[5:0]
31	gz[5]

Bit	Description
32	bz[3]
33	bz[5]
34	bz[4]
40:35	rx[5:0]
44:41	gy[3:0]
50:45	gx[5:0]
54:51	gz[3:0]
60:55	bx[5:0]
64:61	by[3:0]
70:65	ry[5:0]
76:71	rz[5:0]
81:77	partition
127:82	indices

Mode 10: (ONE) Red, Green, Blue: 10-bit endpoints for both, no deltas

Bit	Description
4:0	mode = 00011
14:5	rw[9:0]
24:15	gw[9:0]
34:25	bw[9:0]
44:35	rx[9:0]
54:45	gx[9:0]
64:55	bx[9:0]
127:65	indices

Mode 11: (ONE) Red, Green, Blue: 11-bit endpoints, 9-bit deltas

Bit	Description
4:0	mode = 00111
14:5	rw[9:0]
24:15	gw[9:0]
34:25	bw[9:0]
43:35	rx[8:0]
44	rw[10]
53:45	gx[8:0]
54	gw[10]
63:55	bx[8:0]
64	bw[10]
127:65	indices

Mode 12: (ONE) Red, Green, Blue: 12-bit endpoints, 8-bit deltas

Bit	Description
4:0	mode = 01011
14:5	rw[9:0]
24:15	gw[9:0]
34:25	bw[9:0]
42:35	rx[7:0]
43	rw[11]
44	rw[10]
52:45	gx[7:0]
53	gw[11]
54	gw[10]
62:55	bx[7:0]
63	bw[11]
64	bw[10]
127:65	indices

Mode 13: (ONE) Red, Green, Blue: 16-bit endpoints, 4-bit deltas

Bit	Description
4:0	mode = 01111
14:5	rw[9:0]
24:15	gw[9:0]
34:25	bw[9:0]
38:35	rx[3:0]
39	rw[15]
40	rw[14]
41	rw[13]
42	rw[12]
43	rw[11]
44	rw[10]
48:45	gx[3:0]
49	gw[15]
50	gw[14]
51	gw[13]
52	gw[12]
53	gw[11]
54	gw[10]
58:55	bx[3:0]

Bit	Description
59	bw[15]
60	bw[14]
61	bw[13]
62	bw[12]
63	bw[11]
64	bw[10]
127:65	indices

Undefined mode values (10011, 10111, 11011, and 11111) return zero in the RGB channels.

The "indices" fields are defined as follows:

TWO mode *indices* **field with fix-up index [1] at texel[3][3]**

Bit	Description
83:82	texel[0][0] index
86:84	texel[0][1] index
89:87	texel[0][2] index
92:90	texel[0][3] index
95:93	texel[1][0] index
98:96	texel[1][1] index
101:99	texel[1][2] index
104:102	texel[1][3] index
107:105	texel[2][0] index
110:108	texel[2][1] index
113:111	texel[2][2] index
116:114	texel[2][3] index
119:117	texel[3][0] index
122:120	texel[3][1] index
125:123	texel[3][2] index
127:126	texel[3][3] index

TWO mode *indices* **field with fix-up index [1] at texel[0][2]**

Bit	Description
83:82	texel[0][0] index
86:84	texel[0][1] index
88:87	texel[0][2] index
91:89	texel[0][3] index
94:92	texel[1][0] index
97:95	texel[1][1] index

Bit	Description
100:98	texel[1][2] index
103:101	texel[1][3] index
106:104	texel[2][0] index
109:107	texel[2][1] index
112:110	texel[2][2] index
115:113	texel[2][3] index
118:116	texel[3][0] index
121:119	texel[3][1] index
124:122	texel[3][2] index
127:125	texel[3][3] index

TWO mode *indices* field with fix-up index [1] at texel[2][0]

Bit	Description
83:82	texel[0][0] index
86:84	texel[0][1] index
89:87	texel[0][2] index
92:90	texel[0][3] index
95:93	texel[1][0] index
98:96	texel[1][1] index
101:99	texel[1][2] index
104:102	texel[1][3] index
106:105	texel[2][0] index
109:107	texel[2][1] index
112:110	texel[2][2] index
115:113	texel[2][3] index
118:116	texel[3][0] index
121:119	texel[3][1] index
124:122	texel[3][2] index
127:125	texel[3][3] index

ONE mode *indices* field

Bit	Description
67:65	texel[0][0] index
71:68	texel[0][1] index
75:72	texel[0][2] index
79:76	texel[0][3] index
83:80	texel[1][0] index
87:84	texel[1][1] index

Bit	Description
91:88	texel[1][2] index
95:92	texel[1][3] index
99:96	texel[2][0] index
103:100	texel[2][1] index
107:104	texel[2][2] index
111:108	texel[2][3] index
115:112	texel[3][0] index
119:116	texel[3][1] index
123:120	texel[3][2] index
127:124	texel[3][3] index

Endpoint Computation

The endpoints can be defined in many different ways, as shown above. This section describes how the endpoints are computed from the bits in the compression block. The method used depends on whether the BC6H format is signed (BC6H_SF16) or unsigned (BC6H_UF16).

First, each channel (RGB) of each endpoint is extended to 16 bits. Each is handled identically and independently, however in some modes different channels have different incoming precision which must be accounted for. The following rules are employed:

- If the format is BC6H_SF16 or the endpoint is a delta value, the value is sign-extended to 16 bits
- For all other cases, the value is zero-extended to 16 bits

If there are no endpoints that are delta values, endpoint computation is complete. For endpoints that are delta values, the next step involves computing the absolute endpoint. The "w" endpoint is always absolute and acts as a base value for the other three endpoints. Each channel is handled identically and independently.

```
x = w + x
y = w + y
z = w + z
```

The above is performed using 16-bit integer arithmetic. Overflows beyond 16 bits are ignored (any resulting high bits are dropped).

Palette Color Computation

The next step involves computing the color palette values that provide the available values for each texel's color. The color palette for each line consists of the two endpoint colors plus 6 (TWO mode) or 14 (ONE mode) interpolated colors. Again, each channel is processed independently.

First the endpoints are unquantized, with each channel of each endpoint being processed independently. The number of bits in the original base *w* value represents the precision of the endpoints. The input

endpoint is called *e*, and the resulting endpoints are represented as 17-bit signed integers and called e' below.

For the BC6H_UF16 format:

- if the precision is already 16 bits, e' = e
- if e = 0, e' = 0
- if e is the maximum representable in the precision, e' = 0xFFFF
- otherwise, e' = ((e « 16) + 0x8000) » precision

For the BC6H_SF16 format, the value is treated as sign magnitude. The sign is not changed, e' and e refer only to the magnitude portion:

- if the precision is already 16 bits, e' = e
- if e = 0, e' = 0
- if e is the maximum representable in the precision, e' = 0x7FFF
- otherwise, e' = ((e « 15) + 0x4000) » (precision - 1)

Next, the palette values are generated using predefined weights, using the tables below:

```
palette[i] = (w' * (64 - weight[i]) + x' * weight[i] + 32) » 6
```

TWO mode weights:

palette index	0	1	2	3	4	5	6	7
weight	0	9	18	27	37	46	55	64

ONE mode weights:

palette index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
weight	0	4	9	13	17	21	26	30	34	38	43	47	51	55	60	64

The two end palette indices are equal to the two endpoints given that the weights are 0 and 64. In the above equation w' and x' represent the endpoints e' computed in the previous step corresponding to w and x, respectively. For the second line in TWO mode, w and x are replaced with y and z.

The final step in computing the palette colors is to rescale the final results. For BC6H_UF16 format, the values are multiplied by 31/64. For BC6H_SF16, the values are multiplied by 31/32, treating them as sign magnitude. These final 16-bit results are ultimately treated as 16-bit floats.

Texel Selection

The final step is to select the appropriate palette index for each texel. This index then selects the 16-bit per channel palette value, which is re-interpreted as a 16-bit floating point result for input into the filter. This procedure differs depending on whether the mode is TWO or ONE.

ONE Mode

In ONE mode, there is only one set of palette colors, but the "indices" field is 63 bits. This field consists of a 4-bit palette index for each of the 16 texels, with the exception of the texel at [0][0] which has only 3 bits, the missing high bit being set to zero.

TWO Mode

32 partitions are defined for TWO, which are defined below. Each of the 32 cases shows the 4x4 block of texels, and is indexed by adding its hexadecimal row number (00-1C) to its column number (0-3). Each texel in the 4x4 block indicates whether that texel is part of line 0 (endpoints *w* and *x*) or line 1 (endpoints *y* and *z*). Each case has one texel each of "[0]" and "[1]", the index that this is at is termed the "fix-up index". These texels have one less bit in the index.

	0				1				2				3			
00	[0]	0	1	1	[0]	0	0	1	[0]	1	1	1	[0]	0	0	1
	0	0	1	1	0	0	0	1	0	1	1	1	0	0	1	1
	0	0	1	1	0	0	0	1	0	1	1	1	0	0	1	1
	0	0	1	[1]	0	0	0	[1]	0	1	1	[1]	0	1	1	[1]
04	[0]	0	0	0	[0]	0	1	1	[0]	0	0	1	[0]	0	0	0
	0	0	0	1	0	1	1	1	0	0	1	1	0	0	0	1
	0	0	0	1	0	1	1	1	0	1	1	1	0	0	1	1
	0	0	1	[1]	1	1	1	[1]	1	1	1	[1]	0	1	1	[1]
08	[0]	0	0	0	[0]	0	1	1	[0]	0	0	0	[0]	0	0	0
	0	0	0	0	0	1	1	1	0	0	0	1	0	0	0	0
	0	0	0	1	1	1	1	1	0	1	1	1	0	0	0	1
	0	0	1	[1]	1	1	1	[1]	1	1	1	[1]	0	1	1	[1]
0C	[0]	0	0	1	[0]	0	0	0	[0]	0	0	0	[0]	0	0	0
	0	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
	1	1	1	[1]	1	1	1	[1]	1	1	1	[1]	1	1	1	[1]
10	[0]	0	0	0	[0]	1	[1]	1	[0]	0	0	0	[0]	1	[1]	1
	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1
	1	1	1	0	0	0	0	0	[1]	0	0	0	0	0	0	1
	1	1	1	[1]	0	0	0	0	1	1	1	0	0	0	0	0
14	[0]	0	[1]	1	[0]	0	0	0	[0]	0	0	0	[0]	1	1	1
	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1
	0	0	0	0	[1]	1	0	0	[1]	0	0	0	0	0	1	1
	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	[1]
18	[0]	0	[1]	1	[0]	0	0	0	[0]	1	[1]	0	[0]	0	[1]	1

	0	0	0	1	1	0	0	0	0	1	1	0	0	1	1	0
	0	0	0	1	[1]	0	0	0	0	1	1	0	0	1	1	0
	0	0	0	0	1	1	0	0	0	1	1	0	1	1	0	0
1C	[0]	0	0	1	[0]	0	0	0	[0]	1	[1]	1	[0]	0	[1]	1
	0	1	1	1	1	1	1	1	0	0	0	1	1	0	0	1
	[1]	1	1	0	[1]	1	1	1	1	0	0	0	1	0	0	1
	1	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0

The 46-bit "indices" field consists of a 3-bit palette index for each of the 16 texels, with the exception of the bracketed texels that have only two bits each. The high bit of these texels is set to zero.

BC7

These formats (BC7_UNORM and BC7_UNORM_SRGB) compresses 3-channel and 4-channel fixed point images.

The BC7 block is 16 bytes and represents a 4x4 block of texels. The texels are labeled as texel[row][column] where both row and column range from 0 to 3. Texel[0][0] is the upper left texel. BC7 has 8 different modes, the mode that the block is in is contained in the least significant bits (1-8 bits depending on mode).

The basic scheme consists of interpolating colors and alpha in some modes along either one, two, or three lines, with per-texel indices indicating which color/alpha along the line is chosen for each texel. If a two- or three-line mode is selected, one of 64 partition sets is indicated which selects which of the two lines each texel is assigned to, although some modes are limited to the first 16 partition sets. In the color-only modes, alpha is always returned at its default value of 1.0.

Some modes contain the following fields:

- **P-bits.** These represent shared LSB for all components of the endpoint, which increases the endpoint precision by one bit. In some cases both endpoints of a line share a P-bit.
- **Rotation bits.** For blocks with separate color and alpha, this 2-bit field allows selection of which of the four components has its own indexes (scalar) vs. the other three components (vector).
- **Index selector.** This 1-bit field selects whether the scalar or vector components uses the 3-bit index vs. the 2-bit index.

Field Definition

There are 8 possible modes for a BC7 block, the format of each is indicated in the 8 tables below. The mode is selected by the unique mode bits specified in each table. Each mode has particular characteristics described at the top of the table.

Mode 0: Color only, 3 lines (THREE), 4-bit endpoints with one P-bit per endpoint, 3-bit indices, 16 partitions

Bit	Description
0	mode = 0
4:1	partition
8:5	R0
12:9	R1
16:13	R2
20:17	R3
24:21	R4
28:25	R5
32:29	G0
36:33	G1
40:37	G2
44:41	G3
48:45	G4
52:49	G5
56:53	B0
60:57	B1
64:61	B2
68:65	B3
72:69	B4
76:73	B5
77	P0
78	P1
79	P2
80	P3
81	P4
82	P5
127:83	indices

Mode 1: Color only, 2 lines (TWO), 6-bit endpoints with one shared P-bit per line, 3-bit indices, 64 partitions

Bit	Description
1:0	mode = 10
7:2	partition
13:8	R0
19:14	R1

Bit	Description
25:20	R2
31:26	R3
37:32	G0
43:38	G1
49:44	G2
55:50	G3
61:56	B0
67:62	B1
73:68	B2
79:74	B3
80	P0
81	P1
127:82	indices

Mode 2: Color only, 3 lines (THREE), 5-bit endpoints, 2-bit indices, 64 partitions

Bit	Description
2:0	mode = 100
8:3	partition
13:9	R0
18:14	R1
23:19	R2
28:24	R3
33:29	R4
38:34	R5
43:39	G0
48:44	G1
53:49	G2
58:54	G3
63:59	G4
68:64	G5
73:69	B0
78:74	B1
83:79	B2
88:84	B3
93:89	B4
98:94	B5
127:99	indices

Mode 3: Color only, 2 lines (TWO), 7-bit endpoints with one P-bit per endpoint, 2-bit indices, 64 partitions

Bit	Description
3:0	mode = 1000
9:4	partition
16:10	R0
23:17	R1
30:24	R2
37:31	R3
44:38	G0
51:45	G1
58:52	G2
65:59	G3
72:66	B0
79:73	B1
86:80	B2
93:87	B3
94	P0
95	P1
96	P2
97	P3
127:98	indices

Mode 4: Color and alpha, 1 line (ONE), 5-bit color endpoints, 6-bit alpha endpoints, 16 2-bit indices, 16 3-bit indices, 2-bit component rotation, 1-bit index selector

Bit	Description
4:0	mode = 10000
6:5	rotation
7	index selector
12:8	R0
17:13	R1
22:18	G0
27:23	G1
32:28	B0
37:33	B1
43:38	A0
49:44	A1
80:50	2-bit indices

Bit	Description
127:81	3-bit indices

Mode 5: Color and alpha, 1 line (ONE), 7-bit color endpoints, 8-bit alpha endpoints, 2-bit color indices, 2-bit alpha indices, 2-bit component rotation

Bit	Description
5:0	mode = 100000
7:6	rotation
14:8	R0
21:15	R1
28:22	G0
35:29	G1
42:36	B0
49:43	B1
57:50	A0
65:58	A1
96:66	color indices
127:97	alpha indices

Mode 6: Combined color and alpha, 1 line (ONE), 7-bit endpoints with one P-bit per endpoint, 4-bit indices

Bit	Description
6:0	mode = 1000000
13:7	R0
20:14	R1
27:21	G0
34:28	G1
41:35	B0
48:42	B1
55:49	A0
62:56	A1
63	P0
64	P1
127:65	indices

Mode 7: Combined color and alpha, 2 lines (TWO), 5-bit endpoints with one P-bit per endpoint, 2-bit indices, 64 partitions

Bit	Description
7:0	mode = 10000000
13:8	partition
18:14	R0
23:19	R1
28:24	R2
33:29	R3
38:34	G0
43:39	G1
48:44	G2
53:49	G3
58:54	B0
63:59	B1
68:64	B2
73:69	B3
78:74	A0
83:79	A1
88:84	A2
93:89	A3
94	P0
95	P1
96	P2
97	P3
127:98	indices

Undefined mode values (bits 7:0 = 00000000) return zero in the RGB channels.

The indices fields are variable in length and due to the different locations of the fix-up indices depending on partition set there are a very large number of possible configurations. Each mode above indicates how many bits each index has, and the fix-up indices (one in ONE mode, two in TWO mode, and three in THREE mode) each have one less bit than indicated. However, the indices are always packed into the index fields according to the table below, with the specific bit assignments of each texel following the rules just given.

Bit	Description
LSBs	texel[0][0] index
	texel[0][1] index
	texel[0][2] index
	texel[0][3] index

Bit	Description
	texel[1][0] index
	texel[1][1] index
	texel[1][2] index
	texel[1][3] index
	texel[2][0] index
	texel[2][1] index
	texel[2][2] index
	texel[2][3] index
	texel[3][0] index
	texel[3][1] index
	texel[3][2] index
MSBs	texel[3][3] index

Endpoint Computation

The endpoints can be defined with different precision depending on mode, as shown above. This section describes how the endpoints are computed from the bits in the compression block. Each component of each endpoint follows the same steps.

If a P-bit is defined for the endpoint, it is first added as an additional LSB at the bottom of the endpoint value. The endpoint is then bit-replicated to create an 8-bit fixed point endpoint value with a range from 0x00 to 0xFF.

Palette Color Computation

The next step involves computing the color palette values that provide the available values for each texel's color. The color palette for each line consists of the two endpoint colors plus 2, 6, or 14 interpolated colors, depending on the number of bits in the indices. Again each channel is processed independently.

The equation to compute each palette color with index i, given two endpoints is as follows, using the tables below to determine the weight for each palette index:

```
palette[i] = (E0 * (64 - weight[i]) + E1 * weight[i] + 32) » 6
```

2-bit index weights:

palette index	0	1	2	3
weight	0	21	43	64

3-bit index weights:

palette index	0	1	2	3	4	5	6	7
weight	0	9	18	27	37	46	55	64

4-bit index weights:

palette index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
weight	0	4	9	13	17	21	26	30	34	38	43	47	51	55	60	64

The two end palette indices are equal to the two endpoints given that the weights are 0 and 64. In the above equation E0 and E1 represent the even-numbered and odd-numbered endpoints computed in the previous step for the component and line currently being computed.

Texel Selection

The final step is to select the appropriate palette index for each texel. This index then selects the 8-bit per channel palette value, which is interpreted as an 8-bit UNORM value for input into the filter (In BC7_UNORM_SRGB to UNORM values first go through inverse gamma conversion). This procedure differs depending on whether the mode is ONE, TWO, or THREE.

ONE Mode

In ONE mode, there is only one set of palette colors, thus there is only a single "partition set" defined, with all texels selecting line 0 and texel [0][0] being the "fix-up index" with one less bit in the index.

TWO Mode

64 partitions are defined for TWO, which are defined below. Each of the 64 cases shows the 4x4 block of texels, and is indexed by adding its hexadecimal row number (00-3C) to its column number (0-3). Each texel in the 4x4 block indicates whether that texel is part of line 0 (endpoints *0* and *1*) or line 1 (endpoints *2* and *3*). Each case has one texel each of "[0]" and "[1]", the index that this is at is termed the "fix-up index". These texels have one less bit in the index.

	0				1				2				3			
00	[0]	0	1	1	[0]	0	0	1	[0]	1	1	1	[0]	0	0	1
	0	0	1	1	0	0	0	1	0	1	1	1	0	0	1	1
	0	0	1	1	0	0	0	1	0	1	1	1	0	0	1	1
	0	0	1	[1]	0	0	0	[1]	0	1	1	[1]	0	1	1	[1]
04	[0]	0	0	0	[0]	0	1	1	[0]	0	0	1	[0]	0	0	0
	0	0	0	1	0	1	1	1	0	0	1	1	0	0	0	1
	0	0	0	1	0	1	1	1	0	1	1	1	0	0	1	1
	0	0	1	[1]	1	1	1	[1]	1	1	1	[1]	0	1	1	[1]
08	[0]	0	0	0	[0]	0	1	1	[0]	0	0	0	[0]	0	0	0
	0	0	0	0	0	1	1	1	0	0	0	1	0	0	0	0
	0	0	0	1	1	1	1	1	0	1	1	1	0	0	0	1
	0	0	1	[1]	1	1	1	[1]	1	1	1	[1]	0	1	1	[1]
0C	[0]	0	0	1	[0]	0	0	0	[0]	0	0	0	[0]	0	0	0

	0	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
	1	1	1	[1]	1	1	1	[1]	1	1	1	[1]	1	1	1	[1]
10	[0]	0	0	0	[0]	1	[1]	1	[0]	0	0	0	[0]	1	[1]	1
	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1
	1	1	1	0	0	0	0	0	[1]	0	0	0	0	0	0	1
	1	1	1	[1]	0	0	0	0	1	1	1	0	0	0	0	0
14	[0]	0	[1]	1	[0]	0	0	0	[0]	0	0	0	[0]	1	1	1
	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1
	0	0	0	0	[1]	1	0	0	[1]	0	0	0	0	0	1	1
	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	[1]
18	[0]	0	[1]	1	[0]	0	0	0	[0]	1	[1]	0	[0]	0	[1]	1
	0	0	0	1	1	0	0	0	0	1	1	0	0	1	1	0
	0	0	0	1	[1]	0	0	0	0	1	1	0	0	1	1	0
	0	0	0	0	1	1	0	0	0	1	1	0	1	1	0	0
1C	[0]	0	0	1	[0]	0	0	0	[0]	1	[1]	1	[0]	0	[1]	1
	0	1	1	1	1	1	1	1	0	0	0	1	1	0	0	1
	[1]	1	1	0	[1]	1	1	1	1	0	0	0	1	0	0	1
	1	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0
20	[0]	1	0	1	[0]	0	0	0	[0]	1	0	1	[0]	0	1	1
	0	1	0	1	1	1	1	1	1	0	[1]	0	0	0	1	1
	0	1	0	1	0	0	0	0	0	1	0	1	[1]	1	0	0
	0	1	0	[1]	1	1	1	[1]	1	0	1	0	1	1	0	0
24	[0]	0	[1]	1	[0]	1	0	1	[0]	1	1	0	[0]	1	0	1
	1	1	0	0	0	1	0	1	1	0	0	1	1	0	1	0
	0	0	1	1	[1]	0	1	0	0	1	1	0	1	0	1	0
	1	1	0	0	1	0	1	0	1	0	0	[1]	0	1	0	[1]
28	[0]	1	[1]	1	[0]	0	0	1	[0]	0	[1]	1	[0]	0	[1]	1
	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	1
	1	1	0	0	[1]	1	0	0	0	1	0	0	1	1	0	1
	1	1	1	0	1	0	0	0	1	1	0	0	1	1	0	0
2C	[0]	1	[1]	0	[0]	0	1	1	[0]	1	1	0	[0]	0	0	0
	1	0	0	1	1	1	0	0	0	1	1	0	0	1	[1]	0
	1	0	0	1	1	1	0	0	1	0	0	1	0	1	1	0
	0	1	1	0	0	0	1	[1]	1	0	0	[1]	0	0	0	0
30	[0]	1	0	0	[0]	0	[1]	0	[0]	0	0	0	[0]	0	0	0
	1	1	[1]	0	0	1	1	1	0	0	[1]	0	0	1	0	0
	0	1	0	0	0	0	1	0	0	1	1	1	[1]	1	1	0

	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0
34	[0]	1	1	0	[0]	0	1	1	[0]	1	[1]	0	[0]	0	[1]	1
	1	1	0	0	0	1	1	0	0	0	1	1	1	0	0	1
	1	0	0	1	1	1	0	0	1	0	0	1	1	1	0	0
	0	0	1	[1]	1	0	0	[1]	1	1	0	0	0	1	1	0
38	[0]	1	1	0	[0]	1	1	0	[0]	1	1	1	[0]	0	0	1
	1	1	0	0	0	0	1	1	1	1	1	0	1	0	0	0
	1	1	0	0	0	0	1	1	1	0	0	0	1	1	1	0
	1	0	0	[1]	1	0	0	[1]	0	0	0	[1]	0	1	1	[1]
3C	[0]	0	0	0	[0]	0	[1]	1	[0]	0	[1]	0	[0]	1	0	0
	1	1	1	1	0	0	1	1	0	0	1	0	0	1	0	0
	0	0	1	1	1	1	1	1	1	1	1	0	0	1	1	1
	0	0	1	[1]	0	0	0	0	1	1	1	0	0	1	1	[1]

THREE Mode

64 partitions are defined for THREE, which are defined below. Each of the 64 cases shows the 4x4 block of texels, and is indexed by adding its hexadecimal row number (00-3C) to its column number (0-3). Each texel in the 4x4 block indicates whether that texel is part of line 0 (endpoints *0* and *1*), line 1 (endpoints *2* and *3*), or line 2 (endpoints *4* and *5*). Each case has one texel each of "[0]", "[1]", and "[2]", the index that this is at is termed the "fix-up index". These texels have one less bit in the index.

	0				1				2				3			
00	[0]	0	1	[1]	[0]	0	0	[1]	[0]	0	0	0	[0]	2	2	[2]
	0	0	1	1	0	0	1	1	2	0	0	1	0	0	2	2
	0	2	2	1	[2]	2	1	1	[2]	2	1	1	0	0	1	1
	2	2	2	[2]	2	2	2	1	2	2	1	[1]	0	1	1	[1]
04	[0]	0	0	0	[0]	0	1	[1]	[0]	0	2	[2]	[0]	0	1	1
	0	0	0	0	0	0	1	1	0	0	2	2	0	0	1	1
	[1]	1	2	2	0	0	2	2	1	1	1	1	[2]	2	1	1
	1	1	2	[2]	0	0	2	[2]	1	1	1	[1]	2	2	1	[1]
08	[0]	0	0	0	[0]	0	0	0	[0]	0	0	0	[0]	0	1	2
	0	0	0	0	1	1	1	1	1	1	[1]	1	0	0	[1]	2
	[1]	1	1	1	[1]	1	1	1	2	2	2	2	0	0	1	2
	2	2	2	[2]	2	2	2	[2]	2	2	2	[2]	0	0	1	[2]
0C	[0]	1	1	2	[0]	1	2	2	[0]	0	1	[1]	[0]	0	1	[1]
	0	1	[1]	2	0	[1]	2	2	0	1	1	2	2	0	0	1
	0	1	1	2	0	1	2	2	1	1	2	2	[2]	2	0	0
	0	1	1	[2]	0	1	2	[2]	1	2	2	[2]	2	2	2	0

10	[0]	0	0	[1]	[0]	1	1	[1]	[0]	0	0	0	[0]	0	2	[2]
	0	0	1	1	0	0	1	1	1	1	2	2	0	0	2	2
	0	1	1	2	[2]	0	0	1	[1]	1	2	2	0	0	2	2
	1	1	2	[2]	2	2	0	0	1	1	2	[2]	1	1	1	[1]
14	[0]	1	1	[1]	[0]	0	0	[1]	[0]	0	0	0	[0]	0	0	0
	0	1	1	1	0	0	0	1	0	0	[1]	1	1	1	0	0
	0	2	2	2	[2]	2	2	1	0	1	2	2	[2]	2	[1]	0
	0	2	2	[2]	2	2	2	1	0	1	2	[2]	2	2	1	0
18	[0]	1	2	[2]	[0]	0	1	2	[0]	1	1	0	[0]	0	0	0
	0	[1]	2	2	0	0	1	2	1	2	[2]	1	0	1	[1]	0
	0	0	1	1	[1]	1	2	2	[1]	2	2	1	1	2	[2]	1
	0	0	0	0	2	2	2	[2]	0	1	1	0	1	2	2	1
1C	[0]	0	2	2	[0]	1	1	0	[0]	0	1	1	[0]	0	0	0
	1	1	0	2	0	[1]	1	0	0	1	2	2	2	0	0	0
	[1]	1	0	2	2	0	0	2	0	1	[2]	2	[2]	2	1	1
	0	0	2	[2]	2	2	2	[2]	0	0	1	[1]	2	2	2	[1]
20	[0]	0	0	0	[0]	2	2	[2]	[0]	0	1	[1]	[0]	1	2	0
	0	0	0	2	0	0	2	2	0	0	1	2	0	[1]	2	0
	[1]	1	2	2	0	0	1	2	0	0	2	2	0	1	[2]	0
	1	2	2	[2]	0	0	1	[1]	0	2	2	[2]	0	1	2	0
24	[0]	0	0	0	[0]	1	2	0	[0]	1	2	0	[0]	0	1	1
	1	1	[1]	1	1	2	0	1	2	0	1	2	2	2	0	0
	2	2	[2]	2	[2]	0	[1]	2	[1]	[2]	0	1	1	1	[2]	2
	0	0	0	0	0	1	2	0	0	1	2	0	0	0	1	[1]
28	[0]	0	1	1	[0]	1	0	[1]	[0]	0	0	0	[0]	0	2	2
	1	1	[2]	2	0	1	0	1	0	0	0	0	1	[1]	2	2
	2	2	0	0	2	2	2	2	[2]	1	2	1	0	0	2	2
	0	0	1	[1]	2	2	2	[2]	2	1	2	[1]	1	1	2	[2]
2C	[0]	0	2	[2]	[0]	2	2	0	[0]	1	0	1	[0]	0	0	0
	0	0	1	1	1	2	[2]	1	2	2	[2]	2	2	1	2	1
	0	0	2	2	0	2	2	0	2	2	2	2	[2]	1	2	1
	0	0	1	[1]	1	2	2	[1]	0	1	0	[1]	2	1	2	[1]
30	[0]	1	0	[1]	[0]	2	2	[2]	[0]	0	0	2	[0]	0	0	0
	0	1	0	1	0	1	1	1	1	[1]	1	2	2	[1]	1	2
	0	1	0	1	0	2	2	2	0	0	0	2	2	1	1	2
	2	2	2	[2]	0	1	1	[1]	1	1	1	[2]	2	1	1	[2]
34	[0]	2	2	2	[0]	0	0	2	[0]	1	1	0	[0]	0	0	0
	0	[1]	1	1	1	1	1	2	0	[1]	1	0	0	0	0	0

	0	1	1	1	[1]	1	1	2	0	1	1	0	2	1	[1]	2
	0	2	2	[2]	0	0	0	[2]	2	2	2	[2]	2	1	1	[2]
38	[0]	1	1	0	[0]	0	2	2	[0]	0	2	2	[0]	0	0	0
	0	[1]	1	0	0	0	1	1	1	1	2	2	0	0	0	0
	2	2	2	2	0	0	[1]	1	[1]	1	2	2	0	0	0	0
	2	2	2	[2]	0	0	2	[2]	0	0	2	[2]	2	[1]	1	[2]
3C	[0]	0	0	[2]	[0]	2	2	2	[0]	1	0	[1]	[0]	1	1	[1]
	0	0	0	1	1	2	2	2	2	2	2	2	2	0	1	1
	0	0	0	2	0	2	2	2	2	2	2	2	[2]	2	0	1
	0	0	0	[1]	[1]	2	2	[2]	2	2	2	[2]	2	2	2	0

Video Pixel/Texel Formats

This section describes the "video" pixel/texel formats with respect to memory layout. See the Overlay chapter for a description of how the Y, U, V components are sampled.

Packed Memory Organization

Color components are all 8 bits in size for YUV formats. For YUV 4:2:2 formats each DWord will contain two pixels and only the byte order affects the memory organization.

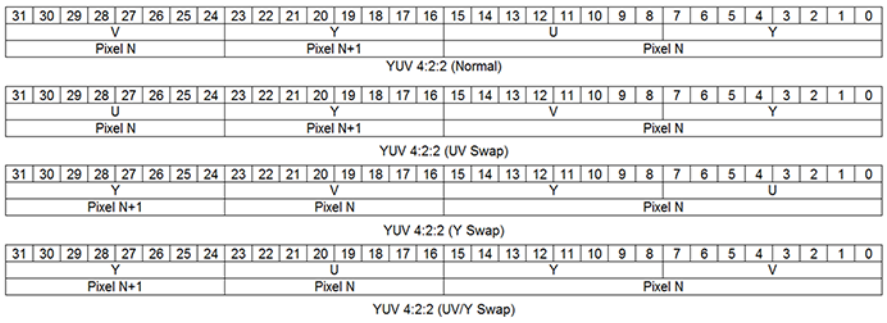
The following four YUV 4:2:2 surface formats are supported, listed with alternate names:

- YCRCB_NORMAL (YUYV/YUY2)
- YCRCB_SWAPUVY (VYUY) (R8G8_B8G8_UNORM)
- YCRCB_SWAPUV(YVYU) (G8R8_G8B8_UNORM)
- YCRCB_SWAPY (UYVY)

The channels are mapped as follows:

Cr (V)	Red
Y	Green
Cb (U)	Blue

Memory layout of packed YUV 4:2:2 formats



Planar Memory Organization

Planar formats use what could be thought of as separate buffers for the three color components. Because there is a separate stride for the Y and U/V data buffers, several memory footprints can be supported.

The 3D sampler supports direct sampling and filtering of planar video surfaces such as YV12 and NV12.

Programming Note	
Context:	Tiling of Planar Surface
Tiling of planar surfaces (tileX, tileY, tileYf, or tileYs) is only supported for planar surfaces where the chroma plane is full-pitch (e.g. NV21). In this case, the field **Y Offset for U or UV Plane** in the RENDER_SURFACE_STATE must be programmed to force the UV plane to be at the start of a tile.	

Programming Note	
Context:	YV12/YV21 Surface Pitch Restriction
The Surface Pitch defined in RENDER_SURFACE_STATE must be a multiple of 64Bytes for YV12 and YV21 surfaces.	

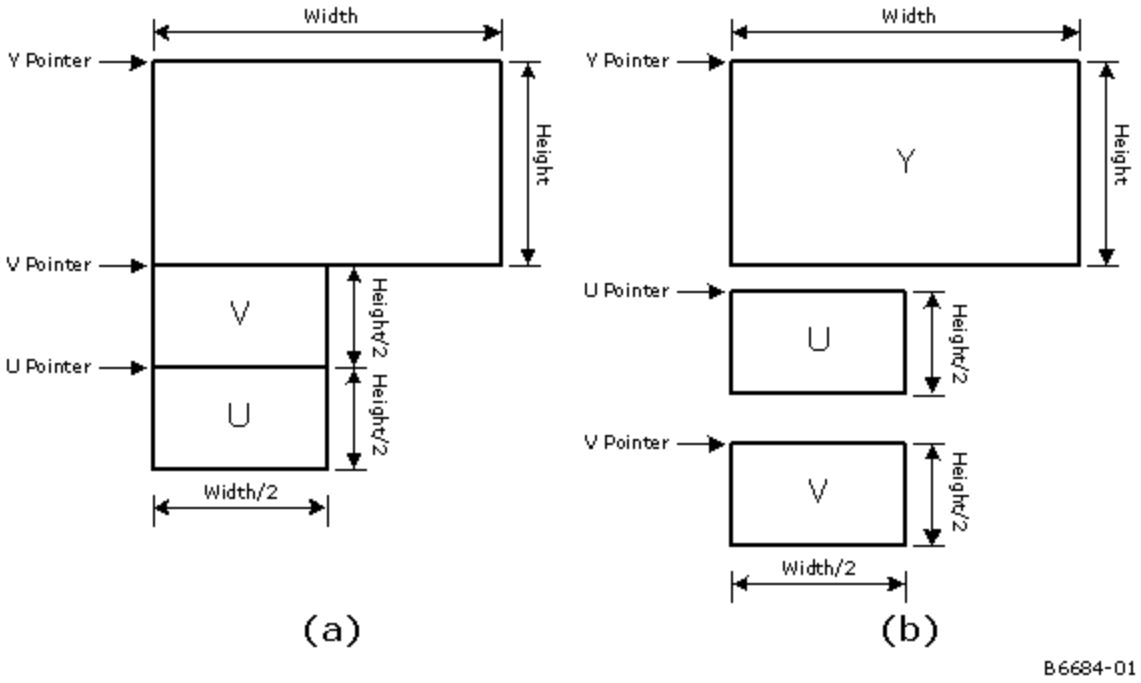
Programming Note	
Context:	NV21 Support
Sampling of NV21 surface format is supported by swapping the U and V channels when sampling the surface. This can be done by programming the **Shader Channel Select** in the RENDER_SURFACE_STATE for the Red and Blue Channels.	

The 3D sampler supports 12, and 16-bit planar video surface formats known collectively as P016. They are only supported for Sample_Unorm.

The following figure shows two types of memory organization for the YUV 4:2:0 planar video data:

1. The memory organization of the common YV12 data, where all three planes are contiguous, and the strides of U and V components are half of that of the Y component.
2. An alternative memory structure that the addresses of the three planes are independent but satisfy certain alignment restrictions.

YUV 4:2:0 Format Memory Organization

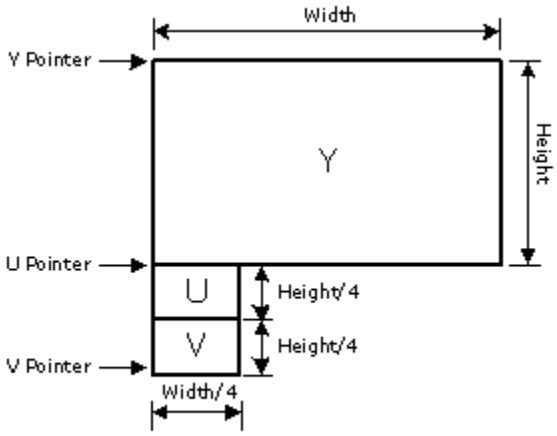


The following figure shows memory organization of the planar YUV 4:1:0 format where the planes are contiguous.

Note: The chroma planes (U and V), when separate (case b above) are treated as half-pitch with respect to the Y plane.

Workaround
When using Planar formats for YUV with half-pitch chroma planes (e.g. YV12), and fenced tiling is not supported
LINEAR filtering of Planar YUV surfaces such as YV12 using the 3D sampler is done after the U and V have been replicated to form a YUV444 texels. This means that the U and V components will effectively be point-sampled rather than filtered. Acheive true filtering of the U and V components, the 3 planes of the YUV surface must be bound as separate surfaces, and the filtering must be done on each individually.

YUV 4:1:0 Format Memory Organization



B6685-01

The table below shows how position within a Planar YUV surface chroma plane is calculated for various cases ot U and V pitch and position. It also shows restrictions on the alignment of the chroma planes in memory for non-interleaved (YV12) and interleaved chroma (e.g. NV12) is used.

Case	Interleave Chroma	Pitch	Vertical U/V Offset
YUV with Half Pitch Chroma	No	Half	When U is below Y Y_Uoffset = Y_Height * 2 Y_Voffset = Y_Height * 2 + V_Height When V is below Y Y_Uoffset = Y_Height * 2 + V_Height Y_Voffset = Y_Height * 2
YUV with Full Pitch Chroma	Yes	Full	When U is below Y Y_Uoffset = Y_Height Y_Voffset = Y_Height + V_Height When V is below Y Y_Uoffset = Y_Height + V_Height Y_Voffset = Y_Height
YUV for Media Sampling	Yes	Always Full	Same as 3D full pitch

Programming Note
Context: Planar YUV surfaces cannot be 1D surface types.
Because there is a requirement that the height of the Y plane of a planar surface must be a greater than 1, it cannot be programmed to be a **Surface Type** of SURFTYPE_1D.

Programming Note	
Context:	MIP Filtering
Surface state cannot have (**MIP Mode Filter** != NONE) for Planar YUV surfaces (e.g. PLANAR_420_8).	

Programming Note	
Context:	Standard Tiling
Planar YUV does not support MIP Tails as part of Standard Tiling. The MIP Tail Start field in RENDER_SURFACE_STATE must be programmed to 15.	

Programming Note	
Context:	Quilted and Planar
Planar YUV is not supported for Quilted surfaces.	

Programming Note	
Context:	
Planar YUV is not supported with Corner Texel Mode	

Additional Video Formats

Additional Video Formats

Dx YUV format name	DxRGB format name	GPGPU Image Type	GPGPU Image Data Type	Packed / Planar	# Surface States	Surface Format #1	Surface Format #2	Surface Format #3	Support By
DXGI_FORMAT_AYUV	DXGI_FORMAT_R8G8B8A8_UNORM (V->R8, U->G8, Y->B8, A->A8)			Packed	1	R8G8B8A8_UNORM	NA	NA	Sampler, PB
DXGI_FORMAT_AYUV	DXGI_FORMAT_R8G8B8A8_UINT (V->R8, U->G8, Y->B8, A->A8)			Packed	1	R8G8B8A8_UINT	NA	NA	Sampler, HDC, PB
DXGI_FORMAT_YUY2	DXGI_FORMAT_R8G8B8A8_UNORM (Y0->R8, U0->G8, Y1->B8, V0->A8)			Packed	1	R8G8B8A8_UNORM	NA	NA	Sampler,
NA	NA	CL_YCbYCr (Y0->R16, U0->G16, Y1->B16, V0->A16	CL_UNORM_INT16	Packed	1	R16G16B16A16_UINT	NA	NA	Sampler, HDC
NA	NA	CL_YCbYCr (Y0->R16, U0->G16, Y1->B16, V0->A16	CL_UNORM_INT12	Packed	1	R16G16B16A16_UINT	NA	NA	Sampler, HDC
NA	NA	CL_YCbYCr (Y0->R16, U0->G16, Y1->B16, V0->A16	CL_UNORM_INT10	Packed	1	R16G16B16A16_UINT	NA	NA	Sampler, HDC

Dx YUV format name	DxRGB format name	GPGPU Image Type	GPGPU Image Data Type	Packed / Planar	# Surface States	Surface Format #1	Surface Format #2	Surface Format #3	Support By
DXGI_FORMAT_YUY2	DXGI_FORMAT_R8G8B8A8_UINT (Y0->R8, U0->G8, Y1->B8, V0->A8)	CL_YCbYCr (Y0->R8, U0->G8, Y1->B8, V0->A8	CL_UNORM_INT8	Packed	1	R8G8B8A8_UINT	NA	NA	Sampler, HDC
DXGI_FORMAT_YUY2	DXGI_FORMAT_R8G8_B8G8_UNORM			Packed	1	R8G8B8A8_UNORM In this case the width of the view will appear to be twice the R8G8B8A8 view, with hardware reconstruction of RGBA done automatically on read (and before filtering).	NA	NA	Sampler
NA	NA	CL_CbYCrY (U0->R16, Y0->G16, V0->B16, Y1->A16	CL_UNORM_INT16	Packed	1	R16G16B16A16_UINT	NA	NA	Sampler, HDC
NA	NA	CL_CbYCrY (U0->R16, Y0->G16, V0->B16, Y1->A16	CL_UNORM_INT12	Packed	1	R16G16B16A16_UINT	NA	NA	Sampler, HDC
NA	NA	CL_CbYCrY (U0->R16, Y0->G16, V0->B16, Y1->A16	CL_UNORM_INT10	Packed	1	R16G16B16A16_UINT	NA	NA	Sampler, HDC

Dx YUV format name	DxRGB format name	GPGPU Image Type	GPGPU Image Data Type	Packed / Planar	# Surface States	Surface Format #1	Surface Format #2	Surface Format #3	Support By
NA	NA	CL_CbYCrY (U0->R8, Y0->G8, V0->B8, Y1->A8	CL_UNORM_INT8	Packed	1	R8G8B8A8_UINT	NA	NA	Sampler, HDC
DXGI_FORMAT_NV12	Y = DXGI_FORMAT_R8_UNORM U/V = DXGI_FORMAT_R8G8_UNORM (U->R8, V->G8)			Planar	2	R8_UNORM	R8G8_UNORM chomra pixel dimensions 1/2 in both x and y from the Luma view	NA	Sampler, PB
NA	NA	CL_Y_Cr_Cb (Y -> R16) (U->R16) (V->R16)	CL_UNORM_INT16	Planar	3	R16_UNIT	R16_UNIT	R16_UNIT	Sampler
NA	NA	CL_Y_Cr_Cb (Y -> R16) (U->R16) (V->R16)	CL_UNORM_INT12	Planar	3	R16_UNIT	R16_UNIT	R16_UNIT	Sampler
NA	NA	CL_Y_Cr_Cb (Y -> R16) (U->R16) (V->R16)	CL_UNORM_INT10	Planar	3	R16_UNIT	R16_UNIT	R16_UNIT	Sampler
NA	NA	CL_Y_Cr_Cb	CL_UNORM_INT8	Planar	3	R8_UNIT	R8_UNIT	R8_UNIT	Sampler

Dx YUV format name	DxRGB format name	GPGPU Image Type	GPGPU Image Data Type	Packed / Planar	# Surface States	Surface Format #1	Surface Format #2	Surface Format #3	Support By
		(Y -> R8) (U->R8) (V->R8)							
NA	NA	CL_Y_CrCb (Y -> R16) (U->R16, V->G16)	CL_UNORM_INT16	Planar	2	R16_UNIT	R16G16_UINT chomra pixel dimensions halved in both x and y from the Luma view	NA	Sampler, HDC
NA	NA	CL_Y_CrCb (Y -> R16) (U->R16, V->G16)	CL_UNORM_INT12	Planar	2	R16_UNIT	R16G16_UINT chomra pixel dimensions halved in both x and y from the Luma view	NA	Sampler, HDC
NA	NA	CL_Y_CrCb (Y -> R16) (U->R16, V->G16)	CL_UNORM_INT10	Planar	2	R16_UNIT	R16G16_UINT chomra pixel dimensions halved in both x and y from the Luma view	NA	Sampler, HDC
DXGI_FORMAT_NV12	Y = DXGI_FORMAT_R8_UINT	CL_Y_CrCb (Y -> R8)	CL_UNORM_INT8	Planar	2	R8_UNIT	R8G8_UINT chomra	NA	Sampler, HDC, PB

Dx YUV format name	DxRGB format name	GPGPU Image Type	GPGPU Image Data Type	Packed / Planar	# Surface States	Surface Format #1	Surface Format #2	Surface Format #3	Support By
	U/V = DXGI_FORMAT_R8G8_UINT (U->R8, V->G8)	(U->R8, V->G8)					pixel dimensions halved in both x and y from the Luma view		
DXGI_FORMAT_NV11	Y = DXGI_FORMAT_R8_UNORM U/V = DXGI_FORMAT_R8G8_UNORM (U->R8, V->G8)			Planar	2	R8_UNORM	R8G8_UNORM chomra pixel dimensions 1/4 in both x and y from the Luma view	NA	Sampler, PB
DXGI_FORMAT_NV11	Y = DXGI_FORMAT_R8_UINT U/V = DXGI_FORMAT_R8G8_UINT (U->R8, V->G8)			Planar	2	R8_UNIT	R8G8_UINT chomra pixel dimensions 1/4 in both x and y from the Luma view	NA	Sampler, HDC, PB
DXGI_FORMAT_P016	Y = DXGI_FORMAT_R16_UNORM U/V = DXGI_FORMAT_R16G16_UNORM (U->R16, V->G16)			Planar	2	R16_UNORM	R16G16_UNORM chomra pixel dimensions 1/2 in both x and y from the Luma view	NA	Sampler, HDC, PB

Dx YUV format name	DxRGB format name	GPGPU Image Type	GPGPU Image Data Type	Packed / Planar	# Surface States	Surface Format #1	Surface Format #2	Surface Format #3	Support By
DXGI_FORMAT_P016	Y = DXGI_FORMAT_R16_UINT U/V = DXGI_FORMAT_R16G16_UINT (U->R16, V->G16)			Planar	2	R16_UNIT	R16G16_UINT chomra pixel dimensions 1/2 in both x and y from the Luma view	NA	Sampler, PB
DXGI_FORMAT_P010	Y = DXGI_FORMAT_R16_UNORM U/V = DXGI_FORMAT_R16G16_UNORM (U->R16, V->G16)			Planar	2	R16_UNORM	R16G16_UNORM chroma pixel dimensions 1/2 in both x and y from the Luma view	NA	Sampler, HDC, PB
DXGI_FORMAT_P010	Y = DXGI_FORMAT_R16_UINT U/V = DXGI_FORMAT_R16G16_UINT (U->R16, V->G16)			Planar	2	R16_UNIT	R16G16_UINT chomra pixel dimensions 1/2 in both x and y from the Luma view	NA	Sampler, HDC, PB
DXGI_FORMAT_Y216	DXGI_FORMAT_R16G16B16A16_UNORM (Y0->R16, U->G16, Y1->B16, V->A16).			Packed	1	R16G16B16A16_UNORM	NA	NA	Sampler,
DXGI_FORMAT_Y216	DXGI_FORMAT_R16G16B16A16_UINT			Packed	1	R16G16B16A16_UINT	NA	NA	Sampler, HDC

Dx YUV format name	DxRGB format name	GPGPU Image Type	GPGPU Image Data Type	Packed / Planar	# Surface States	Surface Format #1	Surface Format #2	Surface Format #3	Support By
	(Y0->R16, U->G16, Y1->B16, V->A16).								
DXGI_FORMAT_Y210	DXGI_FORMAT_R16G16B16A16_UNORM (Y0->R16, U->G16, Y1->B16, V->A16).			Packed	1	R16G16B16A16_UNORM	NA	NA	Sampler,
DXGI_FORMAT_Y210	DXGI_FORMAT_R16G16B16A16_UINT (Y0->R16, U->G16, Y1->B16, V->A16).			Packed	1	R16G16B16A16_UINT	NA	NA	Sampler, HDC
DXGI_FORMAT_Y416	DXGI_FORMAT_R16G16B16A16_UNORM (U->R16, Y->G16, V->B16, A->A16)			Packed	1	R16G16B16A16_UNORM	NA	NA	Sampler, HDC
DXGI_FORMAT_Y416	DXGI_FORMAT_R16G16B16A16_UINT (U->R16, Y->G16, V->B16, A->A16)			Packed	1	R16G16B16A16_UINT	NA	NA	Sampler,
DXGI_FORMAT_Y410	DXGI_FORMAT_R16G16B16A16_UNORM (U->R16, Y->G16, V->B16, A->A16)			Packed	1	R16G16B16A16_UNORM	NA	NA	Sampler, HDC
DXGI_FORMAT_Y410	DXGI_FORMAT_R16G16B16A16_UINT (U->R16, Y->G16, V->B16, A->A16)			Packed	1	R16G16B16A16_UINT	NA	NA	Sampler,
NA	NA	CL_CbCr (U->R16, V->G16)	CL_UNORM_INT16	Packed	1	R16G16_UINT	NA	NA	Sampler, HDC
NA	NA	CL_CbCr (U->R16, V->G16)	CL_UNORM_INT12	Packed	1	R16G16_UINT	NA	NA	Sampler, HDC

Dx YUV format name	DxRGB format name	GPGPU Image Type	GPGPU Image Data Type	Packed / Planar	# Surface States	Surface Format #1	Surface Format #2	Surface Format #3	Support By
NA	NA	CL_CbCr (U->R16, V->G16)	CL_UNORM_INT10	Packed	1	R16G16_UINT	NA	NA	Sampler, HDC
NA	NA	CL_CbCr (U->R8, V->G8)	CL_UNORM_INT8	Packed	1	R8G8_UINT	NA	NA	Sampler, HDC
NA	NA	CL_Cb_Cr (U->R16) (V->R16)	CL_UNORM_INT16	Planar	2	R16_UNIT	R16_UNIT	NA	Sampler, HDC
NA	NA	CL_Cb_Cr (U->R16) (V->R16)	CL_UNORM_INT12	Planar	2	R16_UNIT	R16_UNIT	NA	Sampler, HDC
NA	NA	CL_Cb_Cr (U->R16) (V->R16)	CL_UNORM_INT10	Planar	2	R16_UNIT	R16_UNIT	NA	Sampler, HDC
NA	NA	CL_Cb_Cr (U->R8) (V->R8)	CL_UNORM_INT8	Planar	2	R8_UNIT	R8_UNIT	NA	Sampler, HDC
NA	NA	CL_Y (Y->R16)	CL_UNORM_INT16	Packed	1	R16_UNIT	NA	NA	Sampler, HDC
NA	NA	CL_Y (Y->R16)	CL_UNORM_INT12	Packed	1	R16_UNIT	NA	NA	Sampler, HDC
NA	NA	CL_Y (Y->R16)	CL_UNORM_INT10	Packed	1	R16_UNIT	NA	NA	Sampler, HDC
NA	NA	CL_Y (Y->R8)	CL_UNORM_INT8	Packed	1	R8_UNIT	NA	NA	Sampler, HDC

Dx YUV format name	DxRGB format name	GPGPU Image Type	GPGPU Image Data Type	Packed / Planar	# Surface States	Surface Format #1	Surface Format #2	Surface Format #3	Support By
NA	NA	CL_Cb (V->R16)	CL_UNORM_INT16	Packed	1	R16_UNIT	NA	NA	Sampler, HDC
NA	NA	CL_Cb (V->R16)	CL_UNORM_INT12	Packed	1	R16_UNIT	NA	NA	Sampler, HDC
NA	NA	CL_Cb (V->R16)	CL_UNORM_INT10	Packed	1	R16_UNIT	NA	NA	Sampler, HDC
NA	NA	CL_Cb (V->R8)	CL_UNORM_INT8	Packed	1	R8_UNIT	NA	NA	Sampler, HDC
NA	NA	CL_Cb (U->R16)	CL_UNORM_INT16	Packed	1	R16_UNIT	NA	NA	Sampler, HDC
NA	NA	CL_Cb (U->R16)	CL_UNORM_INT12	Packed	1	R16_UNIT	NA	NA	Sampler, HDC
NA	NA	CL_Cb (U->R16)	CL_UNORM_INT10	Packed	1	R16_UNIT	NA	NA	Sampler, HDC
NA	NA	CL_Cb (U->R8)	CL_UNORM_INT8	Packed	1	R8_UNIT	NA	NA	Sampler, HDC

Raw Format

A format called "RAW" is available that is only supported with the untyped surface read/write, block, scattered, and atomic operation data port messages. It means that the surface has no inherent format. Surfaces of type RAW are addressed with byte-based offsets. The RAW surface format can be applied only to surface types of BUFFER and SCRATCH.

Surface Memory Organizations

See *Memory Interface Functions* chapter for a discussion of tiled vs. linear surface formats.

Display, Overlay, Cursor Surfaces

These surfaces are memory image buffers (planes) used to refresh a display device in non-VGA mode. See the Display chapter for specifics on how these surfaces are defined/used.

2D Render Surfaces

These surfaces are used as general source and/or destination operands in 2D BLT operations.

Note that there is no coherency between 2D render surfaces and the texture cache. Software must explicitly invalidate the texture cache before using a texture that has been modified via the BLT engine.

See the 2D Instruction and 2D Rendering chapters for specifics on how these surfaces are used, restrictions on their size, placement, etc.

2D Monochrome Source

These 1 BPP (bit per pixel) surfaces are used as source operands to certain 2D BLT operations, where the BLT engine expands the 1 BPP source to the required color depth.

The texture cache stores any monochrome sources. There is no mechanism to maintain coherency between 2D render surfaces and texture-cached monochrome sources. Software must explicitly invalidate the texture cache before using a memory-based monochrome source that has been modified via the BLT engine. (Here the assumption is that SW enforces memory-based monochrome source surfaces as read-only surfaces.)

See the 2D Instruction and 2D Rendering chapters for specifics on how these surfaces are used, restrictions on their size, placement, coherency rules, etc.

2D Color Pattern

Color pattern surfaces are used as special pattern operands in 2D BLT operations.

The device uses the texture cache to store color patterns. There is no mechanism to maintain coherency between 2D render surfaces and (texture)-cached color patterns. Software is required to explicitly invalidate the texture cache before using a memory-based color pattern that has been modified via the BLT engine. (Here the assumption is that SW enforces memory-based color pattern surfaces as read-only surfaces.)

See the *2D Instruction* and *2D Rendering* chapters for specifics on how these surfaces are used, restrictions on their size, placement, etc.

3D Color Buffer (Destination) Surfaces

3D Color Buffer surfaces hold per-pixel color values for use in the 3D Pipeline. The 3D Pipeline always requires a Color Buffer to be defined.

See the Non-Video Pixel/Texel Formats section in this chapter for details on the Color Buffer pixel formats. See the 3D Instruction and 3D Rendering chapters for Color Buffer usage details.

The Color Buffer is defined as the BUFFERID_COLOR_BACK memory buffer via the 3DSTATE_BUFFER_INFO instruction. That buffer can be mapped to LM or SM (snooped or unsnooped), and can be linear or tiled. When both the Depth and Color Buffers are tiled, the respective Tile Walk directions must match.

When a linear Color Buffer and a linear Depth Buffer are used together:

- The buffers may have different pitches, though both pitches must be a multiple of 32 bytes.
- The buffers must be co-aligned with a 32-byte region.

3D Depth Buffer Surfaces

Depth Buffer surfaces hold per-pixel depth values and per-pixel stencil values for use in the 3D Pipeline. The 3D Pipeline does not require a Depth Buffer in general, though a Depth Buffer is required to perform non-trivial Depth Test and Stencil Test operations.

The Depth Buffer is specified via the 3DSTATE_DEPTH_BUFFER command. See the description of that instruction in *Windower* for restrictions.

See *Depth Buffer Formats* below for a summary of the possible depth buffer formats. See the Depth Buffer Formats section in this chapter for details on the pixel formats. See the *Windower* and *DataPort* chapters for details on the usage of the Depth Buffer.

Depth Buffer Formats

DepthBufferFormat / DepthComponent	BPP (Bits Per Pixel)	Description
D32_FLOAT_S8X24_UINT	64	32-bit floating point Z depth value in first DWord, 8-bit stencil in lower byte of second DWord
D32_FLOAT	32	32-bit floating point Z depth value
D24_UNORM_S8_UINT	32	24-bit fixed point Z depth value in lower 3 bytes, 8-bit stencil value in upper byte
D16_UNORM	16	16-bit fixed point Z depth value

3D Separate Stencil Buffer Surfaces

Separate Stencil Buffer surfaces hold per-pixel stencil values for use in the 3D Pipeline. Note that the 3D Pipeline does not require a Stencil Buffer to be allocated, though a Stencil Buffer is required to perform non-trivial Stencil Test operations.

Depth Buffer Formats summarizes Stencil Buffer formats. Refer to the Stencil Buffer Formats section in this chapter for details on the pixel formats. Refer to the *Windower* chapters for Stencil Buffer usage details.

The Stencil buffer is specified via the 3DSTATE_STENCIL_BUFFER command. See that instruction description in *Windower* for restrictions.

Depth Buffer Formats

DepthBufferFormat / DepthComponent	BPP (bits per pixel)	Description
R8_ UNIT	8	8-bit stencil value in a byte

Surface Layout and Tiling

This section explains how various surface types (1D, 2D, 3D, and Cube) are laid out in memory. Most of the information in this section is independent of tiling. The concept of tiling can be laid on top of information. Wherever there is a specific difference it will be called out.

For Tiling (TileY, TileYs etc.), see the Address Tiling Function Introduction section which provides detailed information on how tiles are organized and laid out.

Maximum Surface Size in Bytes

In addition to restrictions on maximum height, width, and depth, surfaces are also restricted to a maximum size of 2^44 bytes. All pixels within the surface must be contained within 2^44 bytes of the base address.

NULL Page Support for Media Sampler

NULL support for VA Media sampler will process the returned data for NULL pages as if it was all zeros. Except for feature matching function, in which the NULL page indication will cause the distance function calculation to be ignored and a maximum distance value of 0xFF to be returned.

Tiling

To improve efficiency in memory accesses, most surfaces can be laid out using a tiling scheme.

Supported Legacy Tiling Modes:

- **TileY**
- **TileX**
- **TileW**

Supported Tiled Resource Modes

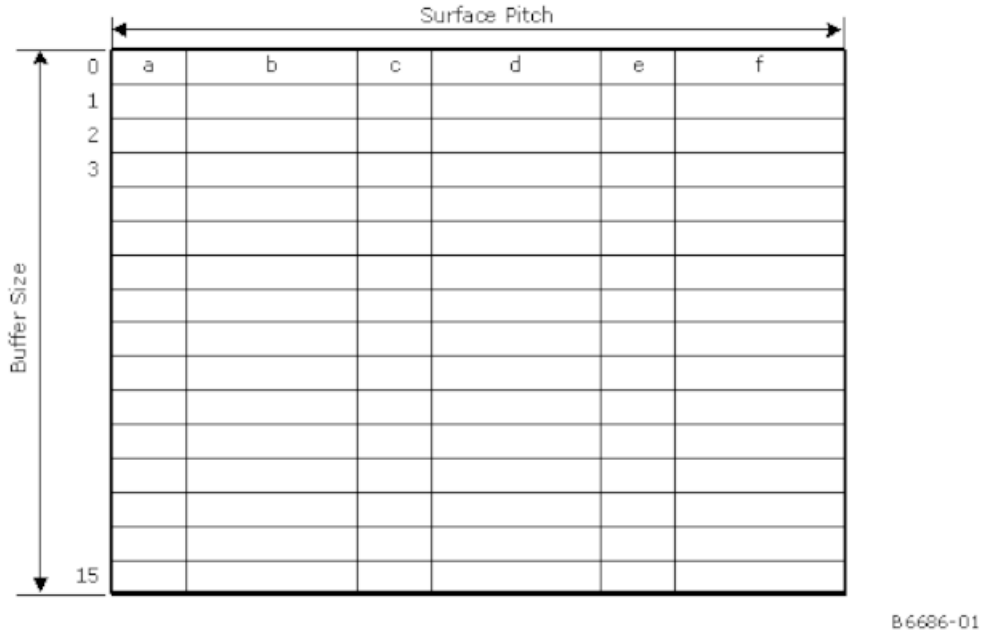
- **TileYF:** 4KB tiling mode based on TileY
- **TileYS:** 64KB tiling mode based on TileY

These modes are described in the Address Tiling Function Introduction volume.

Typed Buffers

A typed buffer is an array of structures. Each structure contains up to 2048 bytes of elements. Each element is a single surface format using one of the supported surface formats depending on how the surface is being accessed. The surface pitch state for the surface specifies the size of each structure in bytes.

The buffer is stored in memory contiguously with each element in the structure packed together, and the first element in the next structure immediately following the last element of the previous structure. Buffers are supported only in linear memory.



Typed buffers are accessed using a surface state for each structure element (a,b,c, etc. in the diagram above). The surface state for element "b" (for example) contains the surface format of element "b" (which may differ from other elements), the base address points to element "b" in the first structure (slice 0 of the array). The pitch for all of the elements in the buffer is the same value, and the surface type of each element is SURFTYPE_BUFFER.

The offset into the typed buffer is given by the following equation:

```
Offset = (V * Pitch) + U
```

MIP Layout

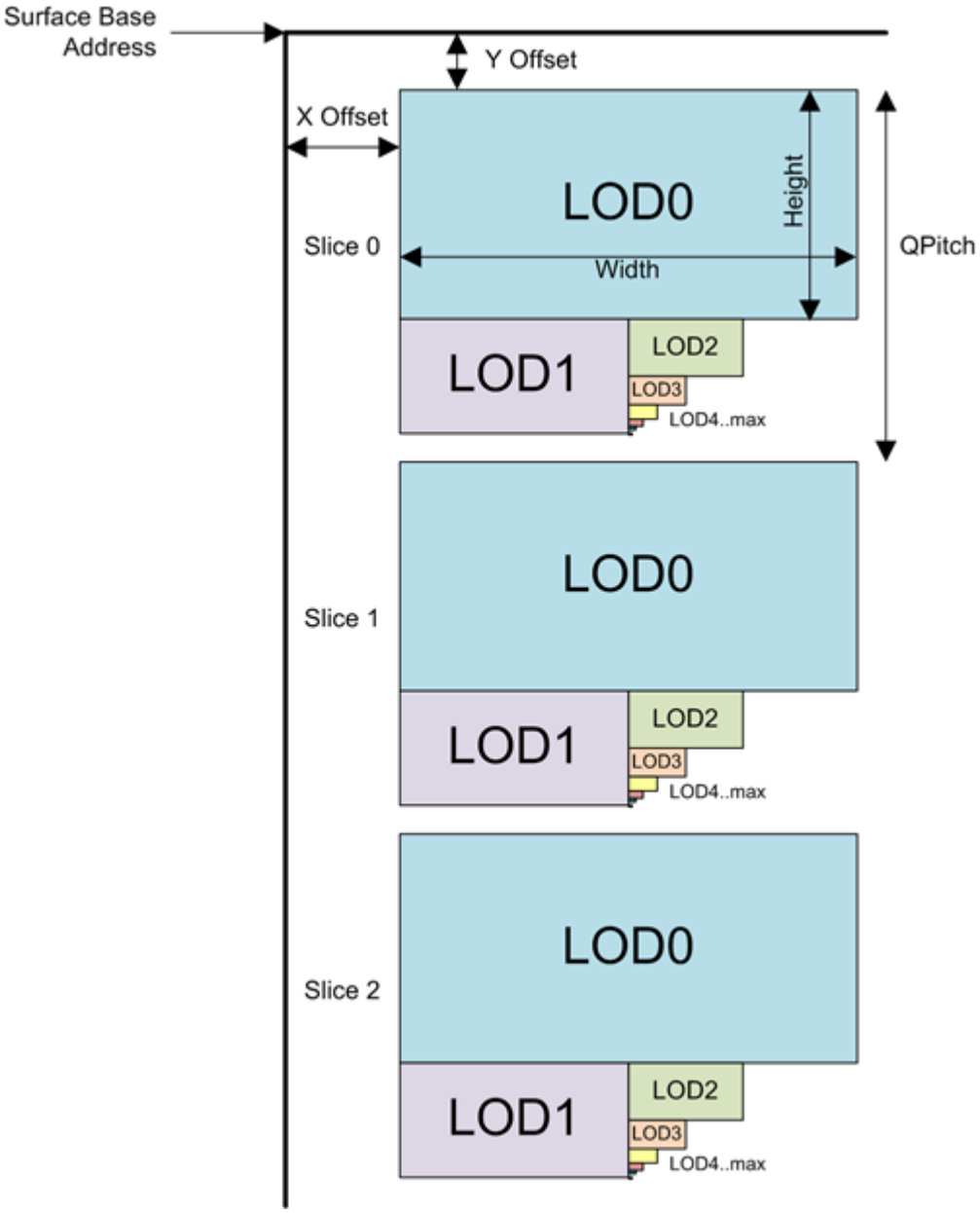
A surface can support multiple levels of details (LODs) or MIPs. The MIPCOUNT field in the RENDER_SURFACE_STATE defines how many MIPs a surface contains.

MIP0 or LOD0 is the largest, highest-detail MIP. The height, width and depth of this LOD is what is defined in the RENDER_SURFACE_STATE for that surface. Each subsequent

MIP is exactly one-half the height and width of the previous, making it 1/4th the size in memory.

The MIPs of a surface a laid out in memory using a 2-dimensional method as shown below. Volumetric and arrayed surfaces use multiple "slices" of this MIP layout, with each slice separated by QPITCH number of rows.

The diagram below shows many of the parameters of a 2D,2D Arrayed and 3D surface.



This 2-dimensional layout implies that there is padding required on the rows below LOD0 in order to ensure each row is the same number of texels.

If Tiling is enabled, then each MIP is laid out using one or more tiles. If TileYf or TileYs tiling is enabled (TR_MODE != NONE), then some of the MIPs may actually be stored in a MIPTail which fits in a single 64K or 4K tile. The layout above, then only applied to MIPs which are not packed in the MIP Tail. Note that, depending on surface height the Vertical Alignment that surface can actually have the last few mips laid out below LOD1. Using MIP Tail (if supported) eliminates this possibility.

Raw (Untyped) Buffers

Raw buffers also use the surface type of SURFTYPE_BUFFER, but the surface format is RAW. These buffers are one-dimensional. They are accessed with a single U parameter which is a byte offset into the buffer. Raw buffers are also supported only in linear memory.

The offset into the raw buffer is given directly by the U parameter.

```
Offset = U
```

Structured Buffers

A structured buffer is a surface type that is accessed by a 2-dimensional coordinate. It can be thought of as an array of structures, where each structure is a predefined number of DWords in size. The first coordinate (U) defines the array index, and the second coordinate (V) is a byte offset into the structure which must be a multiple of 4 (DWord-aligned). A structured buffer must be defined with **Surface Format** RAW.

The structured buffer has only one dimension programmed in SURFACE_STATE which indicates the array size. The byte offset dimension (V) is assumed to be bounded only by the **Surface Pitch**.

The two-dimensional offset into the surface is defined directly by the U and V parameters. Structured buffers are linear.

1D Surfaces

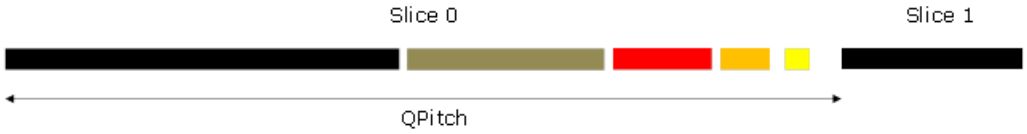
One-dimensional surfaces use a tiling mode of linear. Technically, they are not tiled resources, but the Tiled Resource Mode field in RENDER_SURFACE_STATE is still used to indicate the alignment requirements

for this linear surface (See 1D Alignment requirements for how 4K and 64KB Tiled Resource Modes impact alignment). 1D surfaces are stored linearly in memory.

Programming Note	
Context:	Legacy 1D Tiling
There is a legacy mode for representing a 1D surface as a 2D surface with a height of 1 texel. However, this mode is not recommended due to API compatibility. The **Sampler Legacy 1D Map Layout Disable** MMIO bit (bit 0, E194h) *must* be set to 1h to allow true 1D surfaces.	

Programming Note	
Context:	Legacy 1D Tiling
There is a legacy mode for representing a 1D surface as a 2D surface with a height of 1 texel. However, this mode is not recommended due to API compatibility. The MMIO bit (bit 0, E194h) *must* be set to 1h to allow true 1D surfaces.	

Linear 1D surfaces are stored in a one-dimensional view of memory as follows:



Surface Pitch is ignored for 1D surfaces. **Surface QPitch** specifies the distance in pixels between array slices. QPitch should allow at least enough space for any mips that may be present.

A number of parameters are useful to determine where given pixels will be located on the 1D surface. First, the width for each LOD "L" is computed:

$$W_L = ((width >> L) > 0 ? width >> L : 1$$

When **Corner Texel Mode** is enabled via the RENDER_SURFACE_STATE, the width of a 1D surface is calculated as shown below:

W_L = MAX(1,(W_{L-1}-1)»1)+1

There is a restriction that the smallest map dimension is 2 texels for **Corner Texel Mode** (W_0 > 1)

Next, the aligned width parameter for each LOD "L" is computed. The "i" parameter is the horizontal alignment parameter set by a state field or defined as a constant, depending on the surface. The equation uses the I value that applies to the LOD being computed.

$$w_L = i * ceil\left(\frac{W_L}{i}\right)$$

Next, the offset to each LOD is determined. The offset has one dimension for 1D surfaces. The single element in the LOD_L vector is named LODUL.

$$LOD_0 = (0)$$
$$LOD_1 = (w_0)$$
$$LOD_2 = (w_0 + w_1)$$
$$LOD_3 = (w_0 + w_1 + w_2)$$
$$LOD_4 = (w_0 + w_1 + w_2 + w_3)$$
...

Based on the above parameters and the U and R (pixel address and array index, respectively), and the bytes per pixel of the surface format (Bpp), the offset "u" in bytes from the base address of the surface is given by:

```
u = [(R * QPitch) + LODUL + U] * Bpp
```

Programming Note	
Context:	Packed YUV Surfaces
Packed YUV surface formats such as YCRCB_NORMAL, YCRCB_SWAPUVY etc. will be treated as 16bpp surface, not 32bpp, which may impact how they are laid out in memory.	

Tiling and Mip Tail for 1D Surfaces

There is no MIP Tail allowed for 1D surfaces because they are not allowed to be tiled. They must be declared as linear.

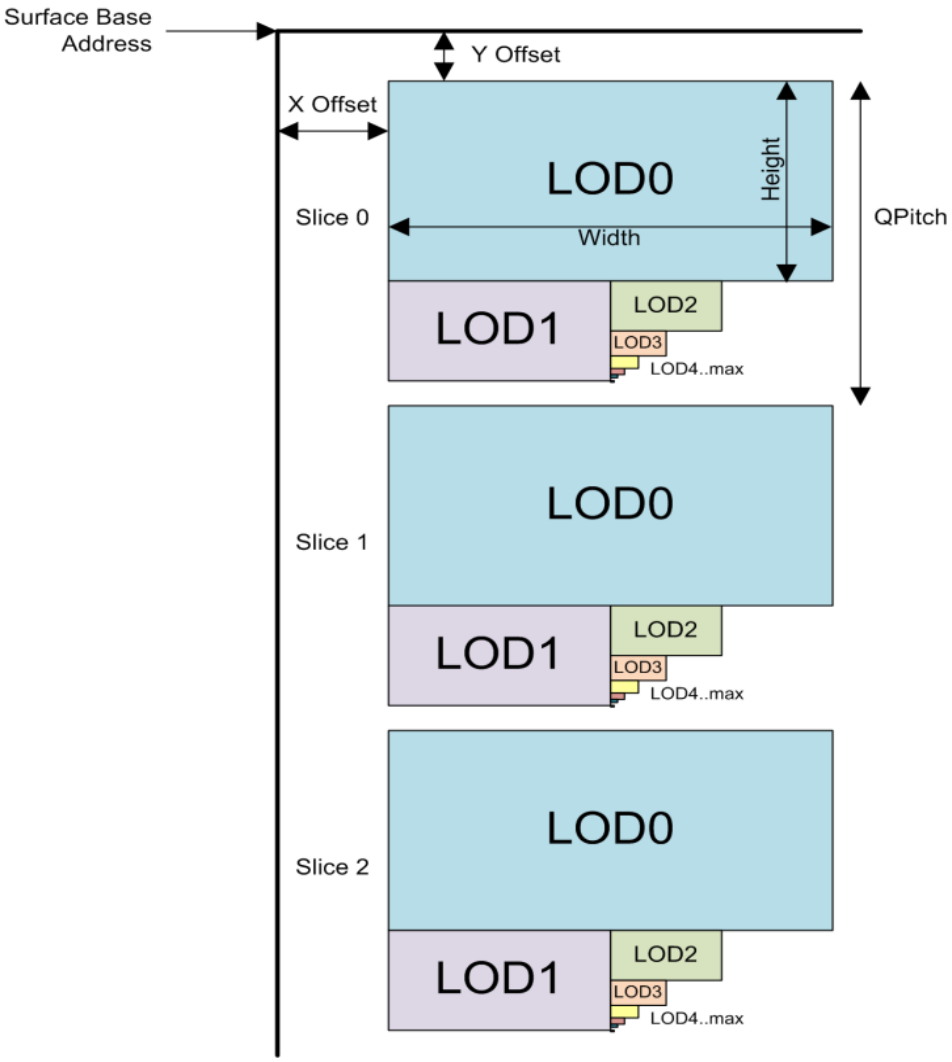
1D Alignment Requirements

1D surfaces are not tiled but laid out linearly in memory.

Tiled Resource Mode	Bits per Element	Horizontal Alignment
TRMODE_NONE	Any	64

2D Surfaces

2D surfaces represent two-dimensional bitmaps, which can also be mip-mapped and/or consist of array slices, effectively representing multiple 2D sub-surfaces within a single surface. The diagram below shows many of the parameters of a 2D surface or Arrayed 2D Surface and what they mean.



All surface parameters are defined in terms of texels (agnostic to the surface format).

Surface Pitch defines the distance in bytes between rows of the surface and is a function of the **Width** of LOD. **QPitch** specifies the distance in rows between array slices and is a function of the **Height**. **QPitch** should allow at least enough space for any Mips that may be present.

There can also be non-zero offsets (**X_Offset** and **Y_Offset)** defined from the base address which can be used to provide padding or provide an offset to a lower-detail LOD.

There are limitations to the physical size of an LOD in the sampler texture cache. An LOD must be aligned to a cache-line except for some special cases related to Planar YUV surfaces. In general, the cache-alignment restriction implies there is a minimum height for an LOD of 4 texels. So, LODs which are smaller than 4 high are padded.

There are limitations to the physical size of an LOD in the texture cache. An LOD must be aligned to a cache-line except for some special cases related to Planar YUV surfaces. In general, the cache-alignment restriction implies there is a minimum height for an LOD of 4 texels. So, LODs which are smaller than 4 high are padded.

For tiled surfaces with **TR_MODE** != TR_NONE this restriction is not significant because the MIP tail will be used for smaller MIPs and the slots are a minimum of 64B. For non-tiled surface or surfaces where **TR_MODE == TR_NONE**, Mips smaller than 4 high start at the top of the region, and they are padded. This padding leads to a case where the smallest LOD starts "below" LOD1 vertically.

Calculating Texel Location

This section describes how the texel location is calculated once the Surface State and LOD are known. A number of parameters are useful to determine where given pixels are located on the 2D surface. The width (**W**$_L$) and height (**H**$_L$) for each LOD "L" is computed by the formula:

W_L = ((width»L) > 0 ? width»L:1)

H_L = ((height»L) > 0? height»L:1)

When **Corner Texel Mode** is enabled via the RENDER_SURFACE_STATE, the width and height of a 2D surface are calculated as shown below:

W_L = MAX(1,(W_0-1)»L)+1

H_L = MAX(1,(H_0-1)»L)+1

There is a restriction that the smallest map dimension is 2 texels for **Corner Texel Mode (**W_0 >1, H_0 > 1). This also applies to 2D arrays and 2D arrays viewed as cubes.

The LOD width and height for each subsequent LOD is one-half the previous LOD, with the minimum dimension being 1 texel. If the surface is multisampled and it is a depth or stencil surface or **Multisampled Surface Storage Format** in SURFACE_STATE is MSFMT_DEPTH_STENCIL, **W**$_L$ and **H**$_L$ must be adjusted as follows:

Number of Multisamples	W_L =	H_L =
2	ceiling(W_L / 2) * 4	H_L [no adjustment]

4	ceiling(W_L / 2) * 4	ceiling(H_L / 2) * 4
8	ceiling(W_L / 2) * 8	ceiling(H_L / 2) * 4
16	ceiling(W_L / 2) * 8	ceiling(H_L / 2) * 8

Next, aligned width, height, and depth parameters for each LOD "L" must be computed. The "i" and "j" parameters are horizontal and vertical alignment parameters set by state fields or defined as constants, depending on the surface. Depth has no alignment parameter (effectively it is 1).

The equation uses the i and j values that apply to the LOD being computed. The "p" and "q" parameters define the width and height in texels of the compression block for compressed surface formats. Both p and q are defined to equal 1 for uncompressed surface formats.

$$w_L = i * p * ceil\left(\frac{W_L}{i * p}\right)$$

$$h_L = j * q * ceil\left(\frac{H_L}{j * q}\right)$$

Once the height (h_i) and width (w_i) of each LOD is computed, the offset to each LOD can be determined. The offset is a vector with two dimensions. The elements in the LOD_L vector are named in order $LODU_L$, $LODV_L$.

LOD offset computation for when no Mip Tail is used or when L < Mip Tail Start LOD:

LOD_0 = (0,0)

LOD_1 = (0,h_0)

LOD_2 = (w_1,h_0)

LOD_3 = (w_1,h_0 + h_2)

LOD_4 = (w_1,h_0 + h_2 + h_3)

...

LOD_N = (w_1, h_0 + h_2 + h_3 ... + h_{N-1})

Where N = MIP_COUNT for the surface. As noted previous in this section, the value of h_2 + h_3... + h_{N-1} may be greater than h_1 due to alignment requirements.

Based on the above parameters and the U, V, and R (two dimensional pixel address U/V and array index R), and the *bytes* per pixel of the surface format (Bpp), the offsets u in bytes and v in rows are given by:

u = (U + $LODU_L$) * Bpp

v = (R * QPitch) + $LODV_L$ + V

For a description of how the Mip Tail is laid out and offsets into the Mip Tail are calculated see the sub-section on 2D Surface Layout for Mip Tails.

Programming Note	
Context:	Packed YUV Surfaces
Packed YUV surface formats such as YCRCB_NORMAL, YCRCB_SWAPUVY etc. will be treated as 16bpp surface, not 32bpp, which may impact how they are layed out in memory.	

The two-dimensional offset into the surface (for non-MipTail cases) is defined by the u and v values computed above. The lower virtual address bits are determined by the following table, based on the bits of u and v. An *element* is defined as a pixel for uncompressed surface formats and a compression block for compressed surface formats. Empty bit positions indicate that the bit is not part of the tile swizzle and is filled in with equations given next (note that linear mode has all bits empty--there is no swizzling in linear mode).

The table below shows the mapping of u and v address bits within a tile for the supported tiling modes for a 2D surface. The u bits are a Byte address within a row and the v bits are a Row address within the tile.

Programming Note	
Context:	Tiling Definition

Tile Mode	Bits per Element	TileID constants		Virtual Address Bits															
		Cv	Cu	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TileYS	64 & 128	6	10	u9	v5	u8	v4	u7	v3	u6	v2	u5	u4	v1	v0	u3	u2	u1	u0
	16 & 32	7	9	u8	v6	u7	v5	u6	v4	u5	v3	u4	v2	v1	v0	u3	u2	u1	u0
	8	8	8	u7	v7	u6	v6	u5	v5	u4	v4	v3	v2	v1	v0	u3	u2	u1	u0
TileYF	64 & 128	4	8					u7	v3	u6	v2	u5	u4	v1	v0	u3	u2	u1	u0
	16 & 32	5	7					u6	v4	u5	v3	u4	v2	v1	v0	u3	u2	u1	u0
	8	6	6					u5	v5	u4	v4	v3	v2	v1	v0	u3	u2	u1	u0
TileY	all	5	7					u6	u5	u4	v4	v3	v2	v1	v0	u3	u2	u1	u0
TileX	all	3	9					v2	v1	v0	u8	u7	u6	u5	u4	u3	u2	u1	u0
TileW	all	6	6					u5	u4	u3	v5	v4	v3	v2	u2	v1	u1	v0	u0
Linear	all	0	0																

The TileID fills the upper bits of the virtual address (starting with the lowest blank bit in the above table):

TileID = (v » Cv) * (Pitch » Cu) + (u » Cu)

Where Pitch is the **Surface_Pitch** field from RENDER_SURFACE_STATE.

Note: Multisampled CMS and UMS surfaces use a modified address bit swizzling table rather than the one above. Refer to the *Multisampled*2D Surfaces section for details.

Tiling and Mip Tails for 2D Surfaces

When surface is Tiled (Tile_Mode=YMAJOR) and Tile Resources are enabled (TR_MODE != TR_NONE), a 2D surface can contain a Mip Tail for smaller Mip sizes.

When LOD (L) is less than the **Mip Tail Start LOD** (S) declared in the Surface State the offset to the start of LOD is calculated as shown above.

If the LOD is greater than or equal to **Mip Tail Start LOD** field in the surface state then the MIP Tail layout below is used..

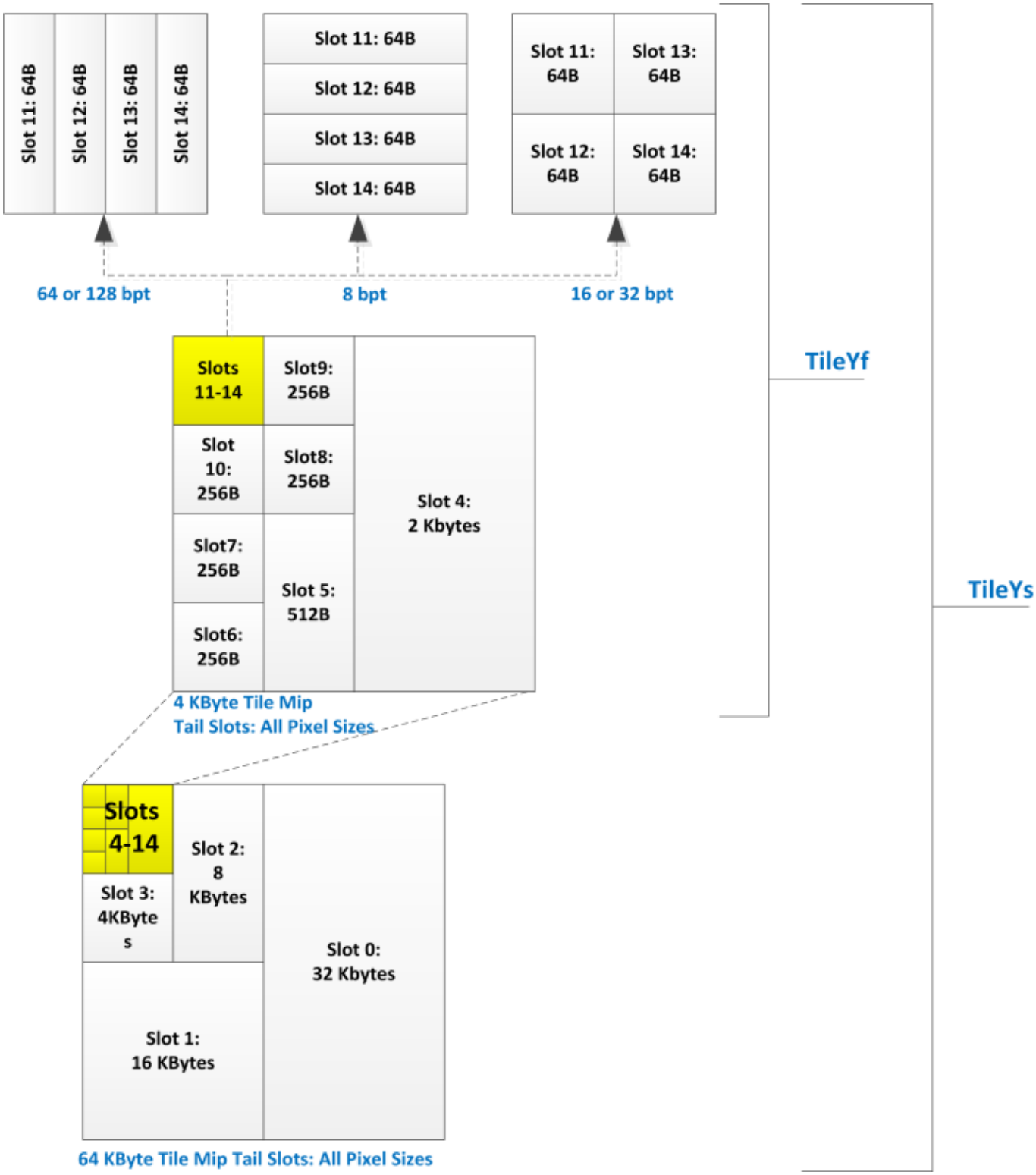
For tiled resources, the mip tail offset is given by the following, where s is the Mip Tail Start LOD:

$LOD_S = (w_1, h_0 + h_2 + h_3 + ... + h_{S-1})$

The LOD's in the Mip Tail are arranged differently than the other LOD's.

The diagram below shows the 64KB TileYS Mip Tail layout of LODs within it, with "slots" indicating the LOD contained within (slot 0 corresponds to LOD_s above). LOD's are aligned to the upper left corner of the space available. The block marked "Slots 4-14" is a 4KB tile arrangement as shown. Within this 4KB tile slots 11 thru 14 are arranged differently depending on the number bits per texel (bpt).

A TileYf (4KByte) Mip Tail will start with the 4KByte tile shown, but the slots will be renumbered to start at Slot0 rather than Slot4. The layout of slots 11 through 14 remain the same. Note that Slots 12-14 are NOT 256-Byte aligned which is not compliant with the standard MIP Tail layout. These slots are not supported for Standard Tiling.



The offsets *into the Mip Tail tile* are given by the following table for each LOD in the Mip tail. Each entry in the table is a horizontal (M_U) and vertical (M_V) position (in texels) from the upper left corner of the Mip Tail. If LOD >= S (starting LOD for MIP Tail), then these Mip Tail offsets must be added to the $LODU_L$ and $LODV_L$ calculated above.

Note that many of the higher LODs are not possible given surface size constraints, but they are listed here for reference. The offsets given here need to be added to the LODs offset computed earlier to obtain the offset into the surface LOD_L.

Slot 11 is 256-byte aligned. Slots 12 through 14 are 64-byte aligned.

TileYS LOD					TileYF LOD					128 bpe	64 bpe	32 bpe	16 bpe	8 bpe
1x	2x	4x	8x	16x	1x	2x	4x	8x	16x					
s										(32,0)	(64,0)	(64,0)	(128,0)	(128,0)
s+1	s									(0,32)	(0,32)	(0,64)	(0,64)	(0,128)
s+2	s+1	s								(16,0)	(32,0)	(32,0)	(64,0)	(64,0)
s+3	s+2	s+1	s							(0,16)	(0,16)	(0,32)	(0,32)	(0,64)
s+4	s+3	s+2	s+1	s	s					(8,0)	(16,0)	(16,0)	(32,0)	(32,0)
s+5	s+4	s+3	s+2	s+1	s+1	s				(4, 8)	(8, 8)	(8, 16)	(16, 16)	(16, 32)
s+6	s+5	s+4	s+3	s+2	s+2	s+1				(0, 12)	(0, 12)	(0, 24)	(0, 24)	(0, 48)
s+7	s+6	s+5	s+4	s+3	s+3	s+2				(0, 8)	(0, 8)	(0, 16)	(0, 16)	(0, 32)
s+8	s+7	s+6	s+5	s+4	s+4	s+3	s			(4, 4)	(8, 4)	(8, 8)	(16, 8)	(16, 16)
s+9	s+8	s+7	s+6	s+5	s+5	s+4	s+1			(4, 0)	(8, 0)	(8, 0)	(16, 0)	(16, 0)
s+10	s+9	s+8	s+7	s+6	s+6	s+5	s+2	s		(0, 4)	(0, 4)	(0, 8)	(0, 8)	(0, 16)
s+11	s+10	s+9	s+8	s+7	s+7	s+6	s+3	s+1	s	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)
s+12	s+11	s+10	s+9	s+8	s+8	s+7	s+4	s+2	s+1	(1, 0)	(2, 0)	(0, 4)	(0, 4)	(0, 4)
s+13	s+12	s+11	s+10	s+9	s+9	s+8	s+5	s+3	s+2	(2, 0)	(4, 0)	(4, 0)	(8, 0)	(0, 8)
s+14	s+13	s+12	s+11	s+10	s+10	s+9	s+6	s+4	s+3	(3, 0)	(6, 0)	(4, 4)	(8, 4)	(0, 12)

If the LOD is located in the MIP Tail then the equation for calculating the byte positions for u and v become:

$$u = (U + LODU_s + M_U) * Bpp$$
$$v = (R * QPitch) + LODV_s + M_V + V$$

where M_U and M_V are the offset parameters from the table above for the given slot in the MIP Tail.

Programming Note	
Context:	Lossless Compression and MIP Tail
Lossless compression must not be used on surfaces which have MIP Tail which contains MIPs for Slots greater than 11.	

Stencil Buffer Layout

This section details the layout of the stencil buffer.

1x Screen space(pixel)							
0	1	4	5	16	17	20	21
2	3	6	7	18	19	22	23
8	9	12	13	24	25	28	29
10	11	14	15	26	27	30	31
32	33	36	37	48	49	52	53
34	35	38	39	50	51	54	55
40	41	44	45	56	57	60	61
42	43	46	47	58	59	62	63

1x Physical space(pixel)															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

4x Screen space(pixel, sample)							
(0,0)	(1,0)	(0,1)	(1,1)	(4,0)	(5,0)	(4,1)	(5,1)
(2,0)	(3,0)	(2,1)	(3,1)	(6,0)	(7,0)	(6,1)	(7,1)
(0,2)	(1,2)	(0,3)	(1,3)	(4,2)	(5,2)	(4,3)	(5,3)
(2,2)	(3,2)	(2,3)	(3,3)	(6,2)	(7,2)	(6,3)	(7,3)
(8,0)	(9,0)	(8,1)	(9,1)	(12,0)	(13,0)	(12,1)	(13,1)
(10,0)	(11,0)	(10,1)	(11,1)	(14,0)	(15,0)	(14,1)	(15,1)
(8,2)	(9,2)	(8,3)	(9,3)	(12,2)	(13,2)	(12,3)	(13,3)
(10,2)	(11,2)	(10,3)	(11,3)	(14,2)	(15,2)	(14,3)	(15,3)

4x Physical space(pixel, sample)															
(0,0)	(1,0)	(2,0)	(3,0)	(0,1)	(1,1)	(2,1)	(3,1)	(0,2)	(1,2)	(2,2)	(3,2)	(0,3)	(1,3)	(2,3)	(3,3)
(4,0)	(5,0)	(6,0)	(7,0)	(4,1)	(5,1)	(6,1)	(7,1)	(4,2)	(5,2)	(6,2)	(7,2)	(4,3)	(5,3)	(6,3)	(7,3)
(8,0)	(9,0)	(10,0)	(11,0)	(8,1)	(9,1)	(10,1)	(11,1)	(8,2)	(9,2)	(10,2)	(11,2)	(8,3)	(9,3)	(10,3)	(11,3)
(12,0)	(13,0)	(14,0)	(15,0)	(12,1)	(13,1)	(14,1)	(15,1)	(12,2)	(13,2)	(14,2)	(15,2)	(12,3)	(13,3)	(14,3)	(15,3)

8x Screen space(pixel, sample)							
(0,0)	(1,0)	(0,1)	(1,1)	(0,4)	(1,4)	(0,5)	(1,5)
(2,0)	(3,0)	(2,1)	(3,1)	(2,4)	(3,4)	(2,5)	(3,5)
(0,2)	(1,2)	(0,3)	(1,3)	(0,6)	(1,6)	(0,7)	(1,7)
(2,2)	(3,2)	(2,3)	(3,3)	(2,6)	(3,6)	(2,7)	(3,7)
(4,0)	(5,0)	(4,1)	(5,1)	(4,4)	(5,4)	(4,5)	(5,5)
(6,0)	(7,0)	(6,1)	(7,1)	(6,4)	(7,4)	(6,5)	(7,5)
(4,2)	(5,2)	(4,3)	(5,3)	(4,6)	(5,6)	(4,7)	(5,7)
(6,2)	(7,2)	(6,3)	(7,3)	(6,6)	(7,6)	(6,7)	(7,7)

8x Physical space(pixel, sample)															
(0,0)	(1,0)	(2,0)	(3,0)	(0,1)	(1,1)	(2,1)	(3,1)	(0,2)	(1,2)	(2,2)	(3,2)	(0,3)	(1,3)	(2,3)	(3,3)
(0,4)	(1,4)	(2,4)	(3,4)	(0,5)	(1,5)	(2,5)	(3,5)	(0,6)	(1,6)	(2,6)	(3,6)	(0,7)	(1,7)	(2,7)	(3,7)
(4,0)	(5,0)	(6,0)	(7,0)	(4,1)	(5,1)	(6,1)	(7,1)	(4,4)	(5,4)	(6,4)	(7,4)	(4,5)	(5,5)	(6,5)	(7,5)
(4,2)	(5,2)	(6,2)	(7,2)	(4,3)	(5,3)	(6,3)	(7,3)	(4,6)	(5,6)	(6,6)	(7,6)	(4,7)	(5,7)	(6,7)	(7,7)

2D/CUBE Alignment Requirement

The vertical and horizontal alignment fields in the RENDER_SURFACE_STATE are ignored for standard tiling formats (TRMODE = NONE). In the case of standard tiling formats the alignment requirements

are fixed and are provided for by the tables below for 2D and CUBE surface.

Tile Mode	Bits per Element	Horizontal Alignment	Vertical Alignment
TileYS	128	64	64
	64	128	64
	32	128	128
	16	256	128
	8	256	256
TileYF	128	16	16
	64	32	16

Tile Mode	Bits per Element	Horizontal Alignment	Vertical Alignment
	32	32	32
	16	64	32
	8	64	64

For MSFMT_MSS type multi-sampled TileYS and TileYF surfaces, the alignments given above must be divided by the appropriate value from the table below.

Number of Multisamples	Horizontal Alignment is divided by	Vertical Alignment is divided by
2	2	1
4	2	2
8	4	2
16	4	4

Multisampled 2D Surfaces

There are three types of multisampled surface layouts designated as follows:

* **IMS** Interleaved Multisampled Surface

* **CMS** Compressed Mulitsampled Surface

* **UMS** Uncompressed Multisampled Surface

These surface layouts are described in the following sections.

Interleaved Multisampled Surfaces

These surfaces contain the samples in an interleaved fashion, with the underlying surface in memory having a height and width that is larger than the non-multisampled surface as follows:

- 4x MSAA: 2x width and 2x height of non-multisampled surface.
- 8x MSAA: 4x width and 2x height of non-multisampled surface.
- 16x MSAA: 4X width and 4X height of the non-multisampled surface.

When sampling from an IMS surface (e.g. ld2dms), the coordinates are automatically scaled to handle the increased physical size of the map.

The tables below shows the layout of 16-bit Depth (Z) values for different IMS formats. It shows layout of each 64-Byte chunk.

1X:

	bit 0							**bit 127**
Bytes 15:0	P(0,0)	P(1,0)	P(2,0)	P(3,0)	P(4,0)	P(5,0)	P(6,0)	P(7,0)
Bytes 16:31	P(0,1)	P(1,1)	P(2,1)	P(3,1)	P(4,1)	P(5,1)	P(6,1)	P(7,1)
Bytes 32:47	P(0,2)	P(1,2)	P(2,2)	P(3,2)	P(4,2)	P(5,2)	P(6,2)	P(7,2)
Bytes 48:63	P(0,3)	P(1,3)	P(2,3)	P(3,3)	P(4,3)	P(5,3)	P(6,3)	P(7,3)

2X:

	bit 0							**bit 127**
Bytes 15:0	P(0,0) s0	P(1,0) s0	P(0,0) s1	P(1,0) s1	P(2,0) s0	P(3,0) s0	P(2,0) s1	P(3,0) s1
Bytes 16:31	P(0,1) s0	P(1,1) s0	P(0,1) s1	P(1,1) s1	P(2,1) s0	P(3,1) s0	P(2,1) s1	P(3,1) s1
Bytes 32:47	P(0,2) s0	P(1,2) s0	P(0,2) s1	P(1,2) s1	P(2,2) s0	P(3,2) s0	P(2,2) s1	P(3,2) s1
Bytes 48:63	P(0,3) s0	P(1,3) s0	P(0,3) s1	P(1,3) s1	P(2,3) s0	P(3,3) s0	P(2,3) s1	P(3,3) s1

4X:

	bit 0							**bit 127**
Bytes 15:0	P(0,0) s0	P(1,0) s0	P(0,0) s1	P(1,0) s1	P(2,0) s0	P(3,0) s0	P(2,0) s1	P(3,0) s1
Bytes 16:31	P(0,1) s0	P(1,1) s0	P(0,1) s1	P(1,1) s1	P(2,1) s0	P(3,1) s0	P(2,1) s1	P(3,1) s1
Bytes 32:47	P(0,0) s2	P(1,0) s2	P(0,0) s3	P(1,0) s3	P(2,0) s2	P(3,0) s2	P(2,0) s3	P(3,0) s3
Bytes 48:63	P(0,1) s2	P(1,1) s2	P(0,1) s3	P(1,1) s3	P(2,1) s2	P(3,1) s2	P(2,1) s3	P(3,1) s3

8X:

	bit 0							**bit 127**
Bytes 15:0	P(0,0) s0	P(1,0) s0	P(0,0) s1	P(1,0) s1	P(0,0) s4	P(1,0) s4	P(0,0) s5	P(1,0) s5
Bytes 16:31	P(0,1) s0	P(1,1) s1	P(0,1) s1	P(1,1) s1	P(0,1) s4	P(1,1) s4	P(0,1) s5	P(1,1) s5
Bytes 32:47	P(0,0) s2	P(1,0) s2	P(0,0) s3	P(1,0) s3	P(0,0) s6	P(1,0) s6	P(0,0) s7	P(1,0) s7

Bytes 48:63	P(0,1) s2	P(1,1) s2	P(0,1) s3	P(1,1) s3	P(0,1) s6	P(1,1) s6	P(0,1) s7	P(1,1) s7

16X:

	bit 0							**bit 127**
Bytes 15:0	P(0,0) s0	P(1,0) s0	P(0,0) s1	P(1,0) s1	P(0,0) s4	P(1,0) s4	P(0,0) s5	P(1,0) s5
Bytes 16:31	P(0,1) s0	P(1,1) s0	P(0,1) s1	P(1,1) s1	P(0,1) s4	P(1,1) s4	P(0,1) s5	P(1,1) s5
Bytes 32:47	P(0,0) s2	P(1,0) s2	P(0,0) s3	P(1,0) s3	P(0,0) s6	P(1,0) s6	P(0,0) s7	P(1,0) s7
Bytes 48:63	P(0,1) s2	P(1,1) s2	P(0,1) s3	P(1,1) s3	P(0,1) s6	P(1,1) s6	P(0,1) s7	P(1,1) s7

	bit 0							**bit 127**
Bytes 64:79	P(0,0) s8	P(1,0) s8	P(0,0) s9	P(1,0) s9	P(0,0) s12	P(1,0) s12	P(0,0) s13	P(1,0) s13
Bytes 80:95	P(0,1) s8	P(1,1) s8	P(0,1) s9	P(1,1) s9	P(0,1) s12	P(1,1) s12	P(0,1) s13	P(1,1) s13
Bytes 96:111	P(0,0) s10	P(1,0) s10	P(0,0) s11	P(1,0) s11	P(0,0) s14	P(1,0) s14	P(0,0) s15	P(1,0) s15
Bytes 112:127	P(0,1) s10	P(1,1) s10	P(0,1) s11	P(1,1) s11	P(0,1) s14	P(1,1) s14	P(0,1) s15	P(1,1) s15

The table below shows the layout of 32-bit Depth (Z) values for different IMS formats. It shows layout of each 64-Byte chunk.

1X:

	bit 0				**bit 127**	
Bytes 15:0	P(0,0)	P(1,0)	P(2,0)		P(3,0)	
Bytes 16:31	P(0,1)	P(1,1)	P(2,1)		P(3,1)	
Bytes 32:47	P(0,2)	P(1,2)	P(2,2)		P(3,2)	
Bytes 48:63	P(0,3)	P(1,3)	P(2,3)		P(3,3)	

2X:

	bit 0			**bit 127**
Bytes 15:0	P(0,0) s0	P(1,0) s0	P(0,0) s1	P(1,0) s1

Bytes 16:31	P(0,1) s0	P(1,1) s0	P(0,1) s1	P(1,1) s1
Bytes 32:47	P(0,2) s0	P(1,2) s0	P(0,2) s1	P(1,2) s1
Bytes 48:63	P(0,3) s0	P(1,3) s0	P(0,3) s1	P(1,3) s1

4X:

	bit 0			**bit 127**
Bytes 15:0	P(0,0) s0	P(1,0) s0	P(0,0) s1	P(1,0) s1
Bytes 16:31	P(0,1) s0	P(1,1) s0	P(0,1) s1	P(1,1) s1
Bytes 32:47	P(0,0) s2	P(1,0) s2	P(0,0) s3	P(1,0) s3
Bytes 48:63	P(0,1) s2	P(1,1) s2	P(0,1) s3	P(1,1) s3

8X:

	bit 0			**bit 127**
Bytes 15:0	P(0,0) s0	P(1,0) s0	P(0,0) s1	P(1,0) s1
Bytes 16:31	P(0,1) s0	P(1,1) s0	P(0,1) s1	P(1,1) s1
Bytes 32:47	P(0,0) s2	P(1,0) s2	P(0,0) s3	P(1,0) s3
Bytes 48:63	P(0,1) s2	P(1,1) s2	P(0,1) s3	P(1,1) s3

	bit 0			**bit 127**
Bytes 64:79	P(0,0) s4	P(1,0) s4	P(0,0) s5	P(1,0) s5
Bytes 80:95	P(0,1) s4	P(1,1) s4	P(0,1) s5	P(1,1) s5
Bytes 96:111	P(0,0) s6	P(1,0) s6	P(0,0) s7	P(1,0) s7
Bytes 112:127	P(0,1) s6	P(1,1) s6	P(0,1) s7	P(1,1) s7

16X:

	bit 0			**bit 127**
Bytes 15:0	P(0,0) s0	P(1,0) s0	P(0,0) s1	P(1,0) s1
Bytes 16:31	P(0,1) s0	P(1,1) s0	P(0,1) s1	P(1,1) s1
Bytes 32:47	P(0,0) s2	P(1,0) s2	P(0,0) s3	P(1,0) s3
Bytes 48:63	P(0,1) s2	P(1,1) s2	P(0,1) s3	P(1,1) s3

	bit 0			**bit 127**
Bytes 64:79	P(0,0) s4	P(1,0) s4	P(0,0) s5	P(1,0) s5
Bytes 80:95	P(0,1) s4	P(1,1) s4	P(0,1) s5	P(1,1) s5
Bytes 96:111	P(0,0) s6	P(1,0) s6	P(0,0) s7	P(1,0) s7
Bytes 112:127	P(0,1) s6	P(1,1) s6	P(0,1) s7	P(1,1) s7

	bit 0			**bit 127**
Bytes 128:143	P(0,0) s8	P(1,0) s8	P(0,0) s9	P(1,0) s9
Bytes 144:159	P(0,1) s8	P(1,1) s8	P(0,1) s9	P(1,1) s9
Bytes 160:175	P(0,0) s10	P(1,0) s10	P(0,0) s11	P(1,0) s11
Bytes 176:191	P(0,1) s10	P(1,1) s10	P(0,1) s11	P(1,1) s11

	bit 0			**bit 127**
Bytes 192:207	P(0,0) s12	P(1,0) s12	P(0,0) s13	P(1,0) s13
Bytes 208:223	P(0,1) s12	P(1,1) s12	P(0,1) s13	P(1,1) s13
Bytes 224:239	P(0,0) s14	P(1,0) s14	P(0,0) s15	P(1,0) s15
Bytes 240:255	P(0,1) s14	P(1,1) s14	P(0,1) s15	P(1,1) s15

The table below shows the layout of Depth Stencil values for different IMS formats. It shows layout of each 64-Byte chunk.

1X:

Bytes 0-7	P(0,0)	P(1,0)	P(0,1)	P(1,1)	P(2,0)	P(3,0)	P(2,1)	P(3,1)
Bytes 8-15	P(0,2)	P(1,2)	P(0,3)	P(1,3)	P(2,2)	P(3,2)	P(2,3)	P(3,3)
Bytes 16-23	P(4,0)	P(5,0)	P(4,1)	P(5,1)	P(6,0)	P(7,0)	P(6,1)	P(7,1)
Bytes 24-31	P(4,2)	P(5,2)	P(4,3)	P(5,3)	P(6,2)	P(7,2)	P(6,3)	P(7,3)
Bytes 32-39	P(0,4)	P(1,4)	P(0,5)	P(1,5)	P(2,4)	P(3,4)	P(2,5)	P(3,5)
Bytes 40-47	P(0,6)	P(1,6)	P(0,7)	P(1,7)	P(2,6)	P(3,6)	P(2,7)	P(3,7)
Bytes 48-55	P(4,4)	P(5,4)	P(4,5)	P(5,5)	P(6,4)	P(7,4)	P(6,5)	P(7,5)
Bytes 56-63	P(4,6)	P(5,6)	P(4,7)	P(5,7)	P(6,6)	P(7,6)	P(6,7)	P(7,7)

2X:

Bytes 0-7	P(0,0) s0	P(1,0) s0	P(0,1) s0	P(1,1) s0	P(0,0) s1	P(1,0) s1	P(0,1) s1	P(1,1) s1
Bytes 8-15	P(0,2) s0	P(1,2) s0	P(0,3) s0	P(1,3) s0	P(0,2) s1	P(1,2) s1	P(0,3) s1	P(1,3) s1
Bytes 16-23	P(2,0) s0	P(3,0) s0	P(2,1) s0	P(3,1) s0	P(2,0) s1	P(3,0) s1	P(2,1) s1	P(3,1) s1
Bytes 24-31	P(2,2) s0	P(3,2) s0	P(2,3) s0	P(3,3) s0	P(2,2) s1	P(3,2) s1	P(2,3) s1	P(3,3) s1
Bytes 32-39	P(0,4) s0	P(1,4) s0	P(0,5) s0	P(1,5) s0	P(0,4) s1	P(1,4) s1	P(0,5) s1	P(1,5) s1
Bytes 40-47	P(0,6) s0	P(1,6) s0	P(0,7) s0	P(1,7) s0	P(0,6) s1	P(1,6) s1	P(0,7) s1	P(1,7) s1
Bytes 48-55	P(2,4) s0	P(3,4) s0	P(2,5) s0	P(3,5) s0	P(2,4) s1	P(3,4) s1	P(2,5) s1	P(3,5) s1
Bytes 56-63	P(2,6) s0	P(3,6) s0	P(2,7) s0	P(3,7) s0	P(2,6) s1	P(3,6) s1	P(2,7) s1	P(3,7) s1

4X:

Bytes 0-7	P(0,0) s0	P(1,0) s0	P(0,1) s0	P(1,1) s0	P(0,0) s1	P(1,0) s1	P(0,1) s1	P(1,1) s1
Bytes 8-15	P(0,0) s2	P(1,0) s2	P(0,1) s2	P(1,1) s2	P(0,0) s3	P(1,0) s3	P(0,1) s3	P(1,1) s3
Bytes 16-23	P(2,0) s0	P(3,0) s0	P(2,1) s0	P(3,1) s0	P(2,0) s1	P(3,0) s1	P(2,1) s1	P(3,1) s1
Bytes 24-31	P(2,0) s2	P(3,0) s2	P(2,1) s2	P(3,1) s2	P(2,0) s3	P(3,0) s3	P(2,1) s3	P(3,1) s3
Bytes 32-39	P(0,2) s0	P(1,2) s0	P(0,3) s0	P(1,3) s0	P(0,2) s1	P(1,2) s1	P(0,3) s1	P(1,3) s1
Bytes 40-47	P(0,2) s2	P(1,2) s2	P(0,3) s2	P(1,3) s2	P(0,2) s3	P(1,2) s3	P(0,3) s3	P(1,3) s3
Bytes 48-55	P(2,2) s0	P(3,2) s0	P(2,3) s0	P(3,3) s0	P(2,2) s1	P(3,2) s1	P(2,3) s1	P(3,3) s1
Bytes 56-63	P(2,2) s2	P(3,2) s2	P(2,3) s2	P(3,3) s2	P(2,2) s3	P(3,2) s3	P(2,3) s3	P(3,3) s3

8X:

Bytes 0-7	P(0,0) s0	P(1,0) s0	P(0,1) s0	P(1,1) s0	P(0,0) s1	P(1,0) s1	P(0,1) s1	P(1,1) s1
Bytes 8-15	P(0,0) s2	P(1,0) s2	P(0,1) s2	P(1,1) s2	P(0,0) s3	P(1,0) s3	P(0,1) s3	P(1,1) s3
Bytes 16-23	P(0,0) s4	P(1,0) s4	P(0,1) s4	P(1,1) s4	P(0,0) s5	P(1,0) s5	P(0,1) s5	P(1,1) s5
Bytes 24-31	P(0,0) s6	P(1,0) s6	P(0,1) s6	P(1,1) s6	P(0,0) s7	P(1,0) s7	P(0,1) s7	P(1,1) s7
Bytes 32-39	P(0,2) s0	P(1,2) s0	P(0,3) s0	P(1,3) s0	P(0,2) s1	P(1,2) s1	P(0,3) s1	P(1,3) s1
Bytes 40-47	P(0,2) s2	P(1,2) s2	P(0,3) s2	P(1,3) s2	P(0,2) s3	P(1,2) s3	P(0,3) s3	P(1,3) s3
Bytes 48-55	P(0,2) s4	P(1,2) s4	P(0,3) s4	P(1,3) s4	P(0,2) s5	P(1,2) s5	P(0,3) s5	P(1,3) s5
Bytes 56-63	P(0,2) s6	P(1,2) s6	P(0,3) s6	P(1,3) s6	P(0,2) s7	P(1,2) s7	P(0,3) s7	P(1,3) s7

16X:

Bytes 0-7	P(0,0) s0	P(1,0) s0	P(0,1) s0	P(1,1) s0	P(0,0) s1	P(1,0) s1	P(0,1) s1	P(1,1) s1
Bytes 8-15	P(0,0) s2	P(1,0) s2	P(0,1) s2	P(1,1) s2	P(0,0) s3	P(1,0) s3	P(0,1) s3	P(1,1) s3
Bytes 16-23	P(0,0) s4	P(1,0) s4	P(0,1) s4	P(1,1) s4	P(0,0) s5	P(1,0) s5	P(0,1) s5	P(1,1) s5
Bytes 24-31	P(0,0) s6	P(1,0) s6	P(0,1) s6	P(1,1) s6	P(0,0) s7	P(1,0) s7	P(0,1) s7	P(1,1) s7
Bytes 32-39	P(0,0) s8	P(1,0) s8	P(0,1) s8	P(1,1) s8	P(0,0) s9	P(1,0) s9	P(0,1) s9	P(1,1) s9
Bytes 40-47	P(0,0) s10	P(1,0) s10	P(0,1) s10	P(1,1) s10	P(0,0) s11	P(1,0) s11	P(0,1) s11	P(1,1) s11
Bytes 48-55	P(0,0) s12	P(1,0) s12	P(0,1) s12	P(1,1) s12	P(0,0) s13	P(1,0) s13	P(0,1) s13	P(1,1) s13
Bytes 56-63	P(0,0) s14	P(1,0) s14	P(0,1) s14	P(1,1) s14	P(0,0) s15	P(1,0) s15	P(0,1) s15	P(1,1) s15

Compressed Multisampled Surfaces

Multisampled render targets can be compressed. If **Auxiliary Surface Mode** in SURFACE_STATE is set to AUX_CCS, hardware handles the compression using a software-invisible algorithm. However, performance optimizations in the multisample resolve kernel using the sampling engine are possible if the internal format of these surfaces is understood by software. This section documents the formats of the Multisample Control Surface (MCS) and Multisample Surface (MSS).

MCS Surface

The MCS surface consists of one element per pixel, with the element size being an 8-bit unsigned integer value for 4x multisampled surfaces, a 32-bit unsigned integer value for 8x multisampled surfaces and a 64-bit unsigned integer value for 16x multisampled surface. Each field within the element indicates which sample slice (SS) the sample resides on.

2x MCS

The 2x MCS is 8 bits per pixel. The 8 bits are encoded as follows:

7:2	1	0
reserved	sample 1 SS	sample 0 SS

Each 1-bit field indicates which sample slice (SS) the sample's color value is stored. An MCS value of 0x00 indicates that both samples are stored in sample slice 0 (thus have the same color). This is the fully compressed case. An MCS value of 0x03 indicates that all samples in the pixel are in the clear state, and none of the sample slices are valid. The pixel's color must be replaced with the surface's clear value.

4x MCS

The 4x MCS is 8 bits per pixel. The 8 bits are encoded as follows:

7:6	5:4	3:2	1:0
sample 3 SS	sample 2 SS	sample 1 SS	sample 0 SS

Each 2-bit field indicates which sample slice (SS) the sample's color value is stored. An MCS value of 0x00 indicates that all four samples are stored in sample slice 0 (thus all have the same color). This is the fully compressed case. An MCS value of 0xff indicates that all samples in the pixel are in the clear state, and none of the sample slices are valid. The pixel's color must be replaced with the surface's clear value. See the section below on Clear Pixel Conditions for additional encoding information.

8x MCS

Extending the mechanism used for the 4x MCS to 8x requires 3 bits per sample times 8 samples, or 24 bits per pixel. The 24-bit MCS value per pixel is placed in a 32-bit footprint, with the upper 8 bits unused as shown below. See the section below on Clear Pixel Conditions for additional encoding information.

31:24	23:21	20:18	17:15	14:12	11:9	8:6	5:3	2:0
reserved (MBZ)	sample 7 SS	sample 6 SS	sample 5 SS	sample 4 SS	sample 3 SS	sample 2 SS	sample 1 SS	sample 0 SS

16x MCS

The 16x MCS is 64 bits per pixel. The 64 bits are encoded as follows:

63:60	59:56	55:52	51:48	47:44	43:40	39:36	35:32
sample 15 SS	sample 14 SS	sample 13 SS	sample 12 SS	sample 11 SS	sample 10 SS	sample 9 SS	sample 8 SS

31:28	27:24	23:20	19:16	15:12	11:8	7:4	3:0
sample 7 SS	sample 6 SS	sample 5 SS	sample 4 SS	sample 3 SS	sample 2 SS	sample 1 SS	sample 0 SS

Other than this, the 16x algorithm is the same as the 8x algorithm. The MCS value indicating clear state is 0xffffffff_ffffffff. See the section below on Clear Pixel Conditions for additional encoding information.

Clear Pixel Conditions

The MCS format allows for the encoding of clear value for one or more planes of the multi-sampled surface. A value of all 1's for defined MCS bits indicates that all planes of the multi-sampled surface are clear. For example, a value of 0x3 for 2X MSAA MCS byte means that both planes of the pixel are clear. Likewise, a value of 0xff for X4, 0xffffffff for X8 and 0xffffffff_ffffffff for X16 MSAA means that all planes of the pixel are clear.

In the case where not all planes are clear, but at least 2 planes are clear the encoding of the MCS given above is changed. If the MCS value for plane 0 is non-zero, then all planes which are at all 1's are clear and all other planes are referencing the plane indicated by their respective MCS value minus 1. For example, a 4X MSAA MCS value of 01 10 11 11 means that MCS 0 is referencing plane 0, and MCS 1 is referencing plane 1, and MCS 2 and 3 are clear.

MSS Surface

The physical MSS surface is stored identically to a 2D array surface, with the height and width matching the pixel dimensions of the logical multisampled surface. The number of array slices in the physical

surface is 2, 4, 8, or 16 times that of the logical surface (depending on the number of multisamples). Sample slices belonging to the same logical surface array slice are stored in adjacent physical slices. The sampling engine ld2dss message gives direct access to a specific sample slice.

Tiling for CMS and UMS Surfaces

Multisampled CMS and UMS use a modified table from non-mulitsampled 2D surfaces.

TileY,

TileX, TileW, and Linear: Treat as 2D array, with the array index "R" modified as follows. "n" is the number of multisamples, "ss" is the sample slice index with range 0..n-1.

```
R(new) = ( R(old) « log2(n) ) | ss
```

TileYS: In addition to u and v, the sample slice index "ss" is included in the address swizzling according to the following table. Because of this, the mip tail holds one less LOD for each successive number of multisamples. Refer to the mip tail table in the previous section for behavior of the mip tail for each number of multisamples.

Number of Multisamples	Bits per Element	TileID constants		Virtual Address Bits									
		Cv	Cu	15	14	13	12	11	10	9	8	7	6
2x	64 & 128	6	9	ss0	v5	u8	v4	u7	v3	u6	v2	u5	u4
	16 & 32	7	8	ss0	v6	u7	v5	u6	v4	u5	v3	u4	v2
	8	8	7	ss0	v7	u6	v6	u5	v5	u4	v4	v3	v2
4x	64 & 128	5	9	ss1	ss0	u8	v4	u7	v3	u6	v2	u5	u4
	16 & 32	6	8	ss1	ss0	u7	v5	u6	v4	u5	v3	u4	v2
	8	7	7	ss1	ss0	u6	v6	u5	v5	u4	v4	v3	v2
8x	64 & 128	5	8	ss2	ss1	ss0	v4	u7	v3	u6	v2	u5	u4
	16 & 32	6	7	ss2	ss1	ss0	v5	u6	v4	u5	v3	u4	v2
	8	7	6	ss2	ss1	ss0	v6	u5	v5	u4	v4	v3	v2
16x	64 & 128	4	8	ss3	ss2	ss1	ss0	u7	v3	u6	v2	u5	u4
	16 & 32	5	7	ss3	ss2	ss1	ss0	u6	v4	u5	v3	u4	v2
	8	6	6	ss3	ss2	ss1	ss0	u5	v5	u4	v4	v3	v2

Note that Cv and Cu are also different that the values for non-multisampled 2D surfaces.

TileYF: In addition to u and v, the sample slice index "ss" is included in the address swizzling according to the following table. Because of this, the mip tail holds one less LOD for each successive number of multisamples. Refer to the mip tail table in the previous section for behavior of the mip tail for each number of multisamples.

Number of Multisamples	Bits per Element	TileID constants		Virtual Address Bits					
		Cv	Cu	11	10	9	8	7	6
2x	128 & 64	4	7	ss0	v3	u6	v2	u5	u4
	32 & 16	5	6	ss0	v4	u5	v3	u4	v2
	8	6	5	ss0	v5	u4	v4	v3	v2
4x	128 & 64	3	7	ss1	ss0	u6	v2	u5	u4
	32 & 16	4	6	ss1	ss0	u5	v3	u4	v2
	8	5	5	ss1	ss0	u4	v4	v3	v2
8x	128 & 64	3	6	ss2	ss1	ss0	v2	u5	u4
	32 & 16	4	5	ss2	ss1	ss0	v3	u4	v2
	8	5	4	ss2	ss1	ss0	v4	v3	v2
16x	128 & 64	2	6	ss3	ss2	ss1	ss0	u5	u4
	32 & 16	3	5	ss3	ss2	ss1	ss0	u4	v2
	8	4	4	ss3	ss2	ss1	ss0	v3	v2

Uncompressed Multisampled Surfaces

UMS surfaces similar to CMS, except that the **Auxiliary Surface Mode** is set to AUX_NONE, meaning that there is no MCS surface. UMS contains only an MSS surface, where each sample is stored on its sample slice (SS) of the same index.

Programming Note	
Context:	3D Sampler
See 3D Sampler Messages section for a description of how the ld2dms and ld2dms_w messages work for UMS surfaces	

Cube Surfaces

The 3D Pipeline supports *cubic environment maps*, conceptually arranged as a cube surrounding the origin of a 3D coordinate system aligned to the cube faces. These maps can be used to supply texel (color/alpha) data of the environment in any direction from the enclosed origin, where the direction is supplied as a 3D "vector" texture coordinate. These cube maps can also be mipmapped.

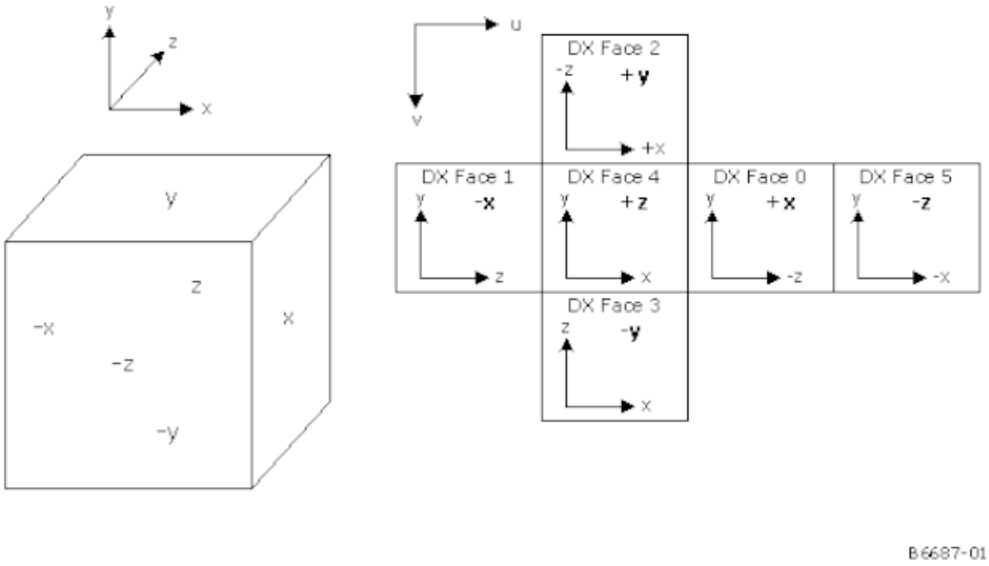
Each texture map level is represented as a group of six, square *cube face* texture surfaces. The faces are identified by their relationship to the 3D texture coordinate system. The subsections below describe the cube maps as described at the API as well as the memory layout dictated by the hardware.

The diagram below describes the cube map faces as they are defined at the DirectX API. It shows the axes on the faces as they would be seen from the inside (at the origin).

The 3D sampler converts the incoming U,V,R coordinates on the sampler.

This will be looking directly at face 4, the +z -face. Y is up by default.

DirectX Cube Map Definition



The coordinates on each face are relative to the center of the cube, and they range from -1.0 to 1.0 rather than the normal 0 to 1.0 normalized coordinate system in a 2D array surface.

Each face has a corresponding face identifier "f" as indicated in the following table:

face	face identifier "f"
+x	0
-x	1
+y	2
-y	3
+z	4
-z	5

A cube surface is stored in memory the same as a 2D array, with the face identifier "f" and array index "ai" being transformed into the "R" coordinate used in storing 2D arrays using the following equation:

```
R = (ai * 6) + f
```

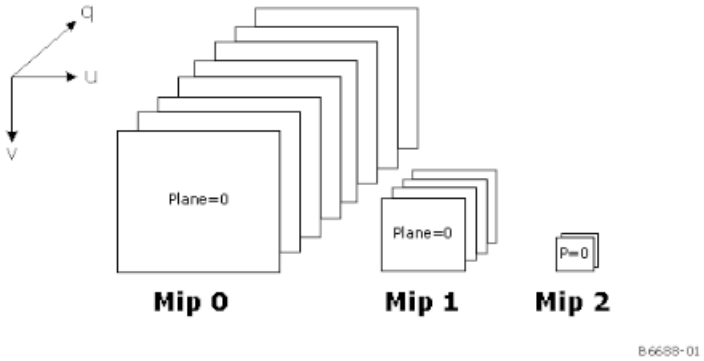
Refer to the "2D Surfaces" section for details on how 2D arrays are stored.

3D Surfaces

Multiple texture map surfaces (and their respective mipmap chains) can be arranged into a structure known as a Texture3D (volume) texture. A volume texture map consists of many *planes* of 2D texture maps.

See *Sampler* for a description of how volume textures are used.

Volume Texture Map



Surface Pitch defines the distance in bytes between rows of the surface. **Surface QPitch** specifies the distance in rows between R-slices. QPitch should allow at least enough space for any mips that may be present.

A number of parameters are useful to determine where given pixels are located on the 3D surface. First, the width, height, and depth for each LOD "L" is computed:

$$W_L = ((width >> L) > 0\,?\,width >> L : 1)$$
$$H_L = ((height >> L) > 0\,?\,height >> L : 1)$$
$$D_L = ((depth >> L) > 0\,?\,depth >> L : 1)$$

When **Corner Texel Mode** is enabled via the RENDER_SURFACE_STATE, the width and height of a 3D surface are calculated as shown below:

$W_L = MAX(1,(W_0-1)»L)+1$

$H_L = MAX(1,(H_0-1)»L)+1$

$D_L = MAX(1,(D_0-1)»L)+1$

There is a restriction that the smallest map dimension is 2 texels for **Corner Texel Mode** ($W_0 > 1$, $H_0 > 1$, $D_0 > 1$)

Next, aligned width, height, and depth parameters for each LOD "L" are computed. The "i", "j", and "k" parameters are the horizontal, vertical, and depth alignment parameters set by state fields or defined as constants. The alignment parameters may change at one point in the mip chain based on **Mip Tail Start LOD**. The equation uses the i/j values that apply to the LOD being computed. The "p", "q", and "s" parameters define the width, height, and depth in texels of the compression block for compressed surface formats. These are all defined to equal 1 for uncompressed surface formats.

$$w_L = i * ceil\left(\frac{W_L}{i}\right)$$
$$h_L = j * ceil\left(\frac{H_L}{j}\right)$$
$$d_L = D_L$$
$$d_L = k * s * ceil\left(\frac{D_L}{k * s}\right)$$

Next, the offset to each LOD is determined. The offset is a vector with three dimensions. The elements in the LOD_L vector are named in order $LODU_L$, $LODV_L$, $LODR_L$.

LOD offset computation for **Tiled Resource Mode** == TR_NONE or when L < **Mip Tail Start LOD**:

$$LOD_0 = (0,0,0)$$
$$LOD_1 = (0,h_0,0)$$
$$LOD_2 = (w_1,h_0,0)$$
$$LOD_3 = (w_1,h_0+h_2,0)$$
$$LOD_4 = (w_1,h_0+h_2+h_3,0)$$
...

For the Primary Surface

Based on the above parameters and the U, V, and R (three dimensional pixel address), and the bytes per pixel of the surface format (Bpp), the offsets u in bytes, v in rows, and r in slices are given by:

```
u = [U + LODUL] * Bpp
v = LODVL + V
r = LODRL + R
```

Programming Note	
Context:	Packed YUV Surfaces
Packed YUV surface formats such as YCRCB_NORMAL, YCRCB_SWAPUVY etc. will be treated as 16bpp surface, not 32bpp, which may impact how they are layed out in memory.	

The three-dimensional offset into the surface is defined by the u, v, and r values computed above. The lower virtual address bits are determined by the following table, based on the bits of u, v, and r. An *element* is defined as a pixel for uncompressed surface formats and a compression block for compressed surface formats.

Empty bit positions indicate that the bit is not part of the tile swizzle and is filled in with the equations given next (note that linear mode has all bits empty--there is no swizzling in linear mode).

		TileID constants			Virtual Address Bits														
Tile Mode	**Bits per Element**	**Cr**	**Cv**	**Cu**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TileYS	128 & 64	4	4	8	u7	v3	r3	u6	v2	r2	u5	u4								
	32	4	5	7	u6	v4	r3	u5	v3	r2	u4	v2	r1	r0	v1	v0	u3	u2	u1	u0
	16 & 8	5	5	6	u5	v4	r4	u4	v3	r3	v2	r2	r1	r0	v1	v0	u3	u2	u1	u0
TileYF	128 & 64	3	3	6					v2	r2	u5	u4	r1	r0	v1	v0	u3	u2	u1	u0
	32	3	4	5					v3	r2	u4	v2	r1	r0	v1	v0	u3	u2	u1	u0
	16 & 8	4	4	4					v3	r3	v2	r2	r1	r0	v1	v0	u3	u2	u1	u0
TileY	all	0	5	7					u6	u5	u4	v4	v3	v2	v1	v0	u3	u2	u1	u0
Linear	all	0	0	0																

The table below is enabled by use of the Tile Address Mapping Mode bi in **RENDER_SURFACE_STATE**.

This mapping must never be used.

Tile Mode	Bits per Element	TileID constants			Virtual Address Bits															
		Cr	Cv	Cu	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TileYS	128 & 64	4	4	8	u7	v3	r3	u6	v2	r2	u5	v1	r1	u4	r0	v0	u3	u2	u1	u0
	32	4	5	7	u6	v4	r3	u5	v3	r2	u4	v2	r1	u3	v1	v0	r0	u2	u1	u0
	16 & 8	5	5	6	u5	v4	r4	u4	v3	r3	u3	v2	r2	u2	v1	v0	r1	r0	u1	u0
TileYF	128 & 64	3	3	6					v2	r2	u5	v1	r1	u4	r0	v0	u3	u2	u1	u0
	32	3	4	5					v3	r2	u4	v2	r1	u3	v1	v0	r0	u2	u1	u0
	16 & 8	4	4	4					v3	r3	u3	v2	r2	u2	v1	v0	r1	r0	u1	u0
TileY	all	0	5	7					u6	u5	u4	v4	v3	v2	v1	v0	u3	u2	u1	u0
Linear	all	0	0	0																

The TileID fills the upper bits of the virtual address (starting with the lowest blank bit in the above table):

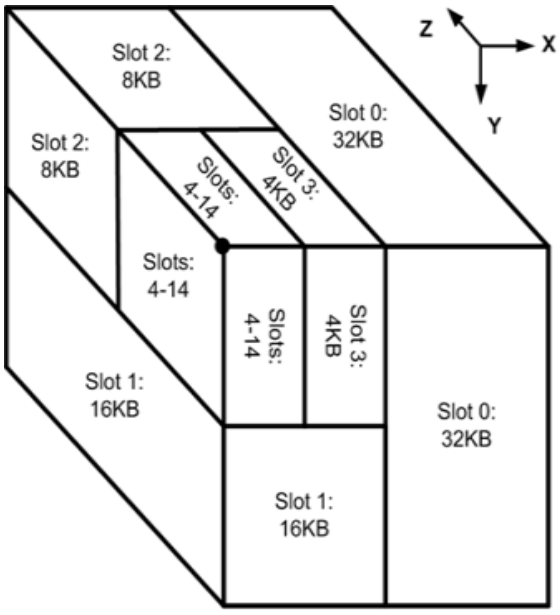
```
TileID = [(r » Cr) * (QPitch » Cv) + (v » Cv)] * (Pitch » Cu) + (u » Cu)
```

Tiling and Mip Tails for 3D Surfaces

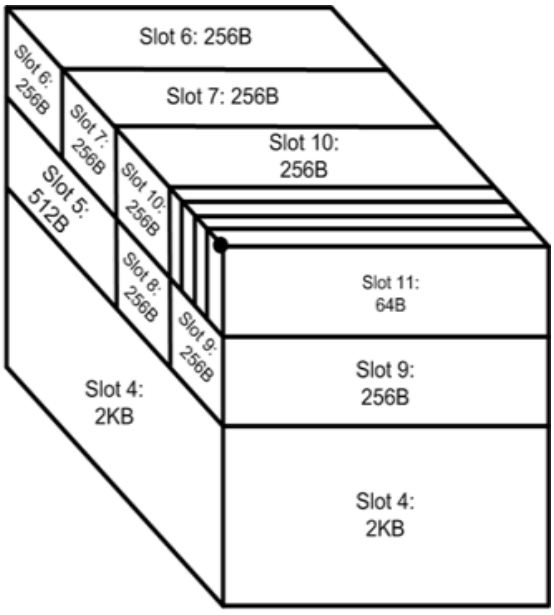
For tiled surfaces where Tiled Resource Mode != TR_NONE, the surface may contain a mip tail. The Mip tail offset is given by the following, where S is the Mip Tail Start LOD:

$$LOD_S = (w_1, h_0+h_2+h_3+ \ldots + h_{S-1}, 0)$$

The mip tail exhibits a different arrangement than the rest of the surface. The diagram below shows the 64KB TileYS mip tail and the arrangement of LODs within it, with "slots" indicating the LOD contained within (slot 0 corresponds to LOD s). LODs are aligned to the front upper left corner of the space available. The block marked "Slots 4-15" contains one of the 4KB tile arrangements within, depending on the surface format bits per element. For TileYF, only the 4KB tile exists, with 4 subtracted from each slot number.



64KB Tile Mip Tail Slots: All Pixel Depths



4KB Tile Mip Tail Slots: 8-bit & 16-bit Pixels

The offsets into the mip tail tile are given by the following table for each LOD in the mip tail. Note that many of the higher LODs are not possible given surface size constraints, but they are listed here for reference. The offsets given here need to be added to the LODs offset computed earlier to obtain the offset into the surface LOD_L.

TileYS LOD	TileYF LOD	128 bpe	64 bpe	32 bpe	16 bpe	8 bpe
s		(8, 0, 0)	(16, 0, 0)	(16, 0, 0)	(16, 0, 0)	(32, 0, 0)
s+1		(0, 8, 0)	(0, 8, 0)	(0, 16, 0)	(0, 16, 0)	(0, 16, 0)
s+2		(0, 0, 8)	(0, 0, 8)	(0, 0, 8)	(0, 0, 16)	(0, 0, 16)
s+3		(4, 0, 0)	(8, 0, 0)	(8, 0, 0)	(8, 0, 0)	(16, 0, 0)
s+4	s	(0, 4, 0)	(0, 4, 0)	(0, 8, 0)	(0, 8, 0)	(0, 8, 0)
s+5	s+1	(2, 0, 4)	(4, 0, 4)	(4, 0, 4)	(4, 0, 8)	(8, 0, 8)
s+6	s+2	(0, 2, 4)	(0, 2, 4)	(0, 4, 4)	(0, 4, 8)	(0, 4, 8)
s+7	s+3	(0, 0, 4)	(0, 0, 4)	(0, 0, 4)	(0, 0, 8)	(0, 0, 8)
s+8	s+4	(2, 2, 0)	(4, 2, 0)	(4, 4, 0)	(4, 4, 0)	(8, 4, 0)
s+9	s+5	(2, 0, 0)	(4, 0, 0)	(4, 0, 0)	(4, 0, 0)	(8, 0, 0)
s+10	s+6	(0, 2, 0)	(0, 2, 0)	(0, 4, 0)	(0, 4, 0)	(0, 4, 0)
s+11	s+7	(1, 0, 2)	(2, 0, 2)	(2, 0, 2)	(2, 0, 4)	(4, 0, 4)
s+12	s+8	(0, 0, 2)	(0, 0, 2)	(0, 0, 2)	(0, 0, 4)	(0, 0, 4)
s+13	s+9	(1, 0, 0)	(2, 0, 0)	(2, 0, 0)	(2, 0, 0)	(4, 0, 0)
s+14	s+10	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)

3D Alignment Requirements

The vertical and horizontal alignment fields in the RENDER_SURFACE_STATE are ignored for standard tiling formats (TRMODE != NONE). In the case of standard tiling formats (TileYs and TileYf) the alignment requirements are fixed and are provided for

by the tables below for 3D (volumetric) surfaces.

Tile Mode	Bits per Element	Horizontal Alignment	Vertical Alignment	Depth Alignment
TileYS	128	16	16	16
	64	32	16	16
	32	32	32	16
	16	32	32	32
	8	64	32	32
TileYF	128	4	8	8
	64	8	8	8
	32	8	16	8
	16	8	16	16
	8	16	16	16

Surface Padding Requirements

This section covers the requirements for padding around surfaces stored in memory, as there are cases where the device will overfetch beyond the bounds of the surface due to implementation of caches and other hardware structures.

Alignment Unit Size

This section documents the alignment (in texels) that the Surface Pitch and Surface Height must be programmed. For most

surface formats it is defined by HAlign and Valign

Alignment Parameters

Surface Defined By	Surface Format	Alignment Unit Width "*i*"	Alignment Unit Height "*j*"
3DSTATE_STENCIL_BUFFER	N/A	16	8
SURFACE_STATE	BC*, ETC*, EAC*	4	4
	FXT1	8	4
	all others	set by **Surface Horizontal Alignment**	set by **Surface Vertical Alignment**

Surface Defined By	Surface Format	MSAA	Alignment Unit Width "*i*"	Alignment Unit Height "*j*"
3DSTATE_DEPTH_BUFFER	D16_UNORM	1x, 4x, 16x	8	8
	D16_UNORM	2x, 8x	16	4
	Not D16_UNORM	1x, 2x, 4x, 8x, 16x	8	4

Surface Defined By	Surface Format	Alignment Unit Width "*i*"	Alignment Unit Height "*j*"
SURFACE_STATE	ASTC	Value of ASTC_2DBlockWidth (4, 5, 6, 8, 10, or 12)	ASTC_2DBlockHeight*4

Sampling Engine Surfaces

The sampling engine accesses texels outside of the surface if they are contained in the same cache line as texels that are within the surface. These texels will not participate in any calculation performed by the sampling engine and will not affect the result of any sampling engine operation, however if these texels lie outside of defined pages in the GTT, a GTT error will result when the cache line is accessed. In order to avoid these GTT errors, "padding" at the bottom and right side of a sampling engine surface is sometimes necessary.

It is possible that a cache line will straddle a page boundary if the base address or pitch is not aligned. All pages included in the cache lines that are part of the surface must map to valid GTT entries to avoid errors. To determine the necessary padding on the bottom and right side of the surface, refer to the table in Alignment Unit Size section for the i and j parameters for the surface format in use. The surface

must then be extended to the next multiple of the alignment unit size in each dimension, and all texels contained in this extended surface must have valid GTT entries.

For example, suppose the surface size is 15 texels by 10 texels and the alignment parameters are i=4 and j=2. In this case, the extended surface would be 16 by 10. Note that these calculations are done in texels, and must be converted to bytes based on the surface format being used to determine whether additional pages need to be defined.

For compressed textures (BC*, FXT1, ETC*, and EAC* surface formats), padding at the bottom of the surface is to an even compressed row. This is equivalent to a multiple of 2*q*, where *q* is the compression block height in texels. Thus, for padding purposes, these surfaces behave as if $j = 2q$ only for surface padding purposes. The value of *j* is still equal to *q* for mip level alignment and QPitch calculation. For cube surfaces, an additional two rows of padding are required at the bottom of the surface. This must be ensured regardless of whether the surface is stored tiled or linear. This is due to the potential rotation of cache line orientation from memory to cache.

The above comments also apply to the ASTC* surface format.

For packed YUV, 96 bpt, 48 bpt, and 24 bpt surface formats, additional padding is required. These surfaces require an extra row plus 16 bytes of padding at the bottom in addition to the general padding requirements.

For linear surfaces, additional padding of 64 bytes is required at the bottom of the surface. This is in addition to the padding required above.

Programming Note	
Context:	Sampling Engine Surfaces.
For SURFTYPE_BUFFER, SURFTYPE_1D, and SURFTYPE_2D non-array, non-MSAA, non-mip-mapped surfaces in linear memory, the only padding requirement is to the next aligned 64-byte boundary beyond the end of the surface. The rest of the padding requirements documented above do not apply to these surfaces.	

Programming Note	
Context:	Sampling Engine Surfaces
For all surface types other than non-mipmapped non-arrayed 2D, 1D, and Buffer, when using linear mode and surface **Height**%4 != 0, the surface must be padded with 4-(**Height**%4)*__Surface_Pitch__ bytes to avoid fetching outside of allocated memory.	

Render Target and Media Surfaces

The data port accesses data (pixels) outside of the surface if they are contained in the same cache request as pixels that are within the surface. These pixels will not be returned by the requesting message, however if these pixels lie outside of defined pages in the GTT, a GTT error will result when the cache request is processed. In order to avoid these GTT errors, "padding" at the bottom of the surface is sometimes necessary.

Address Tiling Function Introduction

When dealing with memory operands (e.g., graphics surfaces) that are inherently rectangular in nature, certain functions within the graphics device support the storage/access of the operands using alternative (tiled) memory formats to increase performance. This section describes these memory storage formats, why and when they should be used, and the behavioral mechanisms within the device to support them.

Legacy Tiling Modes:

- **TileY**: Used for most tiled surfaces when **TR_MODE**=TR_NONE.
- **TileX** : Used primarily for display surfaces.
- **TileW:** Used for Stencil surfaces.

Tiled Resource Tiling Modes

- **TileYF:** 4KB tiling mode based on TileY
- **TileYS:** 64KB tiling mode based on TileY

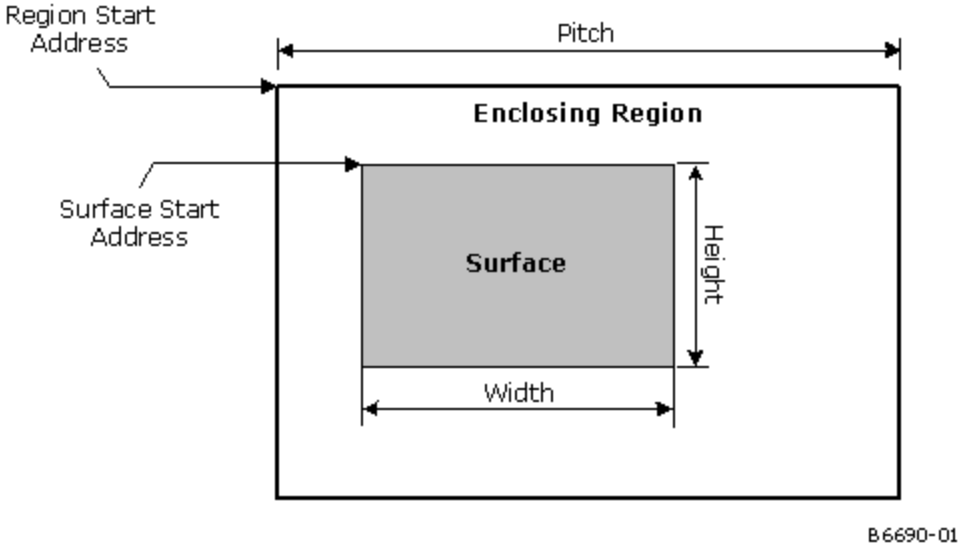
These modes are based on 4KB and 64KB tiles. The 64KB tile is made up of a 4x4 matrix of 4KB tiles. The 4KB tiles in general have a different layout as compared to the legacy modes, with the sub-mode defining the layout within the 4KB tile. The sub-modes are determined by the bits per element of the surface format. The Tiled Resource Mode field in SURFACE_STATE is used to select the new modes.

Tiled surface base addresses must be tile aligned (64KB aligned for TileYS, 4KB aligned for all other tile modes). For 1D surfaces, the base address must be 64KB aligned if **Tiled Resource Mode** is TRMODE_64KB, and 4KB aligned if **Tiled Resource Mode** is TRMODE_4KB. An exception to this tile alignment is when a SURFACE_STATE describes a single MIP within the MIP Tail of another surface, using a 64-bit or 128-bit **Surface Format**--then **Surface Base Address** can refer directly to the given MIP (e.g. to write to a non-renderable **Surface Format** by re-describing as an alternative surface).

Linear vs Tiled Storage

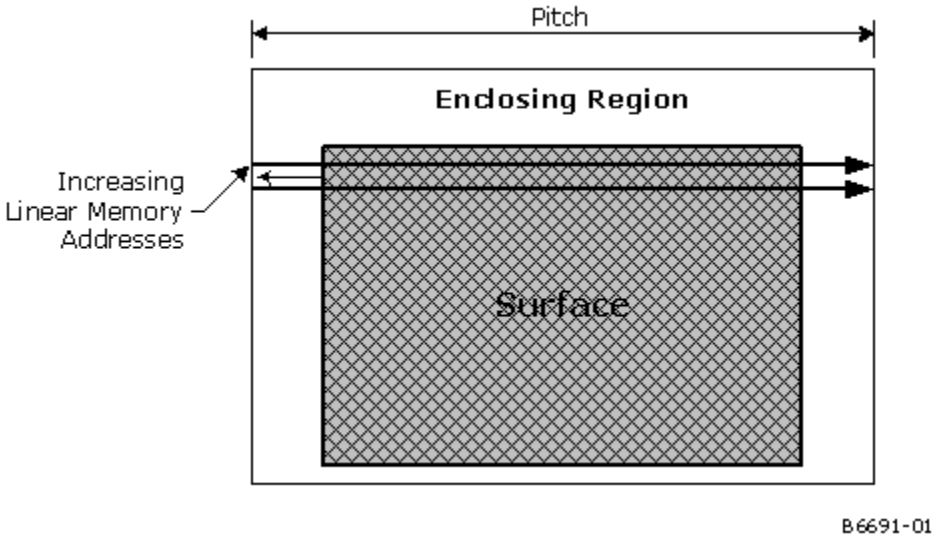
Regardless of the memory storage format, "rectangular" memory operands have a specific *width* and *height*, and are considered as residing within an enclosing rectangular region whose width is considered the *pitch* of the region and surfaces contained within. Surfaces stored within an enclosing region must have widths less than or equal to the region pitch (indeed the enclosing region may coincide exactly with the surface). *Rectangular Memory Operand Parameters* shows these parameters.

Rectangular Memory Operand Parameters



The simplest storage format is the *linear* format (see *Linear Surface Layout*), where each row of the operand is stored in sequentially increasing memory locations. If the surface width is less than the enclosing region's pitch, there will be additional memory storage between rows to accommodate the region's pitch. The pitch of the enclosing region determines the distance (in the memory address space) between vertically-adjacent operand elements (e.g., pixels, texels).

Linear Surface Layout



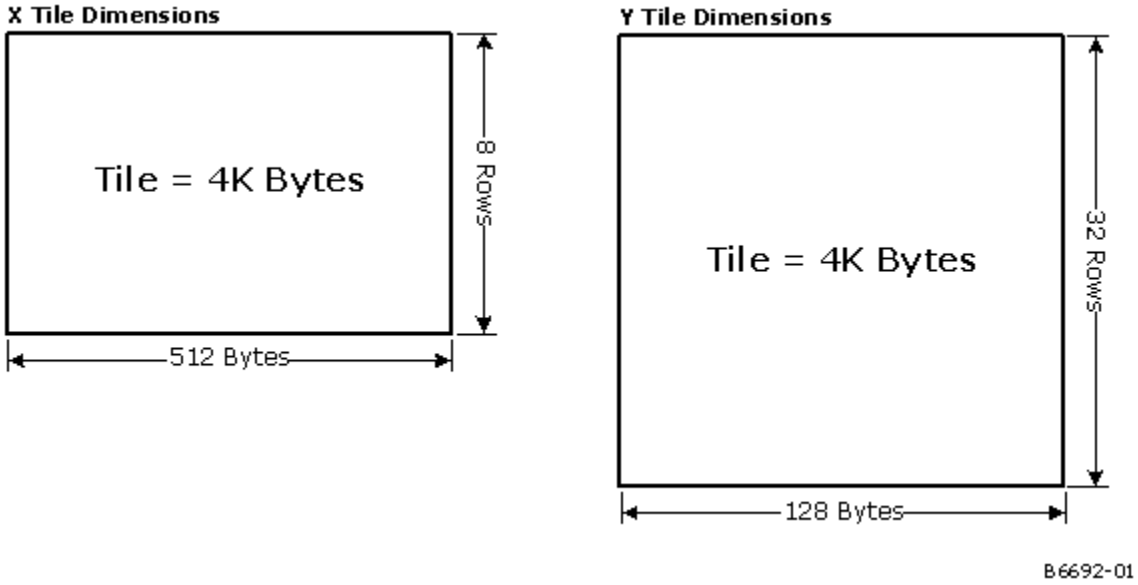
The linear format is best suited for 1-dimensional row-sequential access patterns (e.g., a display surface where each scanline is read sequentially). Here the fact that one object element may reside in a different memory page than its vertically-adjacent neighbors is not significant; all that matters is that horizontally-adjacent elements are stored contiguously. However, when a device function needs to access a 2D subregion within an operand (e.g., a read or write of a 4x4 pixel span by the 3D renderer, a read of a 2x2 texel block for bilinear filtering), having vertically-adjacent elements fall within different memory pages is

to be avoided, as the page crossings required to complete the access typically incur increased memory latencies (and therefore lower performance).

One solution to this problem is to divide the enclosing region into an array of smaller rectangular regions, called memory *tiles*. Surface elements falling within a given tile will all be stored in the same physical memory page, thus eliminating page-crossing penalties for 2D subregion accesses within a tile and thereby increasing performance.

Tiles have a fixed 4KB size and are aligned to physical DRAM page boundaries. They are either 8 rows high by 512 bytes wide or 32 rows high by 128 bytes wide (see *Memory Tile Dimensions*). Note that the dimensions of tiles are irrespective of the data contained within - e.g., a tile can hold twice as many 16-bit pixels (256 pixels/row x 8 rows = 2K pixels) than 32-bit pixels (128 pixels/row x 8 rows = 1K pixels).

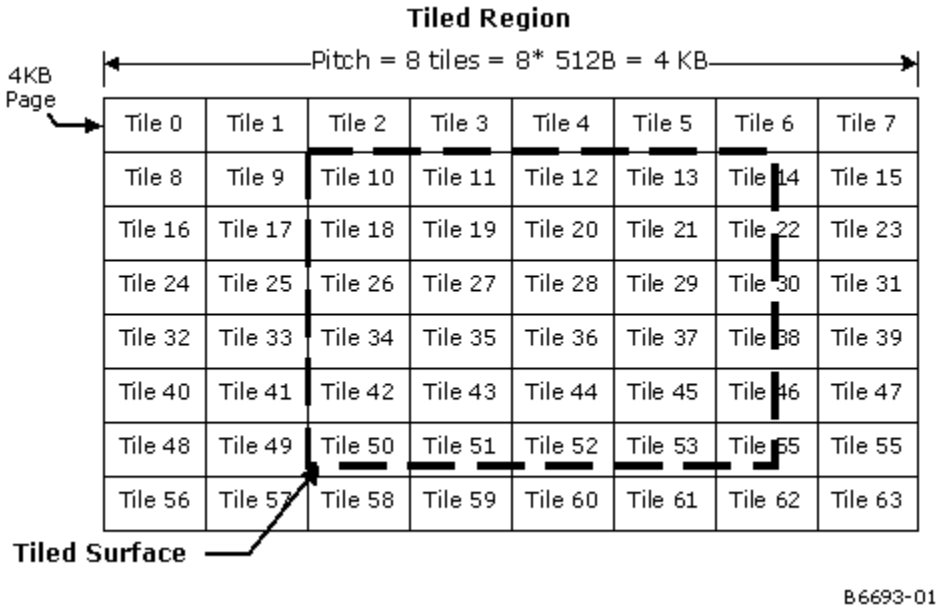
Memory Tile Dimensions



The pitch of a tiled enclosing region must be an integral number of tile widths. The 4KB tiles within a tiled region are stored sequentially in memory in row-major order.

The *Tiled Surface Layout* figure shows an example of a tiled surface located within a tiled region with a pitch of 8 tile widths (512 bytes * 8 = 4KB). Note that it is the *enclosing region* that is divided into tiles - the surface is not necessarily aligned or dimensioned to tile boundaries.

Tiled Surface Layout



Auxiliary Surfaces For Sampled Tiled Resources

For surfaces which are defined as Tiled Resources (TileYs or TileYf format), there may be auxiliary surfaces which are associated with the surface (e.g. HiZ, CCS or MCS). These auxiliary surfaces, while actually not defined as TileYs or TileYf will behave like tiled resources from the hardware perspective. It is possible for software to map and unmap tiles of auxiliary surfaces as tiles of the associated surface are mapped and unmapped. Below is a description how sampling to the mapped/unmapped tile resources is handled for the associated auxiliary surface. Normally, sampling unmapped tiles will return a NULL response to the requesting agen.

For surfaces which are defined as Tiled Resources (TileYs or TileYf format), there may be auxiliary MCS surface which is associated with the surface. These auxiliary surfaces can also be defined as either TileY, TileYs or TileYf. It is possible for software to map and unmap tiles of auxiliary surfaces as tiles of the associated surface are mapped and unmapped. Below is a description how sampling to the mapped/unmapped tile resources is handled for the associated auxiliary surface. Normally, sampling unmapped tiles will return a NULL response to the requesting agen.

MCS

A tile of MCS(Multi-Sample Control Surface) must be mapped to memory whenever MSAA surface pixels associated with the CCS tile are mapped. When all MSAA pixels associated with a MCS tile are unmapped, the MCS may be mapped or unmapped. Below is a table showing the responses for sampling to mapped and unmapped.

Table of Responses for Sampling to MSAA Tiled Resources

MSAA Surface Mapping	MCS Mapping	Sample Response
Mapped	Mapped	Normal Response
Mapped	Unmapped	Undefined Response
Unmapped	Mapped	NULL Response
Unmapped	Unmapped	NULL Response

A "NULL Response" means that the sample returned will be all 0's and the **Null Pixel Mask** (if requested) will indicate the depth pixel is Null.

Tile Formats

Multiple tile formats are supported. The following sections define and describe these formats.

Tiling formats are controlled by programming the fields Tile_Mode and Tiled_Resource_Mode in the RENDER_SURFACE_STATE.

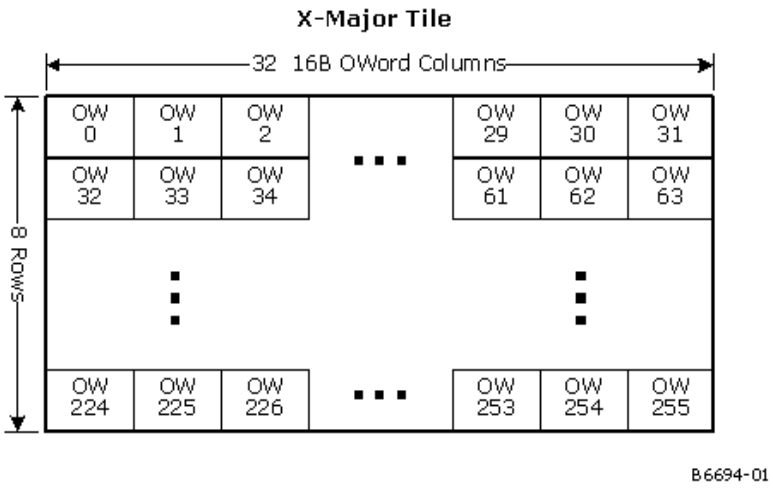
Tile-X Legacy Format

The legacy format Tile-X is a *X-Major* (row-major) storage of tile data units, as shown in the following figure. It is a 4KB tile which is subdivided into an 8-high by 32-wide array of 16-byte OWords . The selection of tile direction only impacts the internal organization of tile data, and does not affect how surfaces map onto tiles. Note that an X-major tiled region with a tile pitch of 1 tile is actually stored in a linear fashion.

Tile-X format is selected for a surface by programming the Tiled_Mode field in RENDER_SURFACE_STATE to XMAJOR.

For 3D sampling operation, a surface using Tile-X layout is generally lower performance the organization of texels in memory.

Tile X-Tile (X-Major) Layout



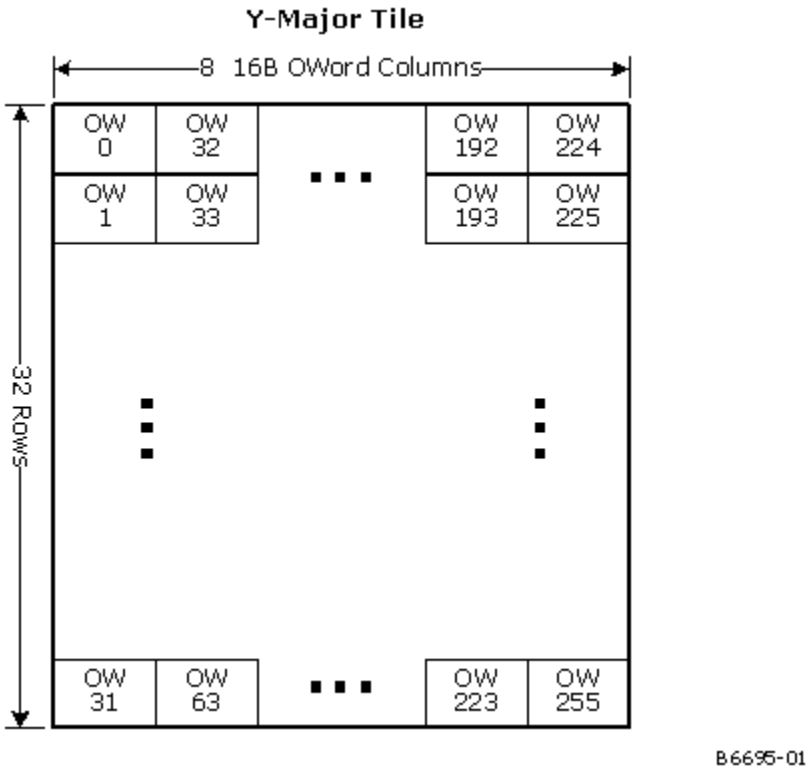
Tile-Y Legacy Format

The device supports Tile-Y legacy format which is *Y-Major* (column major) storage of tile data units, as shown in the following figure. A 4KB tile is subdivided 32-high by 8-wide array of OWords. The selection of tile direction only impacts the internal organization of tile data, and does not affect how surfaces map onto tiles.

Tile-Y surface format is selected by programming the **Tile Mode** field in RENDER_SURFACE_STATE to YMAJOR.

Note that 3D sampling of a surface in Tile-Y format is usually has higher performance due to the layout of pixels.

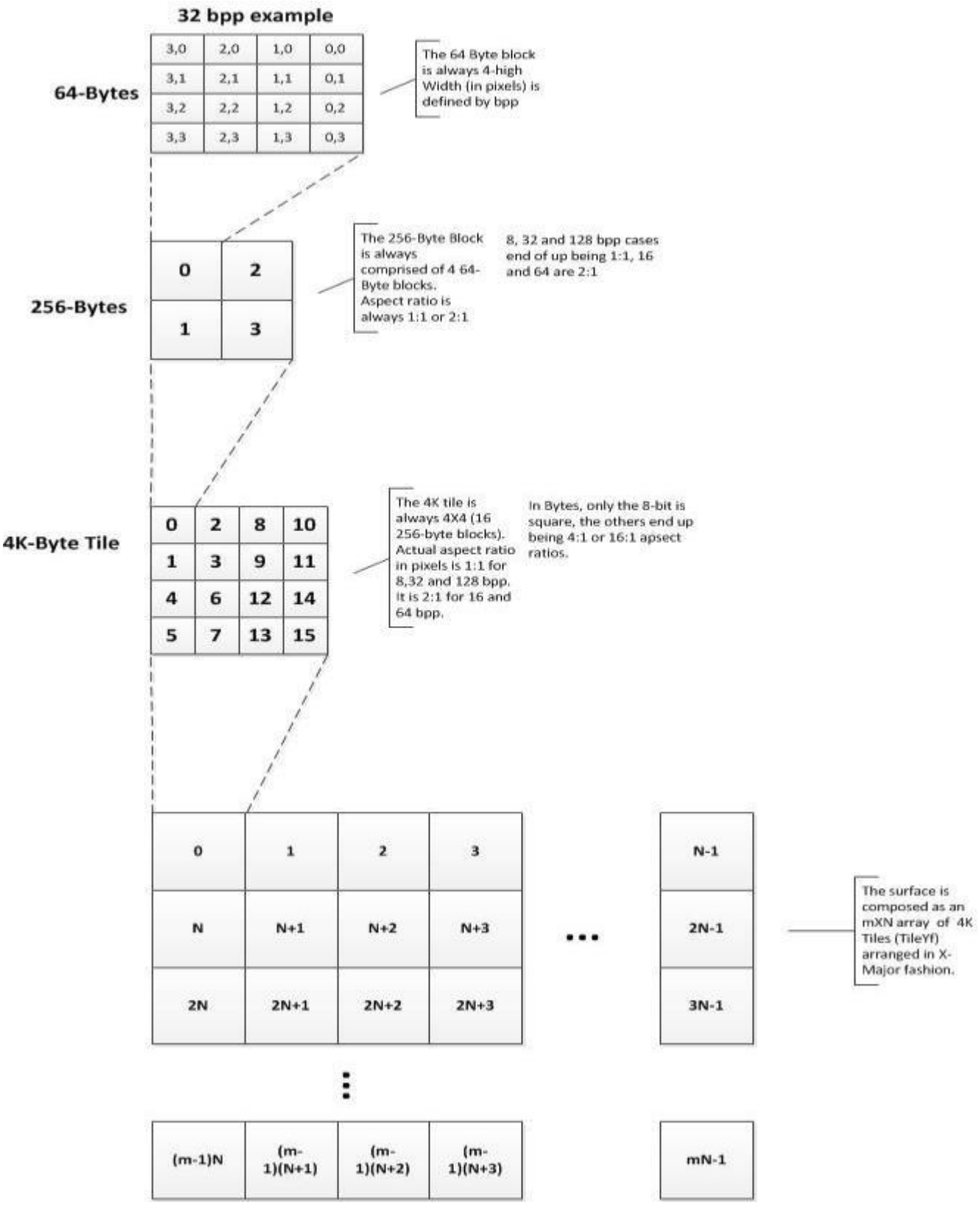
Y-Major Tile Layout



Tile-Yf Format

Tile-Yf is a 4K-Byte tile format (similar to Tile-Y), but organized in a different manner. Tile-Yf is selected by programming the Tile_Mode field in the RENDER_SURFACE_STATE to YMAJOR and the Tiled_Resource_Mode to TILEYF. The diagram below shows how pixels are mapped into the TileYf format for 2D surfaces, and it uses 32Bpp (bits per pixel) surface format as an example on a 2D surface which is N tiles wide and m tiles high. The exact aspect ratio will be dependent on the Bpp of the surface. Note that the TileYf format is identical to the TileYs up to the 4K-Byte tile size.

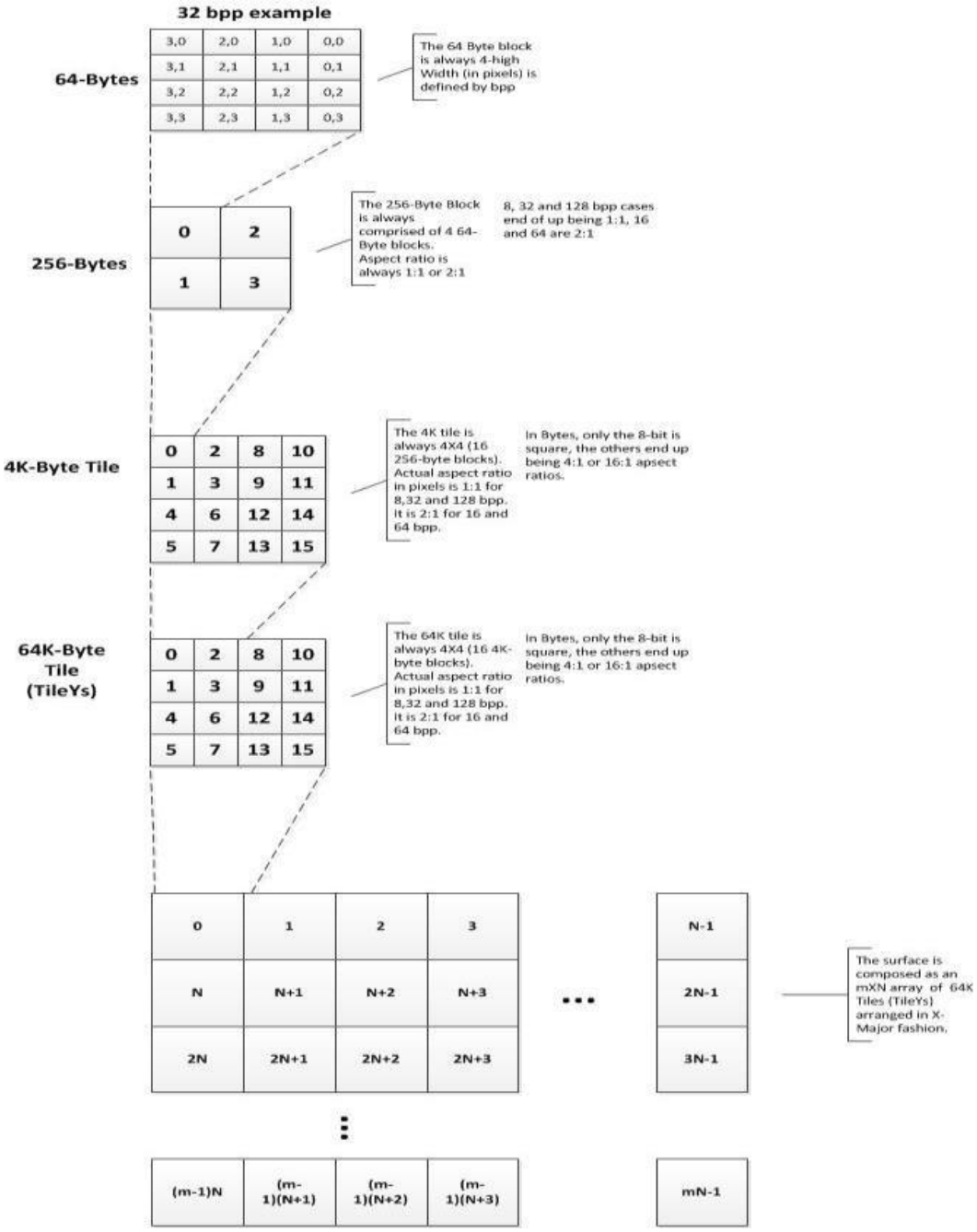
2D Tile Layout for TileYf



Tile-Ys Format

TileYs is a 64K-Byte tile size. It is enabled by programming the Tile_Mode field (in RENDER_SURFACE_STATE) to YMAJOR, and programming the Tiled_Resource_Mode to TILEYS. It is organized as shown below, and is composed of 4KByte blocks which have identical layout to the TileYf format. The diagram below shows how pixels are mapped into the TileYs format, and it uses 32Bpp (bits per pixel) surface format as an example on a 2D surface which is N tiles wide and m tiles high. The exact aspect ratio will be dependent on the Bpp of the surface.

Tile-Ys Layout



Tiling Algorithm

The following pseudo-code describes the algorithm for translating a tiled memory surface in graphics memory to an address in logical space.

The following new modes are supported for Tiled Resources (**TR_MODE** != TR_NONE) defined to enable tiled resources.

For more details about Mip Tails, see Surface Layout and Tiling in the Common Surface Formats section.

- **TileYF:** 4KB tiling mode based on TileY (Standard Tiling)
- **TileYS:** 64KB tiling mode based on TileY (Standard Tiling)

```
Inputs:
      LinearAddress(offset into regular or LT aperture in terms of bytes),
    Pitch(in terms of tiles),
            WalkY (1 for Y and 0 for X)
            WalkW (1 for W and 0 for the rest)

Static Parameters:
      TileH (Height of tile, 8 for X, 32 for Y and 64 for W),
      TileW (Width of Tile in bytes, 512 for X, 128 for Y and 64 for W)
      TileSize = TileH * TileW;
      RowSize = Pitch * TileSize;

If (Fenced) {
     LinearAddress = LinearAddress - FenceBaseAddress
     LinearAddrInTileW = LinearAddress div TileW;
     Xoffset_inTile = LinearAddress mod TileW;
       Y = LinearAddrInTileW div Pitch;
       X = LinearAddrInTileW mod Pitch + Xoffset_inTile;
}

// Internal graphics clients that access tiled memory already have the X, Y
// coordinates and can start here
YOff_Within_Tile = Y mod TileH;
XOff_Within_Tile = X mod TileW;
TileNumber_InY = Y div TileH;
TileNumber_InX = X div TileW;

      TiledOffsetY = RowSize * TileNumber_InY + TileSize * TileNumber_InX + TileH * 16 *
(XOff_Within_Tile div 16) +
             YOff_Within_Tile * 16 +
             (XOff_Within_Tile mod 16);

      TiledOffsetW = RowSize * TileNumber_InY +
             TileSize * TileNumber_InX +
             TileH * 8 * (XOff_Within_Tile div 8) +
             64 * (YOff_Within_Tile div 8) +
             32 * ((YOff_Within_Tile div 4) mod 2) +
             16 * ((XOff_Within_Tile div 4) mod 2) +
              8 * ((YOff_Within_Tile div 2) mod 2) +
              4 * ((XOff_Within_Tile div 2) mod 2) +
              2 * (YOff_Within_Tile mod 2) +
             (XOff_Within_Tile mod 2);

     TiledOffsetX = RowSize * TileNumber_InY + TileSize * TileNumber_InX + TileW *
YOff_Within_Tile + XOff_Within_Tile;

     TiledOffset = WalkW? TiledOffsetW : (WalkY? TiledOffsetY : TiledOffsetX);
```

```
        TiledAddress = Tiled? (BaseAddress + TiledOffset): (BaseAddress + Y*LinearPitch +
X);TiledAddress = (Tiled &&
              (Address Swizzling for Tiled-Surfaces == 01)) ?
              (WalkW || WalkY) ?
              (TiledAddress div 128) * 128 +
              (((TiledAddress div 64) mod 2) ^
              ((TiledAddress div 512) mod 2)) +
              (TiledAddress mod 32)
              :
              (TiledAddress div 128) * 128 +
              (((TiledAddress div 64) mod 2) ^
              ((TiledAddress div 512) mod 2)
              ((TiledAddress Div 1024) mod2) +
              (TiledAddress mod 32)
              :
             TiledAddress;
       }
```

Address Swizzling for Tiled-Surfaces is no longer used because the main memory controller has a more effective address swizzling algorithm.

For Address Swizzling for Tiled-Surfaces see ARB_MODE - Arbiter Mode Control register, ARB_CTL--Display Arbitration Control 1 and TILECTL - Tile Control register

The Y-Major tile formats have the characteristic that a surface element in an even row is located in the same aligned 64-byte cacheline as the surface element immediately below it (in the odd row). This spatial locality can be exploited to increase performance when reading 2x2 texel squares for bilinear texture filtering, or reading and writing aligned 4x4 pixel spans from the 3D Render pipeline.

On the other hand, the X-Major tile format has the characteristic that horizontally-adjacent elements are stored in sequential memory addresses. This spatial locality is advantageous when the surface is scanned in row-major order for operations like display refresh. For this reason, the Display and Overlay memory streams only support linear or X-Major tiled surfaces (Y-Major tiling is not supported by these functions). This has the side effect that 2D- or 3D-rendered surfaces must be stored in linear or X-Major tiled formats if they are to be displayed. Non-displayed surfaces, e.g., "rendered textures", can also be stored in Y-Major order.

The following Psuedo Code Describes the algorithm for mapping TileYs and TileYf Tile Address to Byte Offset within a Tile. It describes the support for 2D for both TileYs and TileYf as well as MSAA 2D For TileYs.

```
/*****************************************************************************\
   BitMask
      Used for masking single bits of x, y, z, ss# when _pdep32 instruction is
    not available
\*****************************************************************************/
enum BitMask
{
    BIT0 = 1,
    BIT1 = (1 « 1),
    BIT2 = (1 « 2),
    BIT3 = (1 « 3),
    BIT4 = (1 « 4),
    BIT5 = (1 « 5),
    BIT6 = (1 « 6),
    BIT7 = (1 « 7),
    BIT8 = (1 « 8),
    BIT9 = (1 « 9),
```

```
    BIT10 = (1 << 10),
    BIT11 = (1 << 11),
    BIT12 = (1 << 12),
    BIT13 = (1 << 13),
    BIT14 = (1 << 14),
    BIT15 = (1 << 15)
};
/*****************************************************************************\
    TileYS/TileYF constant swizzle masks w/o _pdep32 instruction

    Used to mask contiguous x/y/z/sample bit groupings before being shifted into
    their final swizzled bit positions
\*****************************************************************************/
// used for fallback 'manual' bit shifting
static const UINT16 xMaskBits5_4  = 0x0030;
static const UINT16 xMaskBits3_0  = 0x000F;
static const UINT16 yMaskBits4_0  = 0x001F;
static const UINT16 yMaskBits3_0  = 0x000F;
static const UINT16 yMaskBits2_0  = 0x0007;
static const UINT16 yMaskBits1_0  = 0x0003;
static const UINT16 SampleMask3_0 = 0x000F;
static const UINT16 SampleMask2_0 = 0x0007;
static const UINT16 SampleMask1_0 = 0x0003;
static const UINT16 SampleMask0   = 0x0001;

/*****************************************************************************\
    TileYS 2D Tile address swizzling functions w/o _pdep32
\*****************************************************************************/
/*
 ____________________________________________________________________________
|    Num    | Bits per element |           Tiled element offset bits           |
|  Samples  |                  |15|14|13|12|11|10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0|
|___________|__________________|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|     1x    |     64 & 128     |x9|y5|x8|y4|x7|y3|x6|y2|x5|x4|y1|y0|x3|x2|x1|x0|
|           |     16 & 32      |x8|y6|x7|y5|x6|y4|x5|y3|x4|y2|y1|y0|x3|x2|x1|x0|
|           |        8         |x7|y7|x6|y6|x5|y5|x4|y4|y3|y2|y1|y0|x3|x2|x1|x0|
*/
UINT16 TileYS2dElementOffset64_128bpe(UINT16 x, UINT16 y)
{
    UINT16 xSwizzle;
    UINT16 ySwizzle;

    // shift bits in x and y to their respective TileYS swizzled bit locations
    xSwizzle = ((BIT9 & x) << 6) |
               ((BIT8 & x) << 5) |
               ((BIT7 & x) << 4) |
               ((BIT6 & x) << 3) |
               ((xMaskBits5_4 & x) << 2) | // shift to bit positions 7..6
               (xMaskBits3_0 & x);

    ySwizzle = ((BIT5 & y) << 9) |
               ((BIT4 & y) << 8) |
               ((BIT3 & y) << 7) |
               ((BIT2 & y) << 6) |
               ((yMaskBits1_0 & y) << 4);   // shift to bit positions 5..4

    // OR the swizzled bit positions for final offset within a tile
    return xSwizzle | ySwizzle;
}

UINT16 TileYS2dElementOffset16_32bpe(UINT16 x, UINT16 y)
{
    UINT16 xSwizzle;
    UINT16 ySwizzle;
```

```
    // shift bits in x and y to their respective TileYS swizzled bit locations
    xSwizzle = ((BIT8 & x) « 7) |
               ((BIT7 & x) « 6) |
               ((BIT6 & x) « 5) |
               ((BIT5 & x) « 4) |
               ((BIT4 & x) « 3) |
               (xMaskBits3_0 & x);

    ySwizzle = ((BIT6 & y) « 8) |
               ((BIT5 & y) « 7) |
               ((BIT4 & y) « 6) |
               ((BIT3 & y) « 5) |
               ((yMaskBits2_0 & y) « 4); // shift to bit positions 6..4

    // OR the swizzled bit positions for final offset within a tile
    return xSwizzle | ySwizzle;
}

UINT16 TileYS2dElementOffset8bpe(UINT16 x, UINT16 y)
{
    UINT16 xSwizzle;
    UINT16 ySwizzle;

    // shift bits in x and y to their respective TileYS swizzled bit locations
    xSwizzle = ((BIT7 & x) « 8) |
               ((BIT6 & x) « 7) |
               ((BIT5 & x) « 6) |
               ((BIT4 & x) « 5) |
               (xMaskBits3_0 & x);

    ySwizzle = ((BIT7 & y) « 7) |
               ((BIT6 & y) « 6) |
               ((BIT5 & y) « 5) |
               ((yMaskBits4_0 & y) « 4); // shift to bit positions 8..4

    // OR the swizzled bit positions for final offset within a tile
    return xSwizzle | ySwizzle;
}

/*****************************************************************************\
    TileYS 2D MSAA Tile address swizzling functions w/o _pdep32
\*****************************************************************************/
/*
 ____________________________________________________________________________
|    Num    | Bits per element |           Tiled element offset bits           |
|  Samples  |                  |15 |14|13|12|11|10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0|
|___________|__________________|___|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|     2x    |     64 & 128     |ss0|y5|x8|y4|x7|y3|x6|y2|x5|x4|y1|y0|x3|x2|x1|x0|
|           |     16 & 32      |ss0|y6|x7|y5|x6|y4|x5|y3|x4|y2|y1|y0|x3|x2|x1|x0|
|           |        8         |ss0|y7|x6|y6|x5|y5|x4|y4|y3|y2|y1|y0|x3|x2|x1|x0|
*/
UINT16 TileYS2xMsaaElementOffset64_128bpe(UINT16 x, UINT16 y, UINT16 sample)
{
    UINT16 xSwizzle;
    UINT16 ySwizzle;
    UINT16 SampleSwizzle;

    // shift bits in x, y, and sample to their respective TileYS MSAA swizzled bit locations
    xSwizzle = ((BIT8 & x) « 5) |                  // shift to bit position 13
               ((BIT7 & x) « 4) |                  // shift to bit position 11
               ((BIT6 & x) « 3) |                  // shift to bit position 9
               ((xMaskBits5_4 & x) « 2) |          // shift to bit positions 7..6
               (xMaskBits3_0 & x);                  // leave in bits 3..0
```

```
    ySwizzle = ((BIT5 & y) « 9) |                  // shift to bit position 14
               ((BIT4 & y) « 8) |                  // shift to bit position 12
               ((BIT3 & y) « 7) |                  // shift to bit position 10
               ((BIT2 & y) « 6) |                  // shift to bit position 8
               ((yMaskBits1_0 & y) « 4);           // shift to bit positions 5..4

    SampleSwizzle = (sample && SampleMask0) « 15;  // shift to bit position 15

    // OR the swizzled bit positions for final offset within a tile
    return SampleSwizzle | xSwizzle | ySwizzle;
}

UINT16 TileYS2xMsaaElementOffset16_32bpe(UINT16 x, UINT16 y, UINT16 sample)
{
    UINT16 xSwizzle;
    UINT16 ySwizzle;
    UINT16 SampleSwizzle;

    // shift bits in x, y, and sample to their respective TileYS MSAA swizzled bit locations
    xSwizzle = ((BIT7 & x) « 6) |                  // shift to bit position 13
               ((BIT6 & x) « 7) |                  // shift to bit position 11
               ((BIT5 & x) « 6) |                  // shift to bit position 9
               ((BIT4 & x) « 5) |                  // shift to bit position 7
               (xMaskBits3_0 & x);                  // leave in bits 3..0

    ySwizzle = ((BIT6 & y) « 8) |                  // shift to bit position 14
               ((BIT5 & y) « 7) |                  // shift to bit position 12
               ((BIT4 & y) « 6) |                  // shift to bit position 10
               ((BIT3 & y) « 5) |                  // shift to bit position 8
               ((yMaskBits2_0 & y) « 4);           // shift to bit positions 6..4

    SampleSwizzle = (sample && SampleMask0) « 15;  // shift to bit position 15

    // OR the swizzled bit positions for final offset within a tile
    return SampleSwizzle | xSwizzle | ySwizzle;
}

UINT16 TileYS2xMsaaElementOffset8bpe(UINT16 x, UINT16 y, UINT16 sample)
{
    UINT16 xSwizzle;
    UINT16 ySwizzle;
    UINT16 SampleSwizzle;

    // shift bits in x, y, and sample to their respective TileYS MSAA swizzled bit locations
    xSwizzle = ((BIT6 & x) « 7) |                  // shift to bit position 13
               ((BIT5 & x) « 6) |                  // shift to bit position 11
               ((BIT4 & x) « 5) |                  // shift to bit position 9
               (xMaskBits3_0 & x);                  // leave in bits 3..0

    ySwizzle = ((BIT7 & y) « 7) |                  // shift to bit position 14
               ((BIT6 & y) « 6) |                  // shift to bit position 12
               ((BIT5 & y) « 5) |                  // shift to bit position 10
               ((yMaskBits4_0 & y) « 4);           // shift to bit positions 8..4

    SampleSwizzle = (sample && SampleMask0) « 15;  // shift to bit position 15

    // OR the swizzled bit positions for final offset within a tile
    return SampleSwizzle | xSwizzle | ySwizzle;
}

/*
 _____________________________________________________________________________
|    Num    | Bits per element |          Tiled element offset bits            |
```

```
|  Samples  |                   |15 |14 |13|12|11|10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0|
|___________|___________________|___|___|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|     4x    |     64 & 128      |ss1|ss0|x8|y4|x7|y3|x6|y2|x5|x4|y1|y0|x3|x2|x1|x0|
|           |     16 & 32       |ss1|ss0|x7|y5|x6|y4|x5|y3|x4|y2|y1|y0|x3|x2|x1|x0|
|           |        8          |ss1|ss0|x6|y6|x5|y5|x4|y4|y3|y2|y1|y0|x3|x2|x1|x0|
*/
UINT16 TileYS4xMsaaElementOffset64_128bpe(UINT16 x, UINT16 y, UINT16 sample)
{
    UINT16 xSwizzle;
    UINT16 ySwizzle;
    UINT16 SampleSwizzle;

    // shift bits in x, y, and sample to their respective TileYS MSAA swizzled bit locations
    xSwizzle = ((BIT8 & x) « 5) |                  // shift to bit position 13
               ((BIT7 & x) « 4) |                  // shift to bit position 11
               ((BIT6 & x) « 3) |                  // shift to bit position 9
               ((xMaskBits5_4 & x) « 2) |          // shift to bit positions 7..6
               (xMaskBits3_0 & x);                  // leave in bits 3..0

    ySwizzle = ((BIT4 & y) « 8) |                  // shift to bit position 12
               ((BIT3 & y) « 7) |                  // shift to bit position 10
               ((BIT2 & y) « 6) |                  // shift to bit position 8
               ((yMaskBits1_0 & y) « 4);           // shift to bit positions 5..4

    SampleSwizzle = (sample && SampleMask1_0) « 14;// shift to bit positions 15..14

    // OR the swizzled bit positions for final offset within a tile
    return SampleSwizzle | xSwizzle | ySwizzle;
}

UINT16 TileYS4xMsaaElementOffset16_32bpe(UINT16 x, UINT16 y, UINT16 sample)
{
    UINT16 xSwizzle;
    UINT16 ySwizzle;
    UINT16 SampleSwizzle;

    // shift bits in x, y, and sample to their respective TileYS MSAA swizzled bit locations
    xSwizzle = ((BIT7 & x) « 6) |                  // shift to bit position 13
               ((BIT6 & x) « 7) |                  // shift to bit position 11
               ((BIT5 & x) « 6) |                  // shift to bit position 9
               ((BIT4 & x) « 5) |                  // shift to bit position 7
               (xMaskBits3_0 & x);                  // leave in bits 3..0

    ySwizzle = ((BIT5 & y) « 7) |                  // shift to bit position 12
               ((BIT4 & y) « 6) |                  // shift to bit position 10
               ((BIT3 & y) « 5) |                  // shift to bit position 8
               ((yMaskBits2_0 & y) « 4);           // shift to bit positions 6..4

    SampleSwizzle = (sample && SampleMask1_0) « 14;// shift to bit positions 15..14

    // OR the swizzled bit positions for final offset within a tile
    return SampleSwizzle | xSwizzle | ySwizzle;
}

UINT16 TileYS4xMsaaElementOffset8bpe(UINT16 x, UINT16 y, UINT16 sample)
{
    UINT16 xSwizzle;
    UINT16 ySwizzle;
    UINT16 SampleSwizzle;

    // shift bits in x, y, and sample to their respective TileYS MSAA swizzled bit locations
    xSwizzle = ((BIT6 & x) « 7) |                  // shift to bit position 13
               ((BIT5 & x) « 6) |                  // shift to bit position 11
               ((BIT4 & x) « 5) |                  // shift to bit position 9
```

```
                (xMaskBits3_0 & x);                  // leave in bits 3..0

    ySwizzle = ((BIT6 & y) « 6) |                   // shift to bit position 12
               ((BIT5 & y) « 5) |                   // shift to bit position 10
               ((yMaskBits4_0 & y) « 4);            // shift to bit positions 8..4

    SampleSwizzle = (sample && SampleMask1_0) « 14;// shift to bit positions 15..14

    // OR the swizzled bit positions for final offset within a tile
    return SampleSwizzle | xSwizzle | ySwizzle;
}

/*
 ____________________________________________________________________________
|    Num     | Bits per element |           Tiled element offset bits          |
|  Samples   |                  |15 |14 |13 |12|11|10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0|
|____________________________________|___|___|___|__|__|__|__|__|__|__|__|__|__|__|
|     8x     |     64 & 128     |ss2|ss1|ss0|y4|x7|y3|x6|y2|x5|x4|y1|y0|x3|x2|x1|x0|
|            |     16 & 32      |ss2|ss1|ss0|y5|x6|y4|x5|y3|x4|y2|y1|y0|x3|x2|x1|x0|
|            |        8         |ss2|ss1|ss0|y6|x5|y5|x4|y4|y3|y2|y1|y0|x3|x2|x1|x0|
*/
UINT16 TileYS8xMsaaElementOffset64_128bpe(UINT16 x, UINT16 y, UINT16 sample)
{
    UINT16 xSwizzle;
    UINT16 ySwizzle;
    UINT16 SampleSwizzle;

    // shift bits in x, y, and sample to their respective TileYS MSAA swizzled bit locations
    xSwizzle = ((BIT7 & x) « 4) |                   // shift to bit position 11
               ((BIT6 & x) « 3) |                   // shift to bit position 9
               ((xMaskBits5_4 & x) « 2) |           // shift to bit positions 7..6
               (xMaskBits3_0 & x);                  // leave in bits 3..0

    ySwizzle = ((BIT4 & y) « 8) |                   // shift to bit position 12
               ((BIT3 & y) « 7) |                   // shift to bit position 10
               ((BIT2 & y) « 6) |                   // shift to bit position 8
               ((yMaskBits1_0 & y) « 4);            // shift to bit positions 5..4

    SampleSwizzle = (sample && SampleMask2_0) « 13;// shift to bit positions 15..13

    // OR the swizzled bit positions for final offset within a tile
    return SampleSwizzle | xSwizzle | ySwizzle;
}

UINT16 TileYS8xMsaaElementOffset16_32bpe(UINT16 x, UINT16 y, UINT16 sample)
{
    UINT16 xSwizzle;
    UINT16 ySwizzle;
    UINT16 SampleSwizzle;

    // shift bits in x, y, and sample to their respective TileYS MSAA swizzled bit locations
    xSwizzle = ((BIT6 & x) « 7) |                   // shift to bit position 11
               ((BIT5 & x) « 6) |                   // shift to bit position 9
               ((BIT4 & x) « 5) |                   // shift to bit position 7
               (xMaskBits3_0 & x);                  // leave in bits 3..0

    ySwizzle = ((BIT5 & y) « 7) |                   // shift to bit position 12
               ((BIT4 & y) « 6) |                   // shift to bit position 10
               ((BIT3 & y) « 5) |                   // shift to bit position 8
               ((yMaskBits2_0 & y) « 4);            // shift to bit positions 6..4

    SampleSwizzle = (sample && SampleMask2_0) « 13;// shift to bit positions 15..13

    // OR the swizzled bit positions for final offset within a tile
```

```
    return SampleSwizzle | xSwizzle | ySwizzle;
}

UINT16 TileYS8xMsaaElementOffset8bpe(UINT16 x, UINT16 y, UINT16 sample)
{
    UINT16 xSwizzle;
    UINT16 ySwizzle;
    UINT16 SampleSwizzle;

    // shift bits in x, y, and sample to their respective TileYS MSAA swizzled bit locations
    xSwizzle = ((BIT5 & x) « 6) |                  // shift to bit position 11
               ((BIT4 & x) « 5) |                  // shift to bit position 9
               (xMaskBits3_0 & x);                  // leave in bits 3..0

    ySwizzle = ((BIT6 & y) « 6) |                  // shift to bit position 12
               ((BIT5 & y) « 5) |                  // shift to bit position 10
               ((yMaskBits4_0 & y) « 4);           // shift to bit positions 8..4

    SampleSwizzle = (sample && SampleMask2_0) « 13;// shift to bit positions 15..13

    // OR the swizzled bit positions for final offset within a tile
    return SampleSwizzle | xSwizzle | ySwizzle;
}

/*
 ____________________________________________________________________________________
|    Num    | Bits per element |           Tiled element offset bits                |
|  Samples  |                  |15 |14 |13 |12 |11|10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0|
|___________|__________________|___|___|___|___|__|__|__|__|__|__|__|__|__|__|__|__|
|    16x    |     64 & 128     |ss3|ss2|ss1|ss0|x7|y3|x6|y2|x5|x4|y1|y0|x3|x2|x1|x0|
|           |     16 & 32      |ss3|ss2|ss1|ss0|x6|y4|x5|y3|x4|y2|y1|y0|x3|x2|x1|x0|
|           |        8         |ss3|ss2|ss1|ss0|x5|y5|x4|y4|y3|y2|y1|y0|x3|x2|x1|x0|
*/
UINT16 TileYS16xMsaaElementOffset64_128bpe(UINT16 x, UINT16 y, UINT16 sample)
{
    UINT16 xSwizzle;
    UINT16 ySwizzle;
    UINT16 SampleSwizzle;

    // shift bits in x, y, and sample to their respective TileYS MSAA swizzled bit locations
    xSwizzle = ((BIT7 & x) « 4) |                  // shift to bit position 11
               ((BIT6 & x) « 3) |                  // shift to bit position 9
               ((xMaskBits5_4 & x) « 2) |          // shift to bit positions 7..6
               (xMaskBits3_0 & x);                  // leave in bits 3..0

    ySwizzle = ((BIT3 & y) « 7) |                  // shift to bit position 10
               ((BIT2 & y) « 6) |                  // shift to bit position 8
               ((yMaskBits1_0 & y) « 4);           // shift to bit positions 5..4

    SampleSwizzle = (sample && SampleMask3_0) « 12;// shift to bit positions 15..12

    // OR the swizzled bit positions for final offset within a tile
    return SampleSwizzle | xSwizzle | ySwizzle;
}

UINT16 TileYS16xMsaaElementOffset16_32bpe(UINT16 x, UINT16 y, UINT16 sample)
{
    UINT16 xSwizzle;
    UINT16 ySwizzle;
    UINT16 SampleSwizzle;

    // shift bits in x, y, and sample to their respective TileYS MSAA swizzled bit locations
    xSwizzle = ((BIT6 & x) « 7) |                  // shift to bit position 11
               ((BIT5 & x) « 6) |                  // shift to bit position 9
```

```
                ((BIT4 & x) « 5) |                  // shift to bit position 7
                (xMaskBits3_0 & x);                  // leave in bits 3..0

    ySwizzle = ((BIT4 & y) « 6) |                   // shift to bit position 10
                ((BIT3 & y) « 5) |                  // shift to bit position 8
                ((yMaskBits2_0 & y) « 4);           // shift to bit positions 6..4

    SampleSwizzle = (sample && SampleMask3_0) « 12;// shift to bit positions 15..12

    // OR the swizzled bit positions for final offset within a tile
    return SampleSwizzle | xSwizzle | ySwizzle;
}

UINT16 TileYS16xMsaaElementOffset8bpe(UINT16 x, UINT16 y, UINT16 sample)
{
    UINT16 xSwizzle;
    UINT16 ySwizzle;
    UINT16 SampleSwizzle;

    // shift bits in x, y, and sample to their respective TileYS MSAA swizzled bit locations
    xSwizzle = ((BIT5 & x) « 6) |                   // shift to bit position 11
                ((BIT4 & x) « 5) |                  // shift to bit position 9
                (xMaskBits3_0 & x);                  // leave in bits 3..0

    ySwizzle = ((BIT5 & y) « 5) |                   // shift to bit position 10
                ((yMaskBits4_0 & y) « 4);           // shift to bit positions 8..4

    SampleSwizzle = (sample && SampleMask3_0) « 12;// shift to bit positions 15..12

    // OR the swizzled bit positions for final offset within a tile
    return SampleSwizzle | xSwizzle | ySwizzle;
}


/*****************************************************************************\
    TileYF 2D Tile address swizzling functions w/o _pdep32
\*****************************************************************************/
/*
 ____________________________________________________________________________
|     Num     | Bits per element |           Tiled element offset bits          |
|  Samples    |                  |15|14|13|12|11|10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0|
|_____________|__________________|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|     1x      |     64 & 128     |  |  |  |  |x7|y3|x6|y2|x5|x4|y1|y0|x3|x2|x1|x0|
|             |     16 & 32      |  |  |  |  |x6|y4|x5|y3|x4|y2|y1|y0|x3|x2|x1|x0|
|             |        8         |  |  |  |  |x5|y5|x4|y4|y3|y2|y1|y0|x3|x2|x1|x0|
*/
UINT16 TileYF2dElementOffset64_128bpe(UINT16 x, UINT16 y)
{
    UINT16 xSwizzle;
    UINT16 ySwizzle;

    // shift bits in x and y to their respective TileYF swizzled bit locations
    xSwizzle = ((BIT7 & x) « 4) |
                ((BIT6 & x) « 3) |
                ((xMaskBits5_4 & x) « 2) | // shift to bit positions 7..6
                (xMaskBits3_0 & x);

    ySwizzle = ((BIT3 & y) « 7) |
                ((BIT2 & y) « 6) |
                ((yMaskBits1_0 & y) « 4);   // shift to bit positions 5..4

    // OR the swizzled bit positions for final offset within a tile
    return xSwizzle | ySwizzle;
}
```

```
UINT16 TileYF2dElementOffset16_32bpe(UINT16 x, UINT16 y)
{
    UINT16 xSwizzle;
    UINT16 ySwizzle;

    // shift bits in x and y to their respective TileYF swizzled bit locations
    xSwizzle = ((BIT6 & x) « 5) |
               ((BIT5 & x) « 4) |
               ((BIT4 & x) « 3) |
               (xMaskBits3_0 & x);

    ySwizzle = ((BIT4 & y) « 6) |
               ((BIT3 & y) « 5) |
               ((yMaskBits2_0 & y) « 4); // shift to bit positions 6..4

    // OR the swizzled bit positions for final offset within a tile
    return xSwizzle | ySwizzle;
}

UINT16 TileYF2dElementOffset8bpe(UINT16 x, UINT16 y)
{
    UINT16 xSwizzle;
    UINT16 ySwizzle;

    // shift bits in x and y to their respective TileYF swizzled bit locations
    xSwizzle = ((BIT5 & x) « 6) |
               ((BIT4 & x) « 5) |
               (xMaskBits3_0 & x);

    ySwizzle = ((BIT5 & y) « 5) |
               ((yMaskBits4_0 & y) « 4); // shift to bit positions 8..4

    // OR the swizzled bit positions for final offset within a tile
    return xSwizzle | ySwizzle;
}
```

Tiling Support

The rearrangement of the surface elements in memory must be accounted for in device functions operating upon tiled surfaces. (Note that not all device functions that access memory support tiled formats). This requires either the modification of an element's linear memory address or an alternate formula to convert an element's X,Y coordinates into a tiled memory address.

However, before tiled-address-generation can take place, some mechanism must be used to determine whether the surface elements accessed fall in a linear or tiled region of memory, and if tiled, what the tile region pitch is, and whether the tiled region uses X-Major or Y-Major format. There are two mechanisms by which this detection takes place: (a) an implicit method by detecting that the pre-tiled (linear) address falls within a "fenced" tiled region, or (b) by an explicit specification of tiling parameters for surface operands (i.e., parameters included in surface-defining instructions).

The following table identifies the tiling-detection mechanisms that are supported by the various memory streams.

Access Path	Tiling-Detection Mechanisms Supported
Processor access through the Graphics Memory Aperture	Fenced Regions
3D Render (Color/Depth Buffer access)	Explicit Surface Parameters
Sampled Surfaces	Explicit Surface Parameters
Blt operands	Explicit Surface Parameters
Display and Overlay Surfaces	Explicit Surface Parameters

Tiled (Fenced) Regions

The only mechanism to support the access of surfaces in tiled format by the host or external graphics client is to place them within "fenced" tiled regions within Graphics Memory. A fenced region is a block of Graphics Memory specified using one of the sixteen FENCE device registers. (See *Memory Interface Registers* for details). Surfaces contained within a fenced region are considered tiled from an external access point of view. Note that fences cannot be used to untile surfaces in the PGM_Address space since external devices cannot access PGM_Address space. Even if these surfaces (or any surfaces accessed by an internal graphics client) fall within a region covered by an enabled fence register, that enable will be effectively masked during the internal graphics client access. Only the explicit surface parameters described in the next section can be used to tile surfaces being accessed by the internal graphics clients.

Tiled Surface Parameters

Internal device functions require explicit specification of surface tiling parameters via information passed in commands and state. This capability is provided to limit the reliance on the fixed number of fence regions.

The following table lists the surface tiling parameters that can be specified for 3D Render surfaces (Color Buffer, Depth Buffer, Textures, etc.) via SURFACE_STATE.

Surface Parameter	Description
Tiled Surface	If ENABLED, the surface is stored in a tiled format. If DISABLED, the surface is stored in a linear format.
Tile Walk	If Tiled Surface is ENABLED, this parameter specifies whether the tiled surface is stored in Y-Major or X-Major tile format.
Base Address	Additional restrictions apply to the base address of a Tiled Surface vs. that of a linear surface.
Pitch	Pitch of the surface. Note that, if the surface is tiled, this pitch must be a multiple of the tile width.

Tiled Surface Restrictions

Additional restrictions apply to the Base Address and Pitch of a surface that is tiled. In addition, restrictions for tiling via SURFACE_STATE are subtly different from those for tiling via fence regions. The most restricted surfaces are those that will be accessed both by the host (via fence) and by internal device functions. An example of such a surface is a tiled texture that is initialized by the CPU and then sampled by the device.

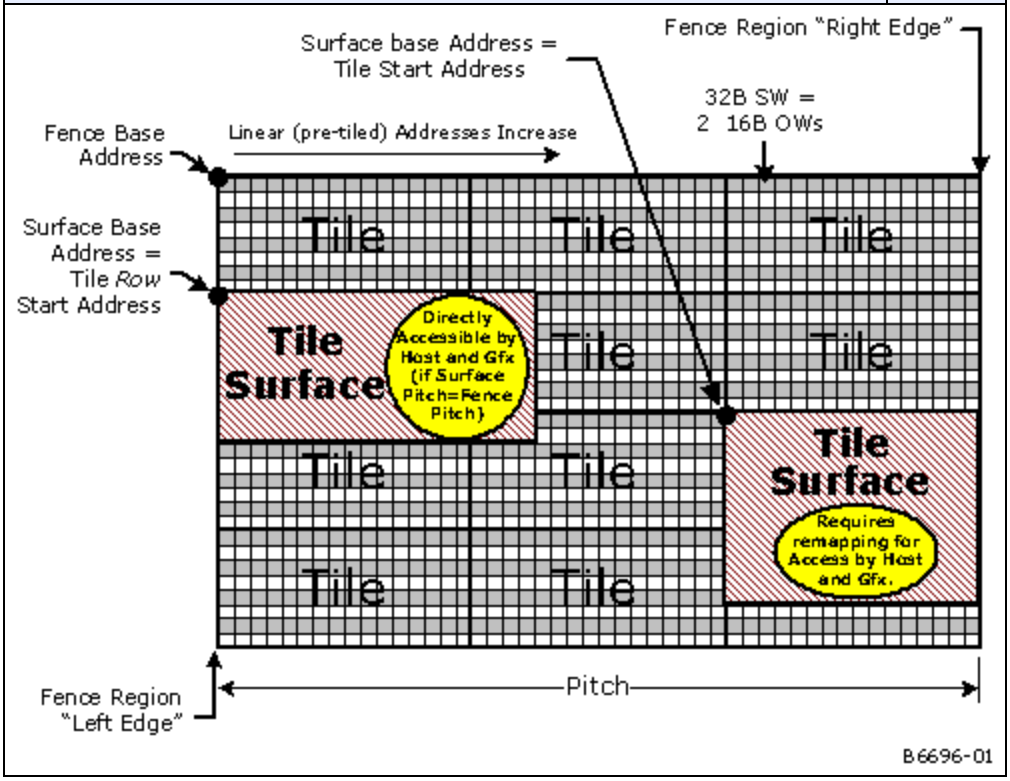
The tiling algorithm for internal device functions is different from that of fence regions. Internal device functions always specify tiling in terms of a surface. The surface must have a base address, and this base address is not subject to the tiling algorithm. Only *offsets* from the base address (as calculated by X, Y addressing within the surface) are transformed through tiling. The base address of the surface must therefore be 4KB-aligned. This forces the 4KB tiles of the tiling algorithm to exactly align with 4KB device pages once the tiling algorithm has been applied to the offset. The width of a surface must be less than or equal to the surface pitch. There are additional considerations for surfaces that are also accessed by the host (via a fence region).

Fence regions have no base address per se. Host linear addresses that fall in a fence region are translated in their entirety by the tiling algorithm. It is as if the surface being tiled by the fence region has a base address in graphics memory equal to the fence base address, and all accesses of the surfaces are (possibly quite large) offsets from the fence base address. Fence regions have a virtual "left edge" aligned with the fence base address, and a "right edge" that results from adding the fence pitch to the "left edge". Surfaces in the fence region must not straddle these boundaries.

Base addresses of surfaces that are to be accessed both by an internal graphics client and by the host have the tightest restrictions. In order for the surface to be accessed without GTT re-mapping, the surface base address (as set in SURFACE_STATE) must be a "Tile Row Start Address" (TRSA). The first address in each tile row of the fence region is a Tile Row Start Address. The first TRSA is the fence base address. Each TRSA can be generated by adding an integral multiple of the row size to the fence base address. The row size is simply the fence pitch in tiles multiplied by 4KB (the size of a tile.)

Tiled Surface Placement

Project:	
Security:	



The pitch in SURFACE_STATE must be set equal to the pitch of the fence that will be used by the host to access the surface if the same GTT mapping will be used for each access. If the pitches differ, a different GTT mapping must be used to eliminate the "extra" tiles (4KB memory pages) that exist in the excess rows at the right side of the larger pitch. Obviously no part of the surface that will be accessed can lie in pages that exist only in one mapping but not the other. The new GTT mapping can be done manually by SW between the time the host writes the surface and the device reads it, or it can be accomplished by arranging for the client to use a different GTT than the host (the PPGTT -- see *Logical Memory Mapping* below).

The width of the surface (as set in SURFACE_STATE) must be less than or equal to both the surface pitch and the fence pitch in any scenario where a surface will be accessed by both the host and an internal graphics client. Changing the GTT mapping will not help if this restriction is violated.

Surface Access	Base Address	Pitch	Width	Tile "Walk"
Host only	No restriction	Integral multiple of tile size <= 256KB	Must be <= Fence Pitch	No restriction
Client only	4KB-aligned	Integral multiple of tile size <= 256KB	Must be <= Surface Pitch	Restrictions imposed by the client (see Per Stream Tile Format Support)
Host and Client, No GTT Remapping	Must be TRSA	Fence Pitch = Surface Pitch = integral multiple of tile size <= 256KB	Width <= Pitch	Surface Walk must meet client restriction, Fence Walk = Surface Walk
Host and Client, GTT Remapping	4KB-aligned for client (will be tile-aligned for host)	Both must be Integral multiple of tile size <=128KB, but not necessarily the same	Width <= Min(Surface Pitch, Fence Pitch)	Surface Walk must meet client restriction, Fence Walk = Surface Walk

Per-Stream Tile Format Support

<table>
<tr><th>MI Client</th><th>Tile Formats Supported</th></tr>
<tr><td>CPU Read/Write</td><td>All</td></tr>
<tr><td>Display/Overlay</td><td>Y-Major not supported
X-Major required for Async Flips</td></tr>
<tr><td>Blt</td><td>Linear and X-Major only
No Y-Major support</td></tr>
<tr><td>3D Sampler</td><td>All Combinations of TileY, TileX and Linear are supported. TileY is the fastest, Linear is the slowest.</td></tr>
<tr><td>3D Color,Depth</td><td>
<table>
<tr><th>Rendering Mode
Color-vs-Depth bpp</th><th>Buffer Tiling Supported</th></tr>
<tr><td>Classical
Same Bpp</td><td>Both Linear
Both TileX
Both TileY
Linear & TileX
Linear & TileY
TileX & TileY</td></tr>
<tr><td>Classical
Mixed Bpp</td><td>Both Linear
Both TileX
Both TileY
Linear & TileX
Linear & TileY
TileX & TileY</td></tr>
</table>
</td></tr>
</table>

Memory Compression

CCS Surface Encodings

This page presents CCS encodings for projects prior to Unified Compression feature.

The compression status is held as a 4 bit encoding in Compression Control Surface. For 3D one 4bit encoding represents status of 128B of data.

For Media, we use 3rd party compressor capable of handling 256b block compression, 4bit encoding is duplicated for left and right 128B.

Table of CCS encodings

Color Code	Notes
x	Unused data
clr	Data at clear, needs to be filled in with clear value, no actual data is transferred
un	Uncompressed data
cp	Compressed data

Unified Compression, **C**ompression **C**ontrol **S**urface encodings																	
		MEDIA										3D					
CCS		Media	Notes	CL3 (64B)		CL2(64 B)		CL1(64 B)		CL0(64 B)		3D	Notes	CL1(64 B)		CL0(64 B)	
00 00	0x 0	Media_1to1	Uncompressed	un	un	un	un	un	un	un	un	3D_1to1	Uncompressed	un	un	un	un
00 01	0x 1	Media_4to1_Mono	128B to 32B					x	x	x	cp	3D_4to1_Mono	128B to 32B	x	x	x	cp
00 10	0x 2	Media_4to3_UCU	not used,128B, upper uncompressed					un	un	x	cp	3D_4to3_UCU	upper uncompressed, lower comp	un	un	x	cp
00 11	0x 3	Media_8to6_UCU	not used, 256B, upper uncompressed	un	un	un	un	x	x	cp	cp	3D_2to1_CCL	upper uncompressed and lower clear	un	un	clr	clr
01	0x	Media_8to1_M	256B to 32B	x	x	x	x	x	x	x	cp	3D_FDV0	FDV0 entire	x	x	x	x

00	4	ono	shared										cache line filled with zero				
0101	0x5	Media_8to2_Mono	256B to 64B shared	x	x	x	x	x	x	cp	cp	3D_Null_Tile	Tile is Null	x	x	x	x
0110	0x6	Media_4to2_Mono	128B to 64B					x	x	cp	cp	3D_4to2_Mono	128B to 64B	x	x	cp	cp
0111	0x7	Media_8to6_UCL	not used, 256B, lower uncompressed	x	x	cp	cp	un	un	un	un	N/A	not used	x	x	x	x
1000	0x8	Media_4to3_UCL	not used,128B, lower uncompressed					x	cp	un	un	3D_4to3_UCL	lower uncompressed	x	cp	un	un
1001	0x9	Media_4to3_Mono	not used, monolithic 4:3 Feature					x	cp	cp	cp	3D_4to3_Mono	not used, monolithic 4:3 Feature	x	cp	cp	cp
1010	0xA	Media_4to2_Individual	not used, 128B, 64B individually comp					x	cp	x	cp	3D_4to2_Individual	64B individually compressed	x	cp	x	cp
1011	0xB	Media_8to3_Mono	256B to 96B	x	x	x	x	x	cp	cp	cp	3D_4to1_CCL	lower clear, upper compressed	x	cp	clr	clr
1100	0xC	Media_8to4_Mono	256B to 128B	x	x	x	x	cp	cp	cp	cp	3D_2to1_CCU	upper clear and lower uncompressed	clr	clr	un	un
1101	0xD	Media_8to5_Mono	256B to 160B	x	x	x	cp	cp	cp	cp	cp	3D_ML	Machine Learning specific	x	x	cp	cp
1110	0xE	Media_8to6_Mono	256B to 192B	x	x	cp	cp	cp	cp	cp	cp	3D_4to1_CCU	upper clear, lower compress	clr	clr	x	cp

													ed				
11 11	0x F	Media_8to7_Mono	not used, 256B to224B shared	x	cp	cp	cp	cp	cp	cp	cp	3D_ClearClear	Both 64B chunks at clear value	clr	clr	clr	clr

Media Memory Compression

The software requirement when using media memory compression is to allocate each compressible surface one memory tile wider than is required based on the surface width plus normal byte padding (this approach is called "pitch+1"). The reason is each compressible surface needs an "extra" tile to the right edge of surface to store important compression control information. For example, if the surface is 1920x1088, this would normally be allocated by the driver to be 2048 bytes wide, or 16 tiles (for NV12 8bpp). Using this "pitch + 1", the pitch would be set to 17 instead of 16 (and the surface width remains unchanged, only pitch is increased).

The largest supported width will be 4K pixels for 2D RGBA 8bpp surfaces and 2x2K for S3D surfaces (for 4KB pages). E.g. the pitch would be set to 129 in these cases (128+1). NV12 4K would be 33 (28+1). The case of 64KB pages is the same: the driver will allocate 1 extra page to the right ("pitch + 1"), however now the 4K wide restriction is relaxed. With 64KB pages, the widest surface that supports memory compression is 16K for 2D RGBA 8bpp or 2x8K for S3D. E.g. the pitch would be set to 129 in these cases (128+1).

Memory Object Overview

Any memory data accessed by the device is considered part of a *memory object* of some memory object type.

The following table lists the various memory objects types and an indication of their role in the system.

Memory Object Type	Role
Graphics Translation Table (GTT)	Contains PTEs used to translate "graphics addresses" into physical memory addresses.
Hardware Status Page	Cached page of sysmem used to provide fast driver synchronization.
Logical Context Buffer	Memory areas used to store (save/restore) images of hardware rendering contexts. Logical contexts are referenced via a pointer to the corresponding Logical Context Buffer.
Ring Buffers	Buffers used to transfer (DMA) instruction data to the device. Primary means of controlling rendering operations.
Batch Buffers	Buffers of instructions invoked indirectly from Ring Buffers.
State Descriptors	Contains state information in a prescribed layout format to be read by hardware. Many different state descriptor formats are supported.
Vertex Buffers	Buffers of 3D vertex data indirectly referenced through "indexed" 3D primitive instructions.

Memory Object Type	Role
VGA Buffer (Must be mapped UC on PCI)	Graphics memory buffer used to drive the display output while in legacy VGA mode.
Display Surface	Memory buffer used to display images on display devices.
Overlay Surface	Memory buffer used to display overlaid images on display devices.
Overlay Register, Filter Coefficients	Memory area used to provide double-buffer for Overlay register and filter coefficient loading.
Cursor Surface	Hardware cursor pattern in memory.
2D Render Source	Surface used as primary input to 2D rendering operations.
2D Render R-M-W Destination	2D rendering output surface that is read in order to be combined in the rendering function. Destination surfaces that accessed via this Read-Modify-Write mode have somewhat different restrictions than Write-Only Destination surfaces.
2D Render Write-Only Destination	2D rendering output surface that is written but not read by the 2D rendering function. Destination surfaces that accessed via a Write-Only mode have somewhat different restrictions than Read-Modify-Write Destination surfaces.
2D Monochrome Source	1 bpp surfaces used as inputs to 2D rendering after being converted to foreground/background colors.
2D Color Pattern	8x8 pixel array used to supply the "pattern" input to 2D rendering functions.
DIB	"Device Independent Bitmap" surface containing "logical" pixel values that are converted (via LUTs) to physical colors.
3D Color Buffer	Surface receiving color output of 3D rendering operations. May also be accessed via R-M-W (aka blending). Also referred to as a Render Target.
3D Depth Buffer	Surface used to hold per-pixel depth and stencil values used in 3D rendering operations. Accessed via RMW.
3D Texture Map	Color surface (or collection of surfaces) which provide texture data in 3D rendering operations.
"Non-3D" Texture	Surface read by Texture Samplers, though not in normal 3D rendering operations (for example, in video color conversion functions).
Motion Comp Surfaces	These are the Motion Comp reference pictures.
Motion Comp Correction Data Buffer	This is Motion Comp intra-coded or inter-coded correction data.

Intel® Iris® Xe and UHD Graphics Open Source

Programmer's Reference Manual

For the 2020-2021 11th Generation Intel Xeon®, Core™, Celeron®, Pentium® Gold Processors based on the "Tiger Lake" Platform

Volume 7: Memory Cache

December 2021, Revision 1.0

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Table of Contents

Memory Cache **1**

L3 Cache 1

Overview 1

L3 Coherency 6

L3 Cache Error Protection 9

Memory Cache

This section describes the GFX L3 Cache, which is a large storage that backs up various L2/L1 caches in the GPU's internal units. It supports a simple, way-based partitioning for segregating the cache among groups of clients.

It also has a provision to dedicate a (programmable) section of the storage for the GFX Unified Return Buffer.

L3 Cache

This volume describes the L3 and the Unified Return Buffer. Much of the design is similar to prior generations, with important changes to enhance performance.

Overview

To cater to the bandwidth demands, the L3 cache is organized as multiple independent banks which can be accessed concurrently. The L3 banks are no longer clocked at 2x and hence support one operation per clock.

The L3 cache and URB data form a single contiguous memory space across all the banks and sub banks in the design. The vision is to build a compute scalable cache where with each additional compute block, both the size and bandwidth of L3 Cache are scaled while maintaining the monolithic cache concept. Each added bank becomes a part of a unified cache rather than an independent localized segment. The concept is to be able to keep a single copy of a line and service all requesters via distributing their accesses over many physical cache banks. The L3 cache can operate concurrently in the non-IA-coherent GFX virtual address space as well as the IA-coherent address space.

- Each logical bank consists of:
    - The Data Array - This array stores the actual data
    - The Tag Array - This array stores the tags for the cachelines above
    - The LRU Array - This array stores information that helps determining the cache line that will be evicted when a fill arrives for a set
    - The State Array - This array stores the cache state information (MESI States of the cache lines and some additional internal information)
    - The SuperQ Buffer - This array stores data temporarily on the way in or out of the data array for each access that is in progress
    - the Atomic Processing Units - This unit houses the ALU and associated logic to perform atomic operations on the data
- The rest of the support logic around L3 consists of:
    - The SuperQ: This is the main scheduler of the micro-sequences to be followed for each access to the cache
    - Ingress/Egress queues: These queues buffer incoming accesses and outgoing data on their way into or out of the cache

- CAM structures: These structures are used to maintain coherency in cases of proximal accesses to the same address
- Crossbars: These are used for routing the data to and from the various sub-banks/arrays

Note: A portion of the L3 cache can be allocated as a Unified Return Buffer (URB) region

L3 Bank Configuration

Each L3 bank is configured as described below.

- Each bank consists of a 480KB data-array organized as 120 logical ways
- Up to 104 ways representing 416KB, tagged for L3$, remaining ways treated as dedicated URB.
- The minimum URB size per L3 Bank is 64KB, but this can be programmed up to 128KB.
- 64B Cacheline storage per cell.
- The total L3 array bandwidth is 64B for each core cycle. The bandwidth can be utilized for either a 64B read cycle or a 64B write cycle
- Data protection via ECC.
- 48b virtual addressing support in TAG.
- 1b LRU implementation for selecting the line to be replaced.

L3 Cache Theory of Operation

Following are the L3/URB clients:

L3 Cache Clients	RW/RO
Data Cluster (i.e., spill/fills, load/stores, Global memory accesses)	RW
Sampler (L2$)	RO
IME (Motion Estimation)	RO
Instruction Cache (I$)	RO
State Arbiter	RO
Constant Cache	RO

URB Clients	RW/RO
Local Thread Dispatcher	RO
SF Backend	RO
Stream Out	RO
Clipper	RO
Geometry shader	RO
Tesselator	RO
VF	RW
Data Cluster	RW

The L3 and the URB are separate address spaces with clients capable of accessing only one or the other except for the data port. L3 access/cacheability is determined via a parameter, part of the surface state or base address programming of L3 clients.

Size of L3 Bank and Allocations

General notes

The L3 Cache has been divided into following client pools:

- URB: Local memory space
- DC: Data Cluster Data type
- Color: Color cache allocation
- Z: Depth cache allocation

In addition to these sub-groups, a collection of groups bundle multiple clients under the same allocation set:

- Read-Only (RO) Clients: Inst/State, Constants & Textures (I/S/C/T)
- All L3 Clients (a.k.a "Rest of L3"): DC, Inst/State, Constants & Textures
- Unified Tile Cache: Color/Z combined

Multi-Bank Allocation Options with Tile Cache and Command buffer support

The L3 cache no longer supports the highly banked SLM mode of operations. Out of the total L3 space per bank, a minimum of 64Kbytes will be earmarked for URB storage. The rest can be used as tagged L3 cache. The L3 Cache allocation is done on a per way basis. The Bank programming options allow a varied set of configurations to be programmed in the L3. Broadly, the number of ways allocated to the various sections (URB, DC, RO, Z, C and the command buffer) is selectable. Each of these sections can be allocated a subset of ways out of the total ways. Apart from that, in lieu of the DC and RO sections a single unified “Rest of L3” section can be used. Similarly, in lieu of separate Z and C partitions in the L3, a single Unified Tile Cache programming allows all Z and C streams to share a common section of the L3.

L3 Allocation programming (KBytes per bank)								
URB	**Rest (DC+RO)**	**DC**	**RO (I/S/C/T)**	**Z**	**Color**	**Unified Tile Cache**	**Command Buffer/ State**	**Sum**
64KB or 128KB	0 to 416 in increments of 8KB	0 to 416 in increments of 8KB	0 to 416 in increments of 8KB	0 to 416 in increments of 8KB	0 to 416 in increments of 8KB	0 to 416 in increments of 8KB	0 to 416 in increments of 8KB	480

Note: The granularity of 8KB delta for each section arises from the fact that the L3 is implemented as a sectored cache with 2 ways per sector. Due to this, the number of ways programmed for each section of the L3 cache should be an even number. The programming of the **L3ALLOCREG** to configure the sections should be determined based on the size of the cache per way. The total number of ways implemented in the design is available in the **L3 Parameter Information register**.

However, several restrictions apply.

- It is not allowed to allocate the entire cache to DC (Data space) with 0KB for reads.
- It is not allowed to have Rest and DC to be "0KB" at the same time. Cache requires either Rest or DC to have at least non-0KB allocation.
- It is not allowed to have Rest and RO to be "0KB" at the same time. Cache requires either Rest or RO to have at least non-0KB allocation.

State is now stored as part of Command Streamer allocation. If specific allocation is 0KB, it will be placed in the Rest section.

L3 Allocation programming (KBytes per bank)									
Config	URB	Rest (DC+RO)	DC	RO (I/S/C/T)	Z	Color	Unified Tile Cache	Command Buffer	Sum
0 (def)	64	416	0	0	0	0	0	0	480
1	128	240	0	0	48	48	0	16	480
2	64	0	32	240	64	64	0	16	480
3	64	144	0	0	128	128	0	16	480
4	64	48	0	0	0	0	352	16	480
5	64	272	0	0	0	0	128	16	480
6	64	400	0	0	0	0	0	16	480
7	128	352	0	0	0	0	0	0	480
8	128	336	0	0	0	0	0	16	480

The number of L3 Banks will vary for different products and SKUs. The number of banks supported for each product is defined in the Configurations section of this document. The total amount of L3$ and URB supported by a product can be calculated by multiplying the number of banks by the values in the above tables.

Note that any allocation for <64KB will limit the bank allocation and proper set distribution. Such programming will lead to set allocation to be less efficient and will not be sufficient to contain the surface unless it is programmed to be natively aligned.

Performance configurations are listed as #5 (deferred rendering) and #6 (immediate and deferred rendering).

L3 Ordering Restrictions

The Super Q will enforce specific ordering requirements on the accesses to the L3/URB but will still allow out-of-order accesses where possible.

Basic Ordering Requirements

The primary purpose of ordering of transactions is to ensure coherency and causality for software accesses to memory. For example, if a thread writes to a memory address, it can expect that any subsequent read issued by the same thread to the same address should read back what it wrote. Conversely, if the read is before the write, it shall receive the value prior to the write.

L3 Cache Allocation Policy

The L3 cache allocation policy is "Allocate on fill". i.e., a line in the cache storage is allocated only upon receipt of the data from the external memory and not at the time that the cache detects a miss. The "Allocate on Fill" policy eliminates many boundary cases by regulating the entry invalidation at the last phase of the data servicing.

If the data already residing in the allocated entry is not modified, then the incoming Fill will overwrite the location. If the allocated entry carries a dirty data, then an eviction is generated, and the dirty data will be moved to the super queue for writing out to memory.

Cache replacement algorithm

The L3 cache uses a 1b LRU algorithm for cache replacement. An N-way 1b LRU has an N-bit vector for each set. The vector is initialized to all 0's. As each Fill request arrives, the first bit in the vector with a 0 is selected, that way is used, and the bit is flipped. Each subsequent fill will search for the first 0 and replace that line and flip its bit. Eventually, when all N bits are 1 and there are no candidates, all bits are cleared.

Any hits to lines present in the cache will update the LRU vector to set the bit corresponding to the way. In this manner, after the first set of allocations are done, any hits will mark the way as "recently used" and avoid replacement. If during a hit operation, a set becomes fully "recent" i.e., all ways have a '1' in the LRU vector, the vector is cleared making way for fresh marking for recent uses.

Memory Object Control State on Cacheability

This 7-bit field is used in various state commands and indirect state objects to define L3/LLC cacheability, memory type, and graphics data type for memory objects.

Note that memory type information from state is used for non-IA compatible paging structures (legacy context). For new context definition where IA compatible (IA32e) paging structures are used, memory typing follows the IOMMU defined structures.

MOCS[6:1] in L3 is used as an index to a set of programmable tables starting with address xB020h. GFX Software can set up the tables as part of the h/w context, and program various index values in surfaces to point to a table that best suits for that particular surface.

L3 Coherency

Coherency is one of the crucial topics within L3, there are multiple levels of coherency that are checked and ensured via L3. The list of domains and flows is dependent on the usage models however basic premise is always the same.

The coherency levels:

1. Thread Level Coherency within a Thread Group
2. Thread Group Coherency between multiple domains
3. GPU/IA level coherency

Besides these special domains basic producer/consumer models are followed which are listed as:

1. Fixed function is producing
2. Data Port is producing

When textures are stored as compressed in L3, the expectation is to ensure that the compressed RO-content is invalidated before same surface can produced as R/W.

Thread Level Coherency

A given thread group is contained within a sub-slice, where its writes and reads target the L3 for global memory and SLM for shared local memory. Given the shared local memory view is the same for all sub-slice accesses, coherency or data sharing is guaranteed within the thread group. Local syncs are executed up to Data Port boundary and not exposed to L3.

Thread Group Coherency

Thread groups can be distributed to multiple sub-slices that are physically far from each other. The coherency between thread groups can only be maintained for their global memory accesses. There are two implications of this coherency depending on the mode we are operating at:

1. Non IA-Coherent L3 mode: This is the same methodology that was introduced with the addition of GT4 support. The cross-thread group coherency is maintained via Sync Global which is processed by L3 as well as introducing WT mode to be able to update global memory with latest data. This mechanism is not expected to be used.
2. IA-Coherent L3 mode: For the new mode of operation, there is no need to have WT behavior. The data in L3 is already visible to all consumers (i.e. other thread groups of GPU or IA cores). Sync'global has no affect given the data in L3 is already globally visible to all consumers.

GPUIA Level Coherency

In non-coherent L3 mode, data sharing between GPU and IA happens via a s/w controlled flow which requires the internal GPU caches to be flushed in order to make the data visible, similarly same caches need to be invalidated when IA produces data. This mode of operation is still available. Within the L3,

non-coherent accesses use "virtual" addresses and are tagged in the L3 Tag array using the Virtual/Physical bit.

Coherent memory is also supported where L3 content is visible to IA via snoops. This allows certain streams (i.e. data port) to be GO (Globally Observable) for IA once posted to L3, allowing shorter loop for completions and eliminating the need to do an entire pipeline flush over L3 to get it sync'ed to IA coherent domain. Coherency enhances the general programmability of GPU when cooperating with IA. Within the L3, coherent accesses use a full 48-bit physical address and are tagged using the Virtual/Physical bit in the l3 Tag array.

Coherency Usage Models

This section is to give some examples of usage models and high-level handling within L3. They are specific to L3 flows and not meant to represent over coherency usage models on the system level.

Fixed Func Producing (Push Constants)

Push constants are also similarly processed; in fact, the target is the same: URB. However, URB accesses are processed with their URB offsets via TAG, their original address is based on virtual memory pointing to buffer location in memory. The delivery mechanism makes the CAMs not possible hence the completions are managed once access is deemed towards SuperQ of the corresponding L3 Bank.

There are additional constraints independent of L3 handling of push constants which are defined as part of the Global Arbitration fabric.

EUs Producing via HDC

Data port is the producer for Global and Shared Local memory types. The shared local memory is specific to a data port, hence L3 does not have to do anything special to maintain coherency but to keep accesses to SLM in-order.

For Global memory coherency is maintained via combination of many mechanisms.

1. Completion tracking: Each Data Port tracks their writes towards L3 and wait for them to reach to GO. GO message is given by L3

a. For Non-coherent/virtual addressed Write: Once the ingress queue retires the write in the corresponding node to targeted bank.
b. For Coherent/physical addressed Write: Once the ownership is obtained for the write (i.e. read-for-ownership or invalid-to-modified) which means write is GO with respect to IA cores.

GO information is used within data port to make sure handle releases are gated until the producer's updates are globally visible.

2. Thread Level Flush: EU threads have the capability to push globally tagged data from L3 to next level caches. Usage is not recommended.

Invalidation and Flushes

There are two different sources to initiate flushes and invalidations:

- Command Streamer initiated
- EU/thread initiated

Both cases have two modes of operation

- Non-IA coherent L3
- IA coherent L3

In addition, there are side flows that back up the various flows and they do require invalidation/flush sequences to be executed for their purpose. Their behavior has no impact between two modes of L3 operation.

- Global Invalidation
- Power Management Invalidation

Command Streamer Invalidation Flows

The Command streamer can initiate the following flush/invalidation flows.

1. **Top of the pipe invalidations**: These are direct, asynchronous commands from the command streamer unit to L3 in respective slices.
2. **Pipeline flush**: These commands flow through the pipe and are issued through the data port to the L3 cache in each slice.

Non-IA Coherent Flows

This is the traditional flow where the content of L3 needs to be invalidated or flushed.

Top of the Pipe Invalidations

For this case, the invalidation causes the cache to drain all FIFOs and pipelines, and the node performs all invalidations. Only after completion in all slices will new transactions be sent to the L3.

The Top of the Pipe invalidation is intended for cases where invalidation and completion need to be coordinated between slices. Therefore, each slice performs invalidation, but will not proceed until all slices have completed the invalidation.

Pipeline Flush

Pipeline invalidation is much simpler and optimized for performance. In this case there is no need to coordinate between slices and the operation is more like a "sync" point. Transactions arriving after the invalidation request will still be queued in the SQ but will only be processed after the invalidation of all candidate lines in the cache is complete. The invalidation acts like a fence.

The pipeline flush send via the dataport will only flush dataport generated allocations. In previous generations this was not optimized and used flush all content of L3 cache.

IA-Coherent Flows

In principle IA-coherent flow is same from the L3 cache perspective, the only difference is within the banks given the data that we are trying to flush or invalidate is already coherent with IA. Such case eliminates the need to push any bank content explicitly out to LLC.

The L3 bank will ensure that the transactions in flight are all globally observable when a flush/invalidation is received.

EUThread Flows

There is continued support for the EUs to perform flush/invalidation events that are not coordinated with other EUs. The data cluster will relate the request to the corresponding L3 and require the entire content of this buffer to be invalidated or flushed to relative coherency domain.

The IA-coherent vs non-IA coherent treatment is same as command streamer flows and still applicable for EU/Thread Flows for invalidation/flush.

Power Management Invalidation

Given we have a IA-coherent mode where a standard pipeline flush does not push the modified lines to outside of GT and we are progressing to deferred invalidations, we need a way to ensure L3 content is cleared and deferred events are complete. This is where PM interaction will have to be introduced.

Corresponding PM will message to L3 config agent to request a flush when needed. Main usage mode is prior to entering RC6. The rest of the treatment in the nodes is the same. Once flush/invalidate event is complete, message(s) will be sent back to PM with the completion status.

L3 Cache Error Protection

L3 cache error protection is covered via ECC (SECDED). All accesses are subject to ECC protection where single bit errors are fixed silently. Double bit errors are reported via a register structure and communicated by an interrupt to GFX driver. L3 cache HW is additionally capable of stalling execution upon a double bit error.

Intel® Iris® Xe and UHD Graphics Open Source

Programmer's Reference Manual

For the 2020-2021 11th Generation Intel Xeon®, Core™, Celeron®, Pentium® Gold Processors based on the "Tiger Lake" Platform

Volume 8: Command Stream Programming

December 2021, Revision 1.0

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Table of Contents

Command Stream Programming 1
Introduction 1
Workload Submission and Execution Status 2
Scheduling and Execlists 2
Preemption 10
Execution Status 11
Commands and Programming Interface 14
Command Buffers 14
Command Streamer Command Formats 17
Execution Control Infrastructure 20
MI Commands for Graphics Processing Engines 29
Register Access and User Mode Privileges 30
Context Management 43
Global State 44
Context State 49
Producer-Consumer Data ordering for MI Commands 53
Memory Data Ordering 53
MMIO Data Ordering 55
Command Fetch 56
Advanced Command Prefetch 56
Self Modifying Code 57

Command Stream Programming

Introduction

Command Streamer is the primary interface to the various engines that are part of the graphics hardware.

The graphics HW consists of multiple parallel engines that can execute different kinds of workloads. E.g Graphics, Video Decode engine, Video Enhancement Engine and Blitter engine.

Some product SKU's have multiple instances of an engine (e.g 2 Video Decode engines).

As shown in figure 1, each of these engines have their own Command Streamer that is responsible for processing the commands in the workload and enabling execution of the task.

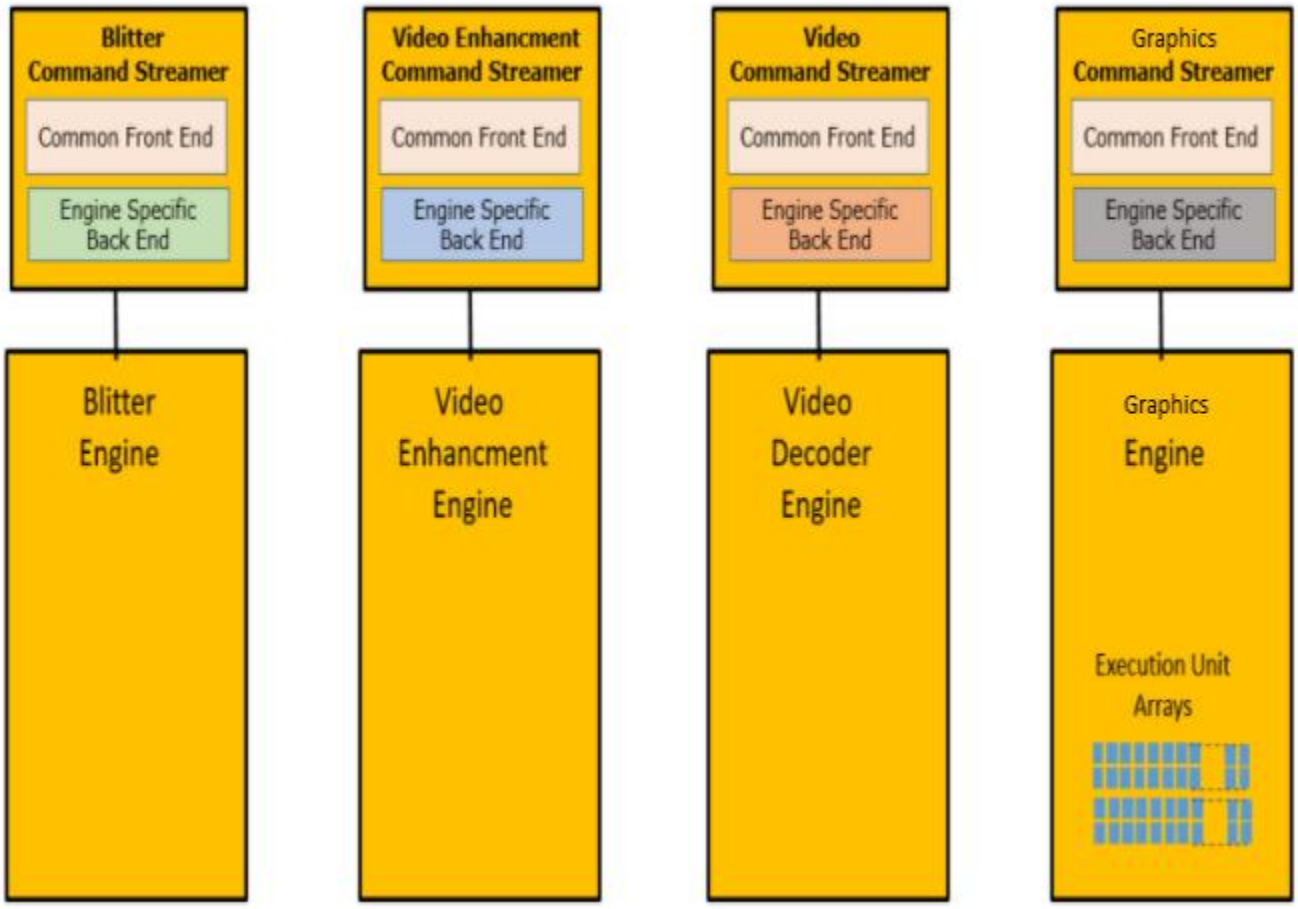


Figure 1: High level view of Command Streamer

As shown in the figure, the command streamer is comprised of a Common Front end and an engine specific backend.

The common front end allows each engine to provide a uniform software interface (e.g infrastructure for submission of commands, sychronization, etc).

The back ends handle the engine specific commands and the protocols required to control the execution of the underlying engine.

Workload Submission and Execution Status

This section describes the interface to submit work and obtain status

Scheduling and Execlists

Execution-List provides a HW-SW interface mechanism to schedule context as a fundamental unit of submission to GFX-device for execution. GFX-device has multiple engines (Graphics, Blitter, Video, Video Enhancement) with each of them having an execution list for context submission. At any given time all engines could be concurrently running different contexts.

A context is identified with a unique identifier called Context ID. Each context is associated with an address space for memory accesses and is assigned a unique ring buffer for command submission.

SW submits workload for a context by programming commands in to its assigned ring buffer prior to submitting context to HW (engine) for execution.

Context State:

Each context programs the engine state according to its workload requirements. All the hardware state variables of an engine required to execute a context is called context state. Each context has its own context state. Context state gets programmed on execution of commands from the context ring buffer. All the contexts designated to run on an engine have the same context format, however the values may differ based on the individual state programming.

Logical Context Address:

Each context is assigned a Logical Context Address to which the context state is saved by the engine on a context getting switched out from execution. Similarly, engine restores the context state from the logical context address of a context on getting switched in for execution.

Logical context address is an absolute graphics virtual address in global virtual memory. Context state save/restore mechanism by the engine avoids SW from re-programming the state across context switches.

Each engine has its own hardware state variables and hence they have different context state formats. A context run on a Render engine can't be submitted to Blitter engine and vice-versa and holds true for any other engines.

Context Submission:

A context is submitted to an engine for execution by writing the context descriptor to the Execlist Submit Port (ELSP). Refer ELSP for more details. Context descriptor provides the Context ID, Address space, Logical Context Address and context valid. Refer context descriptor for more details.

Logical context address points to the context state in global virtual memory which has ring buffer details, address space setup details and other important hardware state initialization for the corresponding context. Refer Logical Context Format for more details.

Note that this mechanism cannot be used when the **Execlist Enable** bit in the corresponding engines MODE register is not set, i.e GFX_MODE register for Render Engine, BLT_MODE register for Blitter Engine, VCS_MODE register for Video Engine, or VECS_MODE register for Video Enhancement Engine.

Execution List Submission Port (ELSP)

Execution List Submit Port is a MMIO register in every engine. Contexts are queued in for submission to the engine by writing the context descriptor to the engine's ELSP. ELSP provides flexibility to queue up to eight contexts at a time for submission called Submission Queue (SQ). SQ is a staging buffer in the engine which can hold up to eight contexts (elements) for submission. ELSP provides a mechanism to load the elements of the SQ in a cyclic order which wraps around to E0 on writing to E7 (E0, E1 ... E7, E0 ..). Each element of the SQ is also individually MMIO mapped which can be written to or read from, this provides an alternate flexible mechanism to independently modify any element of the SQ in any order avoiding ELSP port. The valid bits of the unused context descriptors should be set to '0', in a pathological case all the eight elements can have the valid bits of the context descriptors set to '0' (Empty Submission Queue).

Writing to SQ through ELSP or direct MMIO itself doesn't trigger the engine to start executing the elements from the SQ. Engine has to be explicitly notified to start executing the elements from the SQ by writing to the "Load" bit in the Execlist Control Register. Engine on detecting Load notification will sample the SQ to its internal execution staging buffer called Execution Queue (EQ). On loading Execution Queue becomes valid. Engine will start executing elements with valid context descriptors in serial order from the valid EQ, starting from E0 followed by E1 followed by E2 and so on to E7, invalid elements are simply skipped by the engine. EQ becomes invalid following execution of E7 making engine idle.

Moving from one element to the other element in an EQ is called synchronous context switch. Once a context is switched out, the relevant context state and context descriptor doesn't exist in EQ, only way the context can be brought back for execution in to EQ is through a new "Load" from SQ. Refer context switch section for more details.

SW can modify SQ as many times as needed before issuing the "Load" command to the engine for execution. SQ contents are retained and not destructed on issuing "Load" and SQ contents are also retained across power flows.

Issuing a "Load" command while there is an ongoing element execution from EQ will result in immediate sampling of SQ to EQ and also results in preemption of the executing context on appropriate boundary. Once preemption of the ongoing executing context is complete, engine will start executing from Element-0 of the updated EQ. Preemption of a context is called asynchronous context switch and refer context switch section for more details. Multiple loads occurring while there is an ongoing preemption of an executing context will result in EQ getting updated multiple times, engine will only execute the latest EQ available upon completion of the preemption and will not get to see the intermediate updates.

Execlist Submit Port Register

Execlist Submission Queue Contents

Execlist Control Register

Context Descriptor Format

Context Descriptor Format

Before submitting a context for the first time, the context image must be properly initialized. Proper initialization includes the ring context registers (ring location, head/tail pointers, etc.) and the page directory.

If supported in register programming, the **Context Restore Inhibit** bit in the Context/Save image in memory can be set to prevent restoring garbage engine context. See the Logical Ring Context Format section for details.

Programming Note on Context ID field in the Context Descriptor

This section describes the current usage by SW.

General Layout:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Eng. ID				**SW Counter**					**HW Use**		**SW Context ID**																					

Eng. ID = Engine ID (a software defined enum to identify RCS, BCS etc..)

SW Counter = Submission Counter. (SW generates an unique counter value on every submission to ensure GroupID + PASID is unique to avoid ambiguity in fault reporting & handling)

Bit 20 = Is Proxy submission. If Set to true, SW Context ID[19:0] = LRCA [31:20], else it is an index into the Context Pool.

Direct Submission (Ring 3 to GuC)

Every application gets one context ID of their own.

SW Context ID + Engine ID + SW Counter forms the unique number

The Engine ID is used to identify which engine of a given context needs to be put into wait or ready state based on Semphore/Page Fault ID value in Semaphore/Page fault FIFO.

This method allows the context to submit work to other engines while its blocked on one.

Proxy Submission (In order submission from Kernel Mode Driver to GuC)

KMD creates one context for submitting work on behalf of various user mode contexts (user mode application is not using direct submission model).

This method has certain key restrictions and behaviors:

- Work (LRCA) submitted will be scheduled on the CS in the order it was received.
- KMD uses its SW Context ID in [63:32] but uses the LRCA of the user mode context.
    - KMD's LRCA is not used for any work submission.

- If a workload hits a wait event, it does not lose its position in the schedule queue.
    - Enforces "in order" ness.
- Due to in order execution, same engine - different context semaphore synchronization is not possible.
    - Therefore, cross engine sync is simple because it clears the semaphore of the head.
- Due to in order execution, page fault on a context cannot allow a different context on same engine to execute (may preempt to idle as a power optimization).

This method allows a clean SW architecture to have KMD submissions and Ring 3 submissions to co-exist.

Logical Ring Context Format

Context descriptor has the graphics virtual address pointing to the logical context in memory. Logical context has all the details required for an engine to execute a context. This is the only means through which software can pass on all the required information to hardware for executing a context. Engine on selecting a context for execution will restore (fetch-context restore) the logical context from memory to setup the appropriate state in the hardware. Engine on switching out the context from execution saves (store- context save) the latest updated state to logical context in memory, the updated state is result of the command buffer execution.

The Logical Context of each engine (Render, Video, Blitter, Video Enhancement ..etc) primarily consists of the following sections:

- Per-Process HW Status Page (4K)
- Ring Context (Ring Buffer Control Registers, Page Directory Pointers, etc.)
- Engine Context (PipelineState, Non-pipelineState, Statistics, MMIO)

Per-Process of HW status Page (PPHWSP)

This is a 4KB scratch space memory allocated for each of the context in global address space. First few cachelines are used by the engine for implicit reports like auto-report of head pointer, timestamp statistics associated with a context execution, rest of the space is available for software as scratch space for reporting fences through MI commands. Context descriptor points to the base of Per-Process HW status page. See the PPHWP format in **PPHWSP_LAYOUT**.

Logical Ring Context

Logical Ring Context starts immediately following the PPHWSP in memory. Logical ring context is five cachelines in size. This is the minimal set of hardware state required to be programmed by SW for setting up memory access and the ring buffer for a context to be executed on an engine. Memory setup is required for appropriate address translation in the memory interface. Ring buffer details the location of the ring buffer in global graphics virtual address space with its corresponding head pointer and the tail pointer. Ring context also has "Context Save/Restore Control Register-CTXT_SR_CTL" which details the engine context save/restore format. Engine first restores the Logical Ring Context and upon processing CTXT_SR_CTL it further decides the due course of Engine Context restore. Logical Ring Context is mostly

identical across all engines. Logical ring context is saved to memory with the latest up to date state when a context is switched out.

Engine Context

Engine context starts immediately following the logical ring context in memory. This state is very specific to an engine and differs from engine to engine. This part of the context consists of the state from all the units in the engine that needs to be save/restored across context switches. Engine restores the engine context following the logical ring context restore. It is tedious for software to populate the engine context as per the requirements, it is recommended to implicitly use engine to populate this portion of the context. Below method can be followed to achieve the same:

- When a context is submitted for the first time for execution, SW can inhibit engine from restoring engine context by setting the "Engine Context Restore Inhibit" bit in CTXT_SR_CTL register of the logical ring context. This will avoid software from populating the Engine Context. Software must program all the state required to initialize the engine in the ring buffer which would initialize the hardware state. On a subsequent context save engine will populate the engine context with appropriate values.
- Above method can be used to create a complete logical context with engine context populated by the hardware. This Logical context can be used as an Golden Context Image or template for subsequently created contexts.

Engine saves the engine context following the logical ring context on switching out a context.

The detailed format of the logical ring context for Blitter, Video, and VideoEnhancement is documented in the Memory Object Overview/Logical Contexts chapter.

The detailed formats of the Render Logical Ring and Engine Context, including their size, is mentioned in the **Engine Register and State Context** topic for each product.

RINGBUF -- Ring Buffer Registers

Register
RING_BUFFER_TAIL - Ring Buffer Tail
RING_BUFFER_HEAD - Ring Buffer Head
RING_BUFFER_START - Ring Buffer Start
RING_BUFFER_CTL - Ring Buffer Control

Command Stream Virtual Memory Control

Per-Process GTT (PPGTT) is setup for an engine (Graphics, Blitter, Video and Video Enhancement) by programming corresponding Page Directory Pointer (PDP) registers listed below. Refer "Graphics Translation Tables" in "Memory Overview" for more details on Per-Process page table entries and related translations.

Context Status

Hardware reports the change in state of context execution to software (scheduler) through Context Status Dword. Soft-Ware can read the context status dword from time to time to track the staet of context execution in hardware. A context switch reason (Context Switch Status) quad-word (64bits) is reported to the Soft-Ware (scheduler) on a valid context getting switched out. Context switch could be a synchronous context switch (from one valid element to the other valid element in the EQ) or asynchronous context switch (Load-switching from the current executing context to the very first valid element of the newly updated EQ or on Preempt to Idle). Context switch reason is also reported on HW executing the very first valid element from EQ coming out of idle indicating hardware has gone busy from idle state (Idle to Active). Context ID reported in Context Status Dword on Idle-to-Active context switch is undefined and note that there aren't any active contexts running in hardware coming out of reset, power-on or idle.

A context switch reason reported is always followed by generation of a context switch interrupt to notify the Soft-Ware about the context switch. Soft-Ware can selectively mask the context switch status being reported and the corresponding interrupt due to a specific context switch reason. Refer Context Status Report controls section for more details.

- A status QW for the context that was just switched away from will be written to the Context Status Buffer in the Global Hardware Status Page. Context Status Buffer in Global Hardware Status Page is exercised when IA based scheduling is done. The status contains the context ID and the reason for the context switch.
- A context switch status QW (8 bytes) for the context that was just switched away (including Idle to Active) will be reported to the GUC. Firmware running on GUC or the SW running on host must read the reported context status QW through "CSB Read Port" and "CSB FIFO Status" MMIO registers implemented in GUC. Refer "Command Streamer Status Information" section under GUC for more details.

Format of Context Status QWord

Context Status

Context Status should be inferred as described in the table below.

IDLE_CTXID Encoding

IDLE_CTXID
0x7FF

S.No	Switch to New Queue	Ctid Away	Ctxid To	Switch Detail	Description
1	1	IDLE_CTXID	0xAB	0	**Idle to Active** Ctxid Away: IDLE_CTXID indicates HW was idle when switched to the new queue. Ctxid To: 0xAB is the context picked form the newly submitted queue to execute.
2	1	0xAB	IDLE_CTXID	0-5	**Preempt to Idle** Ctxid Away: 0xAB is the context that got switched out due to Preempt To Idle. Ctxid To: IDLE_CTXID indicates HW will go Idle following this context switch. Switch To New Queue field status set distinguishes between Preempt To Idle Vs Active To Idle Switch.
3	1	IDLE_CTXID	IDLE_CTXID	X	**Preempt To Idle, Idle to Active** Preempt To Idle has occurred when HW was idle. Ctxid Away: IDLE_CTXID indicates HW was idle when switched to the new queue (Preempt To Idle). Ctxid To: IDLE_CTXID indicates HW will go Idle following this context switch.
4	1	0xAB	0x7BC	0	**Switched to New queue and also the earlier context is complete.** Ctxid Away: 0xAB is the context that got switched out due to submission of new queue and also the context is complete. Ctxid To: 0x7BC is the context picked form the newly submitted queue to execute.
5	1	0xAB	0x7BC	5	**Switched to new queue with the earlier context preempted.** Ctxid Away: 0xAB is the context that got preempted and switched out due to submission of new queue. Ctxid To: 0x7BC is the context picked form the newly submitted queue to execute.
6	1	0xABC	0x7BC	1-4	**Switched to new queue, at the time of switch,**

					executing context was waiting on an un-successful wait. Ctxid Away: 0xAB is the context that got switched out on an un-successful wait due to submission of new queue. Ctxid To: 0x7BC is the context picked form the newly submitted queue to execute.
7	1	0xAB	0xAB	-NA-	**Lite restore.** Switched to new queue.
8	0	0xAB	0x7AC	0	**Element (Synchronous context) switch on context complete.** Ctxid Away: 0xAB is the context that got switched out due to context complete. Ctxid To: 0x7AC is the next element (context) form the execution queue selected to execute.
9	0	0xAB	0x7AC	1-4	**Element (Synchronous context) switch on un-successful wait.** Ctxid Away: 0xAC is the context that got switched out due to un-successful wait. Ctxid To: 0x7AC is the next element (context) form the execution queue selected to execute.
10	0	0xAB	IDLE_CTXID	0-4, 5*	**Active to Idle.** Ctxid Away: 0xAB is the context that got switched out due to context complete or un-successful wait. Ctxid To: IDLE_CTXID indicates HW will go Idle following this context switch. Switch To New Queue field reset status distinguishes between Active To Idle Switch Vs Preempt To Idle. Switch Detail as 5 is possible on Preempt to Idle.

Context Status Buffer in Global Hardware Status Page

Status QWords are written to the Context Status Buffer in Global Hardware Status Page at incrementing locations starting from DWORD offset of 28h. The Context Status Buffer has a limited size (see Table Number of Context Status Entries) and simply wraps around to the beginning when the end is reached. The status QWs can be examined to determine the contexts executed by the hardware and the reason for switching out. The most recent location updated in the Context Status Buffer is indicated by the **Last Written Status Offset** in Global Hardware Status page at DWORD offset 47h.

Number of Context Status Entries

Number of Status Entries
12 (QW) Entries

Format of the Context Status Buffer starting at DWORD offset 28h in Global Hardware Status page

QW	Description
15	**Last Written Status Offset.** The lower byte of this QWord is written on every context switch with the (pre-increment) value of the **Context Status Buffer Write Pointer**. The lower 4 bits increment for every status Qword write; bits[7:4] are reserved and must be '0'. The lowest 4 bits indicate which of the Context Status Qwords was just written. The rest of the bits [63:8] are reserved.
14:12	Reserved: MBZ.
11:0	**Context Status QWords.** A circular buffer of context status QWs. As each context is switched away from, its status is written here at ascending QWs as indicated by the **Last Written Status Offset**. Once QW11 has been written, the pointer wraps around so that the next status is written at QW0. Format = ContextStatusDW

Controls for Context Switch Status Reporting

This section describes various configuration bits available which control the hardware reporting mechanism of Context Switch Status.

Hardware reports context switch reason through context switch status report mechanism on every context switch. "Context Status Buffer Interrupt Mask" register provides mechanism to selectively mask/un-mask the context switch interrupt and the context switch status report for a given context switch reason. Hardware will not generate a context switch interrupt and context switch status report on a context switch reason that is masked in "Context Status Buffer Interrupt Mask" register. Every context switch reason reported by hardware may not be of interest to the scheduler. Scheduler may selectively mas/un-mask the context switch reasons of its interest to get notified.

Context Status Buffer Interrupt Mask Register

Preemption

Preemption is a means by which HW is instructed to stop executing an ongoing workload and switch to the new workload submitted. Preemption flows are different based on the mode of scheduling.

ExecList Scheduling

In ExecList mode of scheduling SW triggers preemption by submitting a new pending execlist to ELSP (ExecList Submit Port). HW triggers preemption on a preemptable command on detecting the availability of the new pending execlist, following preemption context switch happens to the newly submitted execlist. As part of the context switch preempted context state is saved to the preempted context LRCA, context state contains the details such that on resubmission of the preempted context HW can resume execution from the point where it was preempted.

Example:

```
Ring Buffer

MI_ARB_ON_OFF    // OFF
MI_BATCH_START   // Media Workload
MI_ARB_ON_OFF    // ON
MI_ARB_CHK       // Preemptable command outside media command buffer.
```

The following table lists Preemptable Commands in ExecList mode of scheduling:

Command Streamer Preemptable Commands

Preemptable Command	Condition
MI_ARB_CHECK	AP
Element Boundary	AP (if allowed)
Semaphore Wait	Unsuccessful & AP
Wait for Event	Unsuccessful & AP (if allowed)

Table Notes:

AP - Allow Preemption if arbitration is enabled.

For additional preemptable commands specific to any engine type, refer to the engine specific command interface documentation.

Execution Status

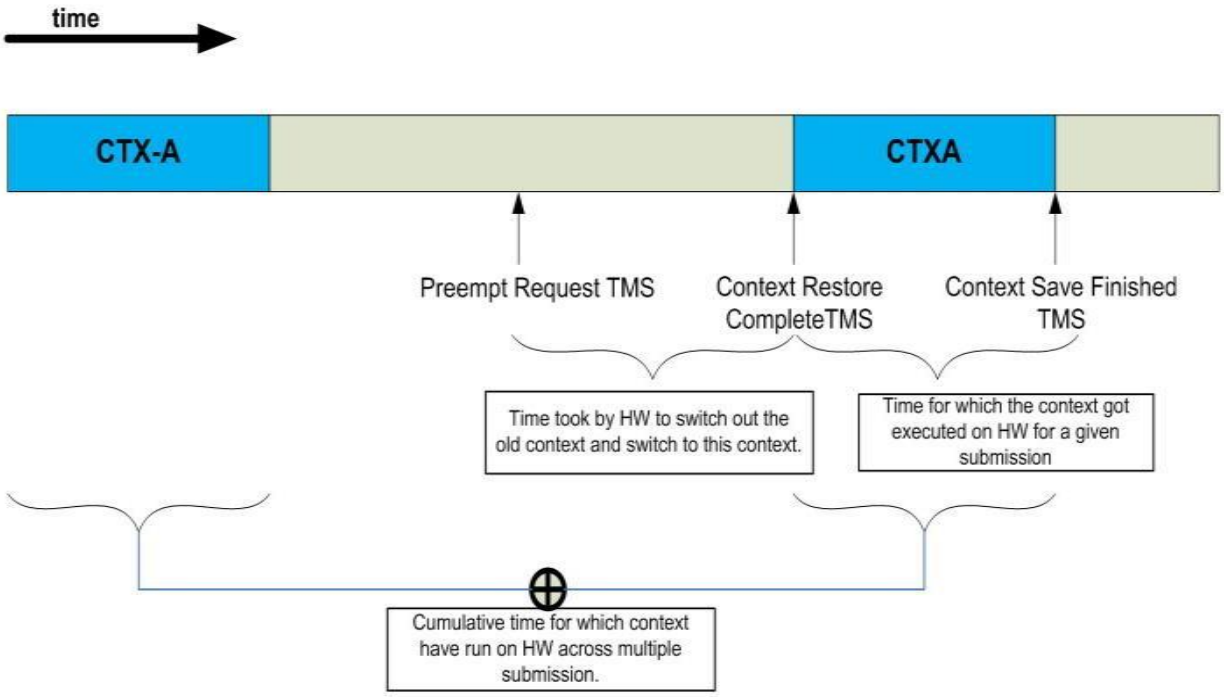
This section describes the infrastructure used to report status that the hardware provides

The Per-Process Hardware Status Page

The layout of the Per-Process Hardware Status Page is defined at **PPHWSP_LAYOUT**.

The DWord offset values in the PPHWSP_LAYOUT are in decimal.

Figure below explains the different timestamp values reported to PPHWSP on a context switch.



This page is designed to be read by SW to glean additional details about a context beyond what it can get from the context status.

Accesses to this page are automatically treated as cacheable and snooped. It is therefore illegal to locate this page in any region where snooping is illegal (such as in stolen memory).

Hardware Status Page

The hardware status page is a naturally-aligned 4KB page residing in memory. This page exists primarily to allow the device to report status via GGTT writes.

The address of this page is programmed via the HWS_PGA MI register.

Hardware Status Page Address Register

Interrupt Control Registers

The Interrupt Control Registers described in this section all share the same bit definition. The bit definition is as follows:

Bit Defintion for Interrupt Control Registers:

Engine Interrupt Vector Definition Table

Blitter Interrupt Vector
Render Engine Interrupt Vector
VideoDecoder Interrupt Vector
VideoEnhancement Interrupt Vector
Compute Engine Interrupt Vector

The following table specifies the settings of interrupt bits stored upon a "Hardware Status Write" due to ISR changes:

Bit	Interrupt Bit	ISR Bit Reporting Via Hardware Status Write (When Unmasked Via HWSTAM)
9	Reserved	
8	**Context Switch Interrupt.** Set when a context switch has just occurred.	Not supported to be unmasked.
7	**Page Fault.** This bit is set whenever there is a pending PPGTT (page or directory) fault. This interrupt is for handling Legacy Page Fault interface for all Command Streamers (BCS, RCS, VCS, VECS). When Fault Repair Mode is enabled, Interrupt mask register value is not looked at to generate interrupt due to page fault. Please refer to vol1c "Page Fault Support" section for more details.	Set when event occurs, cleared when event cleared. Not supported to be unmasked.
6	**Media Decode Pipeline Counter Exceeded Notify Interrupt.** The counter threshold for the execution of the media pipeline is exceeded. Driver needs to attempt hang recovery.	Not supported to be unmasked. Only for Media Pipe.
5	**L3 Parity interrupt**	Only for Render Pipe
4	**Flush Notify Enable**	0
3	**Master Error**	Set when error occurs, cleared when error cleared.
2	Reserved	
0	**User Interrupt**	0

Hardware Status Mask Register

Hardware-Detected Error Bit Definitions (for EIR EMR ESR)

This section defines the Hardware-Detected Error bit definitions and ordering that is common to the EIR, EMR, and ESR registers. The EMR selects which error conditions (bits) in the ESR are reported in the EIR. Any bit set in the EIR will cause the error bit in the ISR to be set. EIR bits will remain set until the appropriate bit(s) in the EIR is cleared by writing the appropriate EIR bits with 1 (except for the unrecoverable bits described below).

The following structures describe the Hardware-Detected Error bits:

The following structures describe the Hardware-Detected Error bits:

Error Bits
RCS Hardware-Detected Error Bit Definitions Structure
BCS Hardware-Detected Error Bit Definitions Structure
VCS Hardware-Detected Error Bit Definitions Structure
VECS Hardware-Detected Error Bit Definitions Structure
ComputeCS Hardware-Detected Error Bit Definitions Structure

The following are the EIR, EMR and ESR registers:

Registers
EIR - Error Identity Register
EMR - Error Mask Register
ESR - Error Status Register

Commands and Programming Interface

This section describes the command supported by command streamer and the programming interface.

Command Buffers

Instructions to be executed by an engine are submitted to the hardware using command buffers.

Command Ring Buffers

Command ring buffers are the memory areas used to pass instructions to the device. Refer to the Programming Interface chapter for a description of how these buffers are used to transport instructions.

The RINGBUF register sets (defined in Memory Interface Registers) are used to specify the ring buffer memory areas. The ring buffer must start on a 4KB boundary and be allocated in linear memory. The length of any one ring buffer is limited to 2MB.

Command Batch Buffers

Command batch buffers are contiguous streams of instructions referenced via an MI_BATCH_BUFFER_START and related instructions (see Memory Interface Instructions, Programming Interface). They are used to transport instructions external to ring buffers.

Programming Note	
Context:	Command batch buffers in memory objects
Batch buffers can be tagged with any memory type when produced by IA. If WB memory type is used, it should be tagged with "snoop required" for GPU consumption (to trigger snoop from CPU cache).	

Programming Note	
Context:	Command batch buffers in memory objects
The batch buffer must be QWord aligned and a multiple of QWords in length. The ending address is the address of the last valid QWord in the buffer. The length of any single batch buffer is "virtually unlimited" (i.e., could theoretically be 4GB in length).	

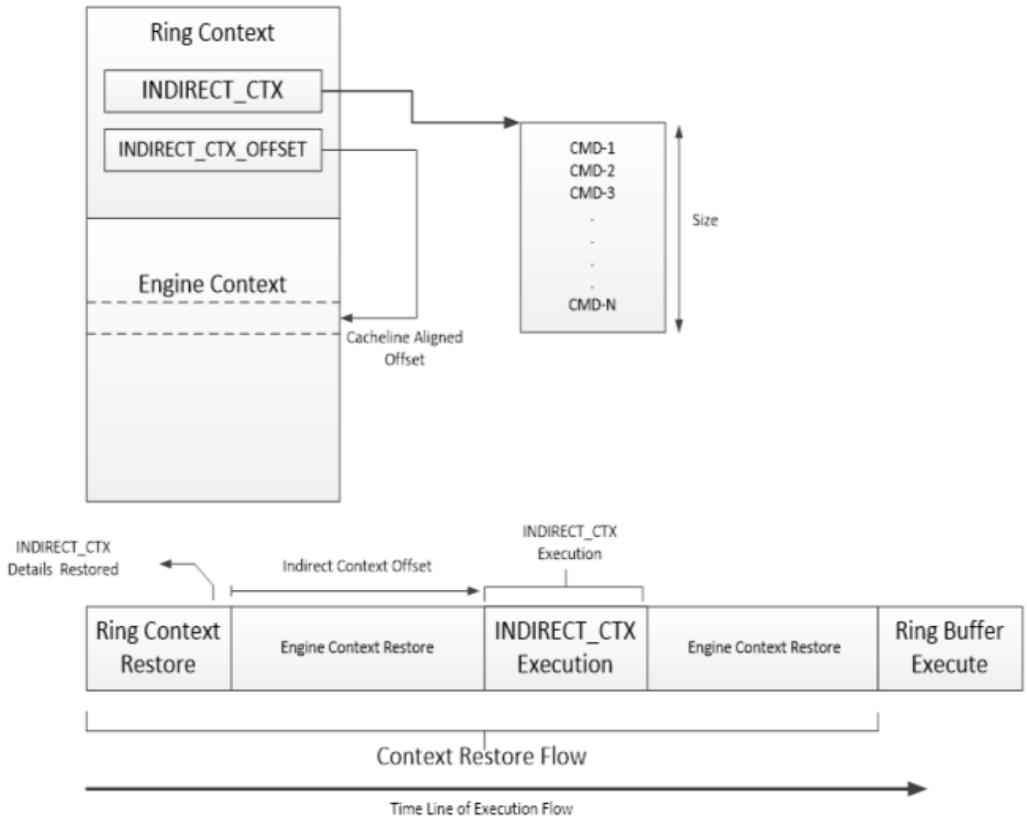
Workaround Batch Buffers

A Workaround batch buffer is a set of commands that is run by the hardware during context load time. i.e when Command Streamer hardware is restoring the state of the context that it is about to execute (before execution of any command in the ring buffer). The Workaround batch buffer uses pointers to command buffers that are setup by the Kernel Mode driver in the context image.

Two flavors of Workaround batch buffers are supported by the hardware. They differ in terms of exactly when the supplied workaround commands are executed in the context restore process. The mechanisms supported are:

Indirect Context Pointer (INDIRECT_CTX)

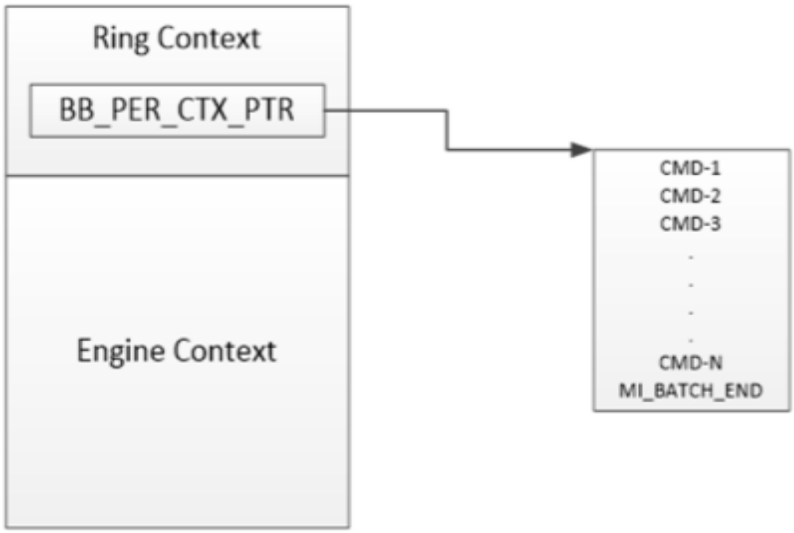
As shown in the figure below, this workaround buffer can be invoked at any cacheline aligned offset in the engine context.

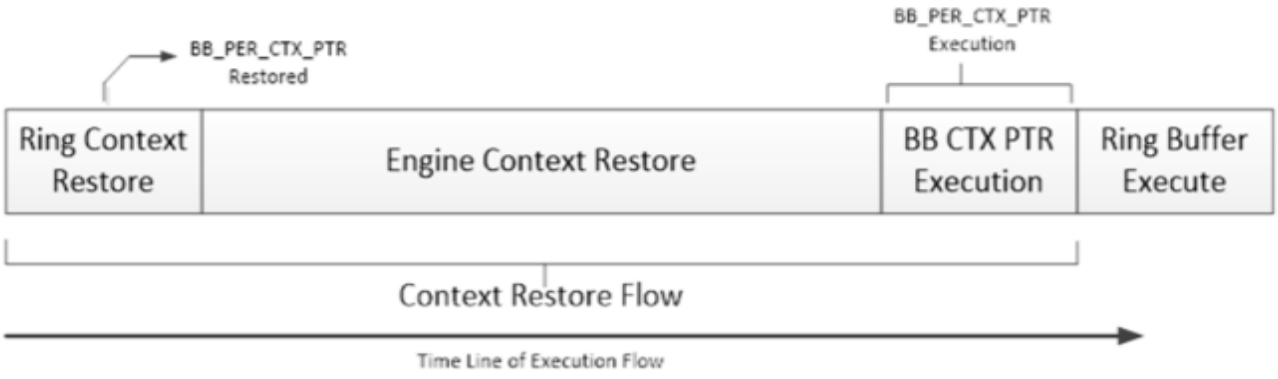


Command streamer, when enabled through "INDIRECT_CTX" provides a mechanism to pause executing context restore on a given cacheline aligned offset in the engine context image and execute a command sequence from a command buffer before resuming context restore flow. This command buffer execution during context restore is referred to as "Indirect Context Pointer" execution. The start address and the size of the command buffer to be executed is provided through "INDIRECT_CTX" register and the offset in the engine context restore is provided through "INDIRECT_CTX_OFFSET". "INDIRECT_CTX" and "INDIRECT_CTX_OFFSET" registers are part of the context image and gets restored as part of the given context's context restore flow, these registers are part of the ring context image which are prior to engine context restore and hence the requirement of the offset being in engine context restore. "Indirect Context Pointer" is always in the GGTT address space of the virtual function or physical function from which the context is submitted. "Indirect context pointer" can be programmed differently for each context providing flexibility to execute different command sequence as part of "Indirect Context Pointer" execution during context restore flows.

Post Context Restore Workaround Batch Buffer

As shown in the figure, this workaround buffer is invoked at the end of the context restore.





Command streamer, when enabled through "BB_PER_CTX_PTR" provides a mechanism to execute a command sequence from a batch buffer at the end of the context restore flow during context switch process. This batch buffer is referred to as "Context Restore Batch Buffer". The batch start address for the "Context Restore Batch Buffer" gets programmed through "BB_PER_CTX_PTR", which is part of the context image and gets restored as part of the given context's context restore flow. "Context Restore Batch Buffer" execution begins (like a regular batch buffer) after the completion of fetching and execution of all the commands for the context restore flow. "Context Restore Batch Buffer" execution ends on executing MI_BATCH_BUFFER_END in the command sequence. "Context Restore Batch Buffer" is always in the GGTT address space of the virtual function or physical function from which the context is submitted. "BB_PER_CTX_PTR" can be programmed differently for every context giving flexibility to execute different command sequence (batch buffers) as part of "Context Restore Batch Buffer" execution or can be programmed to disable execution of the "Context Restore Batch Buffer" for a given context.

This mechanism is especially helpful in programming a set of commands/state that has to be always executed prior to executing a workload from a context every time it is submitted to HW for execution. Limited capability is built for "Context Restore Batch Buffer" unlike a regular MI_BATCH_BUFFER_START due to envisioned usage model, refer BB_PER_CTX_PTR for detailed programming notes.

Command Streamer Command Formats

This section describes the general format of the command streamer commands.

Command streamer commands are defined with various formats. The first DWord of all commands is called the *header* DWord. The header contains the only field common to all commands, the *client* field that determines the device unit that processes the command data. The Command Parser examines the client field of each command to condition the further processing of the command and route the command data accordingly.

Command streamer commands vary in length, though are always multiples of DWords. The length of a command is either:

- Implied by the client/opcode
- Fixed by the client/opcode yet included in a header field (so the Command Parser explicitly knows how much data to copy/process)
- Variable, with a field in the header indicating the total length of the command

Note that command *sequences* require QWord alignment and padding to QWord length to be placed in Ring and Batch Buffers.

The following subsections provide a brief overview of the command streamer commands by client type provides a diagram of the formats of the header DWords for all commands. Following that is a list of command mnemonics by client type.

Command Header

Engine Command Header Format

Type	Bits	
	31:29	28:0
Memory Interface (MI)	000	
Engine Command	010, 011	
Reserved	001, 100, 101, 110, 111	

Memory Interface Commands

Memory Interface (MI) commands are basically those commands which do not require processing by the 2D or 3D Rendering/Mapping engines. The functions performed by these commands include:

- Control of the command stream (e.g., Batch Buffer commands, breakpoints, ARB On/Off, etc.)
- Hardware synchronization (e.g., flush, wait-for-event)
- Software synchronization (e.g., Store DWORD, report head)
- Graphics buffer definition (e.g., Display buffer, Overlay buffer)
- Miscellaneous functions

All of the following commands are defined in *Memory Interface Commands*.

Memory Interface Commands for RCP

Opcode (28:23)	Command	Pipes
1 DWord		
00h	MI_NOOP	All
01h	MI_SET_PREDICATE	All
02h	MI_USER_INTERRUPT	All
03h	MI_WAIT_FOR_EVENT	Render, Blitter
04h	MI_WAIT_FOR_EVENT_2	Render, Blitter
05h	MI_ARB_CHECK	All
07h	MI_REPORT_HEAD	All
08h	MI_ARB_ON_OFF	All except Blitter
0Ah	MI_BATCH_BUFFER_END	All
0Bh	MI_SUSPEND_FLUSH	All
0Ch	MI_PREDICATE	Render
2+ DWord		
10h	Reserved	
12h	MI_LOAD_SCAN_LINES_INCL	Render and Blitter

Opcode (28:23)	Command	Pipes
13h	MI_LOAD_SCAN_LINES_EXCL	Render and Blitter
14h	MI_DISPLAY_FLIP	Render and Blitter
15h	Reserved	
17h	Reserved	
18h	MI_SET_CONTEXT	Render
1Ah	MI_MATH	All
1Bh	MI_SEMAPHORE_SIGNAL	All
1Ch	MI_SEMAPHORE_WAIT	All
1Dh	MI_FORCE_WAKEUP	All except Render
1Fh	Reserved	
Store Data		
20h	MI_STORE_DATA_IMM	All
21h	MI_STORE_DATA_INDEX	All
22h	MI_LOAD_REGISTER_IMM	All
23h	MI_UPDATE_GTT	All
24h	MI_STORE_REGISTER_MEM	All
26h	MI_FLUSH_DW	All except Render
27h	MI_CLFLUSH	Render
29h	MI_LOAD_REGISTER_MEM	All
2Ah	MI_LOAD_REGISTER_REG	All
2Eh	MI_MEM_TO_MEM	All
2Fh	MI_ATOMIC	All
Ring/Batch Buffer		
30h	Reserved	
31h	MI_BATCH_BUFFER_START	Render
32h-35h	Reserved	
36h	MI_CONDITIONAL_BATCH_BUFFER_END	All
37h-38h	Reserved	
39h	Reserved	All
39h-3Fh	Reserved	

Execution Control Infrastructure

This section describes the hardware infrastructure that can be used to control command execution.

Watchdog Timers

Watchdog Counter Control

The Watchdog Counter Control determines if the watchdog is enabled, disabled and count mode. The watchdog is enabled is when the value of the register [30:0] is equal to zero([30:0] = 'd0). If enabled, then the Watchdog Counter is allowed to increment. The watchdog is disabled is when the value of the register [30:0] is equal to one where only bit zero is a value of '1'([30:0] = 0x00000001). If disabled, then the value of Watchdog Counter is reset to a value of zero. Bit 31, specifies the counting mode. If bit 31 is zero, then we will count based timestamp toggle(refer to Reported Timestamp Count register for toggle time). If bit 31 is one, then we will count every ungated GPU clock.

This register is context saved as part of engine context.

Watchdog Counter Threshold

If the Watchdog Counter Threshhold is equal to Watchdog Counter, then the interrupt bit is set in the IIR(bit 6) and the Watchdog Counter is reset to zero.

This register is context saved as part of engine context.

Watchdog Counter

The Watchdog Counter is the count value of the watchdog timer. The Counter can be reset due to the Watchdog Counter Control being disabled or being equal to the Watchdog Counter Threshhold. The increment of the Watchdog counter is enabled when the Watchdog Counter Control is enabled and the current context is valid and execlist is enabled which includes the time to execute, flush and save the context.

The increment of the Watchdog counter is under the following conditions:

- Watchdog timer is enabled.
- Context is valid

The increment granularity is based controlled by Watchdog Counter Control mode(bit 31).

This register is not context saved and restored.

Predication

Predicate Render Registers

Register
MI_PREDICATE_SRC0 - Predicate Rendering Temporary Register0
MI_PREDICATE_SRC1 - Predicate Rendering Temporary Register1
MI_PREDICATE_DATA - Predicate Rendering Data Storage
MI_PREDICATE_RESULT - Predicate Rendering Data Result
MI_PREDICATE_RESULT_1 - Predicate Rendering Data Result 1
MI_PREDICATE_RESULT_2 - Predicate Rendering Data Result 2

MI_SET_PREDICATE

MI_SET_PREDICATE is a command that allows the driver to conditionally execute or skip a command during execution time, as detailed in the instruction definition:

The following is a list of commands that can be programmed when the PREDICATE ENABLE field in MI_SET_PREDICATE allows predication. Commands not listed here will have undefined behavior when executed with predication enabled:

Command
3DSTATE_URB_VS
3DSTATE_URB_HS
3DSTATE_URB_DS
3DSTATE_URB_GS
3DSTATE_PUSH_CONSTANT_ALLOC_VS
3DSTATE_PUSH_CONSTANT_ALLOC_HS
3DSTATE_PUSH_CONSTANT_ALLOC_DS
3DSTATE_PUSH_CONSTANT_ALLOC_GS
3DSTATE_PUSH_CONSTANT_ALLOC_PS
MI_LOAD_REGISTER_IMM
MI_STORE_DATA_IMM
3DSTATE_WM_HZ_OP
MEDIA_VFE_STATE
MEDIA_OBJECT
MEDIA_OBJECT_WALKER
MEDIA_INTERFACE_DESCRIPTOR_LOAD

MI_PREDICATE

The MI_PREDICATE command is used to control the Predicate state bit, which in turn can be used to enable/disable the processing of 3DPRIMITIVE commands.

MI_PREDICATE

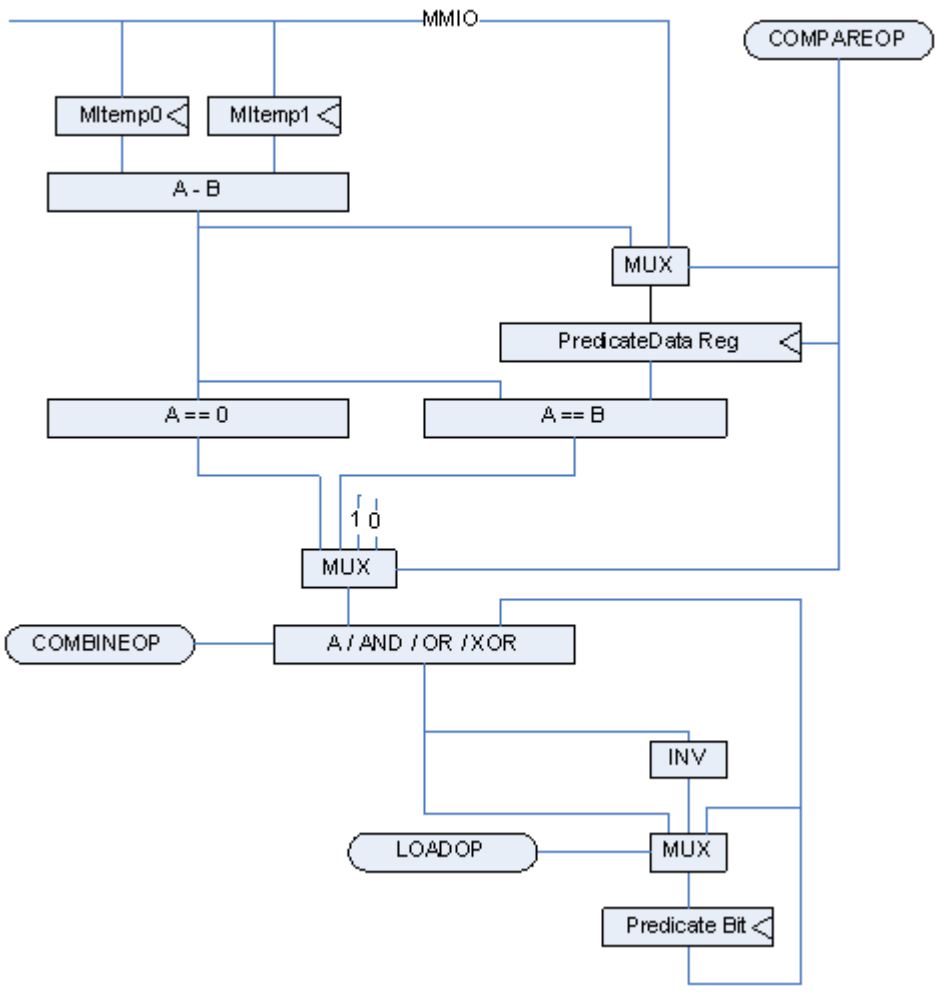
Predicated Rendering Support in HW

DX10 defines predicated rendering, where sequences of rendering commands can be discarded based on the result of a previous predicate test. A new state bit, Predicate, has been added to the command stream. In addition, a PredicateEnable bit is added to 3DPRIMITIVE. When the PredicateEnable bit is set, the command is ignored if the Predicate state bit is set.

A new command, MI_PREDICATE, is added. It contains several control fields which specify how the Predicate bit is generated.

Refer to the diagram below and the command description (linked above) for details.

MI_PREDICATE Function



MI_LOAD_REGISTER_MEM commands can be used to load the MItemp0, MItemp1, and PredicateData registers prior to MI_PREDICATE. To ensure the memory sources of the MI_LOAD_REGISTER_MEM commands are coherent with previous 3D_PIPECONTROL store-DWord operations, software can use the new **Pipe Control Flush Enable** bit in the PIPE_CONTROL command.

CS ALU Programming and Design

Command streamer implements a rudimentary ALU which supports basic Arithmetic (Addition and Subtraction) and logical operations (AND, OR, XOR) on two 64bit operands. ALU has two 64bit registers at the input SRCA and SRCB to which the operands should be loaded on which operations will be performed and outputted to a 64 bit Accumulator. Zero Flag and Carry Flag are set based on accumulator output.

Access to this ALU is thru the **MI_MATH** command.

CS_GPR - Command Streamer General Purpose Registers

Following are Command Streamer General Purpose Registers:

CS_GPR - General Purpose Register

Command Streamer (CS) ALU Programming

The command streamer implements a rudimentary Arithmetic Logic Unit (ALU) which supports basic arithmetic (Addition and Subtraction) and logical operations (AND, OR, XOR) on two 64-bit operands.

The ALU has two 64-bit registers at the input, SRCA and SRCB, to which source operands are loaded. The ALU result is written to a 64-bit accumulator. The Zero Flag and Carry Flag are assigned based on the accumulator output.

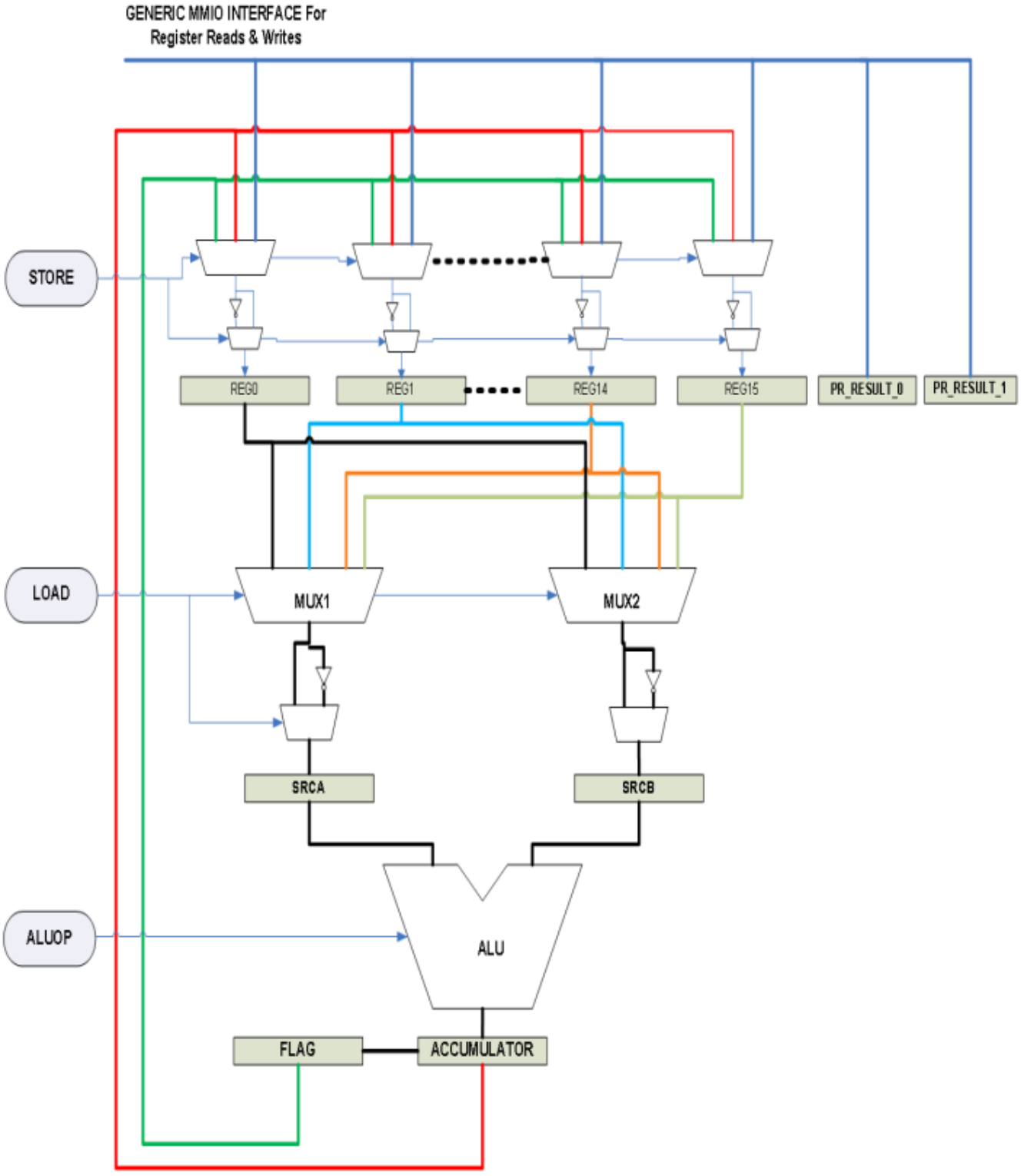
See the ALU Programming section in the Render Engine Command Streamer, for a description of the ALU programming model. Programming model is the same for all command streamers that support ALU, but each command streamer uses its own MMIO address range to address the registers. The following subsections describe the ALU registers and the programming details.

CS ALU Programming and Design

Generic Purpose Registers

Command streamer implements sixteen 64 bit General Purpose Registers which are MMIO mapped. These registers can be accessed similar to any other MMIO mapped registers through LRI, SRM, LRR, LRM or CPU access path for reads and writes. These registers will be labeled as R0, R1, ... R15 throughout the discussion. Refer table in the B-spec update section mapping these registers to corresponding MMIO offset. A selected GPR register can be moved to SRCA or SRCB register using "LOAD" instruction. Outputs of the ALU, Accumulator, ZF and CF can be moved to any of the GPR using "STORE" instruction.

ALU BLOCK Diagram



Instruction Set

The instructions supported by the ALU can be broadly categorized into three groups:

- To move data from GPR to SRCA/SRCB - LOAD instruction.
- To move data from ACCUMULATOR/CF/ZF to GPR - STORE Instruction.
- To do arithmetic/Logical operations on SRCA and SRCB of ALU - ADD/SUB/AND/XOR/OR. Note: Accumulator is loaded with value of SRCA - SRCB on a subtraction.

Instruction Format

Each instruction is one Dword in size and consists of an ALU OPCODE, OPERAND1 and OPERAND2 in the format shown below.

ALU OPCODE	Operand-1	Operand-2
12 bits	10 bits	10 bits

NOOP and FENCE Operations

NOOP operation has does no operation but will delay and add operation idle time between commands.

Opcode	Operand1	Operand2
31:20	19:10	9:0
NOOP	N/A	N/A

Arithmetic/Logical Operations

ADD, SUB, AND, OR, and XOR are the Arithmetic and Logical operations supported by Arithmetic Logic Unit (ALU). When opcode corresponding to a logical operation is performed on SRCA and SRCB, the result is sent to ACCUMULATOR (ACCU), CF and ZF. Note that ACCU is 64-bit register. A NOOP when submitted to the ALU doesn't do anything, it is meant for creating bubble or kill cycles.

Opcode	Operand1	Operand2
31:20	19:10	9:0
ADD	N/A	N/A
SUB*	N/A	N/A
OR	N/A	N/A
XOR	N/A	N/A

*Note: Accumulator is loaded with value of SRCA - SRCB on a subtraction.

LOAD Operation

The LOAD instruction moves the content of the destination register (Operand2) into the source register (Operand1). The destination register can be any of the GPR (R0, R1, ..., R15) and the source registers are SRCA and SRCB of the ALU. This is the only means SRCA and SRCB can be programmed.

LOAD has different flavors, wherein one can load the inverted version of the source register into the destination register or a hard coded value of all Zeros and All ones.

```
// Loads any of Reg0 to Reg15 into the SRCA or SRCB registers of ALU.

    LOAD <SRCA, SRCB>, <REG0..REG15>

// Loads inverted (bit wise) value of the mentioned Reg0 to 15 into SRCA or SRCB registers of
ALU.

    LOADINV <SRCA, SRCB>, <REG0..REG15>

// Loads "0" into SRCA or SRCB

    LOAD0 <SRCA, SRCB>

// Loads "1" into SRCA or SRCB

    LOAD1 <SRCA, SRCB>
```

Opcode	Operand1	Operand2
31:20	19:10	9:0
LOAD	SRCA/SRCB	R0,R1..R15
LOADINV	SRCA/SRCB	R0,R1..R15
LOAD0	SRCA/SRCB	N/A
LOAD1	SRCA/SRCB	N/A

STORE Operation

The STORE instruction moves the content of the destination register (Operand2) into the source register (Operand1). The source register can be accumulator (ACCU), CF or ZF. STORE has different flavors, wherein one can load the inverted version of the source register into destination register via STOREINV. When CF or ZF are stored, the same value is replicated on all 64 bits.

```
// Loads ACCMULATOR or Carry Flag or Zero Flag in to any of the generic registers
// Reg0 to Reg16. In case of CF and ZF same value is replicated on all the 64 bits.

    STORE   <R0.. R15>, <ACCU, CF, ZF >

// Loads inverted (ACCMULATOR or Carry Flag or Zero Flag) in to any of the
// generic registers Reg0 to Reg15.

    STOREINV <R0.. R15>, <ACCU, CF, ZF>
```

Opcode	Operand1	Operand2
31:20	19:10	9:0
STORE	R0,R1..R15	ACCU/ZF/CF
STOREINV	R0,R1..R15	ACCU/ZF/CF

Summary for ALU

Total Opcodes Supported: 12

Total Addressable Registers as source or destination: 21

- 16 GPR (R0, R1 ...R15)
- 1 ACCU
- 1ZF
- 1CF
- SRCA, SRCB

Summary of Instructions Supported

31 20	19 10	9 0
Opcode	**Operand1**	**Operand2**
LOAD	SRCA/SRCB	REG0..REG15
LOADINV	SRCA/SRCB	REG0..REG15
LOAD0	SRCA/SRCB	N/A
LOAD1	SRCA/SRCB	N?A
ADD	N/A	N/A
SUB	N/A	N/A
AND	N/A	N/A
OR	N/A	N/A
XOR	N/A	N/A
NOOP	N/A	N/A
STORE	REG0..REG15	ACCU/CF/ZF
STOREINV	REG0..REG15	ACCU/CF/ZF

Table for ALU OPCODE Encodings

In the above mentioned table, ALU Opcode Encodings look like random numbers. The rationale behind those encodings is because the ALU Opcode is further broken down into sub-sections for ease-of-design implementation.

PREFIX		**OPCODE**		**SUBOPCODE**	
11	10	9	7	6	0
PREFIX VALUE		**Description**			
0		Regular			
1		Invert			
OPCODE VALUE		**Description**			
0		NOOP			

PREFIX	OPCODE		SUBOPCODE	
1	LOAD			
2	ALU			
3	STORE			

ALU OPCODE	OPCODE ENCODING	PREFIX(11:10)	OPCODE(9:7)	SUB-OPCODE(6:0)
NOOP	0x000	0	0	0
LOAD	0x080	0	1	0
LOADINV	0x480	1	1	0
LOAD0	0x081	0	1	1
LOAD1	0x481	1	1	1
ADD	0x100	0	2	0
SUB	0x101	0	2	1
AND	0x102	0	2	2
OR	0x103	0	2	3
XOR	0x104	0	2	4
STORE	0x180	0	3	0
STOREINV	0x580	1	3	0

Table for Register Encodings

Register	Register Encoding
R0	0x0
R1	0x1
R2	0x2
R3	0x3
R4	0x4
R5	0x5
R6	0x6
R7	0x7
R8	0x8
R9	0x9
R10	0xa
R11	0xb
R12	0xc
R13	0xd
R14	0xe
R15	0xf
SRCA	0x20

Register	Register Encoding
SRCB	0x21
ACCU	0x31
ZF	0x32
CF	0x33

MI Commands for Graphics Processing Engines

This chapter lists the MI Commands that are supported by Generic Command Streamer Front End implemented.

Command
MI_NOOP
MI_ARB_CHECK
MI_ARB_ON_OFF
MI_BATCH_BUFFER_START
MI_CONDITIONAL_BATCH_BUFFER_END
MI_DISPLAY_FLIP(Graphics/Copy Only)
MI_LOAD_SCAN_LINES_EXCL(Graphics/Copy Only)
MI_LOAD_SCAN_LINES_INCL(Graphics/Copy Only)
MI_MATH
MI_REPORT_HEAD
MI_STORE_DATA_IMM
MI_STORE_DATA_INDEX
MI_ATOMIC
MI_COPY_MEM_MEM
MI_LOAD_REGISTER_REG
MI_LOAD_REGISTER_MEM
MI_STORE_REGISTER_MEM
MI_USER_INTERRUPT
MI_WAIT_FOR_EVENT
MI_SEMAPHORE_SIGNAL
MI_SEMAPHORE_WAIT
MI_SET_PREDICATE
MI_UPDATE_GTT

Register Access and User Mode Privileges

This section describes access to the MMIO internal to the GPU and funny I/O and how to access the ranges. Command streamer limits accesses for commands that are executed out of a PPGTT batch buffer. This is also referred to a non-privilege command buffer.

Below are the Base Addresses of each command streamer and engine blocks. While this is not all the ranges, it is the ones used to reference which registers are accessible or restricted by command streamer.

Unit	MMIO Base Offset	Description
RCS	0x2000	Render Command Streamer
POCS	0x18000	Position Command Streamer
BCS	0x22000	Blitter Command Streamer
CCS0	0x1A000	Compute Command Streamer 0
VCS/MFC	0x1C0000	Video Command Streamer 0
VCS1/MFC	0x1C4000	Video Command Streamer 1
VCS2/MFC	0x1D0000	Video Command Streamer 2
VCS3/MFC	0x1D4000	Video Command Streamer 3
VCS4/MFC	0x1E0000	Video Command Streamer 4
VCS5/MFC	0x1E4000	Video Command Streamer 5
VCS6/MFC	0x1F0000	Video Command Streamer 6
VCS7/MFC	0x1F4000	Video Command Streamer 7
VECS/MFC	0x1C8000	Video Enhancement Command Streamer 0
VECS1	0x1D8000	Video Enhancement Command Streamer 1
VECS2	0x1E8000	Video Enhancement Command Streamer 2
VECS3	0x1F8000	Video Enhancement Command Streamer 3
AV1/VDBOX0	0x1C2B00	AV1/Video Decode Block
AV1/VDBOX1	0x1C6B00	
AV1/VDBOX2	0x1D2B00	
AV1/VDBOX3	0x1D6B00	
AV1/VDBOX4	0x1E2B00	
AV1/VDBOX5	0x1E6B00	
AV1/VDBOX6	0x1F2B00	
AV1/VDBOX7	0x1F6B00	
HEVC	0x1C2800	
HEVC1	0x1C6800	
HEVC2	0x1D2800	
HEVC3	0x1D6800	
HEVC4	0x1E2800	
HEVC5	0x1E6800	

Unit	MMIO Base Offset	Description
HEVC6	0x1F2800	
HEVC7	0x1F6800	
VDENC	0x1c2d00	
VDENC1	0x1c6d00	
VDENC2	0x1d2d00	
VDENC3	0x1d6d00	
VDENC4	0x1e2d00	
VDENC5	0x1e6d00	
VDENC6	0x1f2d00	
VDENC7	0x1f6d00	

Read Only User Mode Privilege MMIO Access

The tables below specify the offsets that are allowed for MMIO reads within a non-privileged batch buffer(PPGTT). This is in addition to what is already whitelisted for writes in the User Mode Privileged Commands section. Refer to Register Access and User Mode Privileges section for Base address for the below offsets.

CS means all command streamers.

Read Only Whitelist

Name	Base Address (default=none)	MMIO Offset (hex)	Size in DW
All Command Streamers			
OAG_PERF_<x>		2700	64
OAG_PERF_<x>		2B00	320
OAG_PERF_<x>		D900	192
OASTATUS		DAFC	1
OAHEADPTR		DB00	1
OATAILPTR		DB04	1
GFXREG_GT		145040	7
GFXREG_IA		145828	13
GFXREG_IO		145928	24
RP_STATUS0		A01C	1
GFXREG_GT_GFX_RC6		138108	1
GFXREG_GT_GFX_RC6P		13810C	1
Perf Profiler Timer Reg		D00	1
Graphics CS(Including POSH)			
GPU_TIMESTAMP		2358	2
GPU_TIMESTAMP		18358	2

Name	Base Address (default=none)	MMIO Offset (hex)	Size in DW
CS_ENGINE_ID		208C	1
CS_ENGINE_ID		1808C	1
OAR_PERF_<x>		2800	192
GFXREG_UNSLICE_FF_CTRL_FLC_THRSHLD1		A288	1
GFXREG_UNSLICE_FF_CTRL_FLC_THRSHLD2		A28C	1
GFXREG_UNSLICE_FF_COUNT1		A538	1
GFXREG_UNSLICE_FF_COUNT2		A53C	1
GFXREG_RPPREVUP		A058	1
GFXREG_RPPREVDN		A064	1
GFXREG_RPUPEI		A068	1
GFXREG_RPDNEI		A06C	1
GFXREG_GT_GFX_RC6		138108	1
GFXREG_GT_GFX_RC6P		13810C	1
CS_CTX_TIMESTAMP		23A8	1
ComputeCS			
GPU_TIMESTAMP	CCS	358	2
CS_ENGINE_ID	CCS	8C	1
GFXREG_UNSLICE_FF_CTRL_FLC_THRSHLD1		A288	1
GFXREG_UNSLICE_FF_CTRL_FLC_THRSHLD2		A28C	1
GFXREG_UNSLICE_FF_COUNT1		A538	1
GFXREG_UNSLICE_FF_COUNT2		A53C	1
GFXREG_RPPREVUP		A058	1
GFXREG_RPPREVDN		A064	1
GFXREG_RPUPEI		A068	1
GFXREG_RPDNEI		A06C	1
GFXREG_GT_GFX_RC6		138108	1
GFXREG_GT_GFX_RC6P		13810C	1
CS_CTX_TIMESTAMP	CCS	3A8	1
OAC_PERF_<x>		15000	160
BlitterCS			
GPU_TIMESTAMP		22358	2
CS_ENGINE_ID		2208C	1
RP_STATUS0		A01C	1
PERFCNT1_LSB		91B8	1
PERFCNT1_MSB		91BC	1
PERFCNT2_LSB		91C0	1
PERFCNT2_MSB		91C4	1

Name	Base Address (default=none)	MMIO Offset (hex)	Size in DW
GFXREG_GT		145040	7
GFXREG_IA		145828	13
GFXREG_IO		145928	24
GFXREG_UNSLICE_FF_CTRL_FLC_THRSHLD1		A288	1
GFXREG_UNSLICE_FF_CTRL_FLC_THRSHLD2		A28C	1
GFXREG_UNSLICE_FF_COUNT1		A538	1
GFXREG_UNSLICE_FF_COUNT2		A53C	1
GFXREG_RPPREVUP		A058	1
GFXREG_RPPREVDN		A064	1
GFXREG_RPUPEI		A068	1
GFXREG_RPDNEI		A06C	1
VideoCS			
GPU_TIMESTAMP	VCS	358	2
PERFCNT1_LSB		91B8	1
PERFCNT1_MSB		91BC	1
PERFCNT2_LSB		91C0	1
PERFCNT2_MSB		91C4	1
GFXREG_UNSLICE_FF_CTRL_FLC_THRSHLD1		A288	1
GFXREG_UNSLICE_FF_CTRL_FLC_THRSHLD2		A28C	1
GFXREG_UNSLICE_FF_COUNT1		A538	1
GFXREG_UNSLICE_FF_COUNT2		A53C	1
GFXREG_RPPREVUP		A058	1
GFXREG_RPPREVDN		A064	1
GFXREG_RPUPEI		A068	1
GFXREG_RPDNEI		A06C	1
CS_ENGINE_ID	VCS	8C	1
VideoEnhancementCS			
GPU_TIMESTAMP	VECS	358	2
PERFCNT1_LSB		91B8	1
PERFCNT1_MSB		91BC	1
PERFCNT2_LSB		91C0	1
PERFCNT2_MSB		91C4	1
GFXREG_UNSLICE_FF_CTRL_FLC_THRSHLD1		A288	1
GFXREG_UNSLICE_FF_CTRL_FLC_THRSHLD2		A28C	1
GFXREG_UNSLICE_FF_COUNT1		A538	1
GFXREG_UNSLICE_FF_COUNT2		A53C	1
GFXREG_RPPREVUP		A058	1

Name	Base Address (default=none)	MMIO Offset (hex)	Size in DW
GFXREG_RPPREVDN		A064	1
GFXREG_RPUPEI		A068	1
GFXREG_RPDNEI		A06C	1
GFXREG_GT_GFX_RC6		138108	1
GFXREG_GT_GFX_RC6P		13810C	1
CS_ENGINE_ID	VECS	8C	1

User Mode Privileged Commands

A subset of the commands are privileged. These commands may be issued only from a privileged batch buffer or directly from a ring. Batch buffers in GGTT memory space are privileged and batch buffers in PPGTT memory space are non-privileged. On parsing privileged command from a non-privileged batch buffer, a Command Privilege Violation Error is flagged and the command is dropped. Command Privilege Violation Error is logged in Error identity register of command streamer which gets propagated as "Command Parser Violation Error" interrupt to SW. Privilege access violation checks in HW can be disabled by setting "Privilege Check Disable" bit in GFX_MODE register. When privilege access checks are disabled HW executes the Privilege command as expected.

User Mode Privileged Commands

User Mode Privileged Command	Function in Non-Privileged Batch Buffers	Source
MI_UPDATE_GTT	Command is converted to NOOP.	*CS
MI_STORE_DATA_IMM	Command is converted to NOOP if **Use Global GTT** is enabled.	*CS
MI_STORE_DATA_INDEX	Command is converted to NOOP.	*CS
MI_STORE_REGISTER_MEM	Register read is always performed. Memory update is dropped if **Use Global GTT** is enabled.	*CS
MI_BATCH_BUFFER_START	Command when executed from a batch buffer can set its "Privileged" level to its parent batch buffer or lower. Chained or Second level batch buffer can be "Privileged" only if the parent or the initial batch buffer is "Privileged". This is HW enforced.	*CS
MI_LOAD_REGISTER_IMM	Command is converted to NOOP if the register accessed is privileged.	*CS
MI_LOAD_REGISTER_MEM	Command is converted to NOOP if **Use Global GTT** is enabled. Command is converted to NOOP if the register accessed is privileged.	*CS
MI_LOAD_REGISTER_REG	Register write to a **Privileged Register** is discarded.	*CS
MI_REPORT_PERF_COUNT	Command is converted to NOOP if **Use Global GTT** is	Render CS

User Mode Privileged Command	Function in Non-Privileged Batch Buffers	Source
	enabled.	
PIPE_CONTROL	Still send flush down, Post-Sync Operation is NOOP if **Use Global GTT** or Use "Store Data Index" is enabled. Post-Sync Operation LRI to **Privileged Register** is discarded.	Render CS, ComputeCS
MI_ATOMIC	Command is converted to NOOP if **Use Global GTT** is enabled.	*CS
MI_COPY_MEM_MEM	Command is converted to NOOP if **Use Global GTT** is used for source or destination address.	*CS
MI_SEMAPHORE_WAIT	Command is converted to NOOP if **Use Global GTT** is enabled.	*CS
MI_ARB_ON_OFF	Command is converted to NOOP.	*CS
MI_DISPLAY_FLIP	Command is converted to NOOP.	*CS
MI_CONDITIONAL_BATCH_BUFFER_END	Command is converted to NOOP if **Use Global GTT** is enabled.	*CS
MI_FLUSH_DW	Still send flush down, Post-Sync Operation is converted to NOOP if **Use Global GTT** or Use "Store Data Index" is enabled.	Blitter CS, Video CS, Video Enhancement CS

Parsing one of the commands in the table above from a non-privileged batch buffer flags an error and converts the command to a NOOP.

The tables below list the non-privileged registers that can be written to from a non-privileged batch buffer executed from various command streamers.

The tables below also are part of the allowed registers allowed to be read by a non-Privileged(PPGTT) batch buffer. Refer to Read Only User Mode Privilege MMIO Access section for the rest of the allowable registers for read access.

User Mode Non-Privileged Registers for Render Command Streamer (RCS) and POSH Command Streamer (POCS)

MMIO Name	MMIO Offset	Size in DWords
Cache_Mode_0	0x7000	1
Cache_Mode_1	0x7004	1
GT_MODE	0x7008	1
NOPID	0x2094	1
NOPID (POCS)	0x18094	1
INSTPM	0x20C0	1
INSTPM (POCS)	0x180C0	1

MMIO Name	MMIO Offset	Size in DWords
IA_VERTICES_COUNT	0x2310	2
IA_VERTICES_COUNT (POSH)	0x18310	2
IA_PRIMIVTIVES_COUNT	0x2318	2
IA_PRIMITIVES_COUNT (POSH)	0x18318	2
VS_INVOCATION_COUNT	0x2320	2
VS_INVOCATION_COUNT (POSH)	0x18320	2
HS_INVOCATION_COUNT	0x2300	2
DS_INVOCATION_COUNT	0x2308	2
GS_INVOCATION_COUNT	0x2328	2
GS_PRIMITIVES_COUNT	0x2330	2
SO_NUM_PRIMS_WRITTEN0	0x5200	2
SO_NUM_PRIMS_WRITTEN1	0x5208	2
SO_NUM_PRIMS_WRITTEN2	0x5210	2
SO_NUM_PRIMS_WRITTEN3	0x5218	2
SO_PRIM_STORAGE_NEEDED0	0x5240	2
SO_PRIM_STORAGE_NEEDED1	0x5248	2
SO_PRIM_STORAGE_NEEDED2	0x5250	2
SO_PRIM_STORAGE_NEEDED3	0x5258	2
SO_WRITE_OFFSET0	0x5280	1
SO_WRITE_OFFSET1	0x5284	1
SO_WRITE_OFFSET2	0x5288	1
SO_WRITE_OFFSET3	0x528C	1
CL_INVOCATION_COUNT	0x2338	2
CL_INVOCATION_COUNT (POSH)	0x18338	2
CL_PRIMITIVES_COUNT	0x2340	2
CL_PRIMITIVES_COUNT (POSH)	0x18340	2
PS_INVOCATION_COUNT	0x2348	2
PS_DEPTH_COUNT	0x2350	2
PS_INVOCATION_COUNT_0	0x22C8	2
PS_DEPTH_COUNT _0	0x22D8	2
PS_INVOCATION_COUNT_1	0x22F0	2
PS_DEPTH_COUNT _1	0x22F8	2
PS_INVOCATION_COUNT_2	0x2448	2
PS_DEPTH_COUNT_2	0x2450	2
PS_INVOCATION_COUNT_3	0x2458	2
PS_DEPTH_COUNT_3	0x2460	2
PS_INVOCATION_COUNT_4	0x2468	2

MMIO Name	MMIO Offset	Size in DWords
PS_DEPTH_COUNT_4	0x2470	2
PS_INVOCATION_COUNT_5	0x24A0	2
PS_DEPTH_COUNT_5	0x24A8	2
PS_INVOCATION_COUNT_6	0x25D0	2
PS_DEPTH_COUNT_6	0x25B0	2
PS_INVOCATION_COUNT_7	0x25D8	2
PS_DEPTH_COUNT_7	0x25B8	2
CPS_INVOCATION_COUNT	0x2478	2
GPUGPU_DISPATCHDIMX	0x2500	1
GPUGPU_DISPATCHDIMY	0x2504	1
GPUGPU_DISPATCHDIMZ	0x2508	1
MI_PREDICATE_SRC0	0x2400	1
MI_PREDICATE_SRC0 (POSH)	0x18400	1
MI_PREDICATE_SRC0	0x2404	1
MI_PREDICATE_SRC0 (POSH)	0x18404	
MI_PREDICATE_SRC1	0x2408	1
MI_PREDICATE_SRC1 (POSH)	0x18408	
MI_PREDICATE_SRC1	0x240C	1
MI_PREDICATE_SRC1 (POSH)	0x1840C	
MI_PREDICATE_DATA	0x2410	1
MI_PREDICATE_DATA (POSH)	0x18410	
MI_PREDICATE_DATA	0x2414	1
MI_PREDICATE_DATA (POSH)	0x18414	
MI_PREDICATE_RESULT	0x2418	1
MI_PREDICATE_RESULT (POSH)	0x18418	
MI_PREDICATE_RESULT_1	0x241C	1
MI_PREDICATE_RESULT_1 (POSH)	0x1841C	
MI_PREDICATE_RESULT_2	0x23BC	1
MI_PREDICATE_RESULT_2 (POSH)	0x183BC	
3DPRIM_END_OFFSET	0x2420	1
3DPRIM_END_OFFSET (POSH)	0x18420	1
3DPRIM_START_VERTEX	0x2430	1
3DPRIM_START_VERTEX (POSH)	0x18430	1
3DPRIM_VERTEX_COUNT	0x2434	1
3DPRIM_VERTEX_COUNT (POSH)	0x18434	1
3DPRIM_INSTANCE_COUNT	0x2438	1
3DPRIM_INSTANCE_COUNT (POSH)	0x18438	1

MMIO Name	MMIO Offset	Size in DWords
3DPRIM_START_INSTANCE	0x243C	1
3DPRIM_START_INSTANCE (POSH)	0x1843C	1
3DPRIM_BASE_VERTEX	0x2440	1
3DPRIM_BASE_VERTEX (POSH)	0x18440	1
3DPRIM_XP0	0x2690	1
3DPRIM_XP0 (POSH)	0x18690	1
3DPRIM_XP1	0x2694	1
3DPRIM_XP1 (POSH)	0x18694	1
3DPRIM_XP2	0x2698	1
3DPRIM_XP2 (POSH)	0x18698	1
GPGPU_THREADS_DISPATCHED	0x2290	2
BB_OFFSET	0x2158	1
BB_OFFSET (POCS)	0x18158	1
CS_GPR (1-16)	0x2600	32
CS_GPR (1-16) (POSH)	0x18600	32
OA_CTX_CONTROL	0x2360	1
OA_CTX_CONTROL_MSG	0x2AA0	1
OACTXID	0x2364	1
OAR_OACONTROL	0x2960	1
OAR_DEBUG_REGISTER	0x2964	1
OAR_OASTATUS	0x2968	1
PR_CTR_CTL_RCSUNIT	0x2178	1
PR_CTR_THRSH_RCSUNIT	0x217C	1
VSR_PUSH_CONSTANT_BASE	0xE518	1
PTBR_PAGE_POOL_SIZE_REGISTER	0x18590	1
OVR_CHICKEN2	0x185A0	1
PSS_MODE	0x7038	1
CMD_BUFF_CTL	0x2084	1
Z_DISCARD_EN	0x7040	1
COMMON_SLICE_CHICKEN2	0x7014	1
AUX_TABLE_BASE_ADDR_LOW	0x4200	1
AUX_TABLE_BASE_ADDR_HIGH	0x4204	1
CCS_AUX_INV	0x4208	1
TRTT_CR	0x4400	1
TRTT_VA_RANGE	0x4404	1
TRTT_L3_BASE_LOW	0x4408	1
TRTT_L3_BASE_HIGH	0x440C	1

MMIO Name	MMIO Offset	Size in DWords
TR_NULL_GFX	0x4410	1
TRTT_INVAL	0x4414	1
LSQCREG1	0xB100	1
LSQCREG4	0xB118	1
LSQCREG5	0xB158	1
LSQCREG6	0xB15C	1
L3ALLOCREG	0xB134	1
L3TCCNTLREG	0xB138	1

User Mode Non-Privileged Registers for Compute Command Streamer (CCS)

MMIO Name	MMIO Offset	Size in DWords
NOPID	0x1A094	1
INSTPM	0x1A0C0	1
GPUGPU_DISPATCHDIMX	0x1A500	1
GPUGPU_DISPATCHDIMY	0x1A504	1
GPUGPU_DISPATCHDIMZ	0x1A508	1
MI_PREDICATE_SRC0	0x1A400	1
MI_PREDICATE_SRC0	0x1A404	1
MI_PREDICATE_SRC1	0x1A408	1
MI_PREDICATE_SRC1	0x1A40C	1
MI_PREDICATE_DATA	0x1A410	1
MI_PREDICATE_DATA	0x1A414	1
MI_PREDICATE_RESULT	0x1A418	1
MI_PREDICATE_RESULT_1	0x1A41C	1
MI_PREDICATE_RESULT_2	0x1A3BC	1
GPGPU_THREADS_DISPATCHED	0x1A290	2
BB_OFFSET	0x1A158	1
CS_GPR (1-16)	0x1A600	32
PR_CTR_CTL_RCSUNIT	0x1A178	1
PR_CTR_THRSH_RCSUNIT	0x1A17C	1
CMD_BUFF_CTL	0x1A084	1
COMPCS0_AUX_TABLE_BASE_ADDR_LOW	0x42C0	1
COMPCS0_AUX_TABLE_BASE_ADDR_HIGH	0x42C4	1
COMPCS0_CCS_AUX_NV	0x42C8	1
COMP_CTX0_TRTT_CR	0x4580	1
COMP_CTX0_TRTT_VA_RANGE	0x4584	1

MMIO Name	MMIO Offset	Size in DWords
COMP_CTX0_TRTT_L3_BASE_LOW	0x4588	1
COMP_CTX0_TRTT_L3_BASE_HIGH	0x458C	1
COMP_CTX0_TRTT_NULL	0x4590	1
COMP_CTX0_TRTT_INVAL	0x4594	1

User Mode Non-Privileged Registers for Blitter Command Streamer(BCS)

MMIO Name	MMIO Offset	Size in DWords
BCS_GPR	0x22600	32
BCS_SWCTRL	0x22200	1
BLIT_CCTL	0x22204	1
PR_CTR_CTL_BCSUNIT	0x22178	1
PR_CTR_THRSH_BCSUNIT	0x2217C	1
BLT_TRTT_CR	0x4480	1
BLT_TRTT_VA_RANGE	0x4484	1
BLT_TRTT_L3_BASE_LOW	0x4488	1
BLT_TRTT_L3_BASE_HIGH	0x448C	1
BLT_TRTT_NULL	0x4490	1
BLT_TRTT_INV	0x4494	1
NOPID	0x22094	1
MI_PREDICATE_RESULT_1	0x2241C	1
MI_PREDICATE_RESULT_2	0x223BC	1
INSTPM	0x220C0	1

Refer to Register Access and User Mode Privileges section for Base address for the below offsets.

User Mode Non-Privileged Registers for Video Enhancement Command Streamer (VECS)

MMIO Name	MMIO Base	MMIO Offset	Size in DWords
VECS_GPR	VECS	0x600	32
PR_CTR_CTL_VECSUNIT	VECS	0x178	1
PR_CTR_THRSH_VECSUNIT	VECS	0x17C	1
NOPID	VECS	0x094	1
MI_PREDICATE_RESULT_1	VECS	0x41C	1
MI_PREDICATE_RESULT_2	VECS	0x3BC	1
INSTPM	VECS	0x0C0	1

* These registers are not at a standard offset from their corresponding CS MMIO base address and hence are stated individually per CS in a separate table below.

User Mode Non-Privileged Registers for Video Command Streamer (ALL VCS)

MMIO Name	Unit Base	MMIO Range	Size in DWords
VCS_GPR	VCS	0x600	32
PR_CTR_CTL_VCSUNIT	VCS	0x178	1
PR_CTR_THRSH_VCSUNIT	VCS	0x17C	1
MFC_VDBOX1	VCS	0x800	512
HEVC	HEVC	0x00	64
VDENC	VDENC	0x00	64
NOPID	VCS	0x094	1
MI_PREDICATE_RESULT_1	VCS	0x41C	1
MI_PREDICATE_RESULT_2	VCS	0x3BC	1
INSTPM	VCS	0x0C0	1

* These registers are not at a standard offset from their corresponding CS MMIO base address and hence are stated individually per CS in a separate table below.

VEBOX-0

MMIO Name	MMIO Offset	Size in DWords
AUX_TABLE_BASE_ADDR_LOW*	0x4230	1
AUX_TABLE_BASE_ADDR_HIGH*	0x4234	1
CCS_AUX_INV*	0x4238	1
TRTT_CR*	0x4460	1
TRTT_VA_RANGE*	0x4464	1
TRTT_L3_BASE_LOW*	0x4468	1
TRTT_L3_BASE_HIGH*	0x446C	1
TRTT_NULL*	0x4470	1
TRTT_INVAL*	0x4474	1

VDBOX-0

MMIO Name	MMIO Offset	Size in DWords
AUX_TABLE_BASE_ADDR_LOW*	0x4210	1
AUX_TABLE_BASE_ADDR_HIGH*	0x4214	1
CCS_AUX_INV*	0x4218	1
TRTT_CR*	0x4420	1
TRTT_VA_RANGE*	0x4424	1
TRTT_L3_BASE_LOW*	0x4428	1
TRTT_L3_BASE_HIGH*	0x442C	1
TRTT_NULL*	0x4430	1
TRTT_INVAL*	0x4434	1

MMIO Name	MMIO Offset	Size in DWords

VDBOX-1

MMIO Name	MMIO Offset	Size in DWords
AUX_TABLE_BASE_ADDR_LOW*	0x4220	1
AUX_TABLE_BASE_ADDR_HIGH*	0x4224	1
CCS_AUX_INV*	0x4228	1
TRTT_CR*	0x4440	1
TRTT_VA_RANGE*	0x4444	1
TRTT_L3_BASE_LOW*	0x4448	1
TRTT_L3_BASE_HIGH*	0x444C	1
TRTT_NULL*	0x4450	1
TRTT_INVAL*	0x4454	1

VEBOX-1

MMIO Name	MMIO Base	MMIO Offset	Size in DWords
AUX_TABLE_BASE_ADDR_LOW*	n/a	0x42B0	1
AUX_TABLE_BASE_ADDR_HIGH*	n/a	0x42B4	1
CCS_AUX_INV*	n/a	0x42B8	1
TRTT_CR*	n/a	0x4560	1
TRTT_VA_RANGE*	n/a	0x4564	1
TRTT_L3_BASE_LOW*	n/a	0x4568	1
TRTT_L3_BASE_HIGH*	n/a	0x456C	1
TRTT_NULL*	n/a	0x4570	1
TRTT_INVAL*	n/a	0x4574	1

VDBOX-2

MMIO Name	MMIO Base	MMIO Offset	Size in DWords
AUX_TABLE_BASE_ADDR_LOW*	n/a	0x4290	1
AUX_TABLE_BASE_ADDR_HIGH*	n/a	0x4294	1
CCS_AUX_INV*	n/a	0x4298	1
TRTT_CR*	n/a	0x4520	1
TRTT_VA_RANGE*	n/a	0x4524	1
TRTT_L3_BASE_LOW*	n/a	0x4528	1
TRTT_L3_BASE_HIGH*	n/a	0x452C	1
TRTT_NULL*	n/a	0x4530	1

MMIO Name	MMIO Base	MMIO Offset	Size in DWords
TRTT_INVAL*	n/a	0x4534	1

VDBOX-3

MMIO Name	MMIO Base	MMIO Offset	Size in DWords
AUX_TABLE_BASE_ADDR_LOW*	n/a	0x42A0	1
AUX_TABLE_BASE_ADDR_HIGH*	n/a	0x42A4	1
CCS_AUX_INV*	n/a	0x42A8	1
TRTT_CR*	n/a	0x4540	1
TRTT_VA_RANGE*	n/a	0x4544	1
TRTT_L3_BASE_LOW*	n/a	0x4548	1
TRTT_L3_BASE_HIGH*	n/a	0x454C	1
TRTT_NULL*	n/a	0x4550	1
TRTT_INVAL*	n/a	0x4554	1

Context Management

When the scheduler submits a list of workloads through the execution list, the Command streamer hardware executes one context at a time.

An engine starts executing a context by loading the state (LRCA) in memory that is pointed to by the context descriptor.

The structure of the LRCA are described in subsequent sections.

Global State

There is only one copy of state variables across contexts running on an engine and changing the settings of these variables requires explicit programming of these state variables. Typically, global state variables are programmed only once either at the time of power-on or at the time of GFX driver initialization. Examples of global sate include:

- MI registers (HWSTAM, SEMA_WAIT_POLL ..etc)
- Configuration Registers (GFX_MODE ..etc)

The global state of an engine is context save/restored during power-off/on regimes.

Following subsections describe the power context images of engines across generations.

Power Context Image

This section lists the power context image of Video Engine, Copy Engine and Video Enhancement Engine across generations.

CSFE Power context without Display

CSFE Power Context Image with Display

Description	Offset	Unit	# of DW	Address Offset (PWR)	CSFE/CSBE
NOOP		CS	1	0000	CSFE
Load_Register_Immediate header	0x1100_00B5	CS	1	0001	CSFE
GFX_MODE	0x029C	CS	2	0002	CSFE
GHWSP	0x0080	CS	2	0004	CSFE
RC_PWRCTX_MAXCNT	0x0054	CS	2	0008	CSFE
CTX_WA_PTR	0x0058	CS	2	000A	CSFE
NOPID	0x0094	CS	2	000C	CSFE
HWSTAM	0x0098	CS	2	000E	CSFE
IMR	0x00A8	CS	2	0010	CSFE
EIR	0x00B0	CS	2	0012	CSFE
EMR	0x00B4	CS	2	0014	CSFE
CMD_CCTL_0	0x00C4	CS	2	0016	CSFE
PREEMPT_DLY	0x0214	CS	2	0018	CSFE
CTXT_PREMP_DBG	0x0248	CS	2	001A	CSFE
WAIT_FOR_RC6_EXIT	0x00CC	CS	2	001C	CSFE
RCS_CTXID_PREEMPTION_HINT	0x04CC	CS	2	001E	CSFE
CS_PREEMPTION_HINT_UDW	0x04C8	CS	2	0020	CSFE
CS_PREEMPTION_HINT	0x04BC	CS	2	0022	CSFE
CCID Register	0x0180	CS	2	0024	CSFE

Description	Offset	Unit	# of DW	Address Offset (PWR)	CSFE/CSBE
MI_PREDICATE_RESULT_2	0x03BC	CS	2	0026	CSFE
CTXT_ST_PTR	0x03A0	CS	2	0028	CSFE
CTXT_ST_BUF	0x0370	CS	24	002A	CSFE
CTXT_ST_BUF	0x03C0	CS	24	0042	CSFE
SEMA_WAIT_POLL	0x024C	CS	2	005A	CSFE
IDLEDELAY	0x023C	CS	2	005C	CSFE
RCS_FORCE_TO_NONPRIV_0_11	0x04D0	CS	24	005E	CSFE
RCS_FORCE_TO_NONPRIV_12_15	0x010	CS	8	0076	CSFE
RCS_FORCE_TO_NONPRIV_16_19	0x1D0	CS	8	007E	CSFE
EXECLIST_STATUS_REGISTER	0x0234	CS	2	0086	CSFE
CXT_OFFSET	0x01AC	CS	2	008A	CSBE
STOP_PARSER_CONTROL	0x0424	CS	2	008C	CSBE
STOP_PARSER_HINT_ADDR	0x0428	Cs	4	008E	CSBE
EXECLIST_SQ_CONTENTS	0x0510-0x054F	CS	32	0092	CSFE
CSB_INTERRUPT_MASK	0x0218	CS	2	00B2	CSFE
EQ_ELEMENT_MASK	0x056C	CS	2	00B4	CSFE
NOOP		CS	8	00B8	CSFE

CSFE Power Context with Display

CSFE Power Context Image with Display

Description	Offset	Unit	# of DW	Address Offset (PWR)	CSFE/CSBE
NOOP		CS	1	0000	CSFE
Load_Register_Immediate header	0x1100_00C7	CS	1	0001	CSFE
GFX_MODE	0x029C	CS	2	0002	CSFE
GHWSP	0x0080	CS	2	0004	CSFE
RC_PWRCTX_MAXCNT	0x0054	CS	2	0008	CSFE
CTX_WA_PTR	0x0058	CS	2	000A	CSFE
NOPID	0x0094	CS	2	000C	CSFE
HWSTAM	0x0098	CS	2	000E	CSFE
IMR	0x00A8	CS	2	0010	CSFE
EIR	0x00B0	CS	2	0012	CSFE
EMR	0x00B4	CS	2	0014	CSFE
CMD_CCTL_0	0x00C4	CS	2	0016	CSFE
PREEMPT_DLY	0x0214	CS	2	0018	CSFE
CTXT_PREMP_DBG	0x0248	CS	2	001A	CSFE
SYNC_FLIP_STATUS	0x02D0	CS	2	001C	CSFE

Description	Offset	Unit	# of DW	Address Offset (PWR)	CSFE/CSBE
SYNC_FLIP_STATUS_1	0x02D4	CS	2	001E	CSFE
SYNC_FLIP_STATUS_2	0x02EC	CS	2	0020	CSFE
WAIT_FOR_RC6_EXIT	0x00CC	CS	2	0022	CSFE
RCS_CTXID_PREEMPTION_HINT	0x04CC	CS	2	0024	CSFE
CS_PREEMPTION_HINT_UDW	0x04C8	CS	2	0026	CSFE
CS_PREEMPTION_HINT	0x04BC	CS	2	0028	CSFE
CCID Register	0x0180	CS	2	002A	CSFE
MI_PREDICATE_RESULT_2	0x03BC	CS	2	002C	CSFE
CTXT_ST_PTR	0x03A0	CS	2	002E	CSFE
CTXT_ST_BUF	0x0370	CS	24	0030	CSFE
CTXT_ST_BUF	0x03C0	CS	24	0048	CSFE
SEMA_WAIT_POLL	0x024C	CS	2	0060	CSFE
IDLEDELAY	0x023C	CS	2	0062	CSFE
DISPLAY MESSAGE FORWARD STATUS	0x02E8	CS	2	0064	CSFE
RCS_FORCE_TO_NONPRIV_0_11	0x04D0	CS	24	0066	CSFE
RCS_FORCE_TO_NONPRIV_12_15	0x010	CS	8	007E	CSFE
RCS_FORCE_TO_NONPRIV_16_19	0x1D0	CS	8	0086	CSFE
EXECLIST_STATUS_REGISTER	0x0234	CS	2	008E	CSFE
CXT_OFFSET	0x01AC	CS	2	0092	CSBE
STOP_PARSER_CONTROL	0x0424	CS	2	0094	CSBE
STOP_PARSER_HINT_ADDR	0x0428	Cs	4	0098	CSBE
SYNC_FLIP_STATUS_3	0x02B8	CS	2	009A	CSFE
SYNC_FLIP_STATUS_4	0x02C0	CS	2	009C	CSFE
SYNC_FLIP_STATUS_5	0x02C4	CS	2	009E	CSFE
SYNC_FLIP_STATUS_6	0x01F8	CS	2	00A0	CSFE
DISPLAY MESSAGE FORWARD STATUS_2	0x0188	CS	2	00A2	CSFE
DISPLAY MESSAGE FORWARD STATUS_3	0x018C	CS	2	00A4	CSFE
EXECLIST_SQ_CONTENTS	0x0510-0x054F	CS	32	00A6	CSFE
CSB_INTERRUPT_MASK	0x0218	CS	2	00C6	CSFE
EQ_ELEMENT_MASK	0x056C	CS	2	00C8	CSFE
NOOP		CS	4	00D0	CSFE

CSFE Power Context

CSFE Power Context Image

Description	Offset	Unit	# of DW	Address Offset (PWR)	CSFE/CSBE

Description	Offset	Unit	# of DW	Address Offset (PWR)	CSFE/CSBE
NOOP		CS	1	0	CSFE
Load_Register_Immediate header	0x1100_00DB	CS	1	001	CSFE
GFX_MODE	0x029C	CS	2	0002	CSFE
GHWSP	0x0080	CS	2	0004	CSFE
RING_BUFFER_CONTROL (Ring Always Disabled)	0x003C	CS	2	0006	CSFE
Ring Head Pointer Register	0x0034	CS	2	0008	CSFE
Ring Tail Pointer Register	0x0030	CS	2	000A	CSFE
RING_BUFFER_START	0x0038	CS	2	000C	CSFE
RING_BUFFER_CONTROL (Original status)	0x003C	CS	2	000E	CSFE
Batch Buffer Current Head Register (UDW)	0x0168	CS	2	0010	CSFE
Batch Buffer Current Head Register	0x0140	CS	2	0012	CSFE
Batch Buffer State Register	0x0110	CS	2	0014	CSFE
SECOND_BB_ADDR_UDW	0x011C	CS	2	0016	CSFE
SECOND_BB_ADDR	0x0114	CS	2	0018	CSFE
SECOND_BB_STATE	0x0118	CS	2	001A	CSFE
RC_PSMI_CONTROL	0x0050	CS	2	001C	CSFE
RC_PWRCTX_MAXCNT	0x0054	CS	2	001E	CSFE
CTX_WA_PTR	0x0058	CS	2	0020	CSFE
NOPID	0x0094	CS	2	0022	CSFE
HWSTAM	0x0098	CS	2	0024	CSFE
IMR	0x00A8	CS	2	0026	CSFE
EIR	0x00B0	CS	2	0028	CSFE
EMR	0x00B4	CS	2	002A	CSFE
CMD_CCTL_0	0x00C4	CS	2	002C	CSFE
UHPTR	0x0134	CS	2	002E	CSFE
BB_PREEMPT_ADDR_UDW	0x016C	CS	2	0030	CSFE
BB_PREEMPT_ADDR	0x0148	CS	2	0032	CSFE
RING_BUFFER_HEAD_PREEMPT_REG	0x014C	CS	2	0034	CSFE
PREEMPT_DLY	0x0214	CS	2	0036	CSFE
CTXT_PREMP_DBG	0x0248	CS	2	0038	CSFE
SYNC_FLIP_STATUS	0x02D0	CS	2	003A	CSFE
SYNC_FLIP_STATUS_1	0x02D4	CS	2	003C	CSFE
SYNC_FLIP_STATUS_2	0x02EC	CS	2	003E	CSFE
WAIT_FOR_RC6_EXIT	0x00CC	CS	2	0040	CSFE
RCS_CTXID_PREEMPTION_HINT	0x04CC	CS	2	0042	CSFE

Description	Offset	Unit	# of DW	Address Offset (PWR)	CSFE/CSBE
CS_PREEMPTION_HINT_UDW	0x04C8	CS	2	0044	CSFE
CS_PREEMPTION_HINT	0x04BC	CS	2	0046	CSFE
CCID Register	0x0180	CS	2	0048	CSFE
SBB_PREEMPT_ADDRESS_UDW	0x0138	CS	2	004A	CSFE
SBB_PREEMPT_ADDRESS	0x013C	CS	2	004C	CSFE
MI_PREDICATE_RESULT_2	0x03BC	CS	2	004E	CSFE
CTXT_ST_PTR	0x03A0	CS	2	0050	CSFE
CTXT_ST_BUF	0x0370	CS	24	0052	CSFE
CTXT_ST_BUF	0x03C0	CS	24	006A	CSFE
SEMA_WAIT_POLL	0x024C	CS	2	0082	CSFE
IDLEDELAY	0x023C	CS	2	0084	CSFE
DISPLAY MESSAGE FORWARD STATUS	0x02E8	CS	2	0086	CSFE
RCS_FORCE_TO_NONPRIV	0x04D0	CS	24	0088	CSFE
EXECLIST_STATUS_REGISTER	0x0234	CS	2	00A0	CSFE
CXT_OFFSET	0x01AC	CS	2	00A4	CSBE
STOP_PARSER_CONTROL	0x0424	CS	2	00A6	CSBE
STOP_PARSER_HINT_ADDR	0x0428	Cs	4	00A8	CSBE
SYNC_FLIP_STATUS_3	0x02B8	CS	2	00AC	CSFE
SYNC_FLIP_STATUS_4	0x02C0	CS	2	00AE	CSFE
SYNC_FLIP_STATUS_5	0x02C4	CS	2	00B0	CSFE
SYNC_FLIP_STATUS_6	0x01F8	CS	2	00B2	CSFE
DISPLAY MESSAGE FORWARD STATUS_2	0x0188	CS	2	00B4	CSFE
DISPLAY MESSAGE FORWARD STATUS_3	0x018C	CS	2	00B6	CSFE
EXECLIST_SQ_CONTENTS	0x0510-0x054F	CS	32	00B8	CSFE
CSB_INTERRUPT_MASK	0x0218	CS	2	00D8	CSFE
EQ_ELEMENT_MASK	0x056C	CS	2	00DA	CSFE

Context State

Context state is associated with a specific context. Context state can be programmed through Command Stream only when the associated context is being actively executed in the engine.

Context state is save/restored through Logical Context.

Logical Contexts

A logical context is an area in memory used to store hardware context state information and the context is referenced via a context descriptor. Context descriptor carries graphics memory address. Logical Context is always in global virtual memory. GFX device provides means to save and restore the hardware context state to logical context. Context state save/restore mechanism is used by SW to avoid re-programming the HW state across context switches for a given context.

CSFE Execlist Context

This section details the CSFE Execlist Context which is the common layout referred to as part of the VDBOX, Copy Engine and Video Enhancment context images.

CSFE Execlist Context

Programming Note	
Context:	MMIO Offset information
MMIO offset mentioned for the registers in the below table are offset form the units "MMIO Base Offset" mentiond in the table " Base Offset for Video Command Streamers and Media Engine" in the section User Mode Privileged Commands. For Example: VECS has MMIO Base Offset as "0x1C_8000". In the below table "Context Control" register has 0x00244 as offset against it, actual MMIO Offset of "Context Control" register for VECS is 0xx1C_8244. Blitter Engine MMIO base offset must be considered as 0x2_0000.	

EXECLIST CONTEXT
EXECLIST CONTEXT(PPGTT Base)
ENGINE CONTEXT

Description	MMIO Offset/Command	Unit	# of DW	Offset
NOOP		CSEL	1	**0**
Load_Register_Immediate header	0x1108_1019	CSEL	1	**0001**
Context Control	0x00244	CSEL	2	**0002**
Ring Head Pointer Register	0x00034	CSEL	2	**0004**
Ring Tail Pointer Register	0x00030	CSEL	2	**0006**
RING_BUFFER_START	0x00038	CSEL	2	**0008**
RING_BUFFER_CONTROL	0x0003C	CSEL	2	**000A**

Description	MMIO Offset/Command	Unit	# of DW	Offset
Batch Buffer Current Head Register (UDW)	0x00168	CSEL	2	**000C**
Batch Buffer Current Head Register	0x00140	CSEL	2	**000E**
Batch Buffer State Register	0x00110	CSEL	2	**0010**
BB_PER_CTX_PTR	0x001C0	CSEL	2	**0012**
CS_INDIRECT_CTX	0x001C4	CSEL	2	**0014**
CS_INDIRECT_CTX_OFFSET	0x001C8	CSEL	2	**0016**
CCID	0x00180	CSEL	2	**0018**
SEMAPHORE_TOKEN	0x002B4	CSEL	2	**001A**
NOOP		CSEL	4	**001C**
NOOP		CSEL	1	**0020**
Load_Register_Immediate header	0x1108_1011	CSEL	1	**0021**
CTX_TIMESTAMP	0x003A8	CSEL	2	**0022**
PDP3_UDW	0x0028C	CSEL	2	**0024**
PDP3_LDW	0x00288	CSEL	2	**0026**
PDP2_UDW	0x00284	CSEL	2	**0028**
PDP2_LDW	0x00280	CSEL	2	**002A**
PDP1_UDW	0x0027C	CSEL	2	**002C**
PDP1_LDW	0x00278	CSEL	2	**002E**
PDP0_UDW	0x00274	CSEL	2	**0030**
PDP0_LDW	0x00270	CSEL	2	**0032**
NOOP		CSEL	4	**0034**
NOOP		CSEL	8	**0038**
NOOP	{EXISTS IF (VCS, VECS)}	CSEL_BE	16	**0040**
NOOP	{EXISTS IF (BCS)}	CSEL_BE	1	**0040**
Load_Register_Immediate header	0x1100_1003 {EXISTS IF (BCS)}	CSEL_BE	1	**0041**
BCS TILE REGISTER	0x02200 {EXISTS IF (BCS)}	CSEL_BE	2	**0042**
BLIT_CCTL	0x02204 {EXISTS IF (BCS)}	CSEL_BE	2	**0044**
NOOP	{EXISTS IF (BCS)}	CSEL_BE	10	**0046**
NOOP		CSFE	1	**0050**
Load_Register_Immediate header	0x1100_1063	CSFE	1	**0051**
BB_STACK_WRITE_PORT	0x00588	CSFE	12	**0052**
EXCC	0x00028	CSFE	2	**005E**

Description	MMIO Offset/Command	Unit	# of DW	Offset
MI_MODE	0x0009C	CSFE	2	**0060**
INSTPM	0x000C0	CSFE	2	**0062**
PR_CTR_CTL	0x00178	CSFE	2	**0064**
PR_CTR_THRSH	0x0017C	CSFE	2	**0066**
TIMESTAMP Register (LSB)	0x00358	CSFE	2	**0068**
BB_START_ADDR_UDW	0x00170	CSFE	2	**006A**
BB_START_ADDR	0x00150	CSFE	2	**006C**
BB_ADD_DIFF	0x00154	CSFE	2	**006E**
BB_OFFSET	0x00158	CSFE	2	**0070**
MI_PREDICATE_RESULT_1	0x0041C	CSFE	2	**0072**
CS_GPR (1-16)	0x00600	CSFE	64	**0074**
IPEHR	0x00068	CSFE	2	**00B4**
MI_FORCE_WAKEUP	{EXISTS IF (VCS, VECS)}	CSFE	2	**00B6**
NOOP	{EXISTS IF (BCS)}	CSFE	2	**00B6**
AUX_TTRTT_BASE_ADDRESS_SECTION (Seperate section for each engine mentioned below)		CSFE	18	**00B8**
NOOP*		CSFE	2	**00CA**
NOOP		CSFE	4	**00CC**

Video Enhancement Engine: AUX_TRTT_BASE_ADDRESS_SECTION

Description	MMIO Offset/Command	Unit	# of DW	Offset
NOOP		CSFE	1	**00B6**
Load_Register_Immediate header	0x1102_100F	CSFE	1	**00B7**
TRTT_CR	0x4460	CSFE	2	**00BC**
TRTT_VA_RANGE	0x4464	CSFE	2	**00BE**
TRTT_L3_BASE_LOW	0x4468	CSFE	2	**00C0**
TRTT_L3_BASE_HIGH	0x446C	CSFE	2	**00C2**
TRTT_NULL	0x4470	CSFE	2	**00C4**
TRTT_INVAL	0x4474	CSFE	2	**00C6**
AUX_TABLE_BASE_ADDR_LOW	0x4230	CSFE	2	**00B8**
AUX_TABLE_BASE_ADDR_HIGH	0x4234	CSFE	2	**00BA**

Video Decode Engine:AUX_TRTT_BASE_ADDRESS_SECTION

Description	MMIO Offset/Command	Unit	# of DW	Offset
NOOp		CSFE	1	**00B6**
Load_Register_Immediate header	0x1102_100F	CSFE	1	**00B7**
TRTT_CR	0x4420	CSFE	2	**00BC**
TRTT_VA_RANGE	0x4424	CSFE	2	**00BE**
TRTT_L3_BASE_LOW	0x4428	CSFE	2	**00C0**
TRTT_L3_BASE_HIGH	0x442C	CSFE	2	**00C2**
TRTT_NULL	0x4430	CSFE	2	**00C4**
TRTT_INVAL	0x4434	CSFE	2	**00C6**
AUX_TABLE_BASE_ADDR_LOW	0x4210	CSFE	2	**00B8**
AUX_TABLE_BASE_ADDR_HIGH	0x4214	CSFE	2	**00BA**

Copy Engine: AUX_TRTT_BASE_ADDRESS_SECTION

Description	MMIO Offset/Command	Unit	# of DW	Offset
NOOP		CSFE	1	**00B6**
Load_Register_Immediate header	0x1102_100B	CSFE	1	**00B7**
TRTT_CR	0x4480	CSFE	2	**00BC**
TRTT_VA_RANGE	0x4484	CSFE	2	**00BE**
TRTT_L3_BASE_LOW	0x4488	CSFE	2	**00C0**
TRTT_L3_BASE_HIGH	0x448C	CSFE	2	**00C2**
TRTT_NULL	0x4490	CSFE	2	**00C4**
TRTT_INVAL	0x4494	CSFE	2	**00C6**
NOOP		CSFE	2	**00B8**
NOOP		CSFE	2	**00BA**

Producer-Consumer Data ordering for MI Commands

This section details the explicit data ordering enforced by HW for produce-consume of data between MI commands and explicit programming notes for data ordering not explicitly enforced by HW.

This section describes the MI commands that result in modification of data in Graphics memory or MMIO registers. These commands can be treated as producers of data for which consumers can either be SW or subsequent commands (MI or non-MI) executed by HW.

Operations (memory update or MMIO update) resulting from a command execution can be classified in to posted or non-posted.

- An operation is classified as posted if the operation initiated by the command is not guaranteed to complete (data change to be reflected) before HW moves on to the following command to execute, the posted operation is guaranteed to complete eventually. Posted operations can be forced to complete through explicit or implicit means, detailed in following section.
    - For example a memory write is called posted if the hardware moves on to the next command after generating a memory write without waiting for the memory modification to reach a global observable point.
- An operation is classified as non-posted if the operation initiated by the command is completed before HW moves on to execute the following command.
    - For example a memory write is called non-posted if the hardware waits for the memory write to reach a global observable point before it moves on to the next command to execute.

There are certain commands which supported both posted and non-posted operations and can be programmed by SW to select the appropriate behavior based on the usage model.

Memory Data Ordering

This section details the produce-consume data for MI commands accessing memory.

Memory Data Producer

This section describes the MI commands that modify data in graphics memory. Few commands always generate posted memory writes whereas few commands provide programmable option to generate posted Vs non-posted memory writes.

- A memory write is called posted if the hardware moves on to the next command after generating a memory write and doesn't wait for the memory modification to reach a global observable point. Since HW doesn't wait for the memory write completion it can execute the next command immediately without incurring any additional latency. Read after Write hazard is applicable in this scenario.
- A memory write is called non-posted if the hardware waits for the memory write to reach a global observable point before it moves on to the next command to execute. Since HW waits for the memory write completion before it goes on to the next command, it will incur additional latency causing a stall at top of the pipe. Read after write hazard will not happen in this scenario.

A write completion of a non-posted memory write will guarantee all the prior posted memory writes are to global observable (GO) point.

For optimal performance SW must use commands generating non-posted memory writes at the minimal. For example a single non-posted memory write can be used just before the consume point to flush out all the prior posted memory writes to global observable point. Based on the usage model SW can use a combination of commands that generate posted memory writes and non-posted memory writes for optimal performance.

Table below lists the MI Commands that can update/modify the data in graphics memory and the associated type of memory write.

Command	Memory Write Type
MI_STORE_REGISTER_MEM	Posted
MI_COPY_MEM_MEM	Posted
MI_STORE_DATA_INDEX	Posted
MI_STORE_DATA_IMM (with Wr. Completion)	Non-Posted
MI_STORE_DATA_IMM (with out Wr. Completion)	Posted
MI_REPORT_HEAD	Posted
MI_UPDATE_GTT	Posted
MI_REPORT_PERF_COUNT	Posted
MI_ATOMIC	Posted, Non-Posted
MI_FLUSH_DW (With Post-Sync Operation)	Non-Posted
PIPE_CONTROL (non-stalling, with Post-Sync Operation)	Posted
PIPE_CONTROL (Stalling, Post-Sync Operation)	Non-Posted

Apart from the MI commands that generate Non-Posted memory writes listed in the above table, execution of following commands will also implicitly ensure all prior posted writes are to Global Observable point.

Command
PIPE_CONTROL (Stalling)
MI_FLUSH_DWORD
MI_USER_INTERRUPT
PIPE_CONTROL (with Notify Interrupt)

Memory Data - Consumer

Table below lists the MI command that read the data from graphics memory as part of the command execution. Data in memory should be coherent prior to execution of these command to achieve expected functional behavior upon execution of these commands, Graphics memory writes by the earlier executed MI commands must be GO prior to execution of these commands. Hardware has started explicitly enforcing data ordering for few of the commands (based on the prevalent usage models) and mentioned in the table below.

Command	Coherency Requirement
MI_LOAD_REGISTER_MEM	HW implicitly ensures memory writes by the prior MI commands by the corresponding engine are coherent for this command execution.
MI_BATCH_BUFFER_START	SW must ensure the data cohrency.
MI_CONDITIONAL_BATCH_BUFFER_END	SW must ensure the data cohrency.
MI_ATOMIC	HW implicitly ensures memory writes by the prior MI commands by the corresponding engine are coherent for this command execution.
MI_SEMAPHORE_WAIT	HW implicitly ensures memory writes by the prior MI commands by the corresponding engine are coherent for this command execution.
MI_COPY_MEM_MEM	

SW can use any of the MI commands that generate non-posted memory writes or the commands that implicitly force prior memory writes to GO to ensure data is coherent in memory prior to execution of these commands.

MMIO Data Ordering

This section details the produce-consume data for MI commands accessing MMIO registers.

MMIO Data Producer

Table below lists the MI commands that modify data in MMIO registers and also states if the MMIO writes generated are posted Vs non-posted.

- A MMIO write is called non-posted if the hardware waits for the MMIO update to occur before it moves on to the next command to execute.
- A MMIO write is called posted if the hardware moves on to the next command after generating a MMIO write without waiting for the MMIO update to occur.

All the MI commands listed below generate non-posted MMIO writes and hence HW guarantees the MMIO modification has taken place before HW moves on the following command.

MI_LOAD_REGISTER_MEM supports both posted and non-posted behavior and can be configured through "Async Mode Enable" bit in the command header.

Command	MMIO Write Type
MI_LOAD_REGISTER_IMM	Non-Posted
PIPE_CONTROL	Non-Posted
MI_LOAD_REGISTER_MEM	Posted, Non-Posted
MI_MATH	Non-Posted
MI_LOAD_REGISTER_REG	Non-Posted

MMIO Data Consumer

All the commands that modify the MMIO are non-posted and hence any MI command consumer of MMIO data will always get the latest updated value.

Software must take care of appropriately programming the "Async Mode Enable" bit in MI_LOAD_REGISTER_MEM command based on the requirements to enforce data ordering between producer and consumer. Table below lists the MI commands that consume the MMIO data.

Command
MI_STORE_REGISTER_MEM
MI_PREDICATE
MI_LOAD_REGISTER_REG
MI_MATH
MI_SET_PREDICATE
MI_SEMAPHORE_WAIT (register poll)
MI_SEMAPHPORE_SIGNAL

Command Fetch

Command parser implements a DMA engine to fetch the command data from memory. DMA engine pre-fetches eight cacheline worth of command data in to its storage and keeps it ready to be executed, it keeps fetching command data as and when space is available in the storage.

Advanced Command Prefetch

Advanced command pre-fetch is an enhancement to the existing DMA engine to addresses its limitation of not being to stream pre-fetches on instructions causing jumps (Ex: Batch Buffer Start and Batch Buffer End). Advanced command pre-fetch is enabled by default, however software is given flexibility to enable or disable advanced command pre-fetch at its convenience by following means:

- Global state through MMIO (GFX_MODE) which is part of power context. This MMIO bit must be enabled or disabled only through CPU path and must be done when there is no context active in hardware.
- Inline to the command sequence through MI_ARB_CHK, this state is maintained per context. This mechanism must be used by SW to disable pre-fetch around self-modification-code or around selective command sequences of interest.

In order for pre-fetch functionality to be enabled both global and per context state should be set to pre-fetch enable.

Self Modifying Code

Self-modifying-code (SMC) in context to command parser refers to a scenario where in a command sequence executed by the command parser modifies the upcoming commands to be executed by command parser. DMA pre-fetch of command data introduces certain programming restriction on placement of the SMC in the command sequence.

- The modifying commands and the modified commands should be far apart by the number of cachelines fetched by the CS for latency hiding(See Max Command FIFO Depth below) Or
- The modifying commands and the modified command must be executed after a batch buffer start(chained or nested)

Advanced command preparser adds additional limitation to the programming of self-modifying-code. Software must explicitly disable the preparser logic before programming a batch buffer whose contents has been modified by the earlier programmed command sequence (self-modifying-code). Preparser logic must be disabled using MI_ARB_CHECK command prior to programing the MI_BATCH_BUFFER_START command and pre-fetch logic must be enabled using MI_ARB_CHECK as the first command inside the batch buffer.

The depth depends on the parametrization within each command streamer as each have different requirements for latency hiding. Below is a table of the maximum size supported. Refer to the MI_BATCH_BUFFER_START for actual depths per command streamer.

Max Command FIFO Depth(64B)
8

Intel® Iris® Xe and UHD Graphics Open Source

Programmer's Reference Manual

For the 2020-2021 11th Generation Intel Xeon®, Core™, Celeron®, Pentium® Gold Processors based on the "Tiger Lake" Platform

Volume 9: Render Engine

December 2021, Revision 1.0

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Table of Contents

Render Engine 1

Workload Submission 1

Multi-context Submission Overview 1

Render-3D-GPGPU Command Streamer 2

Engine State 47

Software Interface 64

3D Pipeline Stages 70

3D Pipeline-Level State 71

3D Pipeline Geometry 72

3D Pipeline Rasterization 180

Pixel 250

GPGPU Compute Pipeline 309

General Purpose Compute Model 310

GPGPU Context in GPU Hardware 313

Programming the GPGPU Pipeline 314

Commands for GPGPU Pipe 321

Thread Spawner (TS) 331

Thread Dispatch 342

Thread Tracking and Synchronization 348

Context Switch for GPGPU and Media 351

3D and GPGPU Programs 353

EU Overview 353

Shared Functions 521

Render Engine

The Render Engine supports command streams used both for 3D and Compute (GPGPU) workloads. These command streams fetch the data, and dispatch individual work items to many threads that operate in parallel. The threads run small software programs (also called kernels or shaders) on the GPU processors (called Execution Units).

The command streamers control the programmable pipelines in the Render Engine so that the individual programs run in parallel but are synchronized to start only when their required data is available, and complete when all the work is done.

Each pipeline in the Render Engine shares common state with all the threads running in the pipeline. The command streamer manages that state.

Workload Submission

This section describes workload submission on the graphics engine.

Multi-context Submission Overview

Work into the Graphics engine is input using the Command Streamer or multiple command streamers.

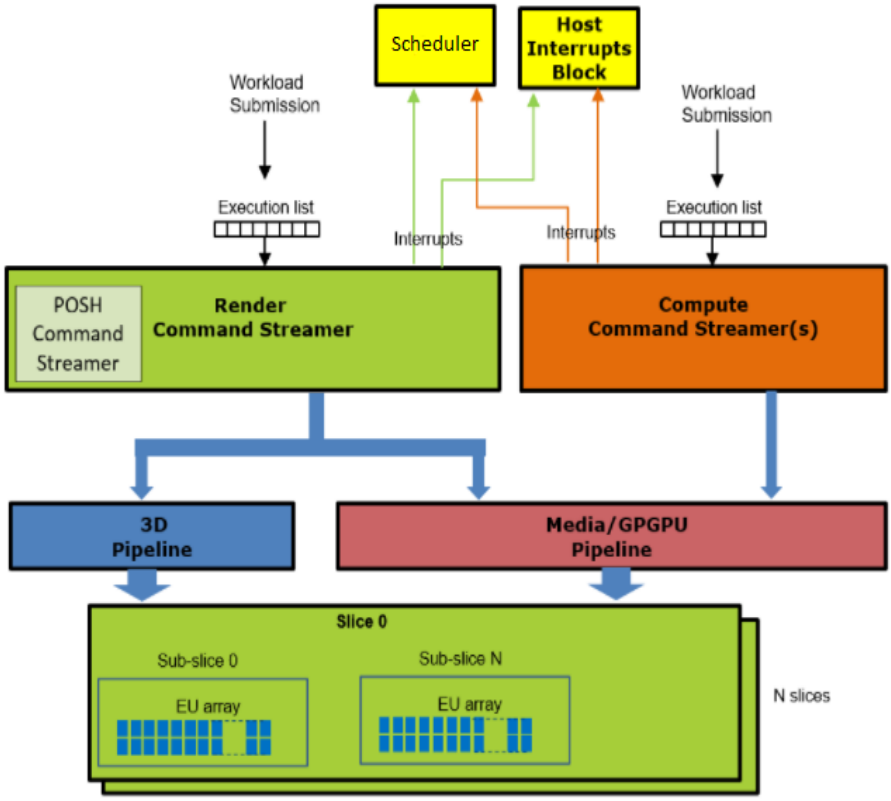
Each command streamers are independent and support their own submit port and execution list. Within a command streamer, the work elements are executed serially.

The Render Command Streamer runs in one of the following modes (that is specified using the PIPELINE_SELECT command):

- 3D

Media/GPGPU

The Compute Command Streamer only supports the Media/GPGPU mode. Number of Compute CS varies per SKU.



Render-3D-GPGPU Command Streamer

This section describes the infrastructure provided by the Command Streamer of the Render engine which supports 3D, Compute and Programmable Media.

Batch Buffer Privilege Register

FORCE_TO_NONPRIV - FORCE_TO_NONPRIV

Context Save Registers

The following are the Context Save Registers:

Register
BB_PREEMPT_ADDR - Batch Buffer Head Pointer Preemption Register
BB_PREEMPT_ADDR_UDW - Batch Buffer Upper Head Pointer Preemption Register
RING_BUFFER_HEAD_PREEMPT_REG - RING_BUFFER_HEAD_PREEMPT_REG
BB_START_ADDR - Batch Buffer Start Head Pointer Register
BB_START_ADDR_UDW - Batch Buffer Start Upper Head Pointer Register
BB_ADDR_DIFF - Batch Address Difference Register
BB_OFFSET - Batch Offset Register
SBB_PREEMPT_ADDR - Second Level Batch Buffer Head Pointer Preemption Register
SBB_PREEMPT_ADDR_UDW - Second Level Batch Buffer Upper Head Pointer Preemption Register

Register
BB_PER_CTX_PTR - Batch Buffer Per Context Pointer

Mode Registers

The following are the Mode Registers:

Register
INSTPM - Instruction Parser Mode Register
EXCC - Execute Condition Code Register
NOPID - NOP Identification Register
CSPREEMPT - CSPREEMPT
IDLEDLY - Idle Switch Delay
SEMA_WAIT_POLL - Semaphore Polling Interval on Wait
HWS_PGA - Hardware Status Page Address Register

Watchdog Timer Registers

These registers together implement a watchdog timer. Writing ones to the control register enables the counter, and writing zeros disables the counter. The second register is programmed with a threshold value which, when reached, signals an interrupt that then resets the counter to 0. Program the threshold value before enabling the counter or extremely frequent interrupts may result.

Note: The counter itself is not observable. It increments with the main render clock.

Programming Notes: When watch dog timer is enabled, HW does not trigger any kind of idle sequences. SW must enable and disable watch dog timer for any given workload within the same command buffer dispatch. SW must disable watch dog timer around semaphore waits and wait for events commands so that HW can trigger appropriate idle sequence for power savings.

Logical Context Support

The following are the Logical Context Support Registers:

Register
BB_ADDR - Batch Buffer Head Pointer Register
BB_ADDR_UDW - Batch Buffer Upper Head Pointer Register
CXT_SIZE - Context Sizes
CXT_EL_OFFSET - Exec-List Context Offset
SYNC_FLIP_STATUS - Wait For Event and Display Flip Flags Register
SYNC_FLIP_STATUS_1 - Wait For Event and Display Flip Flags Register 1
SYNC_FLIP_STATUS_2 - Wait For Event and Display Flip Flags Register 2
WAIT_FOR_RC6_EXIT - Control Register for Power Management
SBB_ADDR - Second Level Batch Buffer Head Pointer Register
SBB_ADDR_UDW - Second Level Batch Buffer Upper Head Pointer Register

Register
SBB_STATE - Second Level Batch Buffer State Register
PS_INVOCATION_COUNT_SLICE0 - PS Invocation Count for Slice0
PS_INVOCATION_COUNT_SLICE1 - PS Invocation Count for Slice1
PS_INVOCATION_COUNT_SLICE2 - PS Invocation Count for Slice2
PS_DEPTH_COUNT_SLICE0 - PS Depth Count for Slice0
PS_DEPTH_COUNT_SLICE1 - PS Depth Count for Slice1
PS_DEPTH_COUNT_SLICE2 - PS Depth Count for Slice2
R_PWR_CLK_STATE - Render Power Clock State Register

MI Commands for Render Engine

This chapter describes the formats of the "Memory Interface" commands, including brief descriptions of their use. The functions performed by these commands are discussed fully in the *Memory Interface Functions* Device Programming Environment chapter.

This chapter describes MI Commands for the original graphics processing engine. The term "for Rendering Engine" in the title has been added to differentiate this chapter from a similar one describing the MI commands for the Media Decode Engine.

The commands detailed in this chapter are used across product families. However, slight changes may be present in some commands (i.e., for features added or removed), or some commands may be removed entirely. Refer to the *Preface* chapter for product specific summary.

Commands
MI_NOOP
MI_ARB_CHECK
MI_ARB_ON_OFF
MI_BATCH_BUFFER_START
MI_BATCH_BUFFER_END
MI_CONDITIONAL_BATCH_BUFFER_END
MI_DISPLAY_FLIP
MI_LOAD_SCAN_LINES_EXCL
MI_LOAD_SCAN_LINES_INCL
MI_MATH
MI_REPORT_HEAD
MI_STORE_DATA_IMM
MI_ATOMIC
MI_COPY_MEM_MEM
MI_LOAD_REGISTER_REG
MI_LOAD_REGISTER_MEM
MI_STORE_REGISTER_MEM
MI_SUSPEND_FLUSH
MI_USER_INTERRUPT

Commands
MI_WAIT_FOR_EVENT
MI_SEMAPHORE_SIGNAL
MI_SEMAPHORE_WAIT
MI_FORCE_WAKEUP

Render Engine Command Streamer

The RCS (Render Command Streamer) unit primarily serves as the software programming interface between the O/S driver and the Render Engine. It is responsible for fetching, decoding, and dispatching of data packets (3D/Media Commands with the header DWord removed) to the front-end interface module of Render Engine.

Logic Functions Included
• MMIO register programming interface. • DMA action for fetching of ring data from memory. • Management of the Head pointer for the Ring Buffer. • Decode of ring data and sending it to the appropriate destination: 3D (Vertex Fetch Unit) & GPGPU. • Handling of user interrupts. • Flushing the 3D and GPGPU Engine. • Handle NOP.
• DMA action for fetching of execlists from memory. • Handling of ring context switch interrupt.

The RCS unit only claims memory mapped I/O cycles that are targeted to its range of 0x2000 to 0x27FF. The Gx and MFX Engines use semaphore to synchronize their operations.

RCS operates completely independent of the MFx CS.

The simple sequence of events is as follows: a ring (say PRB0) is programmed by a memory-mapped register write cycle. The DMA inside RCS is kicked off. The DMA fetches commands from memory based on the starting address and head pointer. The DMA requests cache lines from memory (one cacheline CL at a time). There is guaranteed space in the DMA FIFO (8 CL deep) for data coming back from memory. The DMA control logic has copies of the head pointer and the tail pointer. The DMA increments the head pointer after making requests for ring commands. Once the DMA copy of the head pointer becomes equal to the tail pointer, the DMA stops requesting.

The parser starts executing once the DMA FIFO has valid commands. All the commands have a header DWord packet. Based on the encoding in the header packet, the command may be targeted towards Vertex Fetch Unit or GPPGU engine or the command parser. After execution of every command, the actual head pointer is updated. The ring is considered empty when the head pointer becomes equal to the tail pointer.

POCS and RCS Synchronization

Once POCS is triggered, it executes parallel to RCS, it only stops (doesn't switch out) when it runs out of command (head equals to tail) or on encountering unsuccessful semaphore wait. Command sequence execution of POCS is completely asynchronous to RCS command sequence execution. SW is responsible to explicitly synchronize POCS and RCS command sequence execution whenever required based on the various produce consume model using MI_SEMAPHORE_WAIT command.

GPGPU/Media

POSH pipe supports parallel execution during GPGPU and Media workloads under the condition that the L3 is not reconfigured, and the GPGPU/Media workload will not use more than 32KB of the URB. Currently the POSH and GPGPU on the RCS share the same URB space. However, the first 32KB are not used when POSH is only running.

Reconfiguring the URB cannot occur when switching between 3D and Compute/Media workloads if it is possible for POSH pipeline to be active. SW can ensure POSH is inactive using synchronization commands in the case POSH was enabled earlier in the context.

Mid-thread preemption must be disabled if it is possible for POSH to be running in parallel with compute workloads.

HW Binding Table with RS Disabled

RCS sets up the HW Binding Table functionality when 3DSTATE_BTP_POOL_ALLOC is programmed with RS disabled. POSH pipe uses the mode set by RCS. SW will explicitly synchronize POCS and RCS to ensure they always work in the same mode of operation wither HW BTP Enabled or Disabled. Note that both POCS and RCS will maintain their own copies of 3DSTATE_BTP_POOL_ALLOC.

Protection-On/Off Mode

RenderCS controls the Protection-On/Off mode at all times for both POSH and Render pipes. Protection-on/off mode set by RenderCS applies to memory clients form both render pipe and POSH pipe. based on the protection on signal from RCS. SW must explicitly ensure both POSH and Render pipes are synchronized around Protection and ProtectionOff zones during the command sequencing.

Render Command Formats

3D Commands

The 3D commands are used to program the graphics pipelines for 3D operations.

Refer to the *3D* chapter for a description of the 3D state and primitive commands and the *Media* chapter for a description of the media-related state and object commands.

For all commands listed in **3D Command Map**, the Pipeline Type (bits 28:27) is 3h, indicating the 3D Pipeline.

3D Command Map

Opcode Bits 26:24	Sub Opcode Bits 23:16	Command	Definition Chapter
0h	01h	Reserved	3D Pipeline
0h	02h	Reserved	3D Pipeline
0h	03h	Reserved	
0h	04h	3DSTATE_CLEAR_PARAMS	3D Pipeline
0h	05h	3DSTATE_DEPTH_BUFFER	3D Pipeline
0h	06h	3DSTATE_STENCIL_BUFFER	3D Pipeline
0h	07h	3DSTATE_HIER_DEPTH_BUFFER	3D Pipeline
0h	08h	3DSTATE_VERTEX_BUFFERS	Vertex Fetch
0h	09h	3DSTATE_VERTEX_ELEMENTS	Vertex Fetch
0h	0Ah	3DSTATE_INDEX_BUFFER	Vertex Fetch
0h	0Bh	3DSTATE_VF_STATISTICS	Vertex Fetch
0h	0Ch	3DSTATE_VF	Vertex Fetch
0h	0Dh	3DSTATE_VIEWPORT_STATE_POINTERS	3D Pipeline
0h	0Eh	3DSTATE_CC_STATE_POINTERS	3D Pipeline
0h	10h	3DSTATE_VS	Vertex Shader
0h	11h	3DSTATE_GS	Geometry Shader
0h	12h	3DSTATE_CLIP	Clipper
0h	13h	3DSTATE_SF	Strips & Fans
0h	14h	3DSTATE_WM	Windower
0h	15h	3DSTATE_CONSTANT_VS	Vertex Shader
0h	16h	3DSTATE_CONSTANT_GS	Geometry Shader
0h	17h	3DSTATE_CONSTANT_PS	Windower
0h	18h	3DSTATE_SAMPLE_MASK	Windower
0h	19h	3DSTATE_CONSTANT_HS	Hull Shader
0h	1Ah	3DSTATE_CONSTANT_DS	Domain Shader
0h	1Bh	3DSTATE_HS	Hull Shader
0h	1Ch	3DSTATE_TE	Tesselator
0h	1Dh	3DSTATE_DS	Domain Shader
0h	1Eh	3DSTATE_STREAMOUT	HW Streamout
0h	1Fh	3DSTATE_SBE	Setup
0h	20h	3DSTATE_PS	Pixel Shader
0h	21h	3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP	Strips & Fans
0h	22h	3DSTATE_CPS_POINTER	Course Pixel Shader
0h	23h	3DSTATE_VIEWPORT_STATE_POINTERS_CC	Windower
0h	24h	3DSTATE_BLEND_STATE_POINTERS	Pixel Shader

Opcode Bits 26:24	Sub Opcode Bits 23:16	Command	Definition Chapter
0h	25h	3DSTATE_DEPTH_STENCIL_STATE_POINTERS	Pixel Shader
0h	26h	3DSTATE_BINDING_TABLE_POINTERS_VS	Vertex Shader
0h	27h	3DSTATE_BINDING_TABLE_POINTERS_HS	Hull Shader
0h	28h	3DSTATE_BINDING_TABLE_POINTERS_DS	Domain Shader
0h	29h	3DSTATE_BINDING_TABLE_POINTERS_GS	Geometry Shader
0h	2Ah	3DSTATE_BINDING_TABLE_POINTERS_PS	Pixel Shader
0h	2Bh	3DSTATE_SAMPLER_STATE_POINTERS_VS	Vertex Shader
0h	2Ch	3DSTATE_SAMPLER_STATE_POINTERS_HS	Hull Shader
0h	2Dh	3DSTATE_SAMPLER_STATE_POINTERS_DS	Domain Shader
0h	2Eh	3DSTATE_SAMPLER_STATE_POINTERS_GS	Geometry Shader
0h	2Fh	3DSTATE_SAMPLER_STATE_POINTERS_PS	Pixel Shader
0h	30h	3DSTATE_URB_VS	Vertex Shader
0h	31h	3DSTATE_URB_HS	Hull Shader
0h	32h	3DSTATE_URB_DS	Domain Shader
0h	33h	3DSTATE_URB_GS	Geometry Shader
0h	34h	3DSTATE_GATHER_CONSTANT_VS	Vertex Shader
0h	35h	3DSTATE_GATHER_CONSTANT_GS	Geometry Shader
0h	36h	3DSTATE_GATHER_CONSTANT_HS	Hull Shader
0h	37h	3DSTATE_GATHER_CONSTANT_DS	Domain Shader
0h	38h	3DSTATE_GATHER_CONSTANT_PS	Pixel Shader
0h	39h	3DSTATE_DX9_CONSTANTF_VS	Vertex Shader
0h	3Ah	3DSTATE_DX9_CONSTANTF_PS	Pixel Shader
0h	3Bh	3DSTATE_DX9_CONSTANTI_VS	Vertex Shader
0h	3Ch	3DSTATE_DX9_CONSTANTI_PS	Pixel Shader
0h	3Dh	3DSTATE_DX9_CONSTANTB_VS	Vertex Shader
0h	3Eh	3DSTATE_DX9_CONSTANTB_PS	Pixel Shader
0h	3Fh	3DSTATE_DX9_LOCAL_VALID_VS	Vertex Shader
0h	40h	3DSTATE_DX9_LOCAL_VALID_PS	Pixel Shader
0h	41h	3DSTATE_DX9_GENERATE_ACTIVE_VS	Vertex Shader
0h	42h	3DSTATE_DX9_GENERATE_ACTIVE_PS	Pixel Shader
0h	43h	3DSTATE_BINDING_TABLE_EDIT_VS	Vertex Shader
0h	44h	3DSTATE_BINDING_TABLE_EDIT_GS	Geometry Shader
0h	45h	3DSTATE_BINDING_TABLE_EDIT_HS	Hull Shader
0h	46h	3DSTATE_BINDING_TABLE_EDIT_DS	Domain Shader
0h	47h	3DSTATE_BINDING_TABLE_EDIT_PS	Pixel Shader
0h	48h	3DSTATE_VF_HASHING	Vertex Fetch

Opcode Bits 26:24	Sub Opcode Bits 23:16	Command	Definition Chapter
0h	49h	3DSTATE_VF_INSTANCING	Vertex Fetch
0h	4Ah	3DSTATE_VF_SGVS	Vertex Fetch
0h	4Bh	3DSTATE_VF_TOPOLOGY	Vertex Fetch
0h	4Ch	3DSTATE_WM_CHROMA_KEY	Windower
0h	4Dh	3DSTATE_PS_BLEND	Windower
0h	4Eh	3DSTATE_WM_DEPTH_STENCIL	Windower
0h	4Fh	3DSTATE_PS_EXTRA	Windower
0h	50h	3DSTATE_RASTER	Strips & Fans
0h	51h	3DSTATE_SBE_SWIZ	Strips & Fans
0h	52h	3DSTATE_WM_HZ_OP	Windower
0h	53h	3DSTATE_INT (internally generated state)	3D Pipeline
0h	54h	3DSTATE_RS_CONSTANT_POINTER	Resource Streamer
0h	55h	3DSTATE_VF_COMPONENT_PACKING	Vertex Fetch
0h	56h	3DSTATE_VF_SGVS_2	VertexFetch
0h	58h	3DSTATE_URB_ALLOC_VS	VertexShader
0h	59h	3DSTATE_URB_ALLOC_HS	HullShader
0h	5Ah	3DSTATE_URB_ALLOC_DS	DomainShader
0h	5Bh	3DSTATE_URB_ALLOC_GS	GeometryShader
0h	5Dh-5Fh	Reserved	
0h	60h	3DSTATE_SO_BUFFER_INDEX_0	HW StreamOut
0h	61h	3DSTATE_SO_BUFFER_INDEX_1	HW StreamOut
0h	62h	3DSTATE_SO_BUFFER_INDEX_2	HW StreamOut
0h	63h	3DSTATE_SO_BUFFER_INDEX_3	HW StreamOut
0h	64h-69h	Reserved	
0h	6Ah	3DSTATE_PTBR_MARKER	3D Pipeline
0h	6Bh	3DSTATE_PTBR_TILE_SELECT	Vertex Fetch, Strips & Fans
0h	6Ch	3DSTATE_PRIMITIVE_REPLICATION	3D Pipeline
0h	6Dh	3DSTATE_CONSTANT_ALL	3D Pipeline
0h	6Fh	3DSTATE_AMFS	PSS
0h	70h	3DSTATE_DEPTH_CNTL_BUFFER	WM
0h	71h	3DSTATE_DEPTH_BOUNDS	WM
0h	72h	3DSTATE_AMFS_TEXTURE_POINTERS	WM
0h	73h	3DSTATE_CONSTANT_TS_POINTER	PSS
0h	57h-59h	Reserved	
0h	60h-68h	Reserved	
0h	69h	Reserved	

Opcode Bits 26:24	Sub Opcode Bits 23:16	Command	Definition Chapter
0h	6Eh	Reserved	
0h	74h	Reserved	
0h	75h	Reserved	
0h	76h	Reserved	
0h	77h-82h	Reserved	
0h	83h	Reserved	
0h	83h-FFh	Reserved	
1h	00h	3DSTATE_DRAWING_RECTANGLE	Strips & Fans
1h	02h	Reserved	
1h	03h	Reserved	
1h	04h	3DSTATE_CHROMA_KEY	Sampling Engine
1h	05h	Reserved	
1h	06h	3DSTATE_POLY_STIPPLE_OFFSET	Windower
1h	07h	3DSTATE_POLY_STIPPLE_PATTERN	Windower
1h	08h	3DSTATE_LINE_STIPPLE	Windower
1h	0Ah	3DSTATE_AA_LINE_PARAMS	Windower
1h	0Bh	3DSTATE_GS_SVB_INDEX	Geometry Shader
1h	0Ch	Reserved	
1h	0Dh	3DSTATE_MULTISAMPLE	Windower
1h	0Eh	3DSTATE_STENCIL_BUFFER	Windower
1h	0Fh	3DSTATE_HIER_DEPTH_BUFFER	Windower
1h	10h	3DSTATE_CLEAR_PARAMS	Windower
1h	11h	3DSTATE_MONOFILTER_SIZE	Sampling Engine
1h	12h	3DSTATE_PUSH_CONSTANT_ALLOC_VS	Vertex Shader
1h	13h	3DSTATE_PUSH_CONSTANT_ALLOC_HS	Hull Shader
1h	14h	3DSTATE_PUSH_CONSTANT_ALLOC_DS	Domain Shader
1h	15h	3DSTATE_PUSH_CONSTANT_ALLOC_GS	Geometry Shader
1h	16h	3DSTATE_PUSH_CONSTANT_ALLOC_PS	Pixel Shader
1h	17h	3DSTATE_SO_DECL_LIST	HW Streamout
1h	18h	3DSTATE_SO_BUFFER	HW Streamout
1h	19h	3DSTATE_BINDING_TABLE_POOL_ALLOC	Resource Streamer
1h	1Ah	3DSTATE_GATHER_POOL_ALLOC	Resource Streamer
1h	1Bh	3DSTATE_DX9_CONSTANT_BUFFER_POOL_ALLOC	Resource Streamer
1h	1Ch	3DSTATE_SAMPLE_PATTERN	Windower
1h	1Dh	3DSTATE_URB_CLEAR	3D Pipeline
1h	1Eh	3DSTATE_3D_MODE	3D Pipeline

Opcode Bits 26:24	Sub Opcode Bits 23:16	Command	Definition Chapter
1h	1Fh	3DSTATE_SUBSLICE_HASH_TABLE	3D Pipeline
1h	20h	3DSTATE_SLICE_TABLE_STATE_POINTERS	3D Pipeline
1h	21h	3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS	3D Pipeline
1h	22h	3DSTATE_PTBR_TILE_PASS_INFO	3D Pipeline
1h	23h	3DSTATE_PTBR_RENDER_LIST_BASE_ADDRESS	3D Pipeline
1h	24h	3DSTATE_PTBR_FREE_LIST_BASE_ADDRES	3D Pipeline
1h	23h-2Ah	Reserved	
1h	2Bh-FFh	Reserved	
2h	00h	PIPE_CONTROL	Render/Compute Pipeline
2h	01h	Reserved	
2h	03h-FFh	Reserved	
3h	00h	3DPRIMITIVE	Vertex Fetch
3h	01h	Reserved	
3h	02h	Reserved	
3h	03h-FFh	Reserved	
4h-7h	00h-FFh	Reserved	

Pipeline Type (28:27)	Opcode	Sub Opcode	Command	Definition Chapter
Common (pipelined)	Bits 26:24	Bits 23:16		
0h	0h	04h-FFh	Reserved	
Common (non-pipelined)	Bits 26:24	Bits 23:16		
0h	1h	00h	Reserved	N/A
0h	1h	01h	STATE_BASE_ADDRESS	Graphics Processing Engine
0h	1h	02h	STATE_SIP	Graphics Processing Engine
0h	1h	03h	Reserved	3D Pipeline
0h	1h	04h	GPGPU CSR BASE ADDRESS	Graphics Processing Engine
0h	1h	05h	STATE_COMPUTE_MODE	Compute Pipeline
0h	1h	06h	Reserved	
0h	1h	07h-08h	Reserved	
0h	1h	09h	Reserved	
0h	1h	0Ah-FFh	Reserved	N/A
Reserved	Bits 26:24	Bits 23:16		
0h	2h-7h	XX	Reserved	N/A

Render Command Header Format

Render Command Header

Type	Bits				
	31:29	28:24	23	22	21:0
Memory Interface (MI)	000	Opcode 00h - NOP 0Xh - Single DWord Commands 1Xh - Two+ DWord Commands 2Xh - Store Data Commands 3Xh - Ring/Batch Buffer Cmds		Identification No./DWord Count Command Dependent Data 5:0 - DWord Count 5:0 - DWord Count 5:0 - DWord Count	

Type	Bits					
	31:29	28:24	23	22:17	16:10	9:0

Type	Bits				
	31:29	28:24	23:19	18:16	15:0
Reserved	001, 010	Opcode - 11111	Sub Opcode 00h - 01h	Reserved	DWord Count

Type	Bits					
	31:29	28:27	26:24	23:16	15:8	7:0
Common	011	00	Opcode - 000	Sub Opcode	Data	DWord Count
Common (NP)[1]	011	00	Opcode - 001	Sub Opcode	Data	DWord Count
Reserved	011	00	Opcode - 010 - 111			
Single Dword Command	011	01	Opcode - 000 - 001	Sub Opcode		N/A
Reserved	011	01	Opcode - 010 - 111			
Media State	011	10	Opcode - 000	Sub Opcode		Dword Count
Media Object	011	10	Opcode - 001 - 010	Sub Opcode	Dword Count	
Reserved	011	10	Opcode - 011 - 111			
3DState (Pipelined)	011	11	Opcode - 000	Sub Opcode	Data	DWord Count
3DState (NP)[1]	011	11	Opcode - 001	Sub Opcode	Data	DWord Count
PIPE_Control	011	11	Opcode - 010		Data	DWord Count
3DPrimitive	011	11	Opcode - 011		Data	DWord Count
Reserved	011	11	Opcode - 100			
Reserved	011	11	Opcode - 101			
Reserved	011	11	Opcode - 110 - 111			
Reserved	100	XX				
Reserved	101	XX				
Reserved	110	XX				

Notes:

[1]The qualifier "NP" indicates that the state variable is non-pipelined, and the render pipe is flushed before such a state variable is updated. The other state variables are pipelined (default).

Render Engine Preemption

Render Engine Command Streamer Preemptable Commands

Preemptable Command	Condition
MI_ARB_CHECK	AP
Element Boundary	AP (if allowed)
Semaphore Wait	Unsuccessful & AP
Wait for Event	Unsuccessful & AP (if allowed)
3DPRIMITIVE	Object Level (if enabled[1])
GPGPU_WALKER	Mid-Thread (if enabled[2])
PIPE_CONTROL[3]	PIPESEL_GPGPU MODE / PIPESEL-MEDIA MODE
MEDIA STATE FLUSH	Mid-Thread (if enabled[2])
MEDIA_OBJECT_WALKER / MEDIA_OBJECT	Thread Group
PIPELINE_SELECT	PIPESEL-GPGPU Mode / PIPESEL-MEDIA MODE
Any Non-Pipelined State[4]	PIPESEL-GPGPU Mode / PIPESEL-MEDIA MODE

Table Notes:

AP - Allow Preemption if arbitration is enabled.

1. 0x20EC bit 0 determines whether the level of preemption is command or object level.

2. 0x20E4 bits 2:1 determine the level of preemption for GPGPU workloads.

3. MI_ATOMIC and MI_SEMAPHORE_SIGNAL commands with Post Sync Op bit set are treated as PIPE_CONTROL command with Post Sync Operation as Atomics or Semaphore Signal.

4. Any Header with the value [31:29] = "011", [28:27] = "00" OR "11" and [26:24] = "001". Refer to **Graphics Command Formats**.

Compute Command Streamer

Enabling Multi context

Multi context mode is enabled by setting the *Compute Engine Enable* bit in the **Render Control Unit Mode** Register.

This is a global control bit and hardware responds to submission of workloads to the Compute CS only when this bit is set (render engine must be idle when programming this bit)

Software Interface

Render CS and the Compute CS have their own independent execution list interfaces that SW can schedule to independently.

As these engines are executing asynchronously:

- Each CS sends its interrupts to both GuC and Host Interrupt block.
- Each CS provides its status independently (using the Context Status structure).

Context Priority

Scheduler SW can set priority of a context by specifying the priority in the context descriptor. Context priority can be changed across submissions - for e.g: submitted as low priority at submission n and then high priority at submission n+1

-- Table of priorities and resulting behavior to be added --

Pre-emption Support

RCS Hardware running in Single-Context Mode

Supported modes of Preemption
• 3D mode: Batch level, Draw (command) level, Object level • GPGPU mode: Thread Group level

RCS and CCS Hardware running in Multi-Context Mode

Supported modes of Preemption
• RCS only supports Thread Group pre-emption. • All compute CS-es only support Thread Group pre-emption.

The following table describes the hardware behavior when pre-emption is invoked in different multi-context scenarios.

Context 0 (R0): Running on RCS	Context1 (C0): Running on COMPUTE CS	Pre-empt Trigger	Behavior
3D	Idle	Preempt R0	Batch/Draw/Object
Compute	Idle	Preempt R0	Thread Group
Idle	Compute	Preempt C0	Thread Group
3D	Compute	Preempt R0	Pre-empt 3D at batch/draw/object
3D	Compute	Preempt C0	Thread group preempt C0

Compute	Compute	Preempt R0	Thread group preempt of R0
Compute	Compute	Preempt C0	Thread group preempt of C0

Resets

When Reset is invoked, the entire render engine is reset.

- This reset impacts all the contexts currently running on the machine.
- Hardware does not support a partial reset - where only one context executing on the render engine is reset.

If the hardware is running multiple contexts and if a context needs to be reset (for e.g. because it is hung), then SW has to evict any other normally running context from the machine before it triggers Reset.

Address Space Considerations

Address Space Programming Note	
Context:	Address Space Constraints on Dual Context

Dual context support is implemented by splitting the 48b address space into two halves:

- RCS context occupies the address space corresponding to bit 47 =0
- GPGPU context occupies the address space corresponding to bit 47 = 1

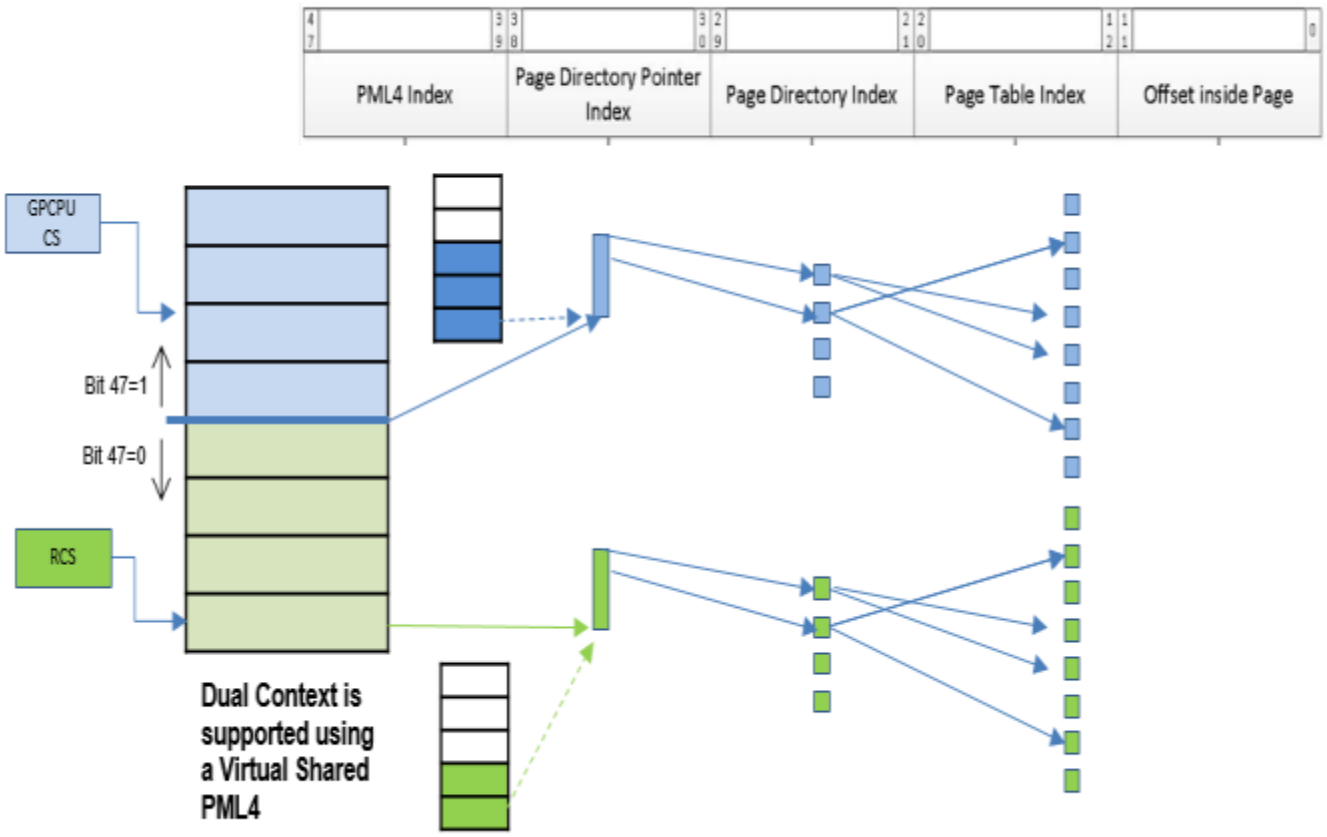
As a result of the above, a context can use the entire 48b address space when not running in dual context mode (matches legacy behavior). When running in dual context mode (i.e Compute Engine Enable bit is set in Graphics Mode Register) contexts are limited to 47b address space.

This model imposes several limitations for SW when running in dual context mode:

- Dual context is supported only in Legacy Context mode with 64b VA
- Applications shall be limited to 47b address space - for e.g: OCL 2.0 fine grained which uses 48b addressing is not supported)

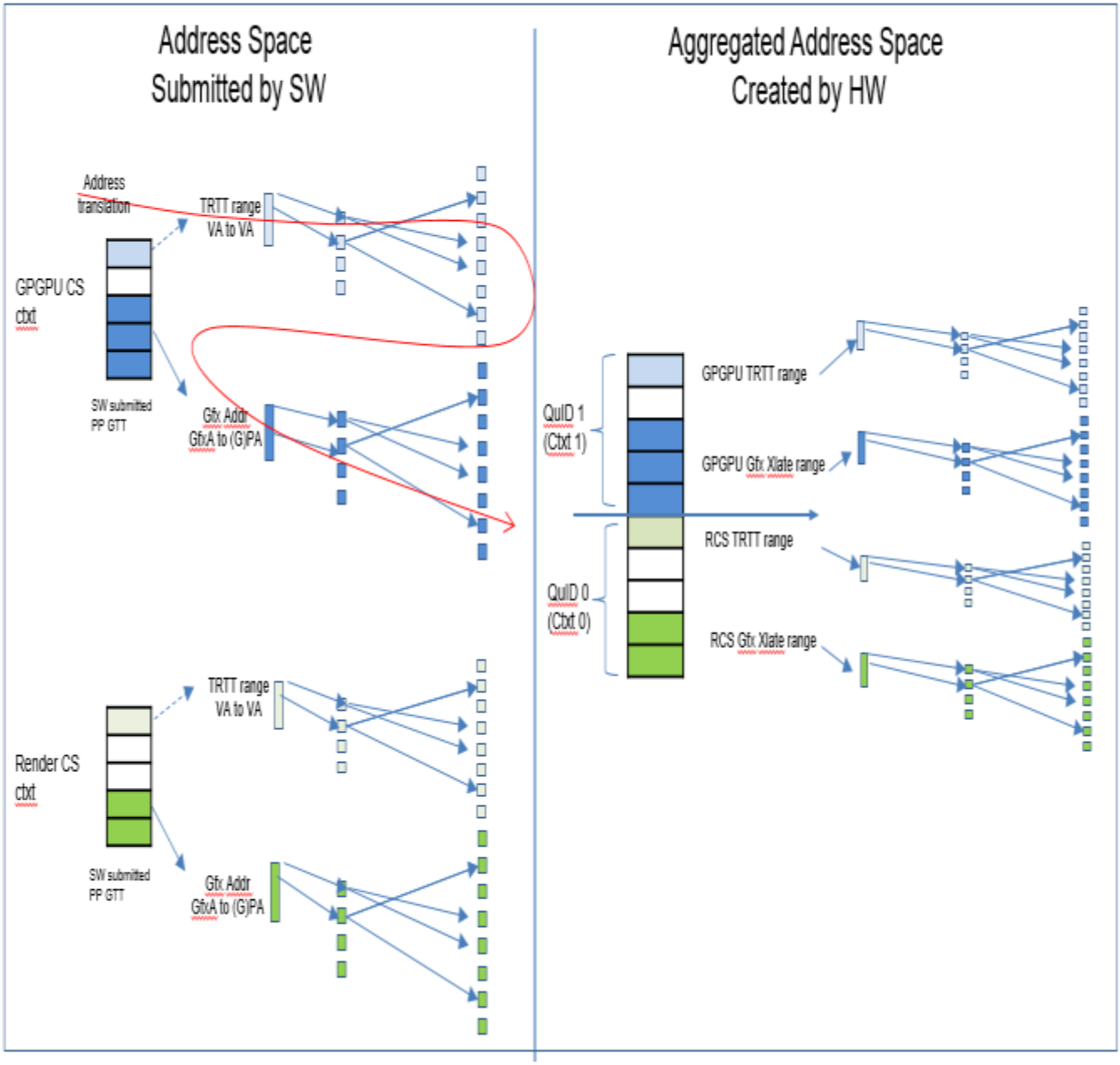
As indicated earlier, dual context is supported by partitioning the address space (by effectively partitioning the PML4) as shown in figure below.

Address Space Programming Note	
Context:	Address Space Constraints on Dual Context



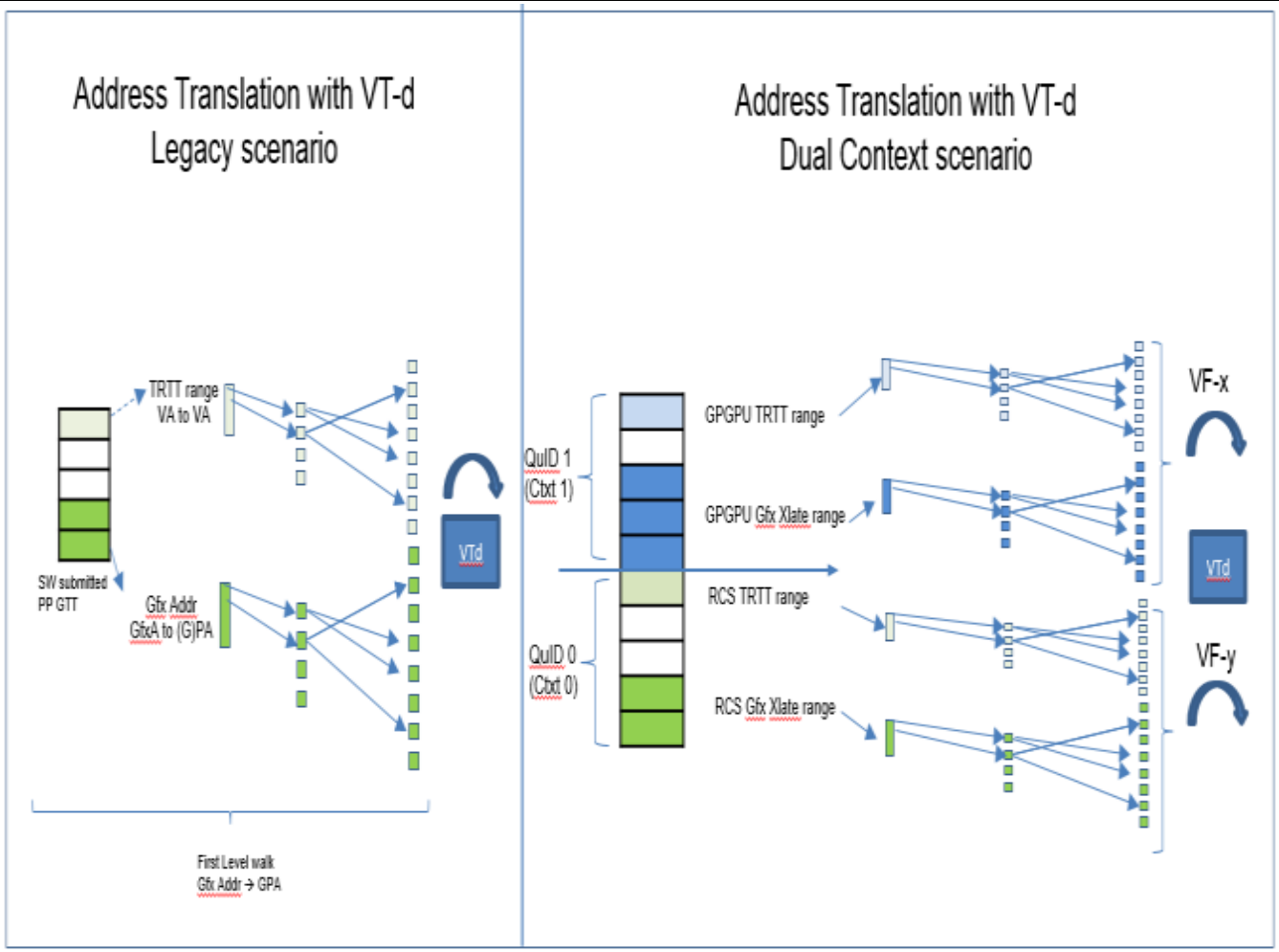
This shared address space needs to be configured to accommodate TR-TT. This is achieved by dropping the TR-TT address space to 46: 44. The following figure depicts the TR-TT spaces in a dual context scenario.

Address Space Programming Note	
Context:	Address Space Constraints on Dual Context



In a virtualized environment, each context can belong to a different Virtual Machine. The hardware allows for independent VT-d page table translation for each context.

Address Space Programming Note	
Context:	Address Space Constraints on Dual Context



CCS Register State Context

This section summarizes the ComputeCS register state context.

Compute Engine Register State Context

Color Coding

EXECLIST CONTEXT
EXECLIST CONTEXT(PPGTT Base)
ENGINE CONTEXT
EXTENDED ENGINE CONTEXT

Description	MMIO Offset/Command	Unit	# of DW
NOOP		CSEL	1
MI_LOAD_REGISTER_IMM	0x1108_1019	CSEL	1
Context Save/Restore Control Register	0x1A244	CSEL	2
Ring Buffer Head	0x1A034	CSEL	2
Ring Tail Pointer Register	0x1A030	CSEL	2
RING_BUFFER_START	0x1A038	CSEL	2
RING_BUFFER_CONTROL	0x1A03C	CSEL	2
Batch Buffer Current Head Register (UDW)	0x1A168	CSEL	2
Batch Buffer Current Head Register	0x1A140	CSEL	2
Batch Buffer State Register	0x1A110	CSEL	2
BB_PER_CTX_PTR	0x1A1C0	CSEL	2
CS_INDIRECT_CTX	0x1A1C4	CSEL	2
CS_INDIRECT_CTX_OFFSET	0x1A1C8	CSEL	2
CCID	0x1A180	CSEL	2
SEMAPHORE_TOKEN	0x1A2B4	CSEL	2
NOOP		CSEL	4
NOOP		CSEL	1
MI_LOAD_REGISTER_IMM	0x1108_1011	CSEL	1
CTX_TIMESTAMP	0x1A3A8	CSEL	2
PDP3_UDW	0x1A28C	CSEL	2
PDP3_LDW	0x1A288	CSEL	2
PDP2_UDW	0x1A284	CSEL	2
PDP2_LDW	0x1A280	CSEL	2
PDP1_UDW	0x1A27C	CSEL	2
PDP1_LDW	0x1A2278	CSEL	2

Description	MMIO Offset/Command	Unit	# of DW
PDP0_UDW	0x1A274	CSEL	2
PDP0_LDW	0x1A270	CSEL	2
POSH_LRCA (DUMMY)	0x1A1B0	CSEL	2
CONTEXT_SCHEDULING_ATTRIBUTES (RESOURCE_MIN_MAX_PRIROITY)	0x1A5A8	CSEL	2
PREEMPTION_STATUS	0x1A5AC	CSEL	2
NOOP		CSEL	4
NOOP		CSEL	1
MI_LOAD_REGISTER_IMM	0x1108_0001	CSEL	1
R_PWR_CLK_STATE	0x1A0C8	CSEL	2
GPGPU_CSR_BASE_ADDRESS		CSEL	3
NOOP		CSEL	9
NOOP		CSFE	1
MI_LOAD_REGISTER_IMM	0x1108_1065	CSFE	1
BB_STACK_WRITE_PORT	0x1A588	CSFE	12
EXCC	0x1A028	CSFE	2
MI_MODE	0x1A09C	CSFE	2
INSTPM	0x1A0C0	CSFE	2
PR_CTR_CTL	0x1A178	CSFE	2
PR_CTR_THRSH	0x1A17C	CSFE	2
TIMESTAMP Register (LSB)	0x1A358	CSFE	2
BB_START_ADDR_UDW	0x1A170	CSFE	2
BB_START_ADDR	0x1A150	CSFE	2
BB_ADD_DIFF	0x1A154	CSFE	2
BB_OFFSET	0x1A158	CSFE	2
MI_PREDICATE_RESULT_1	0x1A41C	CSFE	2
CS_GPR (1-16)	0x1A600	CSFE	64
IPEHR	0x1A068	CSFE	2
CMD_BUF_CCTL	0x1A084	CSFE	2
NOOP		CSFE	1
MI_LOAD_REGISTER_IMM	0x1102_100F	CSFE	1
TRTTCR	0x4580	CSFE	2
TRVADR	0x4584	CSFE	2
TRTT_L3_BASE_LOW	0x4588	CSFE	2
TRTT_L3_BASE_HIGH	0x458C	CSFE	2
TR_NULL_GFX	0x4590	CSFE	2
TR_INV	0xF594	CSFE	2

Description	MMIO Offset/Command	Unit	# of DW
AUX_TABLE_BASE_ADDR_LOW	0x42C0	CSFE	2
AUX_TABLE_BASE_ADDR_HIGH	0x42C4	CSFE	2
NOOP		CSFE	6
NOOP		CSBE	1
MI_LOAD_REGISTER_IMM	0x1108_1023	CSBE	1
CSPAVPMODE	0x1A1F0	CSBE	2
CS_CONTEXT_STATUS1	0x1A184	CSBE	2
GPUGPU_DISPATCHDIMX	0x1A500	CSBE	2
GPUGPU_DISPATCHDIMY	0x1A504	CSBE	2
GPUGPU_DISPATCHDIMZ	0x1A508	CSBE	2
MI_PREDICATE_SRC0	0x1A400	CSBE	2
MI_PREDICATE_SRC0	0x1A404	CSBE	2
MI_PREDICATE_SRC1	0x1A408	CSBE	2
MI_PREDICATE_SRC1	0x1A40C	CSBE	2
MI_PREDICATE_DATA	0x1A410	CSBE	2
MI_PREDICATE_DATA	0x1A414	CSBE	2
MI_PRED_RESULT	0x1A418	CSBE	2
GPGPU_THREADS_DISPATCHED	0x1A290	CSBE	4
CS_CHICKEN2	0x1A194	CSBE	4
NOOP		CSBE	4
NOOP		CSBE	2
PIPELINE_SELECT		CSBE	1
STATE_BASE_ADDRESS		CSBE	22
STATE_SIP		CSBE	3
STATE_COMPUTE_MODE		CSBE	2
3DSTATE_BINDING_TABLE_POOL_ALLOC		CSBE	4
NOOP		CSBE	8
NOOP		CSBE	12
MI_BATCH_BUFFER_END		CSEND	1
NOOP		CSEND	127

CCS Power Context Image

This section summarizes the ComputeCS power context.

Compute Engine Power Context

Compute Engine Power Context Image

Description	Offset	Unit	# of DW	Address Offset (PWR)	CSFE/CSBE
CSFE Power context without Display		CSFE	195	0	CSFE
NOOP		CS	1	00C4	CSBE
Load_Register_Immediate header	0x1100_1009	CS	1	00C5	CSBE
NOOP		CS	10	00D0	CSBE
NOOP		CS	1	00DA	CSBE
MI_BATCH_BUFFER_END		CS	1	00DB	CSBE

MI Commands Supported by ComputeCS

ComputeCS supports all the MI commands supported by RCS except for the below exceptions.

ComputeCS doesn't support below commands and SW must not program them as part of the Compute engine's command sequence.

Commands not supported in ComputeCS executed command buffers:

Commands
MI_DISPLAY_FLIP
MI_LOAD_SCANLINES_INCL/EXCL
MI_WAIT_FOR_EVENT
MI_REPORT_PERF_COUNT
MI_FORCE_WAKEUP

GPGPU-Media State Commands Supported by ComputeCS

The table below lists the GPGPU-MEDIA STATE Commands Supported by Compute Engine (ComputeCS). State commands programmed for ComputeEngine which are not listed in the table below will be gracefully discarded (NOOP'd) by ComputeCS.

GPGPU-MEDIA State Commands
• MEDIA_VFE_STATE • MEDIA_CURBE_LOAD • MEDIA_INTERFACE_DESCRIPTOR_LOAD • MEDIA_OBJECT • MEDIA_OBJECT_* • MEDIA_OBJECT_GRPID • MEDIA_WALKER • GPGPU_WALKER

GPGPU-MEDIA State Commands
• MEDIA_STATE_FLUSH
Common Non-Pipeline State Commands
• STATE_BASE_ADDRESS • GPGPU_CSR_BASE_ADDRESS • PIPELINE_SELECT • STATE_SIP
• STATE_COMPUTE_MODE
Misc Commands
• PIPE_CONTROL • 3DSTATE_BINDING_TABLE_POOL_ALLOC

Position Only Shader Command Streamer (POCS)

Position only shader (POSH) is a new geometry pipeline that has the optional ability to execute the position only vertex shaders and perform the visibility test on these vertices before the actual vertex shader is executed. POSH pipe can run ahead of the original geometry pipe by executing position only vertex shaders and doing visibility test on these vertices and recording this information. Geometry pipe when processing the vertices will use this visibility information outputted by POSH pipe to skip the vertex fetch and shading for vertices that are already marked as culled.

POSH pipe has its own command streamer called Position only command streamer (POCS). A context running on render pipe can exercise POSH capabilities through Render Command Streamer (RCS). RCS manages the POSH pipe through POCS for POSH enabled contexts. Render command streamer loads the context to execute on POCS when a POSH enable context execution begins in render pipe, similarly preempts context executing in POCS when the POSH enabled context switches out of render pipe. Once POCS is loaded with context it starts executing the ring buffer similar to RCS, refer Programming Model section for more details.

Position Only Command Streamer (POCS)

The POCS (Position Only Shader Command Streamer) unit primarily serves as the programming interface between the render command streamer and the POSH pipe. It is responsible for fetching, decoding, and dispatching of data packets (3D Commands with the header DWord removed) for the POSH pipe.

Logic Functions Included

- MMIO register programming interface.
- DMA action for fetching of ring buffer and batch buffer data from memory.
- Management of the Head pointer for the Ring Buffer.
- Decode and execution of command programmed in ring buffer and batch buffers.

- Flushing the POSH pipe.

Handle NOOP.

The POCS unit only claims memory mapped I/O cycles that are targeted to its range of 0x1_8000 to 0x1_9FFF. The POCS and RCS use semaphore to synchronize their operations.

POCS ExecList Scheduling

Position Only Engine Command Streamer Preemptable Commands

Preemptable Command	Condition
MI_ARB_CHECK	AP
Element Boundary	AP (if allowed)
Semaphore Wait	Unsuccessful & AP
Wait for Event	Unsuccessful & AP (if allowed)
3DPRIMITIVE	Object Level (if enabled[1])
3DSTATE_PTBR_TILE_PASS_INFO	AP

Table Notes:

AP - Allow Preemption if arbitration is enabled.

1. 0x20EC bit 0 determines whether the level of preemption is command or object level.

POSH Programming Model

The POSH + Render pipeline will appear as a monolithic engine from SW perspective. Render Command Streamer (RCS) is hardware front end interface to the SW for the modified Render + POSH pipeline. SW will use a single context (and associated LRCA) to submit work to the modified Render + POSH pipeline through its associated ring buffer.

Context submission model should be visualized as context submitted to RCS. RCS will set up the context definition in HW and triggers POSH pipe to execute the same context, resulting in execution of the same ring buffer by render pipe and POSH pipe in parallel. POSH pipe has its own command streamer called POCS (POSH Command Streamer). Similarly, when the context is switched out on the render pipe due to whatever reasons (Wait For Event, Semaphore Wait or Preemption due to pending execlist), RCS will ensure POSH pipe is preempted and its corresponding logic state is saved through POCS.

POCS and RCS get to see the same ring buffer, however the execution of the same ring buffer by POCS and RCS are asynchronous to each other and its SW responsibility to ensure POCS and RCS are synchronized through semaphores as and when required. SW will provide independent command buffers (batch buffers) to be executed by RCS and POCS. Marking of batch buffers for POCS and RCS and execution of ring buffer are detailed in the latter subsections.

This model of execution has the following implications:

- POCS and RCS have to run on the same context definition. RCS sets up context with GAM and POCS runs within this address space.

- Even though the currently running context may not be utilizing the POSH pipe, a waiting context with POSH enabled has to wait for the current context to be evicted. (waiting context cannot take advantage of the idle POSH pipe ahead of getting scheduled on the render engine)

POSH Enabled Context

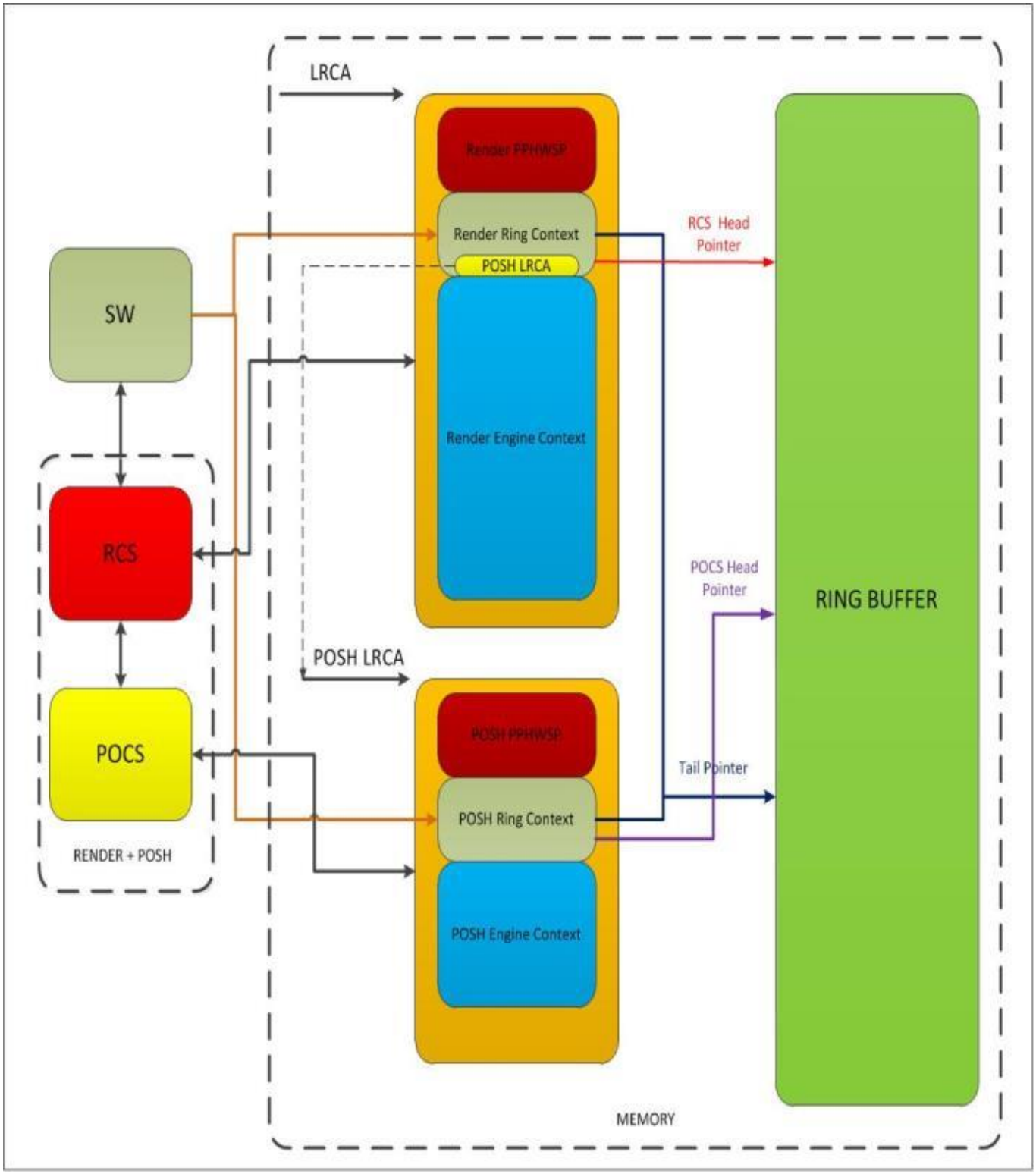
A context submitted to render engine exercising POSH functionality is called "POSH Enabled" context. Application (UMD) decides if a context is POSH Enabled at the time of context creation. A context is indicated as POSH enabled to HW by setting "POSH Enable" bit in CTX_SR_CTL register of RCS. SW allocates additional separate memory space (POSH LRCA) for the POSH Enabled contexts. POSH pipe uses the POSH LRCA for its context state management.

Context Submission and LRCA for POSH

SW will continue to submit POSH enabled contexts to ELSP in RCS. There is no change in the pending execlist submission or context switch status report mechanism to/from RCS.

Listed below are the SW changes required for submission of the POSH enabled context:

- "POSH Enable" bit in CTX_SR_CTL of RCS must be set to indicate POSH enabled context to HW. Refer POSH functionality control section for the bit definition and programming.
- POSH LRCA is provided to RCS through register programming in the ring context of RCS. Refer RCS ring context details below.
- POSH LRCA format is similar to that of RCS, i.e PPHWSP followed by ring context followed by the engine context. However POSH ring context will only have the ring buffer and batch buffer details. POSH ring context will not have the page directory pointers details as the PPGTT is setup by RCS.
- SW does not control POCS context ID independently. The context ID for POCS will be supplied from RCS, and thus will be the same.
- SW must update the ring context of POSH with ring buffer details on the very first submission and whenever the ring buffer start address, control and head pointer details are updated. POSH pipe (POCS) will sample the tail pointer from RCS. Note that the POCS and RCS share the same ring buffer.


LRCA
Render PPHWSP
Render Ring Context
POSH LRCA
RCS Head Pointer
SW
Render Engine Context
RCS
POCS Head Pointer
RING BUFFER
POSH LRCA
POCS
POSH PPHWSP
Tail Pointer
POSH Ring Context
RENDER + POSH
POSH Engine Context
MEMORY

POSH LRCA in RCS Ring Context

The table below highlights the POSH LRCA details in RCS ring context. Ring context listed below is for illustration of the change, "Register State Context" in "Render Logical Context Data" should be referred as the final format for implementation.

Description	Unit	# of DW
NOOP	CSEL	1
MI_LOAD_REGISTER_IMM	CSEL	1
Ring Buffer Head	CSEL	2
Ring Tail Pointer Register	CSEL	2
RING_BUFFER_START	CSEL	2
RING_BUFFER_CONTROL	CSEL	2
Batch Buffer Current Head Register (UDW)	CSEL	2
Batch Buffer Current Head Register	CSEL	2
Batch Buffer State Register	CSEL	2
SECOND_BB_ADDR_UDW	CSEL	2
SECOND_BB_ADDR	CSEL	2
SECOND_BB_STATE	CSEL	2
BB_PER_CTX_PTR	CSEL	2
RCS_INDIRECT_CTX	CSEL	2
RCS_INDIRECT_CTX_OFFSET	CSEL	2
NOOP	CSEL	2
NOOP	CSEL	1
MI_LOAD_REGISTER_IMM	CSEL	1
CTX_TIMESTAMP	CSEL	2
PDP3_UDW	CSEL	2
PDP3_LDW	CSEL	2
PDP2_UDW	CSEL	2
PDP2_LDW	CSEL	2
PDP1_UDW	CSEL	2
PDP1_LDW	CSEL	2
PDP0_UDW	CSEL	2
PDP0_LDW	CSEL	2
MI_LOAD_REGISTER_IMM	CSEL	1
POSH_LRCA	CSEL	2
NOOP	CSEL	9
NOOP	CSEL	1
MI_LOAD_REGISTER_IMM	CSEL	1

Description	Unit	# of DW
R_PWR_CLK_STATE	CSEL	2
GPGPU_CSR_BASE_ADDRESS	CSEL	3
NOOP	CSEL	9

POCS Ring Context

Table below details the POSH ring context. Ring context listed below is for illustration of the change, "Register State Context" in "Render Logical Context Data" should be referred as the final format for implementation.

Description	Unit	# of DW
NOOP	CSEL	1
MI_LOAD_REGISTER_IMM	CSEL	1
Ring Buffer Head	CSEL	2
Ring Tail Pointer Register	CSEL	2
RING_BUFFER_START	CSEL	2
RING_BUFFER_CONTROL	CSEL	2
Batch Buffer Current Head Register (UDW)	CSEL	2
Batch Buffer Current Head Register	CSEL	2
Batch Buffer State Register	CSEL	2
SECOND_BB_ADDR_UDW	CSEL	2
SECOND_BB_ADDR	CSEL	2
SECOND_BB_STATE	CSEL	2
BB_PER_CTX_PTR	CSEL	2
RCS_INDIRECT_CTX(Always Invalid)	CSEL	2
RCS_INDIRECT_CTX_OFFSET	CSEL	2
NOOP	CSEL	2

Description	Unit	# of DW
NOOP	CSEL	48

POSH Command Transport

These following subtopics describe the command transport mechanism from SW to POCS.

"POSH Start" Batch Buffers

Batch buffers dedicated to being executed by POSH pipe are indicated by setting the field "POSH Start" in the MI_BATCH_BUFFER_START command header. Once "POSH Start" is set in a batch buffer all the following chained batch buffers and next level batch buffers will implicitly inherit the "POSH Start" field value. Once "POSH Start" is set in a batch buffer all the following command sequences are to be **executed** by POCS until the corresponding batch buffer sequencing is terminated through MI_BATCH_BUFFER_END/MI_CONDITIONAL_BATCH_BUFFER_END command.

Example:

- Once "POSH Start" is encountered in a first level batch buffer by HW, it will get reset only when the first level batch buffer execution is terminated through batch buffer end and the command execution sequence goes back to the ring buffer,
- Similarly, once "POSH Start" is encountered in a second level batch buffer by HW, it will get reset only when the second level batch buffer execution is terminated through batch buffer end and the command execution sequence goes back to the first level buffer,
- Similarly, once when "POSH Start" is encountered in a third level batch buffer by HW, it will get reset only when the third level batch buffer execution is terminated through batch buffer end and the command execution sequence goes back to the second level batch buffer.

Command sequences executed from the "POSH Start" batch buffer may lead to chained batch buffers or next level batch buffers. Batch buffers executed by POCS may have MI Commands, 3DSATE commands and 3DPRIMTIVE commands for POSH pipe, however these will be a subset of the commands that are supported by render pipe. RCS on parsing MI_BATCH_BUFFER_START command with "POSH Start" enabled NOOPS the command and moves on the following command.

MI Commands Supported by POCS

POCS supports all the MI commands supported by RCS except for the below exceptions.

POCS doesn't support below commands and SW must not program them as part of the POSH command sequence.

Commands not supported in POSH executed command buffers:

Column Title1
MI_DISPLAY_FLIP
MI_LOAD_SCANLINES_INCL/EXCL

Column Title1
MI_WAIT_FOR_EVENT
MI_USER_INTERRUPT
MI_REPORT_PERF_COUNT
MI_SET_CONTEXT
MI_ARB_ON_OFF

POCS can semaphore signal RCS and vice-versa, however POCS will not support semaphore signal forwarding to GUC and also RCS will not forward semaphore signals received from POCS to GUC.

3D State Commands Supported by POCS

The table below lists the 3DSTATE Commands Supported by POSH Pipe. State commands programmed for POSH which are not listed in the table below will be gracefully discarded (NOOP'd) by POCS.

3D State Commands
• 3DSTATE_VF • 3DSTATE_INDEX_BUFFER • 3DSTATE_VERTEX_BUFFER • 3DSTATE_VERTEX_ELEMENTS • 3DSTATE_VF_COMPONENT_PACKING • 3DSTATE_VF_INSTANCING • 3DSTATE_VF_SGVS • 3DSTATE_VF_TOPOLOGY • 3DSTATE_VF_STATISTICS • 3DPRIMTIVE • 3DSTATE_VS • 3DSTATE_PUSH_CONSTANT_ALLOC_VS • 3DSTATE_CONSTANT_VS • 3DSTATE_BINDING_TABLE_POOL_ALLOC • 3DSTATE_BINDING_TABLE_POINTERS_VS • 3DSTATE_SAMPLER_STATE_POINTERS_VS • 3DSTATE_URB_VS • 3DSTATE_CLIP • 3DSTATE_SFFE • 3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP • 3DSTATE_SCISSOR_STATE_POINTERS • 3DSTATE_MULTISAMPLE • 3DSTATE_RASTER • 3DSTATE_DRAWING_RECTANGLE • 3DSTATE_INT

3D State Commands
• PIPECONTROL Command • 3DSTATE_SBE (for PID computation)
• 3DSTATE_PRIMITVE_REPLICATION
• 3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS • 3DSTATE_PTBR_FREE_LIST_BASE_ADDRESS • 3DSTATE_PTBR_RENDER_LIST_BASE_ADDRESS • 3DSTATE_PTBR_TILE_PASS_INFO
Common Non-Pipeline Sate Commands
• STATE_BASE_ADDRESS

"POSH Enable" Batch Buffers

POCS parses/traverses (doesn't execute) the ring buffer to look for batch buffers programmed with "POSH Start" field set. "POSH Enable" field in the MI_BATCH_BUFFER_START command is a hint to POCS to traverse (parse, don't execute) the batch buffer to look for "POSH Start" batch buffers. "POSH Enable" field is only inherited to the chained batch buffer and doesn't get inherit to the next level batch buffers unlike "POSH Start" field. "POSH Enable" field must be explicitly set in the MI_BATCH_BUFFER_START command which calls the next level batch buffers in order for the POCS to parse them to look for "POSH Start" batch buffers. POCS ends the "POSH Enable" batch buffer on executing MI_BATCH_BUFFER_END or on MI_CONDITIONAL_BATCH_BUFFER_END meeting the required condition. "POSH Start" field takes precedence over the "POSH Enable" field in POCS.

Example:

- Once "POSH Enable" is encountered in a first level batch buffer, POCS will traverse the whole of the first level batch buffers (including chained first level) to check for "POSH Start" field in MI_BATCH_BUFFER_START command. POCS by default will not traverse the second level batch buffers. SW must explicitly set the "POSH Enable" field for the second level batch buffer called from first level batch buffer if the second level batch buffer have to be traversed by POCS.
- Similarly, Once "POSH Enable" is encountered in a second level batch buffer, POCS will traverse the whole of the second level batch buffers (including chained second level) to check for "POSH Start" field in MI_BATCH_BUFFER_START command. POCS by default will not traverse the third level batch buffers. SW must explicitly set the "POSH Enable" field for the third level batch buffer called from second level batch buffer if the third level batch buffer have to be traversed by POCS.
- Similarly, Once "POSH Enable" is encountered in a third level batch buffer, POCS will traverse the whole of the third level batch buffers (including chained second level) to check for "POSH Start" field in MI_BATCH_BUFFER_START command.

RCS ignores "POSH Enable" field and has no implications due to the "POSH Enable" field set in the MI_BATCH_BUFFER_START command.

POSH Ring Buffer

POCS and RCS share the same ring buffer. POCS parses the ring buffer to look for batch buffers start commands with "POSH Enable" or "POSH Start" fields set, it doesn't execute any commands programmed in the ring buffer. POCS and RCS executing the same ring buffer results in two different command sequences based on the "POSH Start" and "POSH Enable" fields programmed in various batch buffers.

Examples for Command Buffer Execution by POCS and RCS

There are various possible combinations of command sequences programmable based on the below fields. Example below states few scenarios.

- POSH Start
- POSH Enable
- First Level Batch Buffers
- Second Level Batch Buffers
- Third Level Batch Buffers
- Chaining of First Level Batch Buffers
- Chaining of Second Level Batch Buffers
- Chaining of Third Level Batch Buffers

Acronyms used in the illustrative example below:

BB: Batch Buffer; **BBS**: Second Level batch buffer; **BBT**: Third Level batch buffer; **BBE**: Batch Buffer End; **PS**: POSH Start; **PE**: POSH Enable; **X**: Field programmed is don't care by HW

In the figures below:

- First figure depicts the command sequence programmed in the ring buffer that is submitted to RCS & POCS.
- Second figure depicts the command sequence outputted by POCS on executing the ring buffer submitted by SW.
- Third figure depicts the command sequence outputted by RCS on executing the ring buffer submitted by SW.

Note that the same ring buffer results in different command sequence execution from POCS and RcS.

The figure below depicts the command sequence programmed in the ring buffer that is submitted to RCS & POCS.

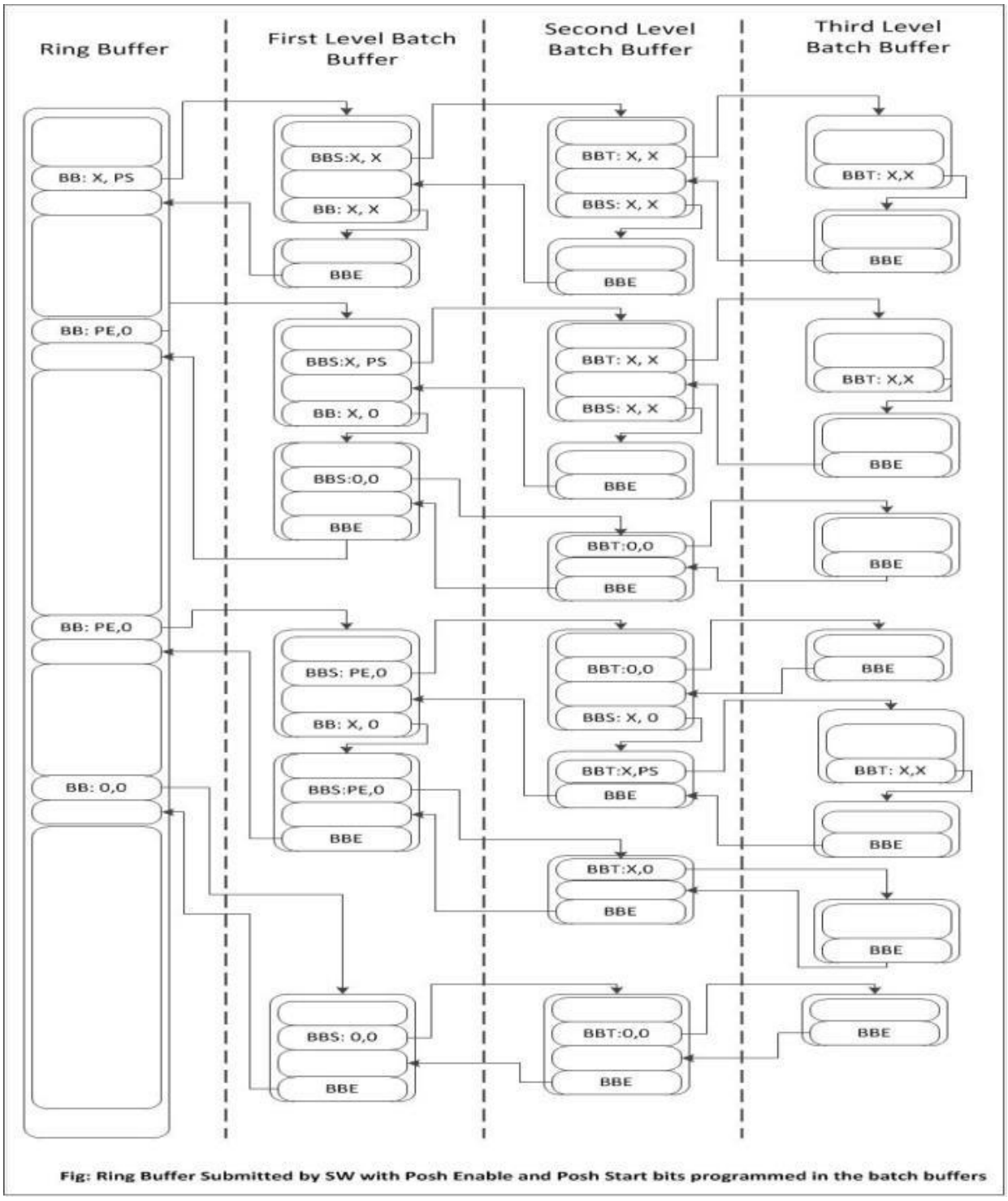


Fig: Ring Buffer Submitted by SW with Posh Enable and Posh Start bits programmed in the batch buffers

The figure below depicts the command sequence outputted by POCS on executing the ring buffer submitted by SW in the above diagram.

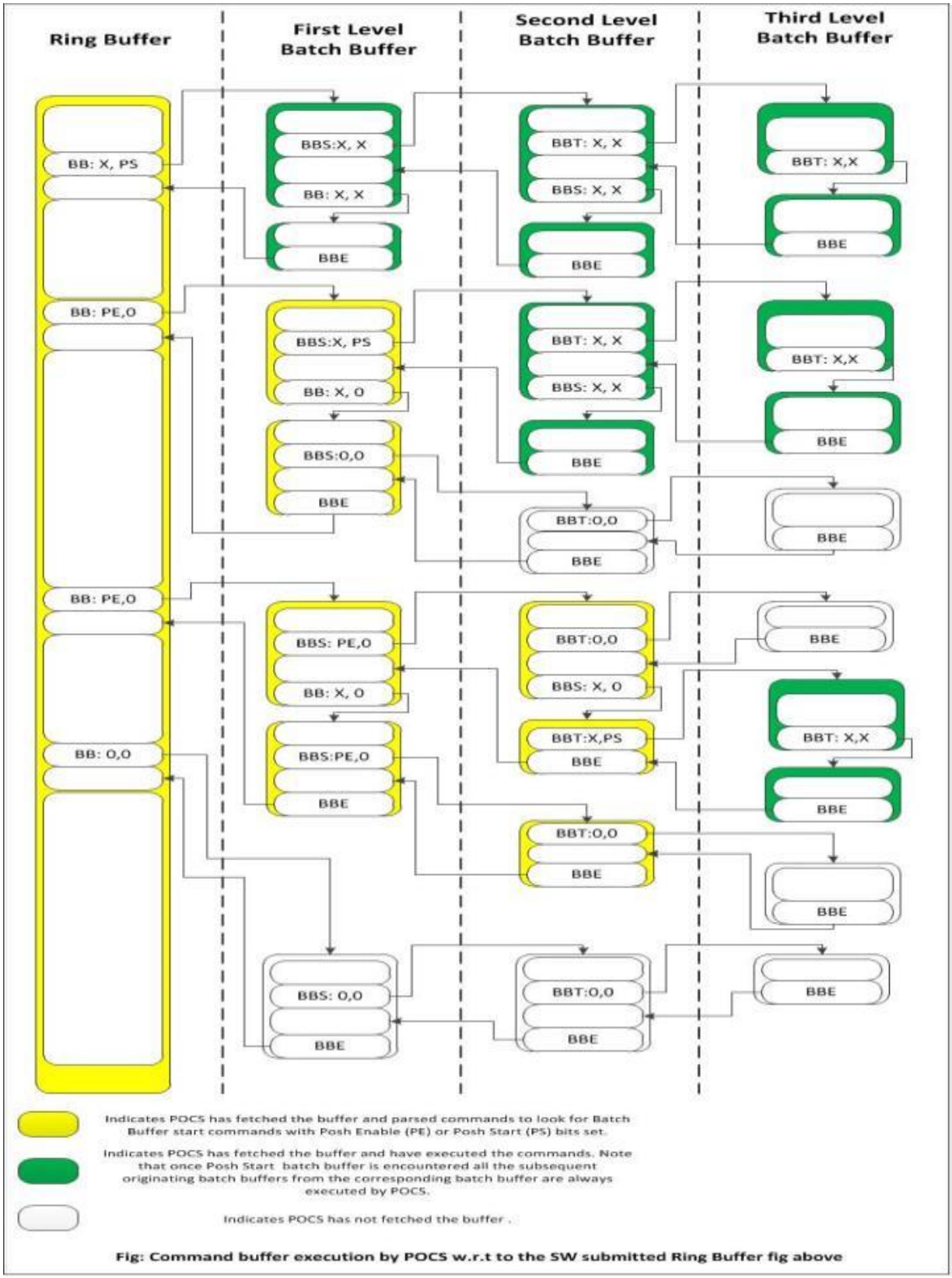


Fig: Command buffer execution by POCS w.r.t to the SW submitted Ring Buffer fig above

The figure below depicts the command sequence outputted by RCS on executing the ring buffer submitted by SW.

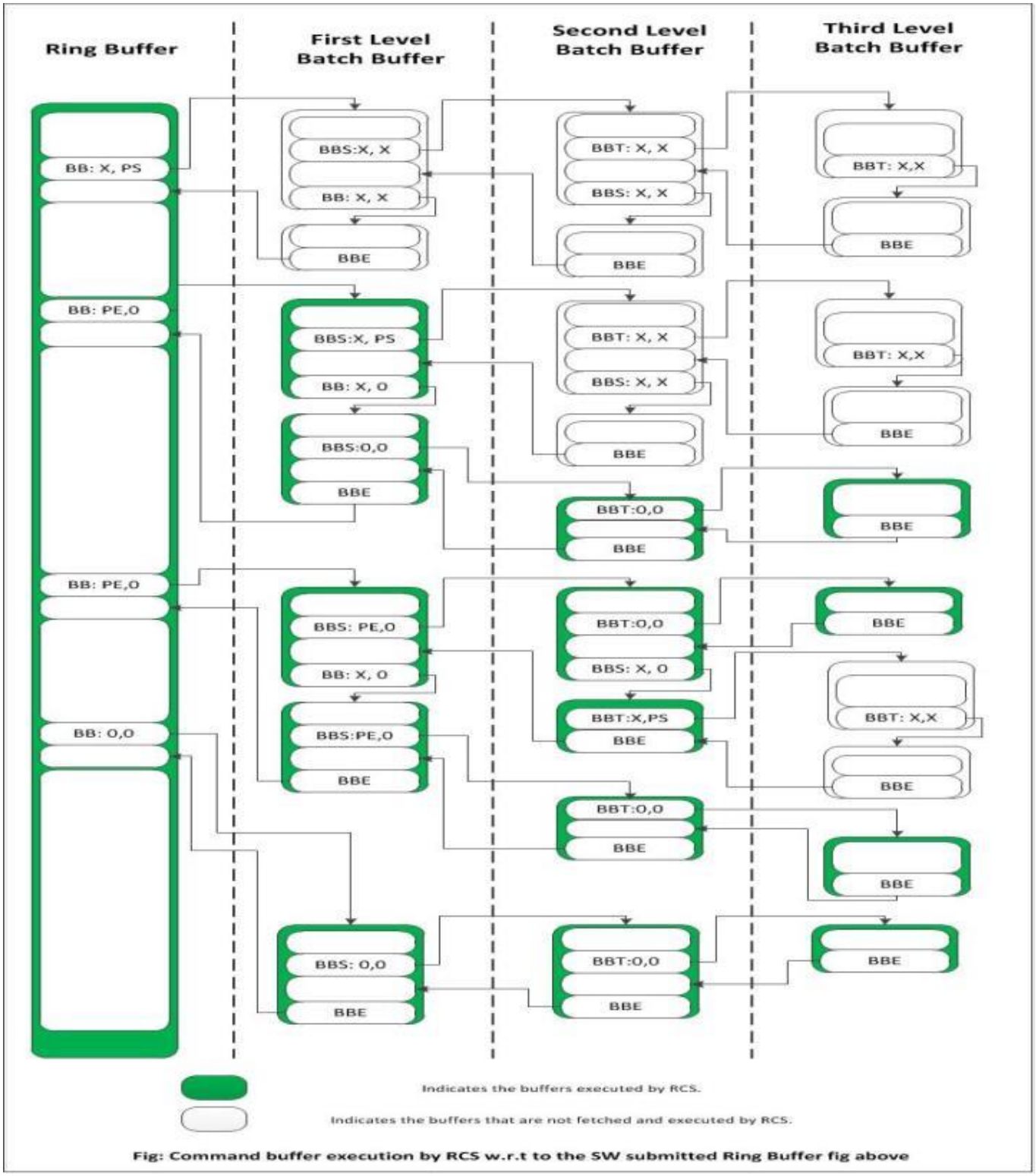


Fig: Command buffer execution by RCS w.r.t to the SW submitted Ring Buffer fig above

POSH Preemption

Once the context is loaded to POCS, only way it can be switched out is through explicit preemption from RCS, POCS doesn't switch out an context on encountering un-successful Wait for Events or Semaphore

Wait or running out of commands on head equal to tail pointer. RCS on switching out the context either due to synchronous context switch or preemption, it also preempts POCS if the context is POSH enabled. POCS receives preemption from RCS and triggers the preemption flow for POSH pipe. POSH pipe supports 3D object level preemption. Preemption from RCS can happen when POCS is in one of the below states:

- POCSFE has executed the context and have Head Equals Tail.
- POCSFE is busy executing commands.

POSH MMIO

POSH pipe implements its own set of MMIO registers similar to render pipe, however POSH pipe implements the registers relevant to the functionality supported in POSH pipeline. Listed below are the only registers that are accessible in POSH pipeline.

Table 1: Registers in POCSFE

MMIO SYMBOL	Suffix
DMA_FADD	POCS
ACTHD	POCS
ACTHD_UDW	POCS
CS_ALU_ACCU	POCS
CS_ALU_CF	POCS
CS_ALU_SRCA	POCS
CS_ALU_SRCB	POCS
CS_ALU_ZF	POCS
BB_ADDR	POCS
BB_ADDR_DIFF	POCS
BB_ADDR_UDW	POCS
BB_OFFSET	POCS
BB_PER_CTX_PTR	POCS
BB_PREEMPT_ADDR	POCS
BB_PREEMPT_ADDR_UDW	POCS
BB_START_ADDR	POCS
BB_START_ADDR_UDW	POCS
BB_STATE	POCS
CCID	POCS
CSFE_CHICKEN1	POCS
CTXT_PREMP_DBG	POCS
CTXT_SR_CTL	POCS
CXT_EL_OFFSET	POCS

MMIO SYMBOL	Suffix
CMD_CCTL_0	POCS
RCS_CTXID_PREEMPTION_HINT	POCS
CTX_TIMESTAMP	POCS
CTX_WA_BB_ADDR	POCS
EXCC	POCS
FORCE_TO_NONPRIV_0	POCS
FORCE_TO_NONPRIV_1	POCS
FORCE_TO_NONPRIV_2	POCS
FORCE_TO_NONPRIV_3	POCS
FORCE_TO_NONPRIV_4	POCS
FORCE_TO_NONPRIV_5	POCS
FORCE_TO_NONPRIV_6	POCS
FORCE_TO_NONPRIV_7	POCS
FORCE_TO_NONPRIV_8	POCS
FORCE_TO_NONPRIV_9	POCS
FORCE_TO_NONPRIV_10	POCS
FORCE_TO_NONPRIV_11	POCS
CS_GPR_R_0	POCS
CS_GPR_R_1	POCS
CS_GPR_R_2	POCS
CS_GPR_R_3	POCS
CS_GPR_R_4	POCS
CS_GPR_R_5	POCS
CS_GPR_R_6	POCS
CS_GPR_R_7	POCS
CS_GPR_R_8	POCS
CS_GPR_R_9	POCS
CS_GPR_R_10	POCS
CS_GPR_R_11	POCS
CS_GPR_R_12	POCS
CS_GPR_R_13	POCS
CS_GPR_R_14	POCS
CS_GPR_R_15	POCS
GFX_MODE	POCS
HWS_PGA	POCS
PWRCTX_MAXCNT	POCS
IPEHR	POCS

MMIO SYMBOL	Suffix
IDLEDLY	POCS
CSCMDOP	POCS
CSCMDVLD	POCS
INSTPM	POCS
INSTPS	POCS
MI_PREDICATE_RESULT_1	POCS
MI_PREDICATE_RESULT_2	POCS
MI_MODE	POCS
NOPID	POCS
PDP0	POCS
PDP1	POCS
PDP2	POCS
PDP3	POCS
PR_CTR_THRSH	POCS
PREEMPTDLY	POCS
PREEMPTION_HINT	POCS
PREEMPTION_HINT_UDW	POCS
DMA_FADD_P_UDW	POCS
RING_BUFFER_CTL	POCS
RING_BUFFER_HEAD	POCS
RING_BUFFER_HEAD_PREEMPT_REG	POCS
RING_BUFFER_START	POCS
RING_BUFFER_TAIL	POCS
TIMESTAMP	POCS
RESET_CTRL	POCS
SBB_ADDR	POCS
SBB_ADDR_UDW	POCS
SBB_PREEMPT_ADDR	POCS
SBB_PREEMPT_ADDR_UDW	POCS
SBB_STATE	POCS
SEMA_WAIT_POLL	POCS
RC_PSMI_CTRL	POCS
CURRENT_LRCA	POCS

Table 2: Registers in POCSBE

MMIO SYMBOL	Suffix
3DPRIM_BASE_VERTEX	POCS
3DPRIM_END_OFFSET	POCS
3DPIM_INSTANCE_COUNT	POCS
3DPRIM_START_INSTANCE	POCS
3DPRIM_START_VERTEX	POCS
3DPRIM_VERTEX_COUNT	POCS
3DPRIM_XP0	POCS
3DPRIM_XP1	POCS
3DPRIM_XP2	POCS
IA_PRIMITIVES_COUNT	POCS
IA_VERTICES_COUNT	POCS
VS_INVOCATION_COUNT	POCS
CL_INVOCATION_COUNT	POCS
CL_PRIMITIVES_COUNT	POCS
MI_PREDICATE_DATA	POCS
MI_PREDICATE_RESULT	POCS
MI_PREDICATE_SRC0	POCS
MI_PREDICATE_SRC1	POCS
CSBEFSM	POCS
CSFLFLAG	POCS
CSFLFSM	POCS
CSFLTRK	POCS
CS_CONTEXT_STATUS1	POCS
CTX_RESTORE_ACK_0	POCS
CTX_RESTORE_ACK_1	POCS
FF_MODE	POCS
STATE_ACK	POCS
STATE_ACK_SLICE1	POCS
STATE_ACK_SLICE2	POCS
STATE_ACK_SLICE3	POCS
State_ACK_Register_Slice_5	POCS
State_Ack_Register_Slice4	POCS

POSH Functionality Controls

POSH functionality enabling and disabling is hierarchically controlled at various levels in the context execution flow.

- Context Granularity
- Batch Buffer Granularity
- 3DPRIMTIVE Granularity

POSH Control	Description
Context Granularity	POSH feature can be enabled or disabled at context level by programming the "POSH Enable" field in CTX_SR_CTL register of the RCS. When POSH is disabled in CTX_SR_CTL register, RCS will not engage POSH. Usage model is one time programming of "POSH Enable" field at context creation time. Dynamic enabling or disabling of POSH during context execution should be achieved through Batch Buffer and 3DPRIMTIVE granularity controls.
Batch Buffer Granularity	**POSH Enable:** "POSH Enable" field in MI_BATCH_BUFFER_START command indicates the possibility of encountering "POSH Start" batch buffer from the corresponding command sequence. **POSH Start:** Commands to be executed by the POCS must be programmed in a dedicated batch buffer and this batch buffer is indicated with a bit "POSH Start" in the MI_BATCH_BUFFER_START command. Once POCS encounters the batch buffer with "POSH Start" it executes all the command in the corresponding batch buffer and also the chained batch buffers from the corresponding buffer. RCS skips the MI_BATCH_BUFFER_START command with "POSH Start" set and goes on the following command. **Programming Notes:** POCS executes only the MI_BATCH_BUFER_START commands programmed in the ring buffer with "POSH Enable" set and NOOPS (predicates) all the other commands in the ring buffer. POCS only parses/traverses the batch buffer with "POSH Enable" to check for any batch buffer programmed with "POSH Start" set. SW must set "POSH Enable" field in the MI_BATCH_BUFFER_START command programmed in ring buffer if the commands in the corresponding batch buffer or the chained batch buffers (includes Second Level and third level) has at least one batch buffer start command with "POSH Start" set (also implies 3DPRIMITIVE command for which POSH is enabled).
3DPRIMTIVE Granularity	"POSH Enable" field in the 3DPRIMTIVE command indicates the POSH pipe to create the visibility recording data and indicates Render pipe to use visibility recording data for the corresponding 3DPRIMTIVE command.

POSH Interrupts

There are no interrupts generated by POSH pipe.

Graphics Pipeline Software Interface

This section covers the Graphics pipeline software interface.

Synchronization of the Graphics Pipeline

Two types of synchronizations are supported for the 3D pipe: top of the pipe and end of the pipe. Top of the pipe synchronization really enforces the read-only cache invalidation. This synchronization guarantees that primitives rendered after such synchronization event fetches the latest read-only data from memory. End of the pipe synchronization enforces that the read and/or read-write buffers do not have outstanding hardware accesses. These are used to implement read and write fences as well as to write out certain statistics deterministically with respect to progress of primitives through the pipeline (and without requiring the pipeline to be flushed.) The PIPE_CONTROL command (see details below) is used to perform all of above synchronizations.

Top-of-Pipe Synchronization

Top-of-pipe synchronization refers to SW actions to prepare HW for new state-binding at the beginning of the rendering sequence in a given context. HW may have residual states cached in the state-caches and read-only surfaces in various caches. With new rendering sequence, read-only surfaces may go through change in the binding. Hence read-only invalidation is required before such new rendering sequence. Read-only cache invalidation is top-of-pipe synchronization. Upon parsing this specific pipe-control command, HW invalidates all caches in GT domain that have read-only surfaces but does not guarantee invalidation beyond GT caches

Upon parsing this specific pipe-control command, HW invalidates all caches in GT domain that have read-only surfaces but does not guarantee invalidation beyond GT caches (i.e. LLC).

Further, HW does not guarantee that all prior accesses to those read-only surfaces have completed. Therefore, SW must guarantee that there are no pending accesses to those read-only surfaces before initializing the top-of-pipe synchronization. PIPE-CONTROL command described below allows for invalidating individual read-only stream type. It is recommended that driver invalidates only the required caches on the need basis so that cache warm-up overhead can be reduced.

End-of-Pipe Synchronization

The driver can use end-of-pipe synchronization to know that rendering is complete (although not necessarily in memory) so that it can deallocate in-memory rendering state, read-only surfaces, instructions, and constant buffers. An end-of-pipe synchronization point is also sufficient to guarantee that all pending depth tests have completed so that the visible pixel count is complete prior to storing it to memory. End-of-pipe completion is sufficient (although not necessary) to guarantee that read events are complete (a "read fence" completion). Read events are still pending if work in the pipeline requires any type of read except a render target read (blend) to complete.

Write synchronization is a special case of end-of-pipe synchronization that requires that the render cache and/or depth related caches are flushed to memory, where the data will become globally visible. This type of synchronization is required prior to SW (CPU) actually reading the result data from memory, or initiating an operation that will use as a read surface (such as a texture surface) a previous render target and/or depth/stencil buffer. Exercising the write cache flush bits (Render Target Cache Flush Enable, Depth Cache Flush Enable, DC Flush) in PIPE_CONTROL only ensures the write caches are flushed and doesn't guarantee the data is globally visible.

SW can track the completion of the end-of-pipe-synchronization by using "Notify Enable" and "Post-Sync Operation - Write Immediate Data" in the PIPE_CONTROL command. "Notify Enable" and "Post-Sync Operation - Write Immediate Data" generate a fence cycle on achieving end-of-pipe-synchronization for the corresponding PIPE_CONTROL command. Fence cycle ensures all the write cycles in front of it are to global visible point before they themselves get processed. It is guaranteed the data flushed out by the PIPE_CONTROL is updated in memory by the time SW receives the corresponding Pipe Control Notify interrupt.

In case the data flushed out by the render engine is to be read back into the render engine in coherent manner, then the render engine has to wait for the fence completion before accessing the flushed data. This can be achieved by following means on various products.

PIPE_CONTROL command with CS Stall and the required write caches flushed with Post-Sync-Operation as Write Immediate Data.

Example:

- WorkLoad-1 (3D/GPGPU/MEDIA)
- PIPE_CONTROL (CS Stall, Post-Sync-Operation Write Immediate Data, Required Write Cache Flush bits set)

WorkLoad-2 (Can use the data produced or output by Workload-1)

Synchronization Actions

In order for the driver to act based on a synchronization point (usually the whole point), the reaching of the synchronization point must be communicated to the driver. This section describes the actions that may be taken upon completion of a synchronization point which can achieve this communication.

Writing a Value to Memory

The most common action to perform upon reaching a synchronization point is to write a value out to memory. An immediate value (included with the synchronization command) may be written. In lieu of an immediate value, the 64-bit value of the PS_DEPTH_COUNT (visible pixel count) or TIMESTAMP register may be written out to memory. The captured value will be the value at the moment all primitives parsed prior to the synchronization commands have been completely rendered, and optionally after all said primitives have been pushed to memory. It is not required that a value be written to memory by the synchronization command.

Visible pixel or TIMESTAMP information is only useful as a delta between 2 values, because these counters are free-running and are not to be reset except at initialization. To obtain the delta, two PIPE_CONTROL commands should be initiated with the command sequence to be measured between

them. The resulting pair of values in memory can then be subtracted to obtain a meaningful statistic about the command sequence.

PS_DEPTH_COUNT

If the selected operation is to write the visible pixel count (PS_DEPTH_COUNT register), the synchronization command should include the **Depth Stall Enable** parameter. There is more than one point at which the global visible pixel count can be affected by the pipeline; once the synchronization command reaches the first point at which the count can be affected, any primitives following it are stalled at that point in the pipeline. This prevents the subsequent primitives from affecting the visible pixel count until all primitives preceding the synchronization point reach the end of the pipeline, the visible pixel count is accurate, and the synchronization is completed. This stall has a minor effect on performance and should only be used in order to obtain accurate "visible pixel" counts for a sequence of primitives.

The PS_DEPTH_COUNT count can be used to implement an (API/DDI) "Occlusion Query" function.

Generating an Interrupt

The synchronization command may indicate that a "Sync Completion" interrupt is to be generated (if enabled by the MI Interrupt Control Registers - see *Memory Interface Registers*) once the rendering of all prior primitives is complete. Again, the completion of rendering can be considered to be when the internal render cache has been updated, or when the cache contents are visible in memory, as selected by the command options.

Invalidating of Caches

If software wishes to use the notification that a synchronization point has been reached in order to reuse referenced structures (surfaces, state, or instructions), it is not sufficient just to make sure rendering is complete. If additional primitives are initiated after new data is laid over the top of old in memory following a synchronization point, it is possible that stale cached data will be referenced for the subsequent rendering operation. In order to avoid this, the PIPE_CONTROL command must be used. (See PIPE_CONTROL Command description).

PIPE_CONTROL Command

The PIPE_CONTROL command provides mechanism to achieve the synchronization of the 3D pipeline and to execute post-synchronization operations as described in the section "Synchronization of the 3D pipeline". Parsing a PIPE_CONTROL command stalls the 3D pipe only if the stall enable bit is set. Commands after PIPE_CONTROL will continue to be parsed and processed in the 3D pipeline. This may include additional PIPE_CONTROL commands. The implementation does enforce a practical upper limit (8) on the number of PIPE_CONTROL commands that may be outstanding at once. Parsing a PIPE_CONTROL command that causes this limit to be reached will stall the parsing of new commands until the first of the outstanding PIPE_CONTROL commands reach the end of the pipe and retires.

Although PIPE_CONTROL is intended for use with the 3D pipe, it is legal to issue PIPE_CONTROL when the Media pipe is selected. In this case PIPE_CONTROL will stall at the top of the pipe until the Media FFs

finish processing commands parsed before PIPE_CONTROL. Post-synchronization operations, flushing of caches and interrupts will then occur if enabled via PIPE_CONTROL parameters. Due to this stalling behavior, only one PIPE_CONTROL command can be outstanding at a time on the Media pipe.

For the invalidate operation of the pipe control, the following pointers are affected. The invalidate operation affects the context restore of these packets. If the pipe control invalidate operation is completed before the context save, the indirect pointers will not be context restored from memory on a context switch.

- Pipeline State Pointer
- Media State Pointer
- Constant Buffer Packet

Programming Note
• SW must ensure to invalidate the Media State and Constant Buffers using "Generic Media State Clear" prior to the releasing the associated resources (memory). • SW must ensure to invalidate the Push Constants using "Indirect State Pointers Disable" prior to the releasing the associated resources (memory).

It is up to software to program the appropriate read-only cache invalidation such as the sampler and constant read caches or the instruction and state caches. Once notification is observed, new data may then be loaded (potentially "on top of" the old data) without fear of stale cache data being referenced for subsequent rendering.

If software wishes to access the rendered data in memory (for analysis by the application or to copy it to a new location to use as a texture, for example), it must also ensure that the write cache (render cache) is flushed after the synchronization point is reached so that memory will be updated. This can be done by setting the **Write Cache Flush Enable** bit. Note that the **Depth Stall Enable** bit must be clear in order for the flush of the render cache to occur. **Depth Stall Enable** is intended only for accurate reporting of the PS_DEPTH counter; the render cache cannot be flushed nor can the read caches be invalidated (except for the instruction/state cache) in conjunction with this operation.

Vertex caches are only invalidated when the VF invalidate bit is set in PIPE_CONTROL (i.e. decision is done in software, not hardware) Note that the index-based vertex cache is always flushed between primitive topologies and of course PIPE_CONTROL can only be issued between primitive topologies. Therefore only the VF ("address-based") cache is uniquely affected by PIPE_CONTROL.

PIPE_CONTROL

PIPE_CONTROL

Description
Hardware supports up to 32 pending PIPE_CONTROL flushes.

The table below explains all the different flush/invalidation scenarios.

Caches Invalidated/Flushed by PIPE_CONTROL Bit Settings

Write Cache Flush	Notification Enabled	Non-VF RO Cache Invalidate	VF RO Cache Invalidate	Marker Sent	Pipeline Marker Enable	Completion Requested	Top of Pipe Invalidate Pulse from CS
0	0	0	0	N/A	N/A	N/A	N/A
0	0	0	1	Yes	No	N/A	No
0	0	1	0	No	N/A	N/A	Yes
0	0	1	1	Yes	No	No	Yes
X	1	0	X	Yes	Yes	Yes	No
X	1	1	X	Yes	Yes	Yes	Yes
1	X	0	X	Yes	Yes	Yes	No
1	X	1	X	Yes	Yes	Yes	Yes

Programming Restrictions for PIPE_CONTROL

PIPE_CONTROL arguments can be split up into three categories:

- Post-sync operations
- Flush Types
- Stall

Post-sync operation is only indirectly affected by the flush type category via the stall bit. The stall category depends on both flush type and post-sync operation arguments. A PIPE_CONTROL with no arguments set is **Invalid**.

Post-Sync Operation

These arguments relate to events that occur after the marker initiated by the PIPE_CONTROL command is completed. The table below shows the restrictions:

Argument	Bits	Restriction
LRI Post Sync Operation	23	Post Sync Operation ([15:14] of DW1) must be set to 0x0.
Global Snapshot Count Reset	19	This bit must not be exercised on any product. Requires stall bit ([20] of DW1) set.
Generic Media State Clear	16	Requires stall bit ([20] of DW1) set.
Generic Media State Clear	16	Must not be set in PIPECONTROL command programmed for POCS.
Indirect State Pointers Disable	9	Requires stall bit ([20] of DW1) set.
Store Data Index	21	Post-Sync Operation ([15:14] of DW1) must be set to something other than '0'.
Sync GFDT	17	Post-Sync Operation ([15:14] of DW1) must be set to something other than '0' or

Argument	Bits	Restriction
		0x2520[13] must be set.
TLB inv	18	Requires stall bit ([20] of DW1) set.
Post Sync Op	15:14	LRI Post Sync Operation ([23] of DW1) must be set to '0'.
Post Sync Op	15:14	Post Sync Operations must not be set to "Write PS Depth Count" in PIPECONTROL command programmed for POCS.
Notify En	8	Must not be set in PIPECONTROL command programmed for POCS.

Flush Types

These are arguments related to the type of read only invalidation or write cache flushing is being requested. Note that there is only intra-dependency. That is, it is not affected by the post-sync operation or the stall bit. The table below shows the restrictions:

Arguments	Bit	Restrictions

Tile Cache Flush	28	**Tile Cache Enabled Mode:**
		• SW must always set CS Stall bit when Tile Cache Flush Enable bit is set in the PIPECONTROL command.
		• SW must ensure level1 depth and color caches are flushed prior to flushing the tile cache. This can be achieved by following means:
		  ○ Single PIPECONTROL command to flush level1 caches and the tile cache. Hardware will sequence the flushing of L1 caches followed by the Tile cache. Attributes listed below must be set. OR
		  ▪ Tile Cache Flush Enable
		  ▪ Render Target Cache Flush Enable
		  ▪ Depth Cache Flush Enable
		  ○ Flushing of L1 caches followed by flushing of tile cache through two different PIEPCONTROL commands. SW must ensure not to issue any rendering commands between the two PIPECONTROL commands.
		Unified Cache (Tile Cache Disabled):
		In unified cache mode of operation Color and Depth (Z) streams are cached in DC space of L2 along with Data Port stream.
		On a "Tile Cache Flush" only Color and Depth (Z) streams from DC space of L2 are flushed to globally observable and where as "DC Flush Enable" will only flush Data Port stream from the DC space of L2 to globally observable. Refer L3 configuration section for Unified cache usage model. In this mode of operation there is no dedicated memory allocated for Tile Cache in L2.
		When the Color and Depth (Z) streams are enabled to be cached in the DC space of L2, Software must use "Render Target Cache Flush Enable" and "Depth Cache Flush Enable" along with "Tile Cache Flush" for getting the color and depth (Z) write data to be globally observable. In this mode of operation it is not required to set "CS Stall" upon setting "Tile Cache Flush" bit.

Arguments	Bit	Restrictions
Depth Stall		Must not set in PIPECONTROL command programmed for POCS.
Render Target Cache Flush		Must not be set in PIPECONTROL command programmed for POCS.
Depth Cache Flush		Must not be set in PIPECONTROL command programmed for POCS.
Stall Pixel Scoreboard	1	No Restriction.
Stall Pixel Scoreboard		Must not be set in PIPECONTROL command programmed for POCS.
DC Flush Enable		Must not be set in PIPECONTROL command programmed for POCS.
Inst invalidate	11	No Restriction.
Tex invalidate	10	Requires stall bit ([20] of DW) set for all GPGPU Workloads.
State Invalidate	2	No Restriction.

Stall

If the stall bit is set, the command streamer waits until the pipe is completely flushed.

Arguments	Bit	Restrictions
Stall Bit	20	No Restrictions.

Engine State

This section describes the state specific to the Graphics Engine

Memory Access Indirection

The GPE supports the indirection of certain graphics (GTT-mapped) memory accesses. This support comes in the form of two *base address* state variables used in certain memory address computations with the GPE.

The intent of this functionality is to support the dynamic relocation of certain driver-generated memory structures after command buffers have been generated but prior to their submittal for execution. For example, as the driver builds the command stream it could append pipeline state descriptors, kernel binaries, etc. to a general state buffer. References to the individual items would be inserted in the command buffers as offsets from the base address of the state buffer. The state buffer could then be freely relocated prior to command buffer execution, with the driver only needing to specify the final base address of the state buffer. Two base addresses are provided to permit surface-related state (binding tables, surface state tables) to be maintained in a state buffer separate from the general state buffer.

While the use of these base addresses is unconditional, the indirection can be effectively disabled by setting the base addresses to zero. The following table lists the various GPE memory access paths and which base address (if any) is relevant.

Base Address Utilization

Base Address Used	Memory Accesses
General State Base Address	**DataPort Read/Write DataPort memory accesses resulting from 'stateless' DataPort Read/Write requests. See** *DataPort* **for a definition of the 'stateless' form of requests.**
Dynamic State Base Address	Sampler reads of SAMPLER_STATE data and associated SAMPLER_BORDER_COLOR_STATE
Dynamic State Base Address	**Viewport states** used by CLIP, SF, and WM/CC
Dynamic State Base Address	COLOR_CALC_STATE, DEPTH_STENCIL_STATE, and BLEND_STATE
Dynamic State Base Address	Push Constants (depending on state of **INSTPM<CONSTANT_BUFFER Address Offset Disable>**)
Instruction Base Address	**Normal EU instruction stream** (non-system routine)
Instruction Base Address	**System routine** EU instruction stream (starting address = SIP)
Surface State Base Address	Sampler and DataPort reads of BINDING_TABLE_STATE, as referenced by BT pointers passed via 3DSTATE_BINDING_TABLE_POINTERS
Surface State Base Address	Sampler and DataPort reads of SURFACE_STATE data
Indirect Object Base Address	**MEDIA_OBJECT Indirect Data** accessed by the CS unit
None	CS unit reads from **Ring Buffers**, **Batch Buffers**
None	CS writes resulting from PIPE_CONTROL command
None	All VF unit memory accesses (**Index Buffers**, **Vertex Buffers**)
None	All Sampler **Surface Memory Data** accesses (texture fetch, etc.)

Base Address Used	Memory Accesses
None	All **DataPort memory accesses** except 'stateless' DataPort Read/Write requests (e.g., RT accesses). See *DataPort* for a definition of the 'stateless' form of requests.
None	Memory reads resulting from **STATE_PREFETCH** commands
None	Any **physical memory access** by the device
None	GTT-mapped accesses not included above (i.e., default)
None	Push Constants (depeding on state of **INSTPM<CONSTANT_BUFFER Address Offset Disable>**)

The following notation is used in the BSpec to distinguish between addresses and offsets:

Notation	Definition
PhysicalAddress[n:m]	Corresponding bits of a physical graphics memory byte address (not mapped by a GTT)
GraphicsAddress[n:m]	Corresponding bits of an absolute, virtual graphics memory byte address (mapped by a GTT)
GeneralStateOffset[n:m]	Corresponding bits of a relative byte offset added to the General State Base Address value, the result of which is interpreted as a virtual graphics memory byte address (mapped by a GTT)
DynamicStateOffset[n:m]	Corresponding bits of a relative byte offset added to the Dynamic State Base Address value, the result of which is interpreted as a virtual graphics memory byte address (mapped by a GTT)
InstructionBaseOffset[n:m]	Corresponding bits of a relative byte offset added to the Instruction Base Address value, the result of which is interpreted as a virtual graphics memory byte address (mapped by a GTT)
SurfaceStateOffset[n:m]	Corresponding bits of a relative byte offset added to the Surface State Base Address value, the result of which is interpreted as a virtual graphics memory byte address (mapped by a GTT)

Context Image

Logical Contexts are memory images used to store copies of the device's rendering and ring context.

Logical Contexts are aligned to 256-byte boundaries.

Logical contexts are referenced by their memory address. The format and contents of rendering contexts are considered ***device-dependent*** and software must not access the memory contents directly. The definition of the logical rendering and power context memory formats is included here primarily for internal documentation purposes.

Power Context Image

POSH Power Context

The table below captures the data from CS power context save/restored by PM. Address offsets in this table are relative to the starting location of CS in the overall power context image managed by PM.

POCS Power Context Image

Description	Offset	Unit	# of DW	Address Offset (PWR)	CSFE/CSBE
NOOP		**POCS**	**1**	**0**	**CSFE**
Load_Register_Immediate header	**0x1100_0055**	POCS	1	001	CSFE
GFX_MODE	**0x1829C**	**POCS**	**2**	**0002**	**CSFE**
GHWSP	**0x18080**	**POCS**	**2**	**0004**	**CSFE**
RC_PSMI_CONTROL	**0x18050**	**POCS**	**2**	**0006**	**CSFE**
RC_PWRCTX_MAXCNT	**0x18054**	**POCS**	**2**	**0008**	**CSFE**
CTX_WA_PTR	**0x18058**	**POCS**	**2**	**000A**	**CSFE**
NOPID	**0x18094**	**POCS**	**2**	**000C**	**CSFE**
CMD_CCTL_0	**0x180C4**	**POCS**	**2**	**000E**	**CSFE**
PREEMPT_DLY	**0x18214**	**POCS**	**2**	**0010**	**CSFE**
CTXT_PREMP_DBG	**0x18248**	**POCS**	**2**	**0012**	**CSFE**
WAIT_FOR_RC6_EXIT	**0x180CC**	**POCS**	**2**	**0014**	**CSFE**
RCS_CTXID_PREEMPTION_HINT	**0x184CC**	**POCS**	**2**	**0016**	**CSFE**
CS_PREEMPTION_HINT_UDW	**0x184C8**	**POCS**	**2**	**0018**	**CSFE**
CS_PREEMPTION_HINT	**0x184BC**	**POCS**	**2**	**001A**	**CSFE**
MI_PREDICATE_RESULT_2	**0x183BC**	**POCS**	**2**	**001C**	**CSFE**
SEMA_WAIT_POLL	**0x1824C**	**POCS**	**2**	**001E**	**CSFE**
IDLEDELAY	**0x1823C**	**POCS**	**2**	**0020**	**CSFE**
RCS_FORCE_TO_NONPRIV_0_11	**0x184D0**	**POCS**	**24**	**0022**	**CSFE**
RCS_FORCE_TO_NONPRIV_12_15	**0x18010**	**POCS**	8	003A	**CSFE**
RCS_FORCE_TO_NONPRIV_16_19	**0x181E0**	**POCS**	8	0042	**CSFE**
EXECLIST_STATUS_REGISTER	**0x18234**	**POCS**	**2**	**004A**	**CSFE**
CXT_OFFSET	**0x181AC**	**POCS**	**2**	**004E**	**CSFE**
STOP_PARSER_CONTROL	**0x18424**	**POCS**	**2**	**0050**	**CSFE**
STOP_PARSER_HINT_ADDR	**0x18428**	**POCS**	**4**	**0052**	**CSFE**
NOOP		POCS	10	00**56**	CSFE
NOOP		**POCS**	**1**	**0060**	**CSBE**
Load_Register_Immediate header	**0x1100_0011**	**POCS**	**1**	**0061**	**CSBE**
FF_THREAD_MODE	**0x180A0**	**POCS**	**2**	**0066**	**CSBE**
VF PREMPTION VERTEX HINT	**0x16EB0**	**VFR**	**2**	**0070**	**CSBE**

Description	Offset	Unit	# of DW	Address Offset (PWR)	CSFE/CSBE
VF PREEMPTION INSTANCE HINT	0x16EB4	VFR	2	0072	CSBE
NOOP		POCS	10	0074	CSBE
NOOP		POCS	1	007E	CSBE
MI_BATCH_BUFFER_END		POCS	1	007F	CSBE

Render Engine Power Context

RCS Power Context Image

Description	Offset	Unit	# of DW	Address Offset (PWR)	CSFE/CSBE
CSFE Power Context with Display			208	0	CSFE
NOOP		CS	1	00D0	CSBE
Load_Register_Immediate header	0x1100_0019	CS	1	00D1	CSBE
GAFS_Mode	0x212C	CS	2	00DC	CSBE
RS_PREEMPTION_HINT_UDW	0x24C4	CS	2	00E2	CSBE
RS_PREEMPTION_HINT	0x24C0	CS	2	00E4	CSBE
FF_STALL_CHK_VFUNIT	0x83B0	VF	2	00E8	CSBE
VFGPCB_VFUNIT	0x83B4	VF	2	00EA	CSBE
NOOP		CS	3	00EC	CSBE
MI_BATCH_BUFFER_END		CS	1	00EF	CSBE

Compute Engine Power Context

Compute Engine Power Context Image

Description	Offset	Unit	# of DW	Address Offset (PWR)	CSFE/CSBE
CSFE Power context without Display		CSFE	195	0	CSFE
NOOP		CS	1	00C4	CSBE
Load_Register_Immediate header	0x1100_0009	CS	1	00C5	CSBE
NOOP		CS	10	00D0	CSBE
NOOP		CS	1	00DA	CSBE
MI_BATCH_BUFFER_END		CS	1	00DB	CSBE

The table below captures the data from TDL power context save/restored by PM.

TDL Power Context Image

Description	Offset	Unit	# of DW
NOOP		TDL	1
MI_LOAD_REGISTER_IMM	0x1100_004D	TDL	1
TD_CTL	E400	TDL	2
TD_CTL2	E404	TDL	2
TD_VF_VS_EMSK	E408	TDL	2
TD_GS_EMSK	E40C	TDL	2
TD_WIZ_EMSK	E410	TDL	2
TD_TS_EMSK	E428	TDL	2
TD_HS_EMSK	E4B0	TDL	2
TD_DS_EMSK	E4B4	TDL	2
EU_PERF_CNT_CTL0	E458	TDL	2
EU_PERF_CNT_CTL1	E558	TDL	2
EU_PERF_CNT_CTL2	E658	TDL	2
EU_PERF_CNT_CTL3	E758	TDL	2
EU_PERF_CNT_CTL4	E45C	TDL	2
EU_PERF_CNT_CTL5	E55C	TDL	2
EU_PERF_CNT_CTL6	E65C	TDL	2
CULLBIT3	E488	TDL	2
CACHE_MODE_SS	E420	TDL	2
VSR_EMASK	E51C	TDL	2
SLM_BANK_HASH	E660	TDL	2
NOOP		TDL	1
MI_BATCH_BUFFER_END		TDL	1

Sampler Power Context

The table below captures the data from TDL power context save/restored by PM.

Sampler Power Context Image

Description	Offset	Unit	# of DW
NOOP		SC	1
MI_LOAD_REGISTER_IMM	0x1100_100D	SC	1
SAMPLER_MODE	0xE18C	SC	2

Engine Register and State Context

This section describes programming requirements for the Register State Context.

EXECLIST CONTEXT(Ring)
EXECLIST CONTEXT (PPGTT Base)
ENGINE CONTEXT(CSFE)
ENGINE CONTEXT(CSBE)
ENGINE CONTEXT(SOL)
ENGINE CONTEXT(VF)
ENGINE CONTEXT(GAMWC)
ENGINE CONTEXT(GAMT)
ENGINE CONTEXT(LNCF)
ENGINE CONTEXT(SVG)
ENGINE CONTEXT
ENGINE CONTEXT(TDL)
ENGINE CONTEXT(WM)
ENGINE CONTEXT(SC)
ENGINE CONTEXT(DM)
ENGINE CONTEXT(VFE)
ENGINE CONTEXT(CS - Footer)

EXECLIST CONTEXT(Ring)
EXECLIST CONTEXT (PPGTT Base)
ENGINE CONTEXT(CSFE)
ENGINE CONTEXT(CSBE)
ENGINE CONTEXT(SOL)
ENGINE CONTEXT(VF)
ENGINE CONTEXT(OAR)
ENGINE CONTEXT(LBCF)
ENGINE CONTEXT(SVG)
ENGINE CONTEXT(SVL)
ENGINE CONTEXT(WM)
ENGINE CONTEXT(SC)
ENGINE CONTEXT(VFE)
ENGINE CONTEXT(CS - Footer)

ComputeCS Context Image

EXECLIST CONTEXT(Ring)
EXECLIST CONTEXT (PPGTT Base)
ENGINE CONTEXT(CSFE)
ENGINE CONTEXT(CSBE)
ENGINE CONTEXT(VFE)
ENGINE CONTEXT(CS - Footer)

POSH Context Image

EXECLIST CONTEXT(Ring)
EXECLIST CONTEXT (PPGTT Base)
ENGINE CONTEXT(CSFE)
ENGINE CONTEXT(CSBE)
ENGINE CONTEXT(VFR)
ENGINE CONTEXT(OVR)
ENGINE CONTEXT(SVGR)
ENGINE CONTEXT(CS - Footer)

Register State Context

Color Coding

EXECLIST CONTEXT
EXECLIST CONTEXT(PPGTT Base)
ENGINE CONTEXT
EXTENDED ENGINE CONTEXT

Description	MMIO Offset/Command	Unit	# of DW
NOOP		CSEL	1
MI_LOAD_REGISTER_IMM	0x1108_1019	CSEL	1
Ring Buffer Head	0x2034	CSEL	2
Ring Tail Pointer Register	0x2030	CSEL	2
RING_BUFFER_START	0x2038	CSEL	2
RING_BUFFER_CONTROL	0x203C	CSEL	2
Batch Buffer Current Head Register (UDW)	0x2168	CSEL	2
Batch Buffer Current Head Register	0x2140	CSEL	2
Batch Buffer State Register	0x2110	CSEL	2
BB_PER_CTX_PTR	0x21C0	CSEL	2

Description	MMIO Offset/Command	Unit	# of DW
RCS_INDIRECT_CTX	0x21C4	CSEL	2
RCS_INDIRECT_CTX_OFFSET	0x21C8	CSEL	2
CCID	0x2180	CSEL	2
SEMAPHORE_TOKEN	0x22B4	CSEL	2
NOOP		CSEL	4
NOOP		CSEL	1
MI_LOAD_REGISTER_IMM	0x1100_1011	CSEL	1
CTX_TIMESTAMP	0x23A8	CSEL	2
PDP3_UDW	0x228C	CSEL	2
PDP3_LDW	0x2288	CSEL	2
PDP2_UDW	0x2284	CSEL	2
PDP2_LDW	0x2280	CSEL	2
PDP1_UDW	0x227C	CSEL	2
PDP1_LDW	0x2278	CSEL	2
PDP0_UDW	0x2274	CSEL	2
PDP0_LDW	0x2270	CSEL	2
MI_LOAD_REGISTER_IMM	0x1108_1005	CSEL	1
POSH_LRCA	0x21B0	CSEL	2
CONTEXT_SCHEDULING_ATTRIBUTES (RESOURCE_MIN_MAX_PRIORITY)	0x25A8	CSEL	2
PREEMPTION_STATUS (RO)	0x25AC	CSEL	2
NOOP		CSEL	5
NOOP		CSEL_BE	1
MI_LOAD_REGISTER_IMM	0x1108_0001	CSEL_BE	1
R_PWR_CLK_STATE	0x20C8	CSEL_BE	2
GPGPU_CSR_BASE_ADDRESS		CSEL_BE	3
NOOP		CSEL_BE	9
NOOP		CSFE	1
MI_LOAD_REGISTER_IMM	0x1108_1065	CSFE	1
BB_STACK_WRITE_PORT	0x2588	CSFE	12
EXCC	0x2028	CSFE	2
MI_MODE	0x209C	CSFE	2
INSTPM	0x20C0	CSFE	2
PR_CTR_CTL	0x2178	CSFE	2
PR_CTR_THRSH	0x217C	CSFE	2
TIMESTAMP Register (LSB)	0x2358	CSFE	2

Description	MMIO Offset/Command	Unit	# of DW
BB_START_ADDR_UDW	0x2170	CSFE	2
BB_START_ADDR	0x2150	CSFE	2
BB_ADD_DIFF	0x2154	CSFE	2
BB_OFFSET	0x2158	CSFE	2
MI_PREDICATE_RESULT_1	0x241C	CSFE	2
CS_GPR (1-16)	0x2600	CSFE	64
IPEHR	0x2068	CSFE	2
CMD_BUF_CCTL	0x2084	CSFE	2
NOOP		CSFE	1
MI_LOAD_REGISTER_IMM	0x1102_100F	CSFE	1
TRTT_CR	0x4400	CSFE	2
TRTT_VA_RANGE	0x4404	CSFE	2
TRTT_L3_BASE_LOW	0x4408	CSFE	2
TRTT_L3_BASE_HIGH	0x440C	CSFE	2
TR_NULL_GFX	0x4410	CSFE	2
TRTT_INVAL	0x4414	CSFE	2
AUX_TABLE_BASE_ADDR_LOW	0x4200	CSFE	2
AUX_TABLE_BASE_ADDR_HIGH	0x4204	CSFE	2
NOOP		CSFE	6
NOOP		CSBE	1
MI_LOAD_REGISTER_IMM	0x1108_10AD		
CS_CONTEXT_STATUS1	0x2184	CSBE	2
IA_VERTICES_COUNT	0x2310	CSBE	4
IA_PRIMITIVES_COUNT	0x2318	CSBE	4
VS_INVOCATION_COUNT	0x2320	CSBE	4
HS_INVOCATION_COUNT	0x2300	CSBE	4
DS_INVOCATION_COUNT	0x2308	CSBE	4
GS_INVOCATION_COUNT	0x2328	CSBE	4
GS_PRIMITIVES_COUNT	0x2330	CSBE	4
CL_INVOCATION_COUNT	0x2338	CSBE	4
CL_PRIMITIVES_COUNT	0x2340	CSBE	4
PS_INVOCATION_COUNT_0	0x22C8	CSBE	4
PS_DEPTH_COUNT _0	0x22D8	CSBE	4
GPUGPU_DISPATCHDIMX	0x2500	CSBE	2
GPUGPU_DISPATCHDIMY	0x2504	CSBE	2
GPUGPU_DISPATCHDIMZ	0x2508	CSBE	2

Description	MMIO Offset/Command	Unit	# of DW
MI_PREDICATE_SRC0	0x2400	CSBE	2
MI_PREDICATE_SRC0	0x2404	CSBE	2
MI_PREDICATE_SRC1	0x2408	CSBE	2
MI_PREDICATE_SRC1	0x240C	CSBE	2
MI_PREDICATE_DATA	0x2410	CSBE	2
MI_PREDICATE_DATA	0x2414	CSBE	2
MI_PRED_RESULT	0x2418	CSBE	2
3DPRIM_END_OFFSET	0x2420	CSBE	2
3DPRIM_START_VERTEX	0x2430	CSBE	2
3DPRIM_VERTEX_COUNT	0x2434	CSBE	2
3DPRIM_INSTANCE_COUNT	0x2438	CSBE	2
3DPRIM_START_INSTANCE	0x243C	CSBE	2
3DPRIM_BASE_VERTEX	0x2440	CSBE	2
Load Indirect Extended Parameter 0	0x2690	CSBE	2
Load Indirect Extended Parameter 1	0x2694	CSBE	2
Load Indirect Extended Parameter 2	0x2698	CSBE	2
GPGPU_THREADS_DISPATCHED	0x2290	CSBE	4
PS_INVOCATION_COUNT_1	0x22F0	CSBE	4
PS_DEPTH_COUNT _1	0x22F8	CSBE	4
RS_PREEMPT_STATUS	0x215C	CSBE	2
PRODUCE_COUNT_BTP	0x2480	CSBE	2
PRODUCE_COUNT_DX9_CONSTANTS	0x2484	CSBE	2
PARSED_COUNT_BTP	0x2490	CSBE	2
PARSED_COUNT_DX9_CONSTANTS	0x2494	CSBE	2
CS_CHICKEN2	0x2194	CSBE	4
OA_CTX_CONTROL	0x2360	CSBE	2
OACTXID	0x2364	CSBE	2
PS_INVOCATION_COUNT_2	0x2448	CSBE	4
PS_DEPTH_COUNT_2	0x2450	CSBE	4
RS_PREEMPT_STATUS_UDW	0x2174	CSBE	2
CPS_INVOCATION_COUNT	0x2478	CSBE	4
PS_INVOCATION_COUNT_3	0x2458	CSBE	4
PS_DEPTH_COUNT_3	0x2460	CSBE	4
PS_INVOCATION_COUNT_4	0x2468	CSBE	4
PS_DEPTH_COUNT_4	0x2470	CSBE	4
PS_INVOCATION_COUNT_5	0x24A0	CSBE	4

Description	MMIO Offset/Command	Unit	# of DW
PS_DEPTH_COUNT_5	0x24A8	CSBE	4
PS_INVOCATION_COUNT_6	0x25D0	CSBE	4
PS_DEPTH_COUNT_6	0x25B0	CSBE	4
PS_INVOCATION_COUNT_7	0x25D8	CSBE	4
PS_DEPTH_COUNT_7	0x25B8	CSBE	4
NOOP		CSBE	1
MI_LOAD_REGISTER_IMM	0x1100_0001	CSBE	
VSR_PUSHCONSTANT_BASE	0xE518	CSBE	2
NOOP		CSBE	2
MI_TOPOLOGY_FILTER		CSBE	1
NOOP		CSBE	2
PIPELINE_SELECT		CSBE	1
STATE_BASE_ADDRESS		CSBE	22
STATE_SIP		CSBE	3
3DSTATE_PUSH_CONSTANT_ALLOC_VS		CSBE	2
3DSTATE_PUSH_CONSTANT_ALLOC_HS		CSBE	2
3DSTATE_PUSH_CONSTANT_ALLOC_DS		CSBE	2
3DSTATE_PUSH_CONSTANT_ALLOC_GS		CSBE	2
3DSTATE_PUSH_CONSTANT_ALLOC_PS		CSBE	2
3DSTATE_BINDING_TABLE_POOL_ALLOC		CSBE	4
3DSTATE_GATHER_POOL_ALLOC		CSBE	4
3DSTATE_DX9_CONSTANT_BUFFER_POOL_ALLOC		CSBE	4
MI_RS_CONTROL		CSBE	1
3DSTATE_PTBR_TILE_PASS_INFO		CSBE	4
STATE_COMPUTE_MODE		CSBE	2
NOOP		CSBE	16
NOOP		SOL	1
MI_LOAD_REGISTER_IMM	0x1100_1027	SOL	1
SO_NUM_PRIMS_WRITTEN0	0x5200	SOL	4
SO_NUM_PRIMS_WRITTEN1	0x5208	SOL	4
SO_NUM_PRIMS_WRITTEN2	0x5210	SOL	4
SO_NUM_PRIMS_WRITTEN3	0x5218	SOL	4
SO_PRIM_STORAGE_NEEDED0	0x5240	SOL	4
SO_PRIM_STORAGE_NEEDED1	0x5248	SOL	4
SO_PRIM_STORAGE_NEEDED2	0x5250	SOL	4
SO_PRIM_STORAGE_NEEDED3	0x5258	SOL	4

Description	MMIO Offset/Command	Unit	# of DW
SO_WRITE_OFFSET0	0x5280	SOL	2
SO_WRITE_OFFSET1	0x5284	SOL	2
SO_WRITE_OFFSET2	0x5288	SOL	2
SO_WRITE_OFFSET3	0x528C	SOL	2
3DSTATE_SO_BUFFER		SOL	32
NOOP		SOL	3
3DSTATE_SO_DECL_LIST		SOL	259
NOOP		SOL	0
3DSTATE_INDEX_BUFFER		VF	5
3DSTATE_VERTEX_BUFFERS		VF	133
3DSTATE_VERTEX_ELEMENTS		VF	69
3DSTATE_VF_STATISTICS		VF	1
3DSTATE_VF		VF	2
3DSTATE_SGVS		VF	2
3DSTATE_VF_INSTANCING		VF	69
3DSTATE_VF_TOPOLOGY		VF	2
NOOP		VF	5
MI_LOAD_REGISTER_IMM	0x1100_10C7	VF	1
INSTANCE CNT	08300 - 08384h	VF	68
INSTANCE INDX	08400 - 08484h	VF	68
COMMITTED VERTEX NUMBER	08390h	VF	2
COMMITTED INSTANCE ID	08394h	VF	2
COMMITTED PRIMITIVE ID	08398h	VF	2
STATUS	0839Ch	VF	2
COMMON VERTEX	083A0h	VF	2
VF_GUARDBAND	083A4h	VF	2
INDEX_OPCODE_DATA00	08490h	VF	2
INDEX_OPCODE_DATA01	08494h	VF	2
INDEX_OPCODE_DATA10	08498h	VF	2
INDEX_OPCODE_DATA11	0849Ch	VF	2
TOKPROC_CULL_COUNT0	084A0h	VF	2
TOKPROC_CULL_COUNT1	084A4h	VF	2
TOKPROC_PID_COUNT0	084A8h	VF	2
TOKPROC_PID_COUNT1	084ACh	VF	2
TOKPROC_CULL_VERTEX	084B0h	VF	2
TOKPROC_PID_OBJECT	084B4h	VF	2

Description	MMIO Offset/Command	Unit	# of DW
TOKPROC_DUMMY_OBJECT	084B8h	VF	2
TOKPROC_CL_PTR	084BCh	VF	2
TOKPROC_CL_MISC	084C0h	VF	2
TOKPROC_STG1_DATA	084C4h	VF	2
TOKPROC_STG1_VERTEX_COUNT	084C8h	VF	2
TOKPROC_STG1_OBJECT_COUNT	084CCh	VF	2
TOKPROC_STG1_VALID	084D0h	VF	2
TOKPROC_STG0_INSTANCE_COUNT	084D4h	VF	2
TOKPROC_STG0_VERTEX_COUNT	084D8h	VF	2
TOKPROC_STG0_COUNT	084DCh	VF	2
TOKPROC_STG0_VALID	084E0h	VF	2
TOKIN_DATA0	084F0h	VF	2
TOKIN_DATA1	084F4h	VF	2
TOKIN_DATA2	084F8h	VF	2
TOKIN_DATA3	084FCh	VF	2
NOOP		VF	7
3DSTATE_VF_COMPONENT_PACKING		VF	5
3DSTATE_VF_SGVS_2		VF	3
3DSTATE_PTBR_TILE_SELECT		VF	2
NOOP		VF	2
NOOP		VF	4
OAR Cotext State		OAR	192
NOOP		LBCF	1
MI_LOAD_REGISTER_IMM	0x1100_100D	LBCF	1
LSQCREG1	0xB100	LBCF	2
LSQCREG4	0xB118	LBCF	2
LSQCREG5	0xB158	LBCF	2
LSQCREG6	0xB15C	LBCF	2
L3ALLOCREG	0xB134	LBCF	2
L3TCCNTLREG	0xB138	LBCF	2
L3RCS0RSVD1	0xB168	LBCF	2
3DSTATE_CONSTANT_VS_Commited		SVG	11
3DSTATE_CONSTANT_HS_Commited		SVG	11
3DSTATE_CONSTANT_DS_Commited		SVG	11
3DSTATE_CONSTANT_GS_Commited		SVG	11
3DSTATE_VS		SVG	9

Description	MMIO Offset/Command	Unit	# of DW
3DSTATE_BINDING_TABLE_POINTERS_VS		SVG	2
3DSTATE_SAMPLER_STATE_POINTERS_VS		SVG	2
3DSTATE_URB_ALLOC_VS		SVG	3
3DSTATE_STREAMOUT		SVG	5
3DSTATE_SO_BUFFER_INDEX_0		SVG	8
3DSTATE_SO_BUFFER_INDEX_1		SVG	8
3DSTATE_SO_BUFFER_INDEX_2		SVG	8
3DSTATE_SO_BUFFER_INDEX_3		SVG	8
3DSTATE_CLIP		SVG	4
3DSTATE_PRIMITIVE_REPLICATION		SVG	6
3DSTATE_SF		SVG	4
3DSTATE_SCISSOR_STATE_POINTERS		SVG	2
3DSTATE_VIEWPORT_STATE_POINTERS_CL_SF		SVG	2
3DSTATE_RASTER		SVG	5
3DSTATE_WM_HZ_OP		SVG	5
3DSTATE_MULTISAMPLE		SVG	2
3DSTATE_HS		SVG	9
3DSTATE_BINDING_TABLE_POINTERS_HS		SVG	2
3DSTATE_SAMPLER_STATE_POINTERS_HS		SVG	2
3DSTATE_URB_ALLOC_HS		SVG	3
3DSTATE_TE		SVG	5
3DSTATE_DS		SVG	11
3DSTATE_BINDING_TABLE_POINTERS_DS		SVG	2
3DSTATE_SAMPLER_STATE_POINTERS_DS		SVG	2
3DSTATE_URB_ALLOC_DS		SVG	3
3DSTATE_GS		SVG	10
3DSTATE_BINDING_TABLE_POINTERS_GS		SVG	2
3DSTATE_SAMPLER_STATE_POINTERS_GS		SVG	2
3DSTATE_URB_ALLOC_GS		SVG	3
NOOP		SVG	1
3DSTATE_CONSTANT_VS_NonComitted		SVG	11
3DSTATE_CONSTANT_HS_NonComitted		SVG	11
3DSTATE_CONSTANT_DS_NonComitted		SVG	11
3DSTATE_CONSTANT_GS_NonComitted		SVG	11
3DSTATE_DRAW_RECTANGULAR		SVG	4
MI_LOAD_REGISTER_IMM	0x1100_1007	SVG	1

Description	MMIO Offset/Command	Unit	# of DW
FF_PERF_REG	0x6b1c	SVG	2
DEREF_CTRL	0x6000	SVG	2
FF_MODE2	0x6604	SVG	2
CULLBIT1	0x6100	SVG	2
NOOP		SVG	8
NOOP		SVG	1
NOOP		SVG	3
NOOP		SVG	2
NOOP		SVG	1
3DSTATE_CONSTANT_PS_comitted		SVL	11
NOOP		SVL	1
3DSTATE_WM		SVL	2
3DSTATE_VIEWPORT_STATE_POINTER_CC		SVL	2
3DSTATE_CC_STATE_POINTERS		SVL	2
3DSATE_WM_SAMPLEMASK		SVL	2
3DSTATE_WM_DEPTH_STENCIL		SVL	4
3DSTATE_WM_CHROMAKEY		SVL	2
3DSTATE_DEPTH_BUFFER		SVL	8
3DSTATE_HIER_DEPTH_BUFFER		SVL	5
3DSTATE_STENCIL_BUFFER		SVL	8
3DSTATE_CLEAR_PARAMS		SVL	3
3DSTATE_CPS_POINTERS		SVL	2
3DSTATE_DEPTH_CNTL_BUFFER		SVL	5
3DSTATE_DEPTH_BOUNDS		SVL	4
3DSTATE_AMFS_TEXTURE_POINTERS		SVL	11
3DSTATE_SBE		SVL	6
3DSTATE_SBE_SWIZ		SVL	11
3DSTATE_PS		SVL	12
3DSTATE_BINDING_TABLE_POINTERS_PS		SVL	2
STATE_SAMPLER_STATE_POINTERS_PS		SVL	2
3DSTATE_BLEND_STATE_POINTERS		SVL	2
3DSTATE_PS_EXTRA		SVL	2
3DSTATE_PS_BLEND		SVL	2
3DSTATE_AMFS		SVL	1
3DSTATE_CONSTANT_TS_POINTERS		SVL	3
NOOP		SVL	1

Description	MMIO Offset/Command	Unit	# of DW
3DSTATE_CONSTANT_PS_NonComitted		SVL	11
3DSTATE_3D_MODE		SVL	2
3DSTATE_SAMPLE_PATTERN		SVL	9
3DSTATE_SUBSLICE_HASH_TABLE		SVL	14
NOOP		SVL	1
NOOP		SVL	12
MI_LOAD_REGISTER_IMM	0x1100_101F	SVL	1
Cache_Mode_0	0x7000	SVL	2
Cache_Mode_1	0x7004	SVL	2
GT_MODE	0x7008	SVL	2
FBC_RT_BASE_ADDR_REGISTER	0x7020	SVL	2
FBC_RT_BASE_ADDR_REGISTER_UPPER	0x7024	SVL	2
OA_CULL	0x7030	SVL	2
PSS_MODE	0x7038	SVL	2
PSS_MODE2	0x703C	SVL	2
PSS_CHICKEN	0x7044	SVL	2
NOOP		SVL	8
NOOP		SVL	14
NOOP		SVL	2
NOOP		WM	1
MI_LOAD_REGISTER_IMM	0x1100_1007	WM	1
WMHWCLRVAL	0x5524	WM	2
3DSTATE_POLY_STIPPLE_PATTERN		WM	33
3DSTATE_AA_LINE_PARAMS		WM	3
3DSTATE_POLY_STIPPLE_OFFSET		WM	2
3DSTATE_LINE_STIPPLE		WM	3
3DSTATE_SLICE_HASH_STATE_POINTERS		WM	2
NOOP		WM	11
3DSTATE_MONOFILTER_SIZE		SC	2
3DSTATE_CHROMA_KEY		SC	16
NOOP		SC	14
MI_BATCH_BUFFER_END		CSEND	1
NOOP		CSEND	127

Software Interface

This chapter describes the memory-mapped registers associated with the Memory Interface, including brief descriptions of their use. Refer to each register's description and related feature for more information on each individual bit.

The registers detailed in this chapter are extensions to previous projects. However, slight changes may be present in some registers (i.e., for features added or removed), or some registers may be removed entirely. These changes are clearly marked within this chapter.

Commands

This section describes the commands specific to Graphics engine

State Commands

This section covers the following commands:

- **STATE_SIP command -** Subroutine Kernel based on interrupt in kernel

Command
STATE_SIP

STATE_BASE_ADDRESS

The STATE_BASE_ADDRESS command sets the base pointers for subsequent state, instruction, and media indirect object accesses by the GPE. (See Memory Access Indirection for details.)

The following commands must be reissued following any change to the base addresses:

- 3DSTATE_PIPELINE_POINTERS
- 3DSTATE_BINDING_TABLE_POINTERS
- MEDIA_STATE_POINTERS

Execution of this command causes a full pipeline flush, thus its use should be minimized for higher performance.

Command
STATE_BASE_ADDRESS
PIPELINE_SELECT

The **Pipeline Select** state is contained within the logical context.

Memory Interface Commands for Rendering Engine

Command
MI_SET_CONTEXT
MI_TOPOLOGY_FILTER

Registers

Context Save Registers

VF Instance Count Registers

VF Instance Count Register Set		
Register Type:	MMIO_VF	
Address:	08300h - 08384h	
Default Value:	0000 0000h	
Access:	RO	
Size:	1088 bits	
Description:	Set of Registers for storing the index count values. In case of preempted drawcalls, these register store index count/number per element. For the non-preempted drawcalls, the values stored are ignored upon restore. These are saved as part of render context.	
DWord	**Bits**	**Description**
0	31:0	**Index Count 0.** Index Count value for Element 0. Format: U32
1	31:0	**Index Count 1.** Index Count value for Element 1. Format: U32
...	31:0	...
33	31:0	**Index Count 33.** Index Count value for Element 33. Format: U32

Mode and Misc Ctrl Registers

This section contains various registers for controls and modes

Controls/Modes
MI_MODE - Mode Register for Software Interface
FF_MODE - Thread Mode Register
GFX_MODE - Graphics Mode Register
GT_MODE - GT Mode Register
SAMPLER_MODE - SAMPLER Mode Register
CACHE_MODE_1 - Cache Mode Register 1
GAFS_MODE - Mode Register for GAFS
FBC_RT_BASE_ADDR_REGISTER - FBC_RT_BASE_ADDR_REGISTER
FBC_RT_BASE_ADDR_REGISTER_UPPER - FBC_RT_BASE_ADDR_REGISTER_UPPER
CACHE_MODE_SS - Cache Mode Subslice Register

L3CNTLREG - L3 Control Register

Pipelines Statistics Counter Registers

These registers keep continuous count of statistics regarding the Graphics pipeline. They are saved and restored with context but should not be changed by software except to reset them to 0 at context creation time. Write access to the statistics counter in this section must be done through MI_LOAD_REGISTER_IMM, MI_LOAD_REGISTER_MEM, or MI_LOAD_REGISTER_REG commands in ring buffer or batch buffer. These registers may be read at any time; however, to obtain a meaningful result, a pipeline flush just prior to reading the registers is necessary to synchronize the counts with the primitive stream.

Registers
IA_VERTICES_COUNT - IA Vertices Count
IA_PRIMITIVES_COUNT - Primitives Generated By VF
VS_INVOCATION_COUNT - VS Invocation Counter
HS_INVOCATION_COUNT - HS Invocation Counter
DS_INVOCATION_COUNT - DS Invocation Counter
GS_INVOCATION_COUNT - GS Invocation Counter
GS_PRIMITIVES_COUNT - GS Primitives Counter
CL_INVOCATION_COUNT - Clipper Invocation Counter
PS_INVOCATION_COUNT - PS Invocation Count
PS_INVOCATION_COUNT_SLICE0 - PS Invocation Count for Slice0
PS_INVOCATION_COUNT_SLICE1 - PS Invocation Count for Slice1
PS_INVOCATION_COUNT_SLICE2 - PS Invocation Count for Slice2
PS_INVOCATION_COUNT_SLICE4 - PS Invocation Count for Slice4
PS_INVOCATION_COUNT_SLICE5 - PS Invocation Count for Slice5
CPS_INVOCATION_COUNT - CPS Invocation Counter
PS_DEPTH_COUNT
PS_DEPTH_COUNT_SLICE0 - PS Depth Count for Slice0
PS_DEPTH_COUNT_SLICE1 - PS Depth Count for Slice1
PS_DEPTH_COUNT_SLICE2 - PS Depth Count for Slice2
PS_DEPTH_COUNT_SLICE3 - PS Depth Count for Slice3
PS_DEPTH_COUNT_SLICE4 - PS Depth Count for Slice4
PS_DEPTH_COUNT_SLICE5 - PS Depth Count for Slice5
TIMESTAMP - Reported Timestamp Count
Stream Output 0 Write Offset
Stream Output 1 Write Offset
Stream Output 2 Write Offset
Stream Output 3 Write Offset
Window Hardware Generated Clear Value

CS_CTX_TIMESTAMP- CS Context Timestamp Count:

This register provides a mechanism to obtain cumulative run time of a GPU context on HW.

Register
CS_CTX_TIMESTAMP - CS Context Timestamp Count

Diagram below details on when CS_CTX_TIMESTAMP run time, save/restored during a GPGPU context switch flow.

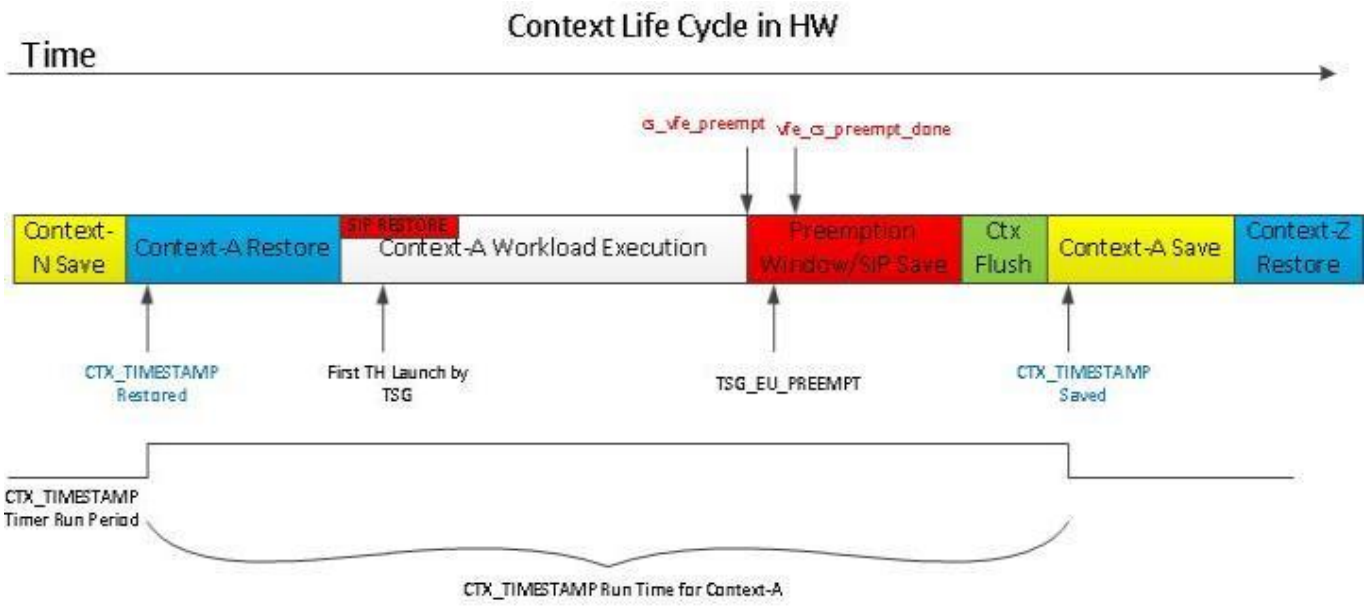


Fig: CTX_TIMESTAMP fucntionality during context execution

AUTO_DRAW Registers

AUTO_DRAW Registers
3DPRIM_END_OFFSET - Auto Draw End Offset
3DPRIM_START_VERTEX - Load Indirect Start Vertex
3DPRIM_VERTEX_COUNT - Load Indirect Vertex Count
3DPRIM_INSTANCE_COUNT - Load Indirect Instance Count
3DPRIM_START_INSTANCE - Load Indirect Start Instance
3DPRIM_BASE_VERTEX - Load Indirect Base Vertex
3DPRIM_XP0 - Load Indirect Extended Parameter 0
3DPRIM_XP1 - Load Indirect Extended Parameter 1
3DPRIM_XP2 - Load Indirect Extended Parameter 2

MMIO Registers for GPGPU Indirect Dispatch

These registers are normally written with the MI_LOAD_REGISTER_MEMORY command rather than from the CPU.

GPGPU_DISPATCHDIMX - GPGPU Dispatch Dimension X

GPGPU_DISPATCHDIMY - GPGPU Dispatch Dimension Y

GPGPU_DISPATCHDIMZ - GPGPU Dispatch Dimension Z

TS_GPGPU_THREADS_DISPATCHED - Count Active Channels Dispatched

Command Ordering Rules

There are several restrictions regarding the ordering of commands issued to the GPE. This subsection describes these restrictions along with some explanation of why they exist. Refer to the various command descriptions for additional information.

PIPELINE_SELECT

The previously active pipeline needs to be flushed immediately before switching to a different pipeline via use of the PIPELINE_SELECT command.

Refer to for details on the PIPELINE_SELECT command.

PIPELINE_SELECT

PIPE_CONTROL

The PIPE_CONTROL command does not require URB fencing/allocation to have been performed, nor does it rely on any other pipeline state. It is intended to be used on both the 3D pipe and the Media pipe. It has special optimizations to support the pipelining capability in the 3D pipe which do not apply to the Media pipe.

Common Pipeline State-Setting Commands

The following commands are used to set state common to the Graphics pipelines.

- STATE_BASE_ADDRESS
- STATE_SIP
- 3DSTATE_CHROMA_KEY
- 3DSTATE_BINDING_TABLE_POOL_ALLOC

The state variables associated with these commands must be set appropriately prior to initiating activity within a pipeline.

3D Pipeline-Specific State-Setting Commands

3D Pipeline-Specific State-Setting Commands

The following commands are used to set state specific to the 3D Pipeline.

- 3DSTATE_PIPELINED_POINTERS
- 3DSTATE_BINDING_TABLE_POINTERS
- 3DSTATE_VERTEX_BUFFERS
- 3DSTATE_VERTEX_ELEMENTS
- 3DSTATE_INDEX_BUFFERS
- 3DSTATE_VF_STATISTICS
- 3DSTATE_DRAWING_RECTANGLE
- 3DSTATE_CONSTANT_COLOR
- 3DSTATE_DEPTH_BUFFER
- 3DSTATE_POLY_STIPPLE_OFFSET
- 3DSTATE_POLY_STIPPLE_PATTERN
- 3DSTATE_LINE_STIPPLE
- 3DSTATE_GLOBAL_DEPTH_OFFSET

The state variables associated with these commands must be set appropriately prior to issuing 3DPRIMITIVE.

Media Pipeline-Specific State-Setting Commands

Media Pipeline-Specific State-Setting Commands

The following command is used to set state specific to the Media pipeline:

- MEDIA_STATE_POINTERS

The state variables associated with this command must be set appropriately prior to issuing MEDIA_OBJECT.

3DPRIMITIVE

Before issuing a 3DPRIMITIVE command, all state (with the exception of MEDIA_STATE_POINTERS) needs to be valid. Thus, the commands used to assign that state must be issued before issuing 3DPRIMITIVE.

MEDIA_OBJECT

Before issuing a MEDIA_OBJECT command, all state (with the exception of 3D-pipeline-specific state) needs to be valid. Therefore, the commands used to set this state need to have been issued at some point prior to the issue of MEDIA_OBJECT.

3D Pipeline Stages

The following table lists the various stages of the 3D pipeline and describes their major functions.

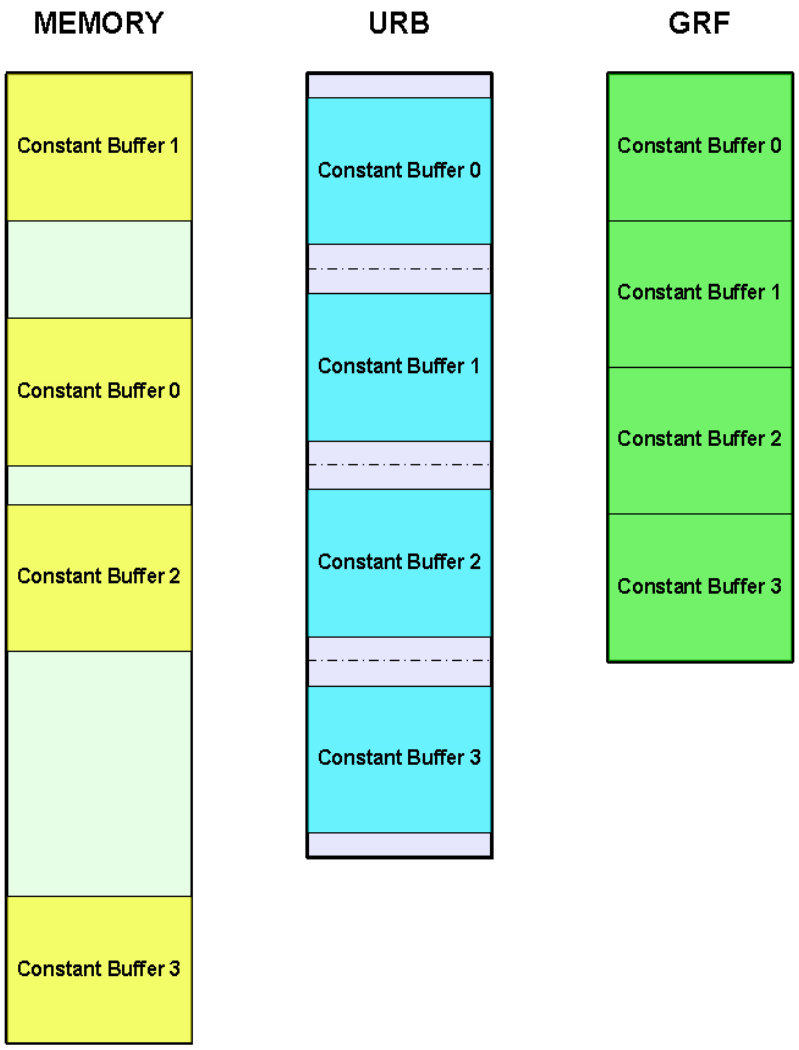
Pipeline Stage	Functions Performed
Command Stream (CS)	The Command Stream stage is responsible for managing the 3D pipeline and passing commands down the pipeline. In addition, the CS unit reads "constant data" from memory buffers and places it in the URB. Note that the CS stage is shared between the 3D, GPGPU and Media pipelines.
Vertex Fetch (VF)	The Vertex Fetch stage, in response to 3D Primitive Processing commands, is responsible for reading vertex data from memory, reformatting it, and writing the results into Vertex URB Entries. It then outputs primitives by passing references to the VUEs down the pipeline.
Vertex Shader (VS)	The Vertex Shader stage is responsible for processing (shading) incoming vertices by passing them to VS threads.
Hull Shader (HS)	The Hull Shader is responsible for processing (shading) incoming patch primitives as part of the tessellation process.
Tessellation Engine (TE)	The Tessellation Engine is responsible for using tessellation factors (computed in the HS stage) to tessellate U,V parametric domains into domain point topologies.
Domain Shader (DS)	The Domain Shader stage is responsible for processing (shading) the domain points (generated by the TE stage) into corresponding vertices.
Geometry Shader (GS)	The Geometry Shader stage is responsible for processing incoming objects by passing each object's vertices to a GS thread.
Stream Output Logic (SOL)	The Stream Output Logic is responsible for outputting incoming object vertices into Stream Out Buffers in memory.
Clipper (CLIP)	The Clipper stage performs Clip Tests on incoming objects and clips objects if required. Objects are clipped using fixed-function hardware.
Strip/Fan (SF)	The Strip/Fan stage performs object setup. Object setup uses fixed-function hardware.
Windower/Masker (WM)	The Windower/Masker performs object rasterization and determines visibility coverage.
CPS	Pipeline stage that gathers coarse pixels (CPs) for Coarse Pixel Shading (CPS).
PS Dispatch (PSD)	PSD assembles and dispatches Pixel Shader (PS) threads at one of these rates: CP, Pixel (P), or Sample (S).

3D Pipeline-Level State

This section contains table commands for the 3D Pipeline Level.

Programming Note	
Context:	3D Pipeline-Level State - Push Constant URB Allocation
The push constants are buffered in the Push Constant section of the URB. Software is required to program the hardware to allocate space in the URB for each shader push constant. The software is limited to the low addresses of the URB and must ensure that none of the shaders have overlapping handles. Software may use some if not all of the Push Constant region of the URB for pr-stage handle allocations as long as none of the push constants and handle allocations overlap. Refer to the various 3DSTATE_PUSH_CONSTANT_ALLOC_xx state commands for details regarding the maximum size of the Push Constant and other state programming information.	

Below is a diagram that represents how the hardware may move and store one CONSTANT_BUFFER command for a fixed function shader:



The bubbles in the URB are caused by the constant buffer in memory starting on a half cacheline and being an even number in length. If the constant buffer starts on an odd cacheline and has an odd number length, then there will only be a bubble at the beginning of the buffer in the URB. If the constant buffer in memory starts on a cache line boundary and has an odd number length then the bubble will only be at the end of the constant buffer in the URB. Once the constant buffer is written to the GRF space then all the bubbles will be removed.

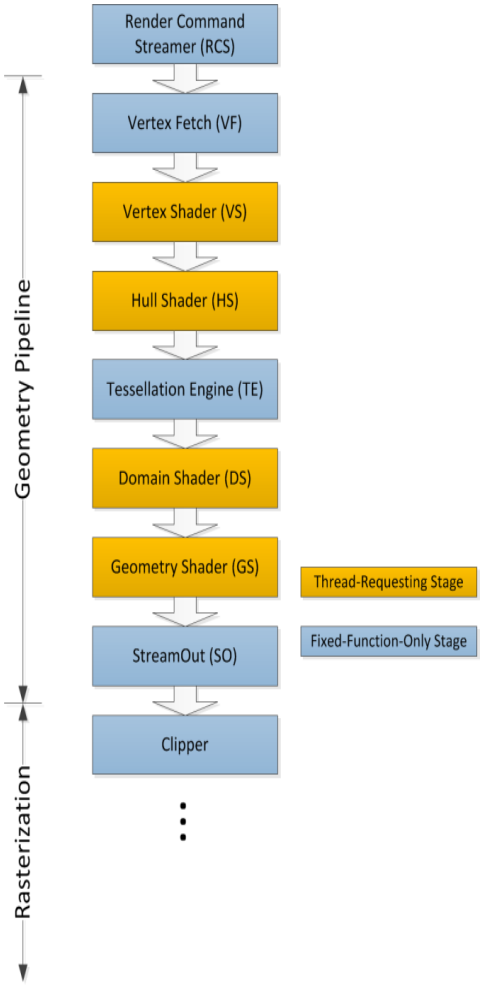
Software must guarantee that there is enough space in the push constant buffer in the URB to hold one constant buffer from memory. This includes any buffering to write the 512b aligned requests from memory into the URB.

3DSTATE_3D_MODE

3D Pipeline Geometry

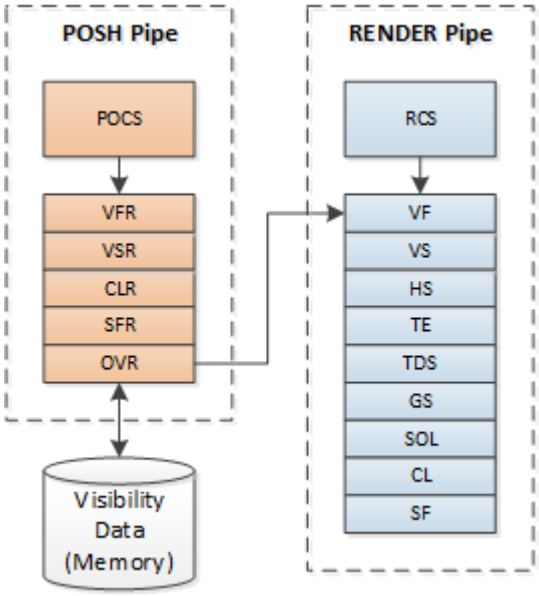
Block Diagram

The following block diagram shows the stages of the Geometry Pipeline and where they are positioned in the overall 3D Pipeline.



POSH Pipeline Overview

The **Position-Only Shading (POSH) pipeline** (aka "Cull Pipe" or "Record Pipe") is utilized to improve 3D rendering performance by removing culled objects from the Render pipeline workload. The POSH pipeline pre-processes geometry objects using simple "position-only" vertex input and vertex shaders. These objects are then subjected to clipper/setup cull tests. The results of these cull tests are then stored (compressed) as streams of "visibility tokens" in memory. Later, when the same geometry work is submitted to the Render pipe, the VF stage of the Render pipe will receive the pre-recorded visibility tokens and use those tokens to skip over culled objects and only process the non-culled objects. The POSH pipe is designed to run ahead of the Render pipe by buffering visibility data for render passes and possibly entire frames before being consumed by the Render pipe.



POSH Pipeline Work Submission

Work is performed on the POSH pipeline by submitting command streams to the POSH CS (POCS) unit which operates similarly to the Render CS (RCS) unit. Refer to *Command Stream Programming* for POCS programming details.

Geometry & Setup Stages of POSH Pipeline

The POSH pipeline contains POSH-specific versions of a subset of the Render pipe stages:

- VFR (POSH VF)
- VSR (POSH VS)
- CLR (POSH CL)
- SFR (POSH SF)

Note that the POSH pipeline does not contain HS, DS, GS or StreamOut stages and therefore does not support those functions. Work submitted to the POSH pipeline shall not contain state commands for those stages do not attempt to enable those functions.

These POSH stages are programmed in a similar manner as the corresponding Render stages. When stage-related state commands are submitted to the POSH pipeline, the corresponding stages in the POSH pipeline are programmed. POSH/Render pipeline programming differences are described in the state command definitions.

OVR Stage of POSH Pipeline

An Object Visibility Recording (OVR) stage is located at the end of the POSH pipe. It is used to compress and store visibility token streams in memory, as well as reading those streams during rendering and passing the tokens to the VF stage. Refer to the *Render Engine Command Streamer BXML* for programming details.

URB Programming when POSH Enabled

When the POSH pipeline is enabled, a URB allocation for the VSR stage is required. This allocation is programmed via execution of 3DSTATE_URB_*_VS commands in the POSH pipeline. Software shall be required to manage this allocation, taking into account the synchronous operation of the RCS and POCS workloads. This programming may require explicit synchronization between the pipelines, e.g., when Render vs. POSH URB allocation boundaries are changed.

When the POSH pipeline is enabled, a URB allocation for the POSH pipeline Push Constants may be defined. Refer to the relevant Push Constant URB commands for details on how this allocation is defined and used.

General Programming of Thread-Generating Stages (VS, HS, DS, GS)

This section provides common programming information for the thread-generating Geometry FF stages (VS, HS, DS, GS). The intent is to include the common description here in order to avoid redundancy in the subsequent stage-specific sections. The stage-specific sections will include any unique or exception information, restrictions, etc. relevant to those stages.

3DSTATE_ Common State Variables

This section describes FF state variables, programmed via 3DSTATE_<FF> commands, that are common to at least two thread-generating FF stages (VS, HS, DS, GS).

The states described in these sections are only used by HW when the given stage is enabled (i.e., can request thread execution), unless specifically called out as an exception.

Thread Management State

These state variables are used by a stage to manage thread request generation.

State	VS	HS	DS	GS
Maximum Number of Threads	Y	Y	Y	Y

Maximum Number Of Threads

This field specifies, for a particular stage, the maximum number of threads allowed to be simultaneously active. Here "active" refers to (a) outstanding in the thread request queue, (b) resident in the EUs, or (c) in the thread retirement queue - up to the point the stage sees the thread retirement. Note that the sum of (a) and (c) above is non-zero, and therefore - depending on configuration - the allowed number of active threads can exceed the total number of thread slots available in the EUs.

There are two main factors to consider when programming this state variable:

- **Scratch space availability:** In the case where threads require scratch space, SW shall allocate enough contiguous scratch space for the stage to allow each active thread (as programmed by this field) to access its full per-thread allocation (as programmed by **PerThreadScratchSpace**). This may require SW to reduce **MaximumNumberOfThreads** to accommodate limitations on scratch space availability.
- **Performance:** For best performance, it is recommended that SW program this field to its maximum value. This will maximize the number of threads available to perform the stage's function. However, SW is free to program a smaller value (as long as it meets any restrictions), e.g., for performance or workaround experimentation.

Thread State Initialization State

The following values are programmed as state, subsequently included by the stage as thread request control information, and eventually loaded into an EU architectural (ARF) register upon thread dispatch. In most instances these initial values can be subsequently overwritten by the thread.

For a complete description of these EU ARF register fields, refer to the EU Execution Environment section.

These values do not appear in the thread payload. (This information may be referred to as the thread's "transparent header", as it is forwarded to the EUs but not visible in the thread payload.)

State	EU State	VS	HS	DS	GS
Kernel Start Pointer	ip[31:6]	Y	Y	Y	Y
Floating Point Mode	cr0.0[0]	Y	Y	Y	Y
Single Program Flow	cr0.0[2]	N	Y	N	Y
Vector Mask Enable	cr0.0[3]	Y	Y	Y	Y
Illegal Opcode Exception Enable	cr0.1[12]	Y	Y	Y	Y
Software Exception Enable	cr0.1[13]	Y	Y	Y	Y
Thread Priority	sr0.0[23]	Y	Y	Y	Y
Binding Table Pointer	see note	Y	Y	Y	Y

Kernel Start Pointer (KSP)

This field specifies bits [31:6] of the value loaded into the EU's **Instruction Pointer** (ip), which in turn specifies the starting offset of the kernel program to be executed. The state is specified as a 64B-granular offset from the **Instruction Base Address** register (programmed via **STATE_BASE_ADDRESS**). Bits[5:3] of the EU 'ip' register (which identify a Dword within a 64B region) are loaded with 0 upon thread dispatch.

Note (below) that **Kernel Start Pointer [47:32]** can be programmed via FF state, but these bits are ignored by HW as the EU 'ip' register only supports a 32-bit value.

A stage may support more than one KSP state, where HW performs an on-the-fly selection of one of the KSPs based on some criteria. Refer to the stage-specific sections for details. For those stages that support multiple dispatch modes but only a single KSP state, SW shall ensure that the KSP value programmed corresponds with the selected dispatch mode.

Floating Point Mode

This state bit is loaded into the EU's **Single Precision Floating Point Mode** (FPMode, cr0.0[0]) which, in turn, controls how certain single-precision floating point operations are performed within the EU subsystem.

Single Program Flow

This state bit is loaded into the EU's **Single Program Flow** (SPF, cr0.0[2]) which, in turn, controls how certain flow control instructions operate across the EU channels.

Vector Mask Enable

This state bit is loaded into the EU's **Vector Mask Enable** (VME, cr0.0[3]) which, in turn, selects whether the EU's Dispatch Mask or Vector Mask register is used as the execution mask for subsequent instructions.

Illegal Opcode Exception

This state bit is loaded into the EU's **Illegal Opcode Exception Enable** (cr0.1[12]) which, in turn, enables or disables the EU's illegal opcode exception mechanism.

Software Exception Enable

This state bit is loaded into the EU's **Software Exception Enable** (cr0.1[13]) which, in turn, enables or disables the EU's software exception mechanism.

Thread Dispatch Priority

This state bit can be used to give thread requests eminating from a Geometry FF stage higher thread dispatch priority than thread request sources that are not marked as high priority.

This state bit is also loaded into the EU's **Priority Class** (sr0.0[23]) which, in turn, determines whether the EU thread is considered as belonging to the high priority class.

Binding Table Pointer (BTP)

Upon thread request, the BTP specified for the relevant FF stage is passed to, and stored in, the EU as part of thread state. This BTP value is subsequently passed to the Shared Functions (e.g., Sampler) that are required to access surfaces specified in the Binding Table. Here the BTP is passed via a side-band channel and not directly in the message descriptor or message header.

Thread State Initialization State (Defaulted)

The following EU state variables are defaulted upon thread dispatch and therefore cannot be controlled via Geometry FF state programming. Refer to the relevant EU sections for an understanding of these state variables and whether the thread can overwrite the defaulted values. Note that this is not an exhaustive list of defaulted EU state variables, only the ones deemed most interesting for Geometry FF threads.

State	EU State	Default Value
FFID	sr0.0[27:24]	see below
Rounding Mode	cr0.0[5:4]	0
Single Precision Denorm Mode	cr0.0[7]	0
Double Precision Denorm Mode	cr0.0[6]	0
Stack Overflow Exception Enable	cr0.1[11]	0
External Halt Exception Enable	cr0.1[14]	0
Breakpoint Exception Enable	cr0.1[15]	0
Instruction Pointer [5:3]	ip[5:3]	0
Stack Pointer	sp.0	0 (see note below)
Stack Pointer Limit	sp_limit	0 (see note below)

FFID

The EU's **Fixed Function Identifier** (FFID, sr0.0[27:24]) is initialized to a value corresponding to the Geometry FF stage that requested the thread dispatch. Note that this simply identifies the source FF unit, not the specific thread dispatched.

Stack Pointer, Stack Pointer Limit

These EU state registers are defaulted to 0 for threads requested by Geometry FF units, as opposed to other thread request sources that may cause them to be initialized differently. The threads can overwrite the defaulted values if so desired.

Prefetch State

The following state variables can be used by SW to attempt the prefetch of certain state from memory into internal state cache. The prefetch is requested as part of the first thread dispatch after these state variables are specified.

Programming Restriction: Software shall not specify a prefetch region that extends into an invalid memory page, otherwise the prefetch may incur page faults.

Performance Note: Early prefetch of the state that will likely be referenced by the thread can improve thread execution performance. This is not guaranteed, especially if the amount of prefetched data is large which may result in state cache thrashing. Also, these prefetch requests are considered low priority hints by HW and may be dropped under conditions of high memory demand.

State	VS	HS	DS	GS
Sampler Count	Y	Y	Y	Y
Binding Table Entry Count	Y	Y	Y	Y

Sampler Count

This field specifies how many SAMPLER_STATE structures are prefetched from memory. The count can be specified as 0 or as a multiple of 4 (4,8,12,16). Refer to the state definition for encodings and further details.

Performance Note: It is recommended that SW program this field to (roughly) equal the number of sampler state structures referenced by the thread.

Binding Table Entry Count

This field specifies how many binding table entries (BTEs) and associated SURFACE_STATE structures are prefetched from memory. The format of this field depends on whether or not HW-generated binding tables are enabled, as determined by 3DSTATE_BINDING_TABLE_POOL_ALLOC::**BindingTablePoolEnable**.

SW Usage Note: When HW-generated binding tables are enabled, it is recommended that the Binding Table Entry Count value be generated when the shader is compiled.

HW-Generated Binding Tables Disabled:

The field has a Format of U8 and specifies a count of BTEs to be prefetched ([0,255]). Each of the SURFACE_STATE structures referenced by the BTEs will also be prefetched.

HW-Generated Binding Tables Enabled:

This field has a Format of Bitmask8 and indicates which 64B cache lines of BTEs will be fetched. Each bit in this field corresponds to a cache line, where a cache line holds 8 16-bit BTEs. Bit 0 refers to the cacheline starting at the Binding Table Pointer, as programmed by 3DSTATE_BINDING_TABLE_POINTER_xx.

By default, only the SURFACE_STATE structures referenced by the first 4 non-zero BTEs of each 64B cacheline will be prefetched.

Common Thread Payload-Related State

The following state variables are either included directly in the thread payload and/or used to control or compute other fields in the thread payload.

State	VS	HS	DS	GS
Sampler State Pointer	Y	Y	Y	Y
Per-Thread Scratch Space	Y	Y	Y	Y
Scratch Space Base Pointer	Y	Y	Y	Y
Include Vertex Handles	N	Y	N	Y

Sampler State Pointer

This state variable specifies the starting, 32B-granular offset of the stage's SAMPLER_STATE table in memory, relative to the **DynamicStateBaseAddress**. It is programmed via 3DSTATE_SAMPLER_STATE_POINTERS_xx commands.

This value is included in thread payloads in R0.3[31:5] and is also directly propagated to the Sampler shared function for use in processing "headerless" messages. If a thread can potentially send any messages to the Sampler shared function that requires the Sampler State Pointer in the message header, that thread shall ensure that it passes along the Sampler State Pointer value passed in the thread payload.

Scratch Space

The **Per-Thread Scratch Space** state variable specifies the amount of scratch memory required by each active thread of a stage. The value is specified as a 4-bit power of two (in excess of 10) bytes, where programmed values in the valid range [0,11] specify scratch space requirements in the range [1KB, 2MB].

When a thread becomes "active" it is allocated a portion of scratch space, sized according to **PerThreadScratchSpace**. The starting location of each thread's scratch space allocation, **ScratchSpaceOffset**, is passed in the thread payload in R0.5[31:10] and is specified as a 1KB-granular offset from the **GeneralStateBaseAddress**. The computation of **ScratchSpaceOffset** includes the starting address of the stage's scratch space allocation, as programmed by **ScratchSpaceBasePointer**. The maximum number of active threads for a stage is specified by the **MaximumNumberOfThreads** state. SW shall abide by the scratch space restrictions included in the description of **MaximumNumberOfThreads**.

This value is also included within thread payloads in R0.3[3:0]. If a thread can potentially send any "A32 Stateless" messages to the DataPort shared function, that thread shall ensure that it passes along the **PerThreadScratchSpace** value passed in the thread payload.

The state command specifies starting offset of the scratch memory region allocated to a stage (**Scratch Space Base Pointer**). It is specified as a 22-bit, 1KB-aligned offset from the **GeneralStateBaseAddress**.

Each thread requested by the FF stage will be allocated it's exclusive portion of this space, with the per-thread allocation size specified by **Per-Thread Scratch Space**. The computed offset of the thread-specific portion is passed in the thread payload as **Scratch Space Offset**. If the thread needs to access this scratch space, it shall utilize "stateless" DataPort read/write message, where the DataPort will cause the **General State Base Address** to be added to the specific scratch space offset passed in the message header.

Include Vertex Handles

This state variable specifies whether input vertex URB handles are included in the thread payload for threads requested by the FF stage. SW shall set this bit if the thread kernel requires access to the data contained input vertex URB entries, either in addition to or instead of the input vertex data pushed into the thread payload.

URB Payload State

The following state variables specify certain parameters related to the amount and location of URB-sourced data in the thread payload. State variables specifying other parameters are found in other state commands. Refer to the Thread Payload Overview subsection for more details.

State	VS	HS	DS	GS
Dispatch GRF Start Register for URB Data	Y	Y	Y	Y
Vertex/Patch URB Entry Read Offset	Y	Y	Y	Y
Vertex/Patch URB Entry Read Length	Y	Y	Y	Y

Dispatch GRF Start Register for URB Data

This state variable specifies a 5b GRF# (32B offset) within the thread payload where URB-sourced data starts. The URB-sourced data starts with some (possibly zero) amount of pushed Constant data, followed by some (possibly zero) amount of Vertex or Patch data.

Programming Restriction: Software shall ensure that it does not cause URB data to overwrite the R0 Header or Extended Header.

Vertex/Patch URB Entry Read Offset

This state variable specifies the 32B offset at which data is to be read from each Vertex or Patch URB entry before being included in the thread payload.

Vertex/Patch URB Entry Read Length

This state variable specifies the number of 16B (vertex elements) to be read from each Vertex or Patch URB entry, starting from the offset specified by the **Vertex/PatchURBEntryReadOffset** state.

If the read length is non-zero, SW shall ensure that the specification of the source (URB) data does not extend beyond the allocated and valid data in the URB entry. Other restrictions are described in the Thread Payload Overview subsection.

Pre-Rasterization Vertex State

The following state variables are implemented in the FF stages whose associated threads generate vertices (therefore the HS stage is excluded). The state variables control some aspects of how the generated ("output") vertices are treated if the pipeline is configured to have the stage's vertices to reach the Clip and Setup stages. Hardware determines which stage produces these "pre-rasterization" vertices as a function of which FF stages are enabled. For example, if the GS and DS stages are disabled, the VS

stage's set of state variables will be used or alternatively, if the GS stage is enabled, the GS stage's set of state variables will be used.

There are "Force" state bits in the Clip & Setup stages that can be used to override use of these per-FF state variables and instead use corresponding state variables programmed in the Clip and/or Setup stages.

State	VS	HS	DS	GS
Vertex URB Entry Output Read Offset	Y	N	Y	Y
Vertex URB Entry Output Read Length	Y	N	Y	Y
User Clip Distance Clip Test Enable Bitmask	Y	N	Y	Y
User Clip Distance Cull Test Enable Bitmask	Y	N	Y	Y

Vertex URB Entry Output Read Offset

This state variable specifies the 32B offset at which attribute data is to be read from each Vertex URB entry for use by the Setup stage.

Vertex URB Entry Output Read Length

This state variable specifies the number of 16B attributes to be read from each Vertex URB entry for use by the Setup stage, starting from the offset specified by the **VertexURBEntryOutputReadOffset** state.

User Clip Distance Clip Test Enable Bitmask

This state variable is used in the Clip stage's clip test functionality. See Clip stage documentation for details.

User Clip Distance Cull Test Enable Bitmask

This state variable is used in the Clip stage's cull test functionality. See Clip stage documentation for details.

UAV Access State

This state variable is used by the HW UAV Coherency mechanism.

State	VS	HS	DS	GS	PS	Compute
Accesses UAV	Y	Y	Y	Y	Y	N

Accesses UAV

This state bit indicates that threads requested by this FF stage may perform accesses to UAV resources. If SW enables the HW UAV Coherency function, it shall set this bit in order to include this stage in the coherency activities. For improved performance, SW should only set this bit for those FF stages that require it. If the HW UAV Coherency function is enabled, this bit is ignored.

Statistics Enable

This state variable is used to enable/disable the statistic counter for a FF stage.

State	VS	HS	DS	GS
Statistics Enable	Y	Y	Y	Y

Statistics Enable

This state bit controls whether or not the statistic counter(s) associated with a FF stage are enabled. Refer to the specific FF stage descriptions for details on the statistics counter(s) supported.

SW shall disable statistics counting via this bit prior to submitting any 3DPRIMITIVE commands that are not to be included in statistics counting. For example, if the statistics counters are to be maintained to only track application-submitted work, SW shall ensure that any driver-generated work is not included in the statistics.

Thread State (Ignored)

The following state variables can be programmed but are ignored in the HW implementation.

State	VS	HS	DS	GS
Kernel Start Pointer [47:32]	Y	Y	Y	Y
Scratch Space Base Offset Upper	Y	Y	Y	Y

URB Allocation Overview

The Geometry FF stages use the URB for temporary storage of vertex and/or patch data as URB Entries, as well as Push Constant (PC) URB Buffers. Software can program the total size of the URB (see URB/L3 documentation). Software can also partition the URB space into FF stage-specific allocations for URB Entries and/or PC URB Buffers. These allocations can be changed dynamically to accommodate changing pipeline configurations and shader data requirements, though such changes may have performance impacts. There shall be no overlap between the individual allocations and no allocation may extend beyond the programmed URB upper limit.

Only the first 32KB of the URB can be used for VS, HS, DS, GS, and PS PC URB Buffer allocations. See *Push Constant Programming*.

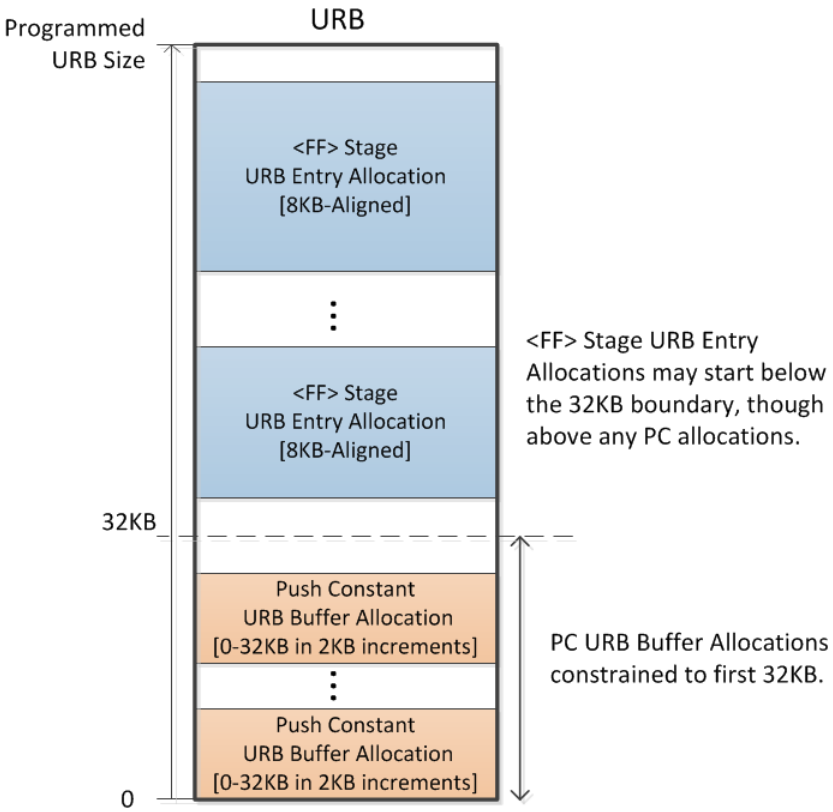
Software can place URB Entry allocations following any PC URB Buffer allocations. Software shall define allocations for all the relevant Geometry FFs (VS, HS, DS, GS), though a subset of these allocations can be "null" allocations that do not consume URB space. The VS stage always requires a non-null allocation. The HS and DS stages only require non-null allocations when tessellation is enabled. Likewise, the GS stage only requires a non-null allocation when GS is enabled.

When POSH is Enabled (via CTXT_SR_CTL), an additional 32KB block of URB is allocated for POCS pipeline Push Constants.

This block is located immediately after the RCS Push Constant URB Buffer Allocation.

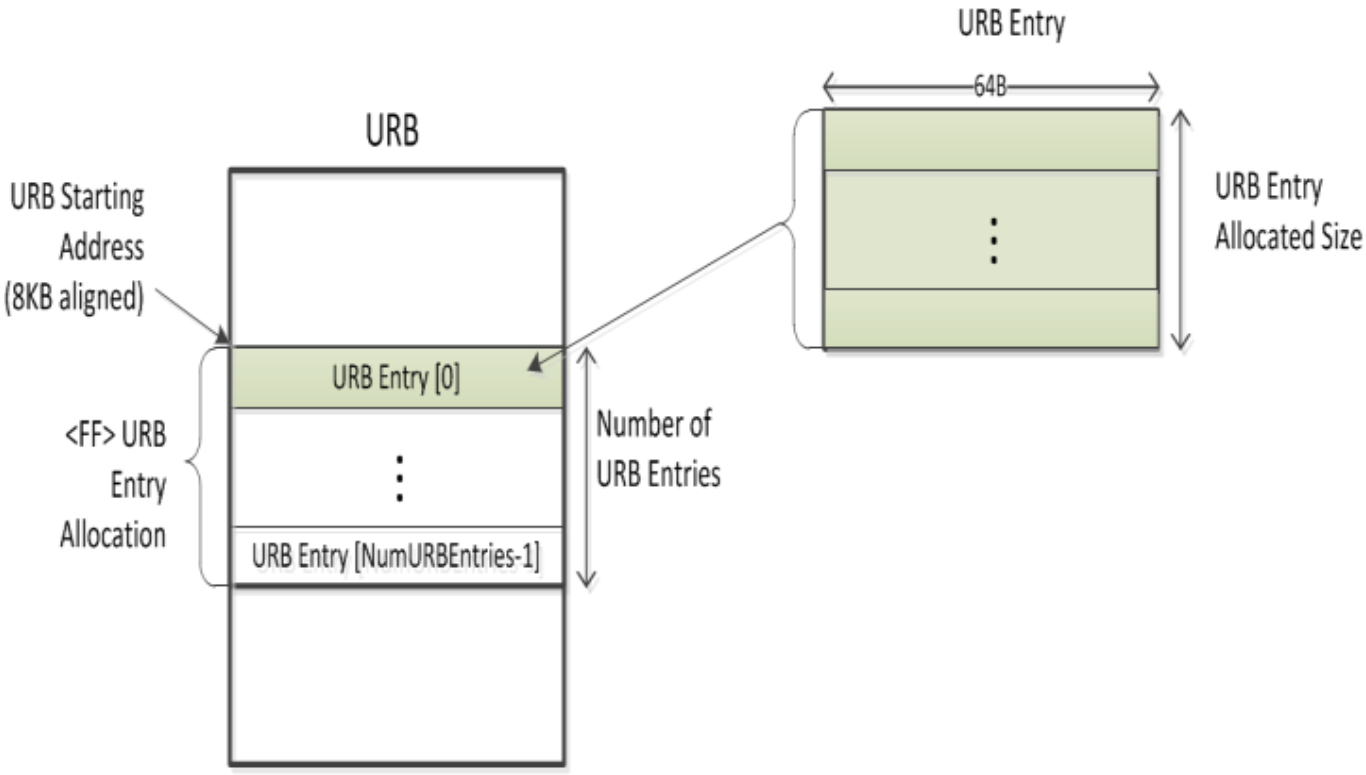
When enabled, the size of the Push Constant URB allocation mentioned in the URB programming information (below) will increase to 64KB total (vs. the 32KB size shown)

URB Space Partitioning



The starting offset (within the URB space) of a FF URB Entry allocation is specified by a **URBStartAddress** state. The size of an allocation is defined by a **NumberOfURBEntries** state and a corresponding **URBEntryAllocationSize** state.

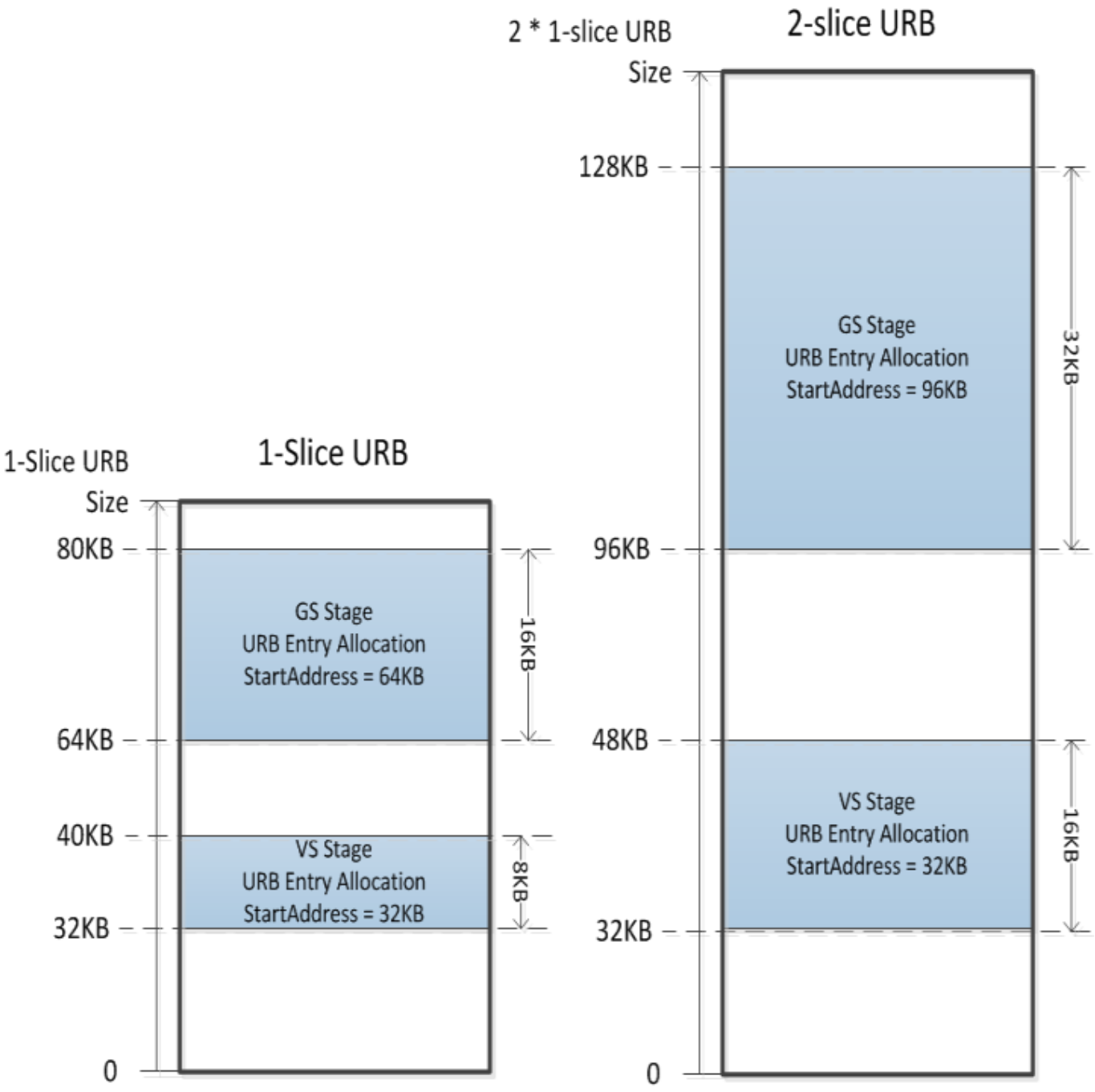
URB Entry and Entry Allocation



Multiple-Slice Implications

The FF URB allocations are programmed based on the URB size for a single slice. If the configuration includes multiple slices, the HW will automatically adjust the **URBStartAddress** and **NumberOfURBEntries** states according to the number of slices. The **URBEntryAllocationSize** states are not affected, nor are the PC URB Buffer allocations affected. The **NumberOfURBEntries** states are simply multiplied by the number of slices. The **URBStartAddress** states are scaled by the number of slices after being first decreased by 32KB, with a 32KB offset added back in after scaling. This scales the allocation start addresses relative to the 32KB boundary versus the start of the URB.

The following diagram provides an example of how this scaling would be applied in a 2-slice configuration.



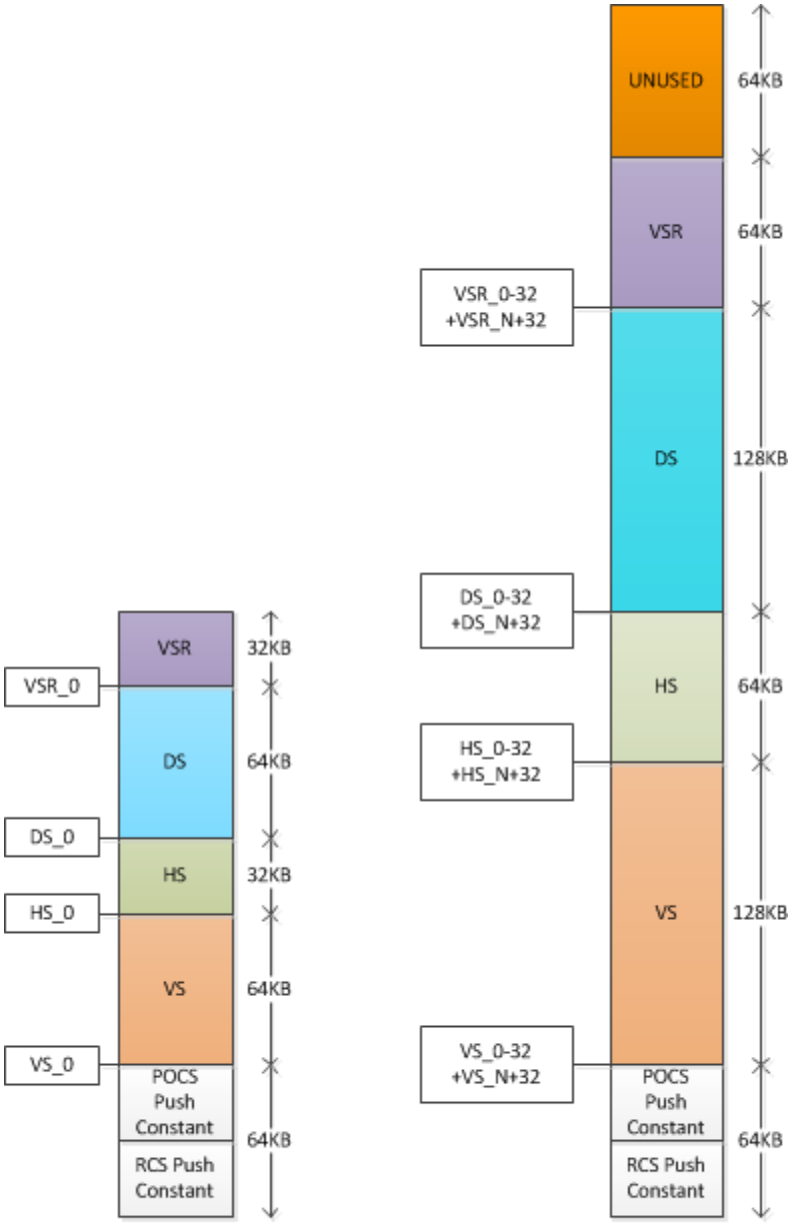
Enhanced URB Allocation Overview

An "enhanced", alternative FF URB allocation scheme is available and offers more efficient URB allocation for multiple-slice configurations when Push Constant URB allocations are required. This scheme uses the 3DSTATE_URB_ALLOC_VS, 3DSTATE_URB_ALLOC_HS, 3DSTATE_URB_ALLOC_DS and 3DSTATE_URB_ALLOC_GS commands. The states programmed by these commands are similar to the legacy 3DSTATE_URB_VS, 3DSTATE_URB_HS, 3DSTATE_URB_DS and 3DSTATE_URB_GS commands, though the new commands support separate URB Starting Address and Number Of URB Entries states

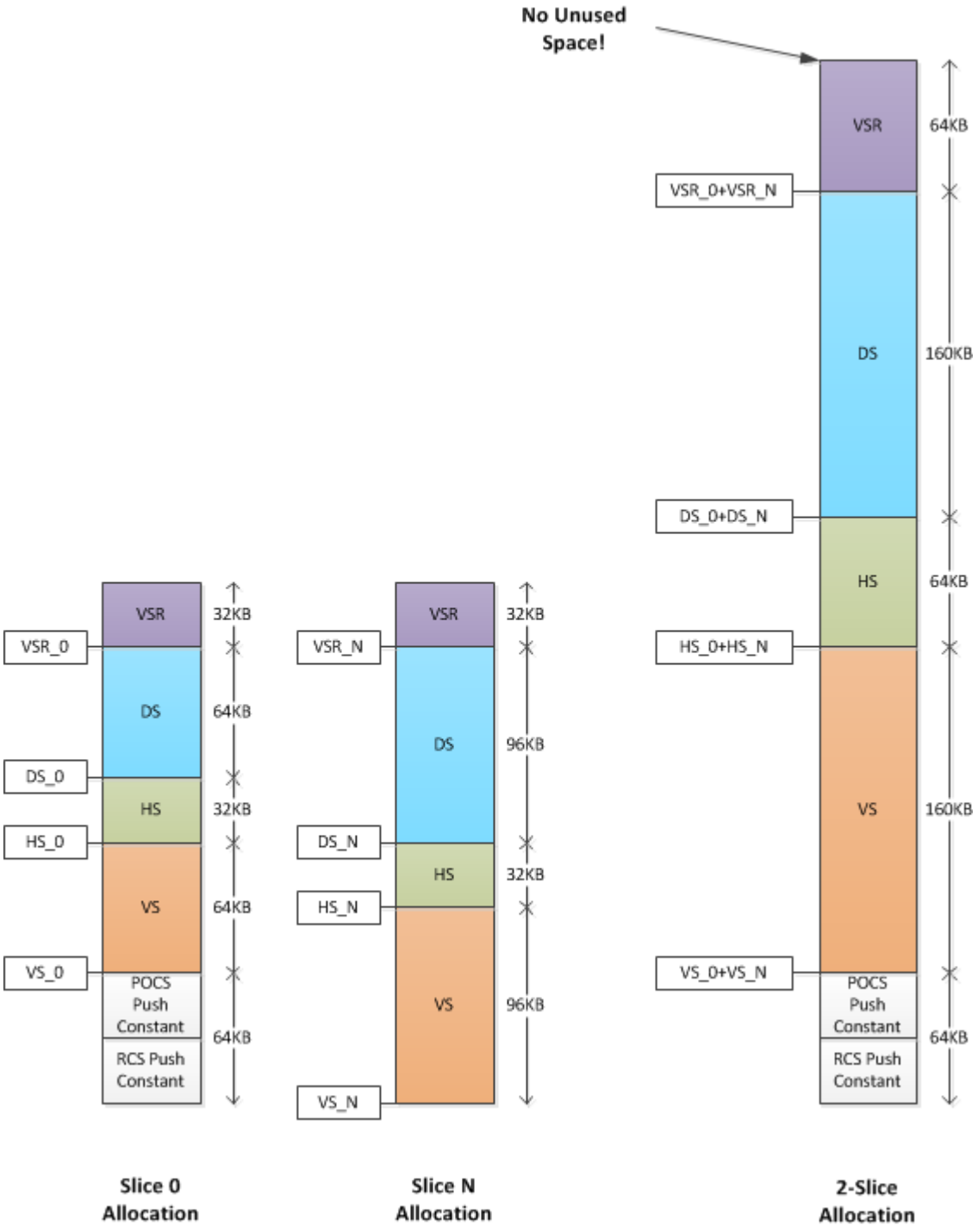
for "Slice0" and "SliceN" (slices beyond Slice0). HW will use both sets of states to scale the URB allocation up to the number of enabled slices, versus the legacy commands that only used the single set of states for Slice0.

The enhanced scheme & commands are available in addition to the legacy scheme & commands. However, within a context, SW shall only utilize one scheme.

The benefit of the enhanced scheme is apparent for multiple-slice configurations when Push Constant URB allocations are required. The legacy scheme only supplied a single set of states for one slice. The size of the FF allocations are therefore impacted by the size of the Push Constant allocations. However, in a multiple-slice configuration, there is only one set of Push Constant allocation. The legacy scheme was unable to take advantage of that fact, resulting in *unused URB storage* at the end of the URB space equal in size to N-1 times the size of the Push Constant allocations (where N is the number of enabled slices). The following example diagram illustrates this condition associated with the legacy scheme:



The enhanced scheme allows a separate, second set of states for "SliceN" (slices beyond Slice0) that can be used to specify allocations that are not impacted by Push Constant URB allocations (as these allocations do not exist in slices beyond Slice0). Therefore the entire URB can be allocated with no wasted space. The following example diagram illustrates how the use of the enhanced scheme leads to more efficient FF URB allocation:


No Unused Space!
VSR
VSR_0+VSR_N
DS
DS_0+DS_N
HS
HS_0+HS_N
VS
VS_0+VS_N
POCS Push Constant
RCS Push Constant
64KB
160KB
64KB
160KB
64KB
VSR
VSR_0
DS
DS_0
HS
HS_0
VS
VS_0
POCS Push Constant
RCS Push Constant
32KB
64KB
32KB
64KB
64KB
VSR
VSR_N
DS
DS_N
HS
HS_N
VS
VS_N
32KB
96KB
32KB
96KB
Slice 0 Allocation
Slice N Allocation
2-Slice Allocation

Allocation State Variables

URB Starting Address

This state variable defines the 8KB-aligned starting offset of the URB allocation for a given FF stage.

URB Entry Allocated Size

Programming Note	
Context:	Allocation State Variables

- This state variable defines the amount of storage allocated for each URB Entry within the allocation. It is 64B-granular. (Note that the contents of a URB entry can be accessed at 32B granularity). The required size of a URB Entry is typically dictated by API parameters and API shader programs.
- Software should attempt to minimize the size of URB entries in order to maximize the number of URB Entries that can be stored in a given allocation. However, as changing URB-related state variable can incur performance penalties, software may decide to employ sizing heuristics that permit a limited amount of wasted space in URB entries as a performance tradeoff.

Number of URB Entries

Programming Note	
Context:	Allocation State Variables

- This state variable defines the number of URB Entries allocated for a given FF stage. If the stage is disabled, the number of entries can be programmed as 0, though this is not required. If the stage is enabled (which is always true for the VS stage), a non-zero number of entries shall be specified. If the **URBEntryAllocatedSize** is less than or equal to 8 64B units, this number shall be 0 or a multiple of 8.
- The minimum number of entries required as well as the maximum number supported are specific to the FF stage and state programming (e.g., a function of Dispatch Mode) - see the documentation for the specific stage.

Thread Request Generation

Once a FF unit determines that a thread can be requested, it must gather all the information required to submit the thread request to the Thread Dispatcher. This information is divided into several categories, as listed below and subsequently described in detail.

- **Thread Control Information:** This is the information required (from the FF unit) to establish the execution environment of the thread.
- **Thread Payload Header:** This is the first portion of the thread payload passed in the GRF, starting at GRF R0. This is information passed directly from the FF unit. It precedes the Thread Payload Input URB Data.
- **Thread Payload Input URB Data:** This is the second portion of the thread payload. It is read from the URB using entry handles supplied by the FF unit.

Thread Control Information

The following table describes the various state variables that a FF unit uses to provide information to the Thread Dispatcher and which affect the thread execution environment. Note that this information is not directly passed to the thread in the thread payload (though some fields may be subsequently accessed by the thread via architectural registers).

State Variables Included in Thread Control Information

State Variable	Usage	FFs
Kernel Start Pointer	This field, together with the **General State Pointer**, specifies the starting location (1st EU core instruction) of the kernel program run by threads spawned by this FF unit. It is specified as a 64-byte-granular offset from the **General State Pointer**.	All FFs spawning threads
GRF Register Block Count	Specifies, in 16-register blocks, how many GRF registers are required to run the kernel. The Thread Dispatcher will only seek candidate EUs that have a sufficient number of GRF register blocks available. Upon selecting a target EU, the Thread DIspatcher will generate a logical-to-physical GRF mapping and provide this to the target EU.	All FFs spawning threads
Single Program Flow (SPF)	Specifies whether the kernel program has a single program flow (SIMDnxm with m = 1) or multiple program flows (SIMDnxm with m > 1). See CR0 description in *ISA Execution Environment*.	All FFs spawning threads
Thread Dispatch Priority	The Thread Dispatcher will give priority to those thread requests with Thread Dispatch Priority of HIGH_PRIORITY over those marked as LOW_PRIORITY. Within these two classes of thread requests, the Thread Dispatcher applies a priority order (e.g., round-robin --- though this algorithm is considered a device implementation-dependent detail).	All FFs spawning threads
Floating Point Mode	This determines the initial value of the **Floating Point Mode** bit of the EU's CR0 architectural register that controls floating point behavior in the EU core. (See ISA.)	All FFs spawning threads
Exceptions Enable	This bitmask controls the exception handing logic in the EU. (See ISA.)	All FFs spawning threads
Sampler Count	This is a hint which specifies how many indirect SAMPLER_STATE structures should be prefetched concurrent with thread initiation. It is recommended that software program this field to equal the number of samplers, though there may be some minor performance impact if this number gets large. This value should not exceed the number of samplers accessed by the thread as there would be no performance advantage. Note that the data prefetch is treated as any other memory fetch (with respect to page faults, etc.).	All stages supporting sampling (VS, GS, WM)
Binding Table Entry Count	This is a hint which specifies how many indirect BINDING_TABLE_STATE structures should be prefetched concurrent with thread initiation. (The notes	All FFs spawning threads

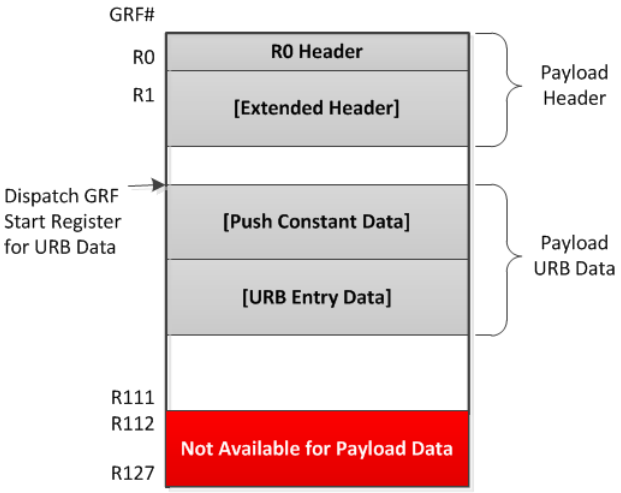
State Variable	Usage	FFs
	included in Sampler Count (above) also apply to this field).	

Thread Payload Overview

Like all threads, the threads spawned by Geometry FF stages have some amount of payload data pre-loaded into the GRF for use as initial input to a thread's kernel. Some of the data is sourced directly from the spawning FF and/or intermediate Thread Dispatch functions, while some is sourced from the URB as specified by the spawning FF. The Geometry FF thread payloads have a similar structure, though the exact payload size/content/layout is unique to each FF stage. This subsection describes the general layout of the payload - refer to the specific FF stage descriptions for details and differences.

The payload data loaded into the GRF starting at R0 and is divided into two main sections: the Payload Header followed by the Payload URB Data. The Payload Header contains information passed from FF units, while the Payload URB Data is obtained from the URB.

Geometry FF Thread Payload Layout (General)



The Payload Header is further subdivided into a leading R0 Header and (if present) a variable-sized Extended Header. The R0 fields are laid out to closely match the message header (M0) of thread-generated messages to shared functions. The Extended Header (if present) starts in R1 and its length varies.

The Payload URB Data section is optional and can contain a variable amount of Push Constant Data and/or a variable amount of (vertex or patch) URB Entry Data. The Payload URB Data starts at the GRF# defined by the **DispatchGRFStartRegisterForURBData** FF state variable. Software can use this state variable to place the Payload URB Data at a common starting GRF# even when the size of the Extended Header varies.

Thread Payload Generation

FF units are responsible for generating a thread *payload* - the data pre-loaded into the target EU's GRF registers (starting at R0) that serves as the primary direct input to a thread's kernel. The general format of these payloads follows a similar structure, though the exact payload size/content/layout is unique to each stage. This subsection describes the common aspects - refer to the specific stage's chapters for details on any differences.

The payload data is divided into two main sections: the *payload header* followed by the *payload URB data*. The payload header contains information passed directly from the FF unit, while the payload URB data is obtained from URB locations specified by the FF unit.

The first 256 bits of the thread payload (the initial contents of R0, aka "the R0 header") is specially formatted to closely match (and in some cases exactly match) the first 256 bits of thread-generated *messages* (i.e., the message header) accepted by shared functions. In fact, the send instruction supports having a copy of a GR's contents (such as R0) used as the message header. Software must take this intention into account (i.e., "don't muck with R0 unless you know what you're doing"). This is especially important given the fact that several fields in the R0 header are considered opaque to SW, where use or modification of their contents might lead to UNDEFINED results.

The payload header is further (loosely) divided into a leading *fixed payload header* section and a trailing, variable-sized *extended payload header* section. In general, the size, content and layout of both payload header sections are FF-specific, though many of the fixed payload header fields are common amongst the FF stages. The extended header is used by the FF unit to pass additional information specific to that FF unit. The extended header is defined to start after the fixed payload header and end at the offset defined by **Dispatch GRF Start Register for URB Data**. Software can cause use the **Dispatch GRF Start Register for URB Data** field to insert padding into the extended header in order to maintain a fixed offset for the start of the URB data.

Fixed Payload Header

The payload header is used to pass *FF pipeline information* required as thread input data. This information is a mixture of SW-provided state information (state table pointers, etc.), primitive information received by the FF unit from the FF pipeline, and parameters generated/computed by the FF unit. Most of the fields of the fixed header are common between the FF stages. These non-FF-specific fields are described in Fixed Payload Header Fields (non-FF-specific). Note that a particular stage's header may not contain all these fields, so they are not "common" in the strictest sense.

Fixed Payload Header Fields (non-FF-specific)

Fixed Payload Header Field (non-FF-specific)	Description	FFs
FF Unit ID	Function ID of the FF unit. This value identifies the FF unit within the GPU. The FF unit uses this field (when transmitted in a Message Header to the URB Function) to detect messages emanating from its spawned threads.	All FFs spawning threads
Snapshot Flag		All FFs spawning threads
Thread ID	This field uniquely identifies this thread within the FF unit over some period.	All FFs spawning threads
Scratch Space Pointer	This is the starting location of the thread's allocated scratch space, specified as an offset from the **General State Base Address**. Note that scratch space is allocated by the FF unit on a per-thread basis, based on the **Scratch Space Base Pointer** and **Per-Thread Scratch Space Size** state variables. FF units assign a thread an arbitrarily-positioned region within this space. The scratch space for multiple (API-visible) entities (vertices, pixels) is interleaved within the thread's scratch space.	All FFs spawning threads
Dispatch ID	This field identifies this thread within the outstanding threads spawned by the FF unit. This field does *not* uniquely identify the thread over any significant period. **Implementation Note:** This field is effectively an "active thread index". It is used on a thread's URB allocation request to identify which thread's handle pool is to source the allocation. It is used upon thread termination to free up the thread's scratch space allocation.	All FFs spawning threads
Binding Table Pointer	This field, together with the **Surface State Base Pointer**, specifies the starting location of the Binding Table used by threads spawned by the FF unit. It is specified as a 64-byte-granular offset from the **Surface State Base Pointer**. See *Shared Functions* for a description of a Binding Table.	All FFs spawning threads
Sampler State Pointer	This field, together with the **General State Base Pointer**, specifies the starting location of the Sampler State Table used by threads spawned by the FF unit. It is specified as a 64-byte-granular offset from the **General State Base Pointer**. See *Shared Functions* for a description of a Sampler State Table.	**All FFs spawning threads which sample (VS, GS, WM)**
Per Thread Scratch Space	This field specifies the amount of scratch space allocated to each thread spawned by the FF unit. The driver must allocate enough contiguous scratch space, starting at the **Scratch Space Base Pointer**, to ensure that the **Maximum Number of Threads** can each get **Per-Thread Scratch Space** size without exceeding the driver-allocated scratch space.	All FFs spawning threads
Handle ID <*n*>	This ID is assigned by the FF unit and links the thread to a specific entry within the FF unit. The FF unit will use this information upon detecting a URB_WRITE message issued by the thread. Threads spawned by the GS, CLIP, and SF units are provided with a single Handle ID / URB Return Handle pair. Threads spawned by the VS are provided	VS, GS, CLIP, SF

Fixed Payload Header Field (non-FF-specific)	Description	FFs
	with one or two pairs (depending on how many vertices are to be processed). Threads spawned by the WM do not write to URB entries, and therefore this info is not supplied.	
URB Return Handle <*n*>	This is an initial destination URB handle passed to the thread. If the thread does output URB entries, this identifies the destination URB entry. Threads spawned by the GS, CLIP, and SF units are provided with a single Handle ID / URB Return Handle pair. Threads spawned by the VS are provided with one or two pairs (depending on how many vertices are to be processed). Threads spawned by the WM do not write to URB entries, and therefore this info is not supplied.	VS, GS, CLIP, SF
Primitive Topology Type	As part of processing an incoming primitive, a FF unit is often required to spawn a number of threads (for example, for each individual triangle in a TRIANGLE_STRIP). This field identifies the type of primitive which is being processed by the FF unit, and which has lead to the spawning of the thread. EU kernels written to process different types of objects can use this value to direct that processing. E.g., when a CLIP kernel is to provide clipping for all the various primitive types, the kernel would need to examine the Primitive Topology Type to distinguish between point, lines, and triangle clipping requests. **Note:** In general, this field is identical to the Primitive Topology Type associated with the primitive vertices as received by the FF unit. Refer to the individual FF unit chapters for cases where the FF unit modifies the value before passing it to the thread. (for example, certain units perform toggling of TRIANGLESTRIP and TRIANGLESTRIP_REV).	GS, CLIP, SF, WM

R0 Header

The R0 header is used to pass various parameters to threads. This information contains SW-provided state information, primitive information received by the FF unit from the FF pipeline, and parameters generated/computed by the FF unit or Thread Dispatch HW.

Below is a list and description of the R0 Header fields common to all Geometry FF thread payloads. Refer to the specific payload definitions for more details and (if relevant) other FF-specific fields.

R0 Header Field	R0 Header Location
Thread ID	R0.6[23:0]
FF Thread ID (FFTID)	R0.5[9:0]
Scratch Space Offset	R0.5[31:10]
Per Thread Scratch Space	R0.3[3:0]
Sampler State Pointer	R0.3[31:5]

Thread ID

This field is a sequence number that identifies this thread within the all threads spawned by the relevant FF stage over some unspecified period of time.

FF Thread ID

This field is assigned by the relevant FF stage and used to identify the thread within the set of currently-outstanding threads spawned by the FF unit. It shall be included in EOT messages sent by the thread as required by the relevant message header.

Scratch Space Offset

This field specifies the starting offset of the 1KB-aligned scratch space region allocated to this particular thread. See the definition of **ScratchSpaceBaseAddressLow**, which specifies the starting offset for scratch space region allocated to the FF stage.

Per Thread Scratch Space

This field is a copy of the **PerThreadScratchSpace** state variable programmed by SW via the 3DSTATE_<FF> commands.

Sampler State Pointer

This field is a copy of the **SamplerStatePointer** state variable programmed by SW via the 3DSTATE_<FF> commands

Extended Payload Header

The extended header is of variable-size, where inclusion of a field is determined by FF unit state programming.

In order to permit the use of common kernels (thus reducing the number of kernels required), the **Dispatch GRF Start Register for URB Data** state variable is supported in all FF stages. This SV is used to place the payload URB data at a specific starting GRF register, irrespective of the size of the extended header. A kernel can therefore reference the payload URB data at fixed GRF locations, while conditionally referencing extended payload header information.

Extended Header (R1+)

In some cases additional FF-sourced information is passed in a variable-size Extended Header, which starts at GRF R1. Some of the field definitions are common across two or more payloads and are described below. Refer to specific payload definitions for more details and (if relevant) other FF-specific fields.

Extended Header Fields
Output URB Handles
Input URB Handles
PrimitiveIDs

Output URB Handles

This set of 16-bit fields contains the 64B-aligned offsets into the URB at which a thread is to write output URB data (i.e., vertex or patch data) via URB Write messages. In the VS thread payload URB Handles are used both for input and output. Up to 8 Output URB Handles can be included in a thread payload. In some SIMD4x2 payloads, these handles are passed in the R0 Header.

Input URB Handles

This set of 16-bit fields contains the 64B-aligned offsets into the URB at which a thread can access input URB data (i.e., vertex or patch data) via URB Read messages. In the VS thread payload URB Handles are used both for input and output. Up to 256 Input URB Handles can be passed in the Extended Header.

As it is often possible for all input URB data to be pushed in the thread payload, the thread may not require Input URB Handles. As these handles may not be needed, a corresponding **IncludeVertexHandles** state bit is typically included in the FF stage's state (via 3DSTATE_<FF>). This state bit controls whether the Input URB Handles are included in the Extended Header.

PrimitiveIDs

This set of 32-bit fields contains the **PrimitiveID** values corresponding to input objects being processed by the thread. See Vertex Fetch for a description of **PrimitiveID**. As PrimitiveID may not be required as input by the thread, a corresponding **IncludePrimitiveID** state bit is typically included in the FF stage's state (via 3DSTATE_<FF>). This state bit controls whether the PrimitiveIDs are included in the Extended Header.

Payload URB Data

In each thread payload, following the payload header, is some amount of URB-sourced data required as input to the thread. This data is divided into an optional *Constant URB Entry* (CURBE), following either by a Primitive URB Entry (WM) or a number of Vertex URB Entries (VS, GS, CLIP, SF). A FF unit only knows the location of this data in the URB, and is never exposed to the contents. For each URB entry, the FF unit will supply a sequence of handles, read offsets and read lengths. The thread dispatch subsystem will read the appropriate 256-bit locations of the URB, optionally perform swizzling (VS only), and write the results into sequential GRF registers (starting at **Dispatch GRF Start Register for URB Data**).

State Variables Controlling Payload URB Data

State Variable	Usage	FFs
Dispatch GRF Start Register for URB Data	This SV identifies the starting GRF register receiving payload URB data. Software is responsible for ensuring that URB data does not overwrite the Fixed or Extended Header portions of the payload.	FFs spawning threads
Vertex URB Entry Read Offset	This SV specifies the starting offset within VUEs from which vertex data is to be read and supplied in this stage's payloads. It is specified as a 256-bit offset into any and all VUEs passed in the payload. This SV can be used to skip over leading data in VUEs that is not required by the stage's threads (e.g., skipping over the Vertex Header data at the SF stage, as that information is not required for setup calculations). Skipping over irrelevant data	VS, GS

State Variable	Usage	FFs
	can only help to improve performance. Specifying a vertex data source extending beyond the end of a vertex entry is UNDEFINED.	
Vertex URB Entry Read Length	This SV determines the amount of vertex data (starting at **Vertex URB Entry Read Offset**) to be read from each VUEs and passed into the payload URB data. It is specified in 256-bit units. A zero value is INVALID (at very least one 256-bit unit must be read). Specifying a vertex data source extending beyond the end of a VUE is UNDEFINED.	

Programming Restrictions: (others may already been mentioned)

- The maximum size payload for any thread is limited by the number of GRF registers available to the thread, as determined by min(128, 16 * **GRF Register Block Count**). Software is responsible for ensuring this maximum size is not exceeded, taking into account:
  - The size of the Fixed and Extended Payload Header associated with the FF unit.
  - The **Dispatch GRF Start Register for URB** Data SV.
  - The amount of CURBE data included (via **Constant URB Entry Read Length**)
  - The number of VUEs included (as a function of FF unit, it's state programming, and incoming primitive types)
  - The amount of VUE data included for each vertex (via **Vertex URB Entry Read Length**)
  - (For WM-spawned PS threads) The amount of Primitive URB Entry data.
- For any type of URB Entry reads:
  - Specifying a source region (via Read Offset, Read Length) that goes past the end of the URB Entry allocation is illegal.
    - The allocated size of Vertex/Primitive URB Entries is determined by the **URB Entry Allocation Size** value provided in the pipeline state descriptor of the FF unit owning the VUE/PUE.
    - The allocated size of CURBE entries is determined by the **URB Entry Allocation Size** value provided in the CS_URB_STATE command.

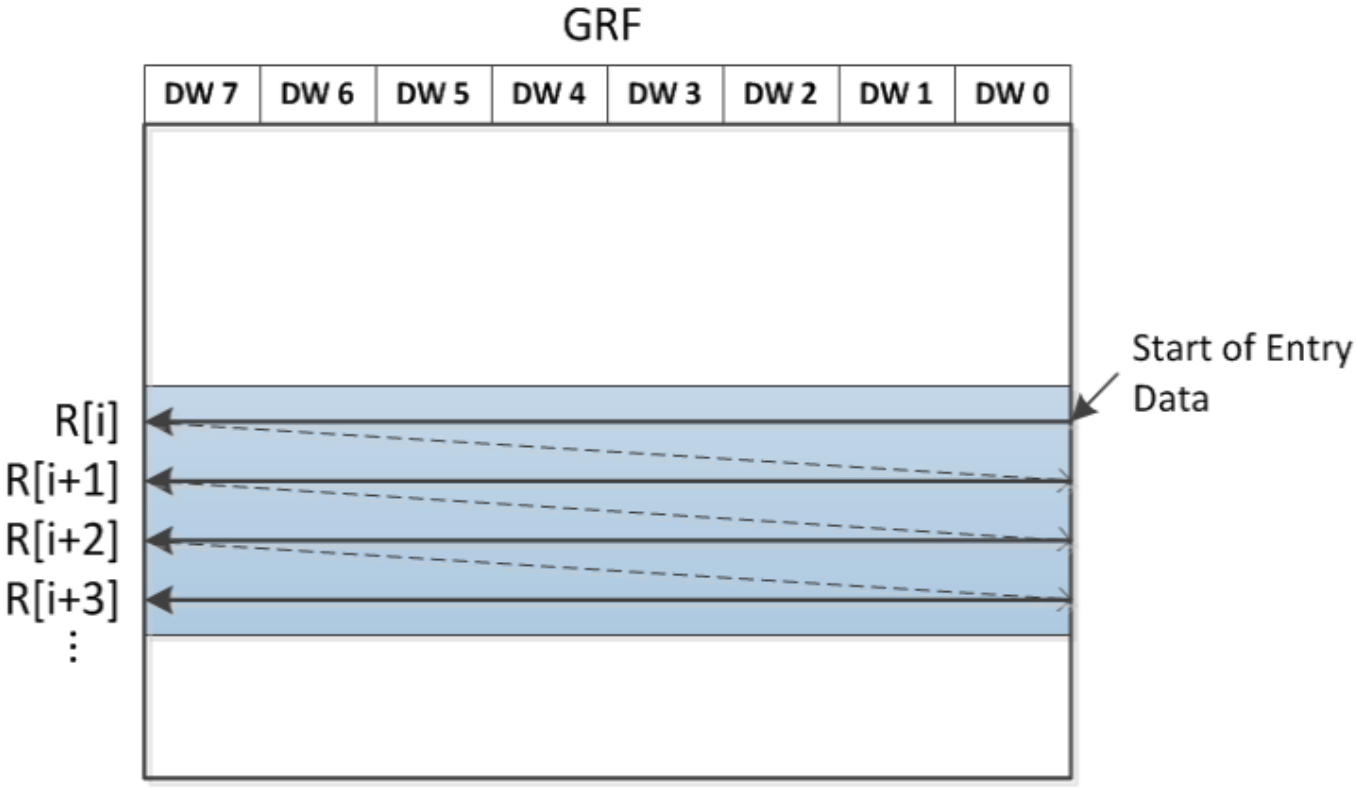
Payload URB Data Layouts

Before going into more detail about URB-sourced payload contents, it is important to discuss the three basic layouts of this data: Linear, SIMD4x2 Interleaved, and SIMD8. These layouts are linked to how data can be accessed by the EU (therefore the EU documentation should be comprehended).

Linear

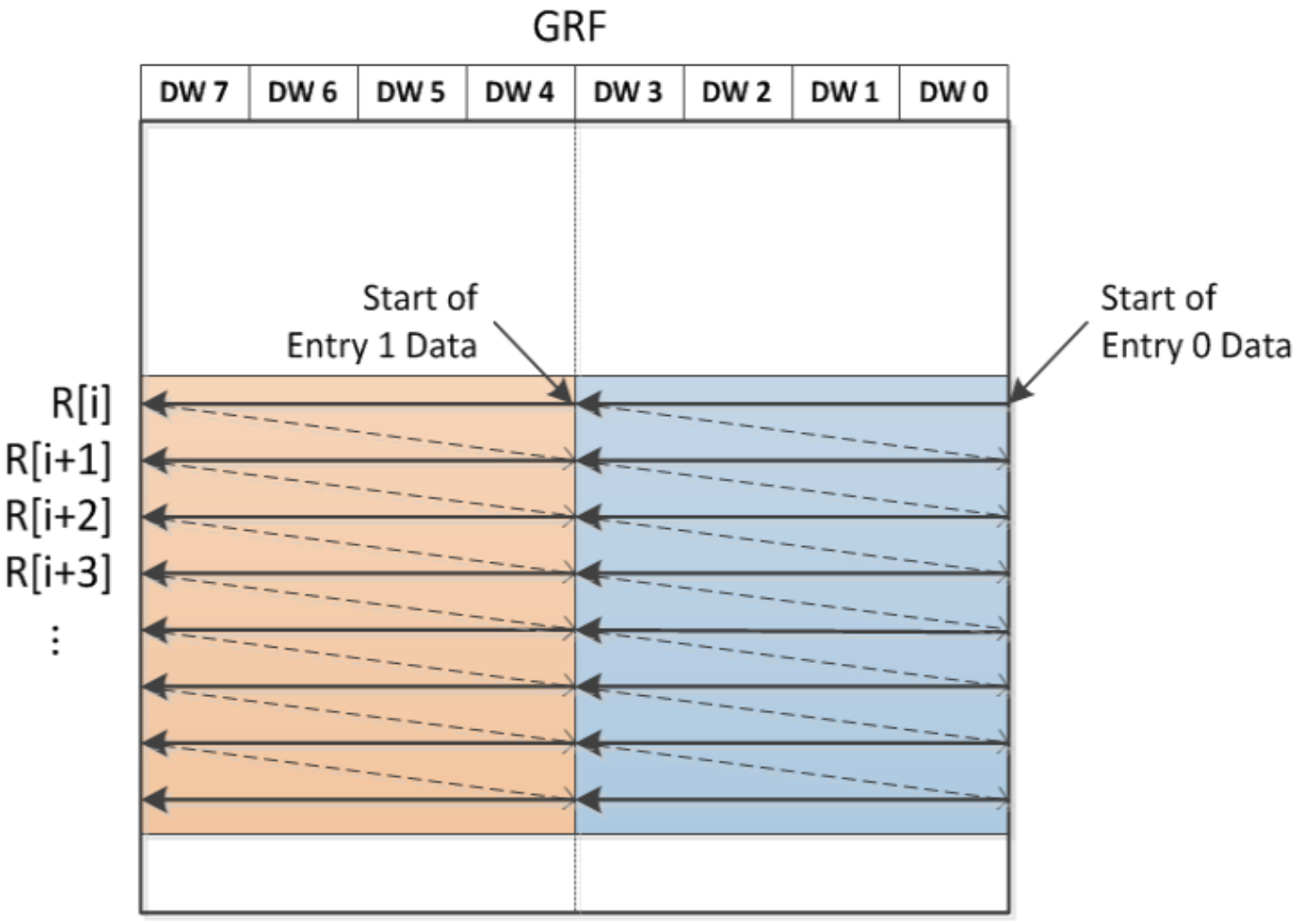
In Linear layout, data is read from the URB and placed in successive GRFs starting at DW0 of the starting destination GRF, as shown below. Data in this layout can be accessed by all EU channels of execution and

it is therefore used to hold "constant" data as well as patch data for DS dispatch modes that work on a single patch at a time.



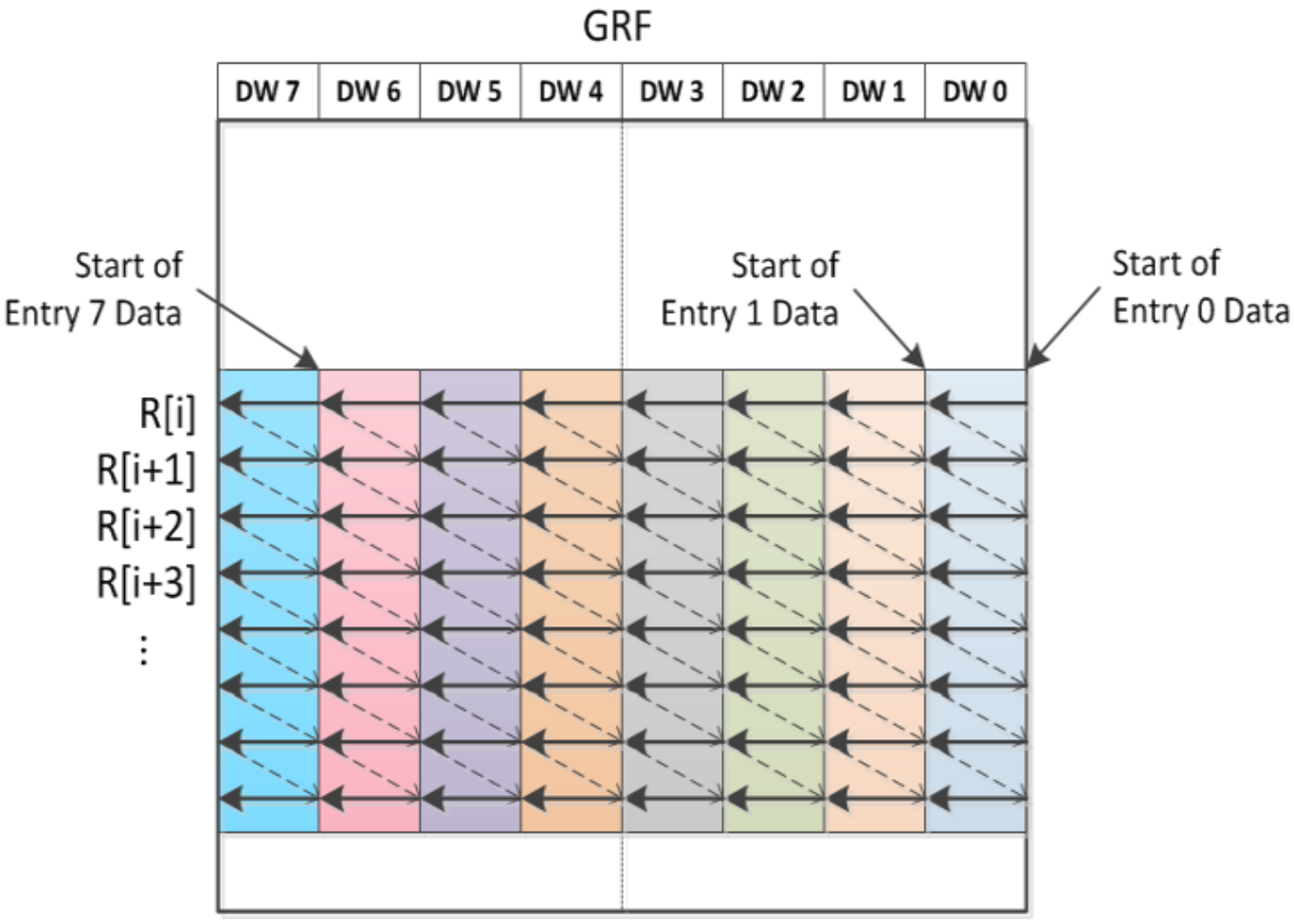
SIMD4x2 Interleaved

In SIMD4x2 Interleaved layout, the GRFs receive data from two URB entries, with the "first" URB entry loaded into the 4 lower DWs of the GRFs and the "second" URB entry loaded into the upper DWs, as shown below. This layout is primarily used to accommodate a kernel executing in SIMD4x2 execution mode (see EU documentation). It is also used to pass data from two input patches into a DS DUAL_PATCH payload, where the kernel may be executing in SIMD8 mode, but with the lower 4 SIMD8 channels operating on one patch and the upper 4 SIMD8 channels operating on another patch.



SIMD8

In SIMD8 layout, each DW position of the target GRFs can receive data from a different URB entry, as shown below. (Note that it may be possible for the data from one source URB entry to be replicated in two or more channels). This layout is used for kernels executing in SIMD8 mode, where each channel operates on independent data.



Payload URB Data

In most Geometry FF thread payloads some amount of URB-sourced data is required as input to the thread. This data is comprised of an optional amount of Push Constant data, immediately followed by an optional amount of URB Entry data (vertex or patch data).

The starting GRF# of the Payload URB Data section is specified by the **DispatchGRFStartRegisterForURBData** per-FF state variable (programmed via 3DSTATE_<FF>). See URB Payload State above for more information on the state variables that affect the Payload URB Data.

Push Constant Data

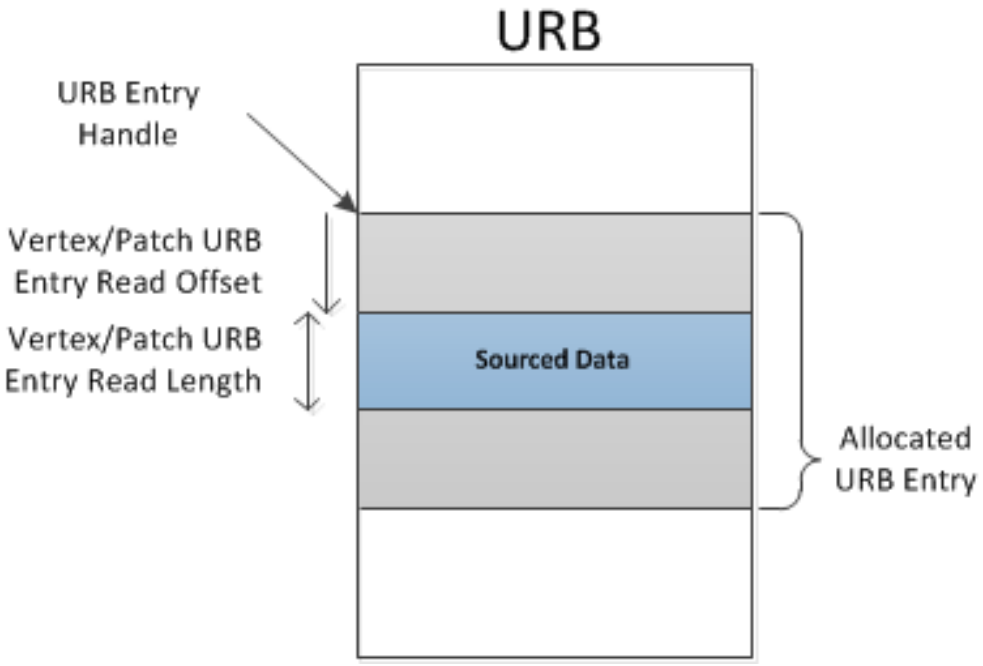
This section of the Payload URB Data is used to pass Push Constant data to the thread kernel. Software can define up to 4 Push Constant Memory Buffer regions for each Geometry FF stage that requests thread dispatches, afterwhich the contents of those memory regions are automatically included in each subsequent payload relevant to the FF stage. A FF stage-specific Push Constant URB Buffer is used to buffer the memory contents, though any padding required by the URB buffer is removed before the data is placed in the payload. See *Push Constant Programming*.

URB Entry Data

All Geometry FF threads have some number of (vertex or patch) Input URB Entries that serve as input to the thread. Some amount (and possibly all) of the data from those Input URB Entries can be pushed into the thread payload for immediate use by the thread's kernel. While the number of Input URB Entries associated with a thread is only indirectly controlled by software (e.g., via Dispatch Mode), the source region within each of the Input URB Entries is directly programmed. This source region definition applies to all of the Input URB Entries pushed into the payload.

The diagram below shows how the **Vertex/Patch URB Entry Read Offset** and **Length** states are used to define the source region of a URB Entry that will be copied into the URB Entry Data area of the thread payload.

Input URB Entry Source Region Definition



The number of Input URB Entries pushed and the layout of the payload data is described in the relevant FF stage descriptions.

Push Constant Programming Overview

Push constants are constant values that are pushed as part of the thread payload. Pushing constants allow for the data to be available to the Execution Units as soon as the thread payload is loaded in the GRF. The alternative to push constants are kernel-fetched constants.

All shaders (VS, HS, DS, GS and PS) have a section of the thread payload for constant data. For the geometry shaders, this is inserted between the R headers and URB Vertex Data. For Pixel Shaders, this is inserted prior to Setup Data. For more information, see the detailed descriptions of each Shader's payload in the corresponding sections.

Below is the format for the constant portion of the thread payload:

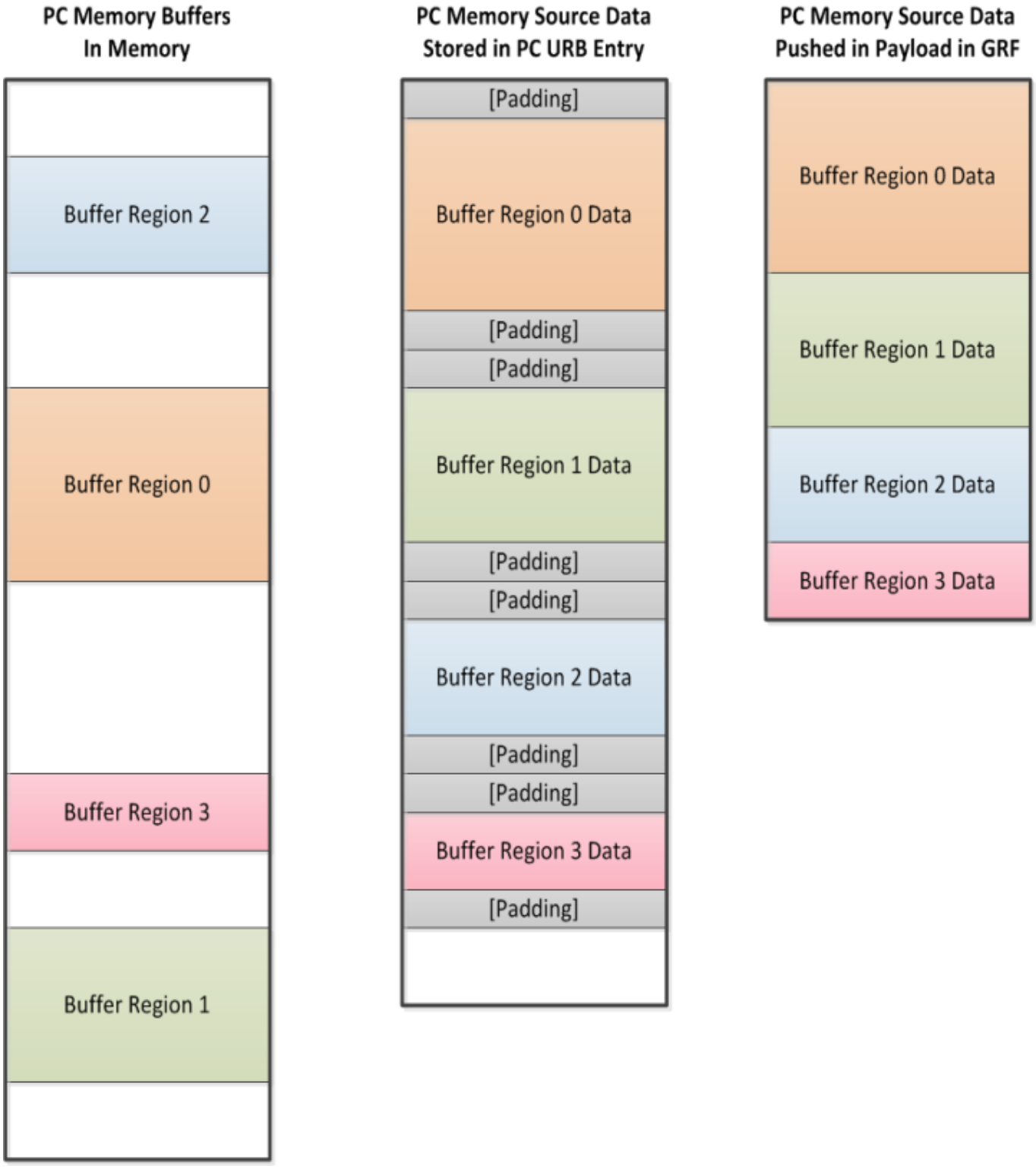
Rn		Registers prior to Push Constants
[Varies] optional	255:0	**Indirect Push Constants:** Push Constant data indirectly fetched from memory based on the 3DSTATE_CONSTANT_* command and read from the URB. The amount of data provided is defined by the sum of the read lengths in the last 3DSTATE_CONSTANT_*command
Data		Vertex or Setup Data

Indirect Push Constant Programming

Note: The Resource Streamer-based "Gather Constant" function is an extension of the Push Constant function and is described in detail in the Resource Streamer section. The Geometry FF state programming aspects are included below, after the basic Push Constant function is described.

3D APIs and their associated shader languages support the access of constant values, typically sourced from memory-resident Constant Buffers. Additionally, shader kernels may require access to compiler and/or driver-generated constants. The device supports a basic Push Constant (PC) mechanism to have a limited amount of constant data to be pushed into GRF registers via the thread payload where they are immediately available to the kernel program. It is up to software to determine which constants (if any) are pushed into the payload versus being dynamically referenced from memory via a shared function. Besides functional restrictions, there are several performance tradeoffs involved in this decision: GRF register pressure, locality of constant references, multiple references, expected shared function latency, etc.

The device supports a basic mechanism where software can specify -- for each FF stage that generates thread requests - up to 4 memory regions as the source for the PC data and one URB allocation used to buffer the data internally to the device. The device will fetch the PC source data from memory and write it into the URB Buffer, and at thread dispatch time the PC data will be read from the URB and inserted into the thread payload GRF registers.



Push Constant Memory Buffers

The 3DSTATE_CONSTANT_<FF> commands specify a set of state variables that define up to 4 PC Memory Buffer regions in memory. The commands also initiate the process of reading the PC source

data (if any) from memory and placing it in the associated PC URB Buffer for inclusion in subsequent thread payloads.

Up to four PC Memory Buffers can be specified. **ConstantBufferReadLength** specifies a 32B-granular amount of PC data residing in the PC Memory Buffer. A length of 0 disables the corresponding buffer. Disabling all four buffers causes no PC data to be inserted in thread payloads. SW shall disable all four buffers whenever the corresponding PC URB Buffer is disabled. If SW disables a buffer, it shall also specify a Pointer value of 0.

The location of a PC Memory Buffer is specified either by:

- 32-Byte granular GraphicsAddress
- 32-Byte granular DynamicStateOffset from the **DynamicStateBaseAddress GraphicsAddress** (programmed via STATE_BASE_ADDRESS)

The **GatherPoolEnable** state bit (programmed via 3DSTATE_GATHER_POOL_ALLOC) is used to enable option (c) for Buffer 1 only. Otherwise, the **CONSTANT_BUFFERAddressOffsetDisable** bit of the INSTPM register controls the use of Pointer state variables:

* If the buffers are specified via a DynamicStateOffset, the **DynamicStateMemoryObjectControlState** (programmed via STATE_BASE_ADDRESS) is used and corresponding DynamicState bounds checking is performed during the memory access.

* If the buffers are specified via a GraphicsAddress, the **ConstantBufferObjectControlState** state variable is used to control the memory accesses, though no bounds checking is performed.

Note that the starting location and length of the PC source data in each PC Memory Buffer is specified via 32B-aligned/granular parameters, while the PC URB Buffer is specified via 64B-aligned/granular parameters. The implications of this are described in the PC URB Buffer description.

State	Command
Constant Buffer Object Control State	3DSTATE_CONSTANT_<FF>
Constant Buffer Read Length [0-3]	3DSTATE_CONSTANT_<FF>
Pointer to Constant Buffer [0-3]	3DSTATE_CONSTANT_<FF>
Pointer to Constant Buffer High [0-3]	3DSTATE_CONSTANT_<FF>

Push Constant URB Buffer Allocation

The 3DSTATE_PUSH_CONSTANT_ALLOC_<FF> commands specify a set of state variables is used to define the PC URB Buffer allocation for each relevant FF stage. Each buffer is used to collect 64B-granular/aligned PC source data prior to use in thread dispatch.

State	Command
Constant Buffer Offset	3DSTATE_PUSH_CONSTANT_ALLOC_<FF>
Constant Buffer Size	3DSTATE_PUSH_CONSTANT_ALLOC_<FF>

ConstantBufferOffset specifies the 2KB-granular offset of a FF stage's PC URB Buffer allocation. If the ConstantBufferSize is zero, this offset is ignored.

ConstantBufferSize specifies the size of a FF stage's PC URB Buffer allocation as a possibly-zero count of 2KB increments. Specifying a size of 0 disables the buffer. It is invalid to specify a non-zero amount of PC source data (via 3DSTATE_CONSTANT_<FF>) when the corresponding PC URB Buffer is disabled.

In order to use PCs for a FF stage, SW shall first program **ConstantBufferSize** to a non-zero value. The buffer shall be large enough to accommodate the worst-case buffering requirements of any single set of PC Memory Buffer definitions (see below). It is invalid to specify more PC source data than can be accommodated in the allocated PC URB Buffer. Additionally, in order to allow the device to pipeline the prefetching of subsequent PC Memory Buffers, it is recommended that SW allocate PC URB Buffers larger than this minimum requirement.

A PC URB Buffer is used to buffer 64B-granular/aligned push constant data from memory, though the PC memory regions are defined as 32B-granular/aligned. In order to accommodate the worst-case alignment, where a specific PC memory region is not 64B aligned but is 64B granular in size, the PC URB Buffer requires 32B of padding at both the beginning and end of the PC data and would therefore need to be sized at least 64B larger than the source data region wrt that source buffer. If this condition holds for all 4 PC source buffers, the PC URB Buffer needs to be sized 256B larger than the worst-case amount of source data. If SW knows a priori that the PC source data is 64B-aligned/granular, then there is no need to allocate additional room for 64B padding.

An important example of this PC URB Buffer sizing restriction is with respect to supporting a maximum amount of PC source data. The per-FF limit on the amount of PC data that can be specified for inclusion in thread payloads at any given time is 2KB (64 * 32B) spread across up to 4 source buffers. Here minimum-sized (2KB) PC URB Buffer could only be used if all the source data was 64B aligned and 64B granular is size, as the PC URB Buffer would have no room for padding. If any 64B padding was required, (at least) a 4KB PC URB Buffer would need to be allocated.

Push Constant URB Buffer Placement: SW shall program all Push Constant URB Buffer allocations to be either disabled or completely contained within the first 32KB of the URB. There are no ordering requirements on the placement of the allocations relative to the particular FF stages (e.g., the VS allocation can come before or after the GS allocation). SW shall not program enabled buffers to overlap. If 32KB is greater than the amount of URB space required for all the Push Constant URB Buffers and SW packs the allocations starting at offset 0, SW can utilize the URB space after the last allocation for URB Entry allocations (e.g., VS VUEs), subject to URB Fence alignment restrictions.

3D Primitives Overview

The 3DPRIMITIVE command (defined in the VF Stage chapter) is used to submit 3D primitives to be processed by the 3D pipeline. Typically, the processing results in the rendering of pixel data into the render targets, but this is not required.

There is considerable confusion surrounding the term 'primitive', e.g., is a triangle strip a 'primitive', or is a triangle within a triangle strip a 'primitive'? Some APIs use the term 'topology' to describe the higher-level construct (e.g., a triangle strip), and uses the term 'primitive' when discussing a triangle within a triangle strip. In this spec, we will try to avoid ambiguity by using the term 'object' to represent the basic

shapes (point, line, triangle), and 'topology' to represent input geometry (strips, lists, etc.). Unfortunately, terms like '3DPRIMITIVE' must remain for legacy reasons.

The following table describes the basic primitive topology types supported in the 3D pipeline.

Programming Note	
Context:	3D Primitives Overview
• There are several variants of the basic topologies. These have been introduced to allow slight variations in behavior without requiring a state change. • Number of vertices and Dangling Vertices: Topologies have an "expected" number of vertices in order to form complete objects within the topologies (e.g., LINELIST is expected to have an even number of vertices). The actual number of vertices specified in the 3DPRIMITIVE command, and as output from the GS unit, is allowed to deviate from this expected number, in which case any "dangling" vertices are discarded. The removal of dangling vertices is initially performed in the VF unit. To filter out dangling vertices emitted by GS threads, the CLIP unit also performs dangling-vertex removal at its input.	

3D Primitive Topology Types

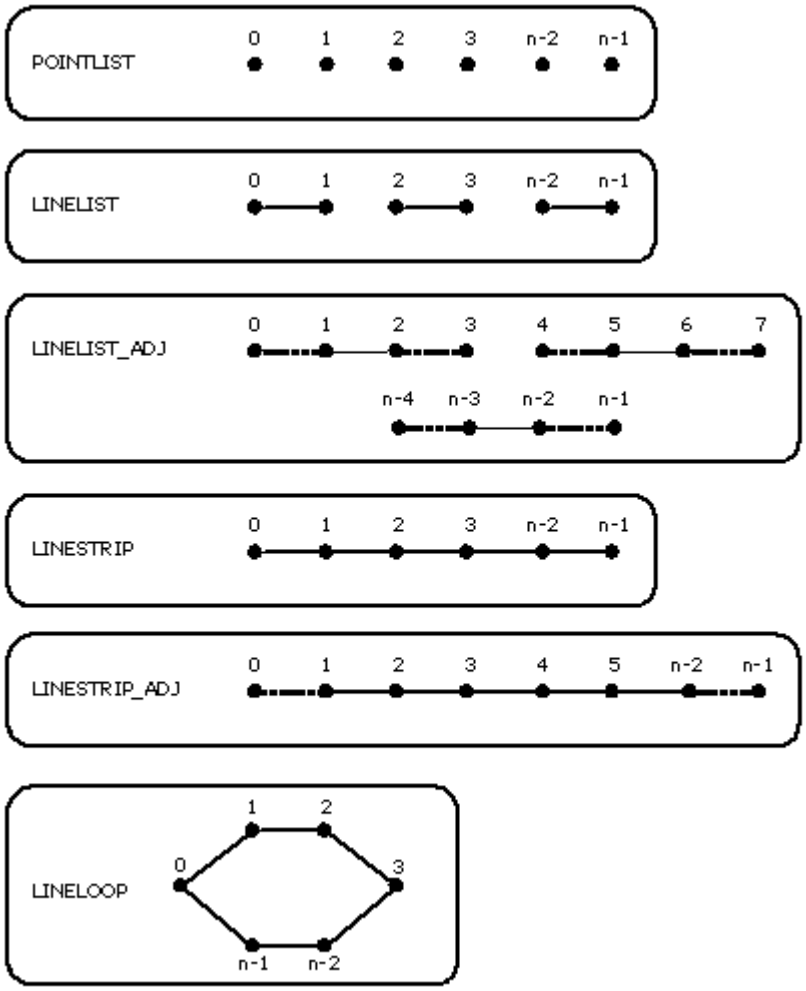
3D Primitive Topology Type (ordered alphabetically)	Description
LINELIST	• A list of independent line objects (2 vertices per line). • Normal usage expects a multiple of 2 vertices, though incomplete objects are silently ignored.
LINELIST_ADJ	• A list of independent line objects with adjacency information (4 vertices per line). • Normal usage expects a multiple of 4 vertices, though incomplete objects are silently ignored. • Not valid as output from GS thread.
LINELOOP	• Similar to a 3DPRIM_LINESTRIP, though the last vertex is connected back to the initial vertex via a line object. The LINELOOP topology is converted to LINESTRIP topology at the beginning of the 3D pipeline. • Normal usage expects at least 2 vertices, though incomplete objects are silently ignored. (The 2-vertex case is required by OGL). • Not valid after the GS stage (i.e., must be converted by a GS thread to some other primitive type).
LINESTRIP	• A list of vertices connected such that, after the first vertex, each additional vertex is associated with the previous vertex to define a connected line object. • Normal usage expects at least 2 vertices, though incomplete objects are silently ignored.
LINESTRIP_ADJ	• A list of vertices connected such that, after the first vertex, each additional

3D Primitive Topology Type (ordered alphabetically)	Description
	vertex is associated with the previous vertex to define connected line object. The first and last segments are adjacent-only vertices. • Normal usage expects at least 4 vertices, though incomplete objects are silently ignored. • Not valid as output from GS thread.
LINESTRIP_BF	• Similar to LINESTRIP, except treated as "backfacing' during rasterization (stencil test). This is used by HW to support "line" polygon fill mode when two-sided stencil is enabled.
LINESTRIP_CONT	• Similar to LINESTRIP, except LineStipple (if enabled) is continued (vs. reset) at the start of the primitive topology. • This can be used to support line stipple when the API-provided primitive is split across multiple topologies.
LINESTRIP_CONT_BF	Combination of LINESTRIP_BF and LINESTRIP_CONT variations.
POINTLIST	A list of point objects (1 vertex per point).
POINTLIST_BF	• Similar to POINTLIST, except treated as "backfacing' during rasterization (stencil test). This is used to support "point" polygon fill mode when two-sided stencil is enabled.
POLYGON	• Similar to TRIFAN, though the first vertex always provides the "flat-shaded" values (vs. this being programmable through state). • Normal usage expects at least 3 vertices, though incomplete objects are silently ignored.
QUADLIST	• A list of independent quad objects (4 vertices per quad). • The QUADLIST topology is converted to POLYGON topology at the beginning of the 3D pipeline. • Normal usage expects a multiple of 4 vertices, though incomplete objects are silently ignored.
QUADSTRIP	• A list of vertices connected such that, after the first two vertices, each additional pair of vertices are associated with the previous two vertices to define a connected quad object. • Normal usage expects an even number (4 or greater) of vertices, though incomplete objects are silently ignored.
RECTLIST	• A list of independent rectangles, where only 3 vertices are provided per rectangle object, with the fourth vertex implied by the definition of a rectangle. V0=LowerRight, V1=LowerLeft, V2=UpperLeft. Implied V3 = V0-V1+V2.

3D Primitive Topology Type (ordered alphabetically)	Description
	• Normal usage expects a multiple of 3 vertices, though incomplete objects are silently ignored. The RECTLIST primitive is supported specifically for 2D operations (e.g., BLTs and "stretch" BLTs) and not as a general 3D primitive. Due to this, a number of restrictions apply to the use of RECTLIST: • Must utilize "screen space" coordinates (VPOS_SCREENSPACE) when the primitive reaches the CLIP stage. The W component of position must be 1.0 for all vertices. The 3 vertices of each object should specify a screen-aligned rectangle (after the implied vertex is computed). • Clipping: Must not require clipping or rely on the CLIP unit's ClipTest logic to determine if clipping is required. Either the CLIP unit should be DISABLED, or the CLIP unit's Clip Mode should be set to a value other than CLIPMODE_NORMAL. • Viewport Mapping must be DISABLED (as is typical with the use of screen-space coordinates).
RECTLIST_SUBPIXEL	The subpixel precise, axis-aligned bounding box of the object's 3 vertices is rendered.
TRIFAN	• Triangle objects arranged in a fan (or polygon). The initial vertex is maintained as a common vertex. After the second vertex, each additional vertex is associated with the previous vertex and the common vertex to define a connected triangle object. • Normal usage expects at least 3 vertices, though incomplete objects are silently ignored.
TRIFAN_NOSTIPPLE	• Similar to TRIFAN, but poylgon stipple is not applied (even if enabled). • This can be used to support "point" polygon fill mode, under the combination of the following conditions: (a) when the frontfacing and backfacing polygon fill modes are different (so the final fill mode is not known to the driver), (b) one of the fill modes is "point" and the other is "solid", (c) point mode is being emulated by converting the point into a trifan, (d) polygon stipple is enabled. In this case, polygon stipple should not be applied to the points-emulated-as-trifans.
TRILIST	• A list of independent triangle objects (3 vertices per triangle). • Normal usage expects a multiple of 3 vertices, though incomplete objects are silently ignored.
TRILIST_ADJ	• A list of independent triangle objects with adjacency information (6 vertices per

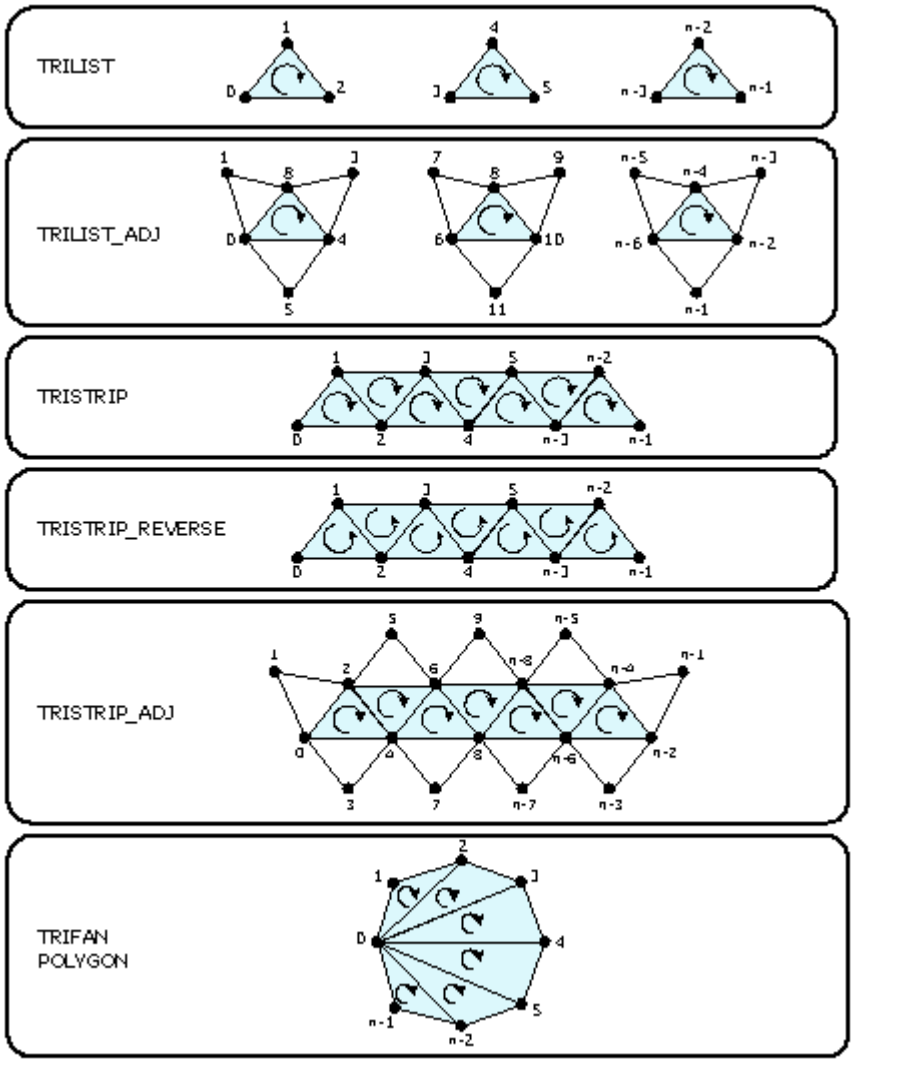
3D Primitive Topology Type (ordered alphabetically)	Description
	triangle). • Normal usage expects a multiple of 6 vertices, though incomplete objects are silently ignored. • Not valid as output from GS thread.
TRISTRIP	• A list of vertices connected such that, after the first two vertices, each additional vertex is associated with the last two vertices to define a connected triangle object. • Normal usage expects at least 3 vertices, though incomplete objects are silently ignored.
TRISTRIP_ADJ	• A list of vertices where the even-numbered (including 0th) vertices are connected such that, after the first two vertex pairs, each additional even-numbered vertex is associated with the last two even-numbered vertices to define a connected triangle object. The odd-numbered vertices are adjacent-only vertices. • VFUNIT will complete a drawcall with the topology of tristrip_adj even if there is a preemption request in the middle of the draw call. • Normal usage expects at least 6 vertices, though incomplete objects are silently ignored. • Not valid as output from GS thread.
TRISTRIP_REVERSE	Similar to TRISTRIP, though the sense of orientation (winding order) is reversed - this allows SW to break long tristrips into smaller pieces and still maintain correct face orientations.
PATCHLIST_n	List of n-vertex "patch" objects. These topologies cannot be rendered directly - the tessellation units must be used to convert them into points, lines, or triangles to produce rasterization results. (VS, GS, and StreamOutput operations can also be performed.)

The following diagrams illustrate the basic 3D primitive topologies. (Variants are not shown if they have the same definition with respect to the information provided in the diagrams).

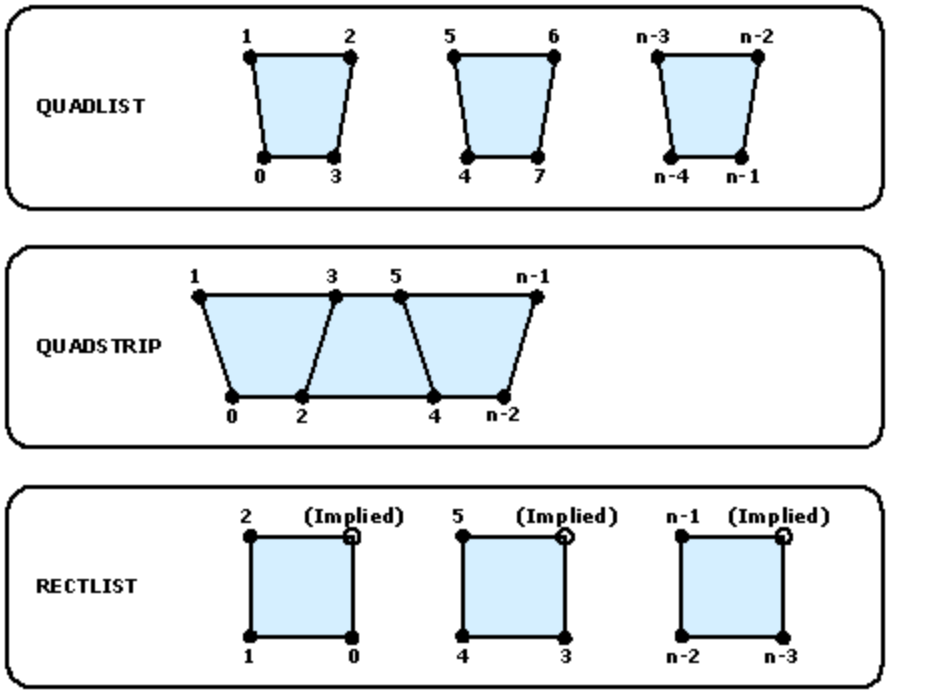


B6815-01

A note on the arrows you see below: These arrows are intended to show the vertex ordering of triangles that are to be considered having "clockwise" winding order in screen space. Effectively, the arrows show the order in which vertices are used in the cross-product (area, determinant) computation. Note that for TRISTRIP, this requires that either the order of odd-numbered triangles be reversed in the cross-product or the sign of the result of the normally-ordered cross-product be flipped (these are identical operations).


TRILIST
TRILIST_ADJ
TRISTRIP
TRISTRIP_REVERSE
TRISTRIP_ADJ
TRIFAN
POLYGON

B6816-01



Vertex Data Overview

The 3D pipeline FF stages (past VF) receive input 3D primitives as a stream of vertex information packets. (These packets are not directly visible to software.) Much of the data associated with a vertex is passed indirectly via a VUE handle. The information provided in vertex packets includes:

- The **URB Handle** of the VUE: This is used by the FF unit to refer to the VUE and perform any required operations on it (e.g., cause it to be read into the thread payload, dereference it, etc.).
- **Primitive Topology Information:** This information is used to identify/delineate primitive topologies in the 3D pipeline. Initially, the VF unit supplies this information, which then passes through the VS stage unchanged. GS and CLIP threads must supply this information with each vertex they produce (via the URB_WRITE message). If a FF unit directly outputs vertices (that were not generated by a thread they spawned), that FF unit is responsible for providing this information.
  - **PrimType:** The type of topology, as defined by the corresponding field of the 3DPRIMITIVE command.
  - **StartPrim:** TRUE only for the first vertex of a topology.
  - **EndPrim:** TRUE only for the last vertex of a topology.
- (Possibly, depending on FF unit) Data read back from the **Vertex Header** of the VUE.

Vertex URB Entry (VUE) Formats

In general, vertex data is stored in Vertex URB Entries (VUEs) in the URB, and only referenced by the pipeline stages indirectly via VUE handles. Therefore (for the most part) the contents/format of the vertex

data is not exposed to 3D pipeline hardware - the FF units are typically only aware of the handles and sizes of VUEs.

VUEs are written in two ways:

- At the top of the 3D Geometry pipeline, the VF's InputAssembly function creates VUEs and initializes them from data extracted from Vertex Buffers as well as internally-generated data.
- VS, GS, HS and DS threads can compute, format, and write new VUEs as thread output.

There are only a few points in the 3D FF pipeline where the FF units are exposed to the VUE data. Otherwise the VUE remains opaque to the 3D pipeline hardware.

- TE stage reads back Patch Headers from Patch URB Entries
- GS stage (optionally) reads back VertexCounts and Control Data Headers from GS VUEs
- StreamOutput stage reads back VUE contents in order to stream the vertices out
- Clip stage reads back VertexHeaders from VUEs

Software must ensure that any VUEs subject to readback by the 3D pipeline start with a valid Vertex Header. This extends to all VUEs with the following exceptions:

- If the VS function is enabled, the VF-written VUEs are not required to have Vertex Headers, as the VS-incoming vertices are guaranteed to be consumed by the VS (i.e., the VS thread is responsible for overwriting the input vertex data).
- If the GS FF is enabled, neither VF-written VUEs nor VS thread-generated VUEs are required to have Vertex Headers, as the GS will consume all incoming vertices.
- If Rendering is disabled, VertexHeaders are not required anywhere.

The following table defines the Vertex Header. The Position fields are described in further detail below.

VUE Vertex Header

DWord	Bits	Description
D0	31:0	Reserved: MBZ
D1	31:0	**Render Target Array Index (RTAIndex).** This value is (eventually) used to index into a specific element of an "array" Render Target. It is read back by the GS unit (for all exiting vertices) and the Clip unit (for all clip-generated vertices), subsequently routed into the PS thread payload, and eventually included in the RTWrite DataPort message header for use by the DataPort shared function. Software is responsible for ensuring this field is zero whenever a programmable index value is not required. When a programmable index value is required , software must ensure that the correct 11-bit value is written to this field. Specifically, the kernels must perform a reange check of computed index values against [0,2047], and output zero if that range is exceeded. Note that the unmodified "renderTargetArrayIndex" must be maintained in the VUE outside of the Vertex Header. Software can force an RTAIndex of 0 to be used (effectively ignoring the setting of this

DWord	Bits	Description
		DWord) by use of the **ForceZeroRTAIndex** bit (3DSTATE_CLIP). Otherwise the read-back value will be used to select an RTArray element, after being clamped to the RTArray surface's [**MinimumArrayElement, Depth**] range (SURFACE_STATE). Format: 0-based U32 index value
D2	31:0	**Viewport Index.** This value is used to select one of a possible 16 sets of viewport (VP) state parameters in the Clip unit's VertexClipTest function and in the SF unit's ViewportMapping and Scissor functions. The Clip unit (if enabled) will read back this value. The Clip unit will range-check the value against [0,**Maximum VPIndex**] (see 3DSTATE_CLIP). Software can force a value of 0 to be used by programming **Maximum VPIndex** to 0. Format: 0-based U32 index value
D3	31:0	**Point Width.** This field specifies the width of POINT objects in screen-space pixels. It is used only for vertices within POINTLIST and POINTLIST_BF primitive topologies, and is ignored for vertices associated with other primitive topologies. This field is read back by the Clip unit. Format: FLOAT32
D4	31:0	**Vertex Position 0 X Coordinate.** This field contains the X component of the vertex's first 4D space position. Format: FLOAT32
D5	31:0	**Vertex Position 0 Y Coordinate.** This field contains the Y component of the vertex's first 4D space position. Format: FLOAT32
D6	31:0	**Vertex Position 0 Z Coordinate.** This field contains the Z component of the vertex's first NDC space position. Format: FLOAT32
D7	31:0	**Vertex Position 0 W Coordinate.** This field contains the Z component of the vertex's first 4D space position. Format: FLOAT32
If (N>0): D8 thru D[8+(4*(N-1))+3]	31:0	**Vertex Position [1].XYZW** through **Vertex Position [N].XYZW**. These fields provide, in the Render pipeline, "replica" (i.e., in addition to Vertex Position 0) 4D Vertex Positions for the vertex if the **Primitive Replication** feature is enabled via 3DSTATE_PRIMITIVE_REPLICATION. Here "**N**" refers to the value programmed in 3DSTATE_PRIMITIVE_REPLICATION::**ReplicationCount**. If N=0, the Primitive Replication feature is disabled, and therefore there shall be no additional replica Vertex Positions, i.e., only

DWord	Bits	Description
		Vertex Position 0.XYWZ shall appear in the Vertex Header with the ClipDistance values (if any) immediately following Vertex Position 0. If N>0, there shall be additional replica Vertex Positions included following Vertex Position 0, and the DWord offsets of the Clip Distance Values (below, if any) shall be offset by the number of DWords required for the replica positions. **Programming Note** Replica Vertex Positions are supported in both the Render and POSH pipelines
D8	31:0	**ClipDistance 0 Value (optional).** If the **UserClipDistance Clip Test Enable Bitmask** bit (3DSTATE_CLIP) is set, this value will be read from the URB in the Clip stage. If the value is found to be less than 0 or a NaN, the vertex's UCF<0> bit will set in the Clip unit's VertexClipTest function. If the **UserClipDistance Clip Test Enable Bitmask** bit is clear, this value will not be read back, and the vertex's UCF<0> bit will be zero by definition. Format: FLOAT32 ClipDistance Values are enabled for clip/cull test in the Clip stage in one of two modes: Normally the corresponding Enable Bitmasks are obtained from the state programmed in the last "vertex-producing" stage (VS/DS/GS) that is enabled prior to the Clip stage. E.g., if VS and DS are enabled but GS is disabled, the masks are obtained from 3DSTATE_DS. Alternatively, the Enable Bitmasks can be obtained directly from corresponding masks programmed via 3DSTATE_CLIP, through use of 3DSTATE_CLIP's Force User Clip Distance [Cull/Clip] Test Enable Bitmask state bits (see description of 3DSTATE_CLIP).
D9	31:0	**ClipDistance 1 Value (optional).** See above.
D10	31:0	**ClipDistance 2 Value (optional).** See above.
D11	31:0	**ClipDistance 3 Value (optional).** See above.
D12	31:0	**ClipDistance 4 Value (optional).** See above.
D13	31:0	**ClipDistance 5 Value (optional).** See above.
D14	31:0	**ClipDistance 6 Value (optional).** See above.
D15	31:0	**ClipDistance 7 Value (optional).** See above.
	31:0	**End of Vertex Header Padding** (if required). The Vertex Header shall be padded at the and so that the header ends on a 32-byte boundary and therefore the Remainder of Vertex Elements (below) starts on a 32B boundary.
	31:0	**(Remainder of Vertex Elements).** The absolute maximum size limit on this data is specified via a maximum limit on the amount

DWord	Bits	Description
		of data that can be read from a VUE (including the Vertex Header) (**Vertex Entry URB Read Length** has a maximum value of 63 256-bit units). Therefore, the Remainder of Vertex Elements has an absolute maximum size of 62 256-bit units. Of course, the actual allocated size of the VUE can and will limit the amount of data in a VUE.

Vertex Positions

(For brevity, the following discussion uses the term map as a shorthand for "compute screen space coordinate via perspective divide followed by viewport transform".)

The "Position" fields of the Vertex Header are the only vertex position coordinates exposed to the 3D Pipeline. The CLIP and SF units are the only FF units which perform operations using these positions. The VUE will likely contain other position attributes for the vertex outside of the Vertex Header, though this information is not directly exposed to the FF units. For example, the Clip Space position will likely be required in the VUE (outside of the Vertex Header) to perform correct and robust 3D Clipping in the CLIP thread.

CLIP unit uses the **3DSTATE_CLIP.PerspectiveDivideDisable** bit to determine whether to perform a perspective projection (divide by w) of the read-back 4D Position.

When Perspective Divide is enabled, the Clip Space position is defined in a homogeneous 4D coordinate space (pre-perspective divide), where the visible "view volume" is defined by the APIs. The API's VS, GS or DS shader program will include geometric transforms in the computation of this clip space position such that the resulting coordinate is positioned properly in relation to the view volume (i.e., it will include a "view transform" in this computation path). When Perspective Divide is enabled, the 3D FF pipeline will perform a perspective projection (division of x,y,z by w), perform clip-test on the resulting NDC (Normalized Device Coordinates), and eventually perform viewport mapping (in the SF unit) to yield screen-space (pixel) coordinates.

When Perspective Divide is disabled, the read-back Position does not undergo perspective projection by the 3D FF pipeline.

Clip Space Position

The *clip-space* position of a vertex is defined in a homogeneous 4D coordinate space where, after perspective projection (division by W), the visible "view volume" is some canonical (3D) cuboid. Typically the X/Y extents of this cuboid are [-1,+1], while the Z extents are either [-1,+1] or [0,+1]. The API's VS or GS shader program will include geometric transforms in the computation of this clip space position such that the resulting coordinate is positioned properly in relation to the view volume (i.e., it will include a "view transform" in this computation path).

Note that, under typical perspective projections, the clip-space W coordinate is equal to the view-space Z coordinate.

A vertex's clip-space coordinates must be maintained in the VUE up to 3D clipping, as this clipping is performed in clip space.

Vertex clip-space positions must be included in the Vertex Header, so that they can be read-back (prior to Clipping) and then subjected to perspective projection (in hardware) and subsequent use by the FF pipeline.

NDC Space Position

A perspective divide operation performed on a clip-space position yields a [X,Y,Z,RHW] NDC (Normalized Device Coordinates) space position. Here "normalized" means that visible geometry is located within the [-1,+1] or [0,+1] extent view volume cuboid (see clip-space above).

- The NDC X,Y,Z coordinates are the clip-space X,Y,Z coordinates (respectively) divided by the clip-space W coordinate (or, more correctly, the clip-space X,Y,Z coordinates are multiplied by the reciprocal of the clip space W coordinate).
    - Note that the X,Y,Z coordinates may contain INFINITY or NaN values (see below).
- The NDC RHW coordinate is the reciprocal of the clip-space W coordinate and therefore, under normal perspective projections, it is the reciprocal of the view-space Z coordinate. Note that NDC space is really a 3D coordinate space, where this RHW coordinate is retained in order to perform perspective-correct interpolation, etal. Note that, under typical perspective projections.
    - Note that the RHW coordinate make contain an INFINITY or NaN value (see below).

Screen-Space Position

Screen-space coordinates are defined as:

- X,Y coordinates are in absolute screen space (pixel coordinates, upper left origin). See Vertex X,Y Clamping and Quantization in the SF section for a discussion of the limitations/restrictions placed on screenspace X,Y coordinates.
- Z coordinate has been mapped into the range used for DepthTest.
- RHW coordinate is actually the reciprocal of clip-space W coordinate (typically the reciprocal of the view-space Z coordinate).

Vertex Fetch (VF) Stage

The Vertex Fetch Stage performs one major function: executing 3DPRIMITIVE commands. This is handled by the VF's InputAssembly function.

The following subsections describe some high-level concepts associated with the VF stage:

- State
- 3D Primitive Command
- Functions

Vertex Buffers as a 2D Array

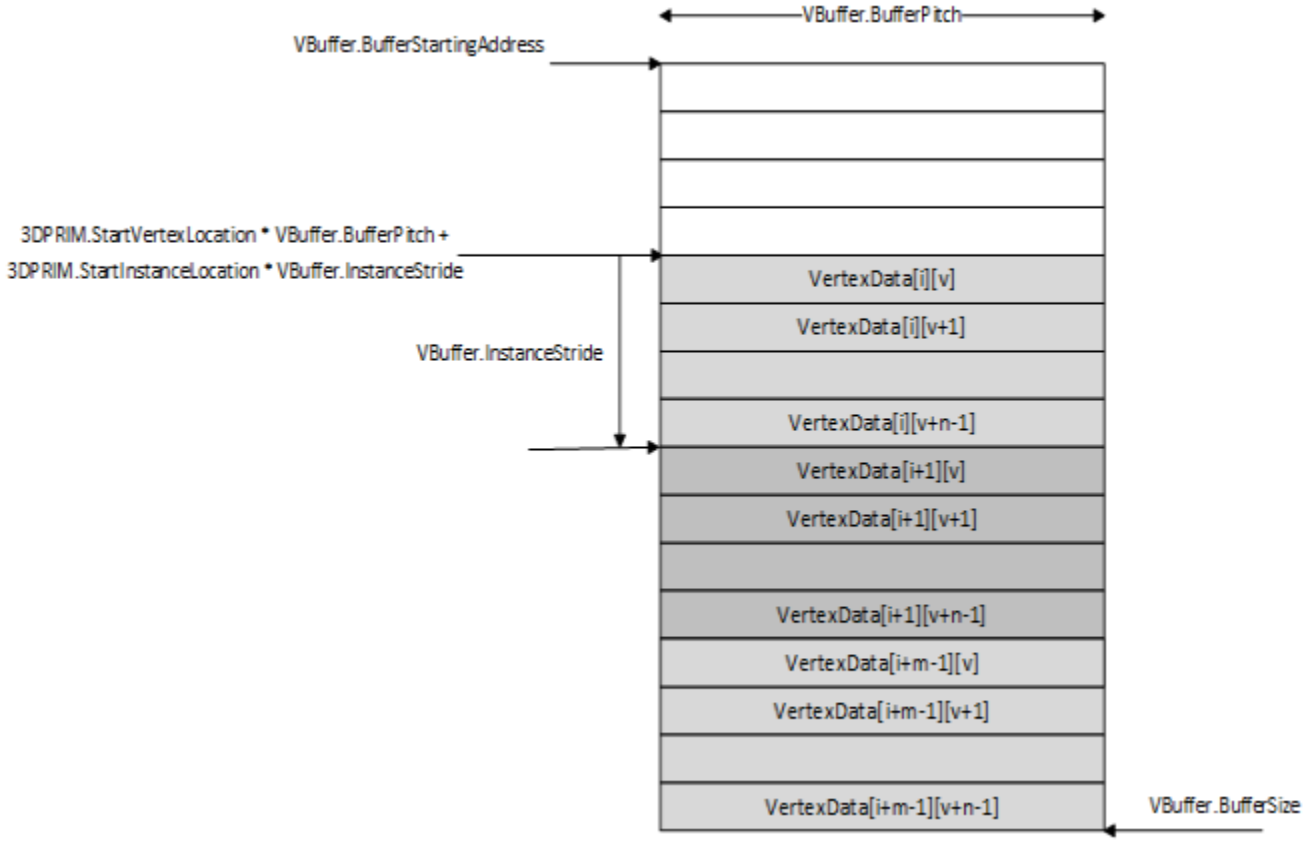
Vertex Buffers (VB) as a 2D Array

The Vertex Buffer can be accessed as a 2D array. The first dimension describes vertices where the size of the vertex is defined by the VB's **BufferPitch**. The second dimension describes instances where the size of the instance is defined by the **InstanceStride** in 3DSTATE_VF_INSTANCING.

Using the VB's as a 2D array is enabled by setting the 3DSTATE_VF_INSTANCING state **InstanceStrideEnable** to ENABLED. **InstanceStrideEnable** cannot be enabled at the same time as **InstancingEnable**.

The SGV InstanceID is used as the index for the second dimension. The state **InstanceIDOffsetEnable** can be used to provide an initial offset to InstanceID. This offset will apply to the InstanceID SGV itself and when InstanceID is used as an index for the second dimension of the VB 2D Array.

The **VertexAccessType** can be RANDOM or SEQUENTIAL when reading VB's as 2D arrays.



State

This section contains various state registers.

Control State

Register
3DSTATE_VF
3DSTATE_VF_TOPOLOGY

Index Buffer (IB) State

The 3DSTATE_INDEX_BUFFER command is used to define an *Index Buffer* (IB) used in subsequent 3DPRIMITIVE commands.

The RANDOM access mode of the 3DPRIMITIVE command involves the use of a memory-resident IB. The IB, defined via the 3DSTATE_INDEX_BUFFER command described below, contains a 1D array of 8, 16 or 32-bit index values. These index values will be fetched by the InputAssembly function, and subsequently used to compute locations in VERTEXDATA buffers from which the actual vertex data is to be fetched. (This is opposed to the SEQUENTIAL access mode where the vertex data is simply fetched sequentially from the buffers).

The following table lists which primitive topology types support the presence of Cut Indices.

Definition	Cut Index?
3DPRIM_POINTLIST	Y
3DPRIM_LINELIST	Y
3DPRIM_LINESTRIP	Y
3DPRIM_TRILIST	Y
3DPRIM_TRISTRIP	Y
3DPRIM_TRIFAN	Y
3DPRIM_QUADLIST	Y
3DPRIM_QUADSTRIP	Y
3DPRIM_LINELIST_ADJ	Y
3DPRIM_LINESTRIP_ADJ	Y
3DPRIM_TRILIST_ADJ	Y
3DPRIM_TRISTRIP_ADJ	Y
3DPRIM_TRISTRIP_REVERSE	Y
3DPRIM_POLYGON	Y
3DPRIM_RECTLIST	N
3DPRIM_LINELOOP	Y
3DPRIM_POINTLIST_BF	Y
3DPRIM_LINESTRIP_CONT	Y
3DPRIM_LINESTRIP_BF	Y
3DPRIM_LINESTRIP_CONT_BF	Y
3DPRIM_TRIFAN_NOSTIPPLE	N

Definition	Cut Index?
3DPRIM_PATCHLIST_n	Y

3DSTATE_INDEX_BUFFER

Vertex Buffers (VB) State

The 3DSTATE_VERTEX_BUFFERS and 3DSTATE_VF_INSTANCING commands are used to define *Vertex Buffers* (VBs) used in subsequent 3DPRIMITIVE commands.

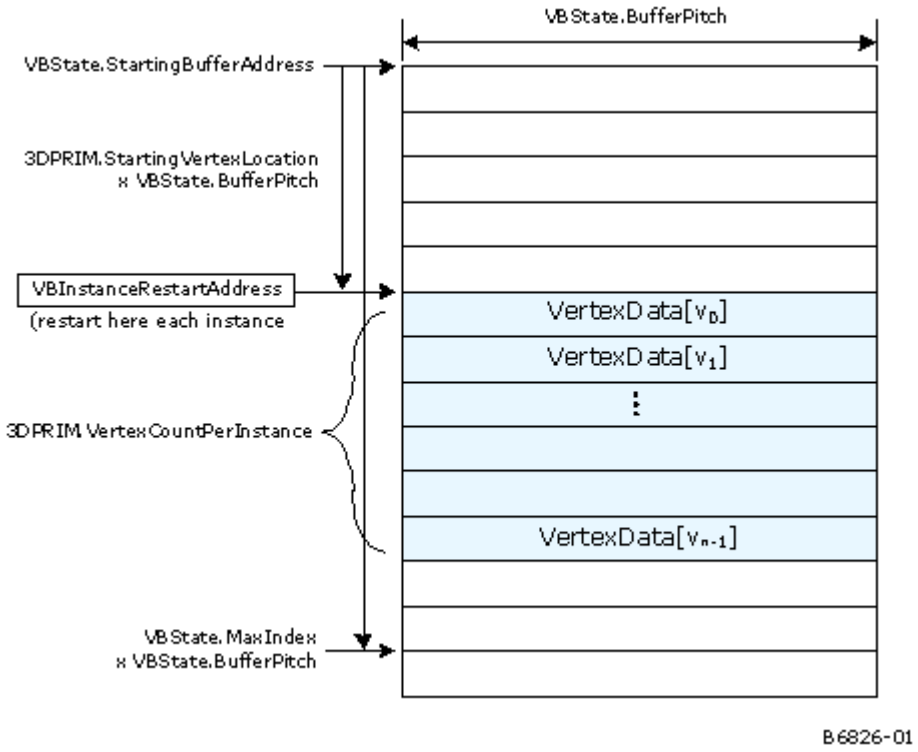
Most input vertex data is sourced from memory-resident VBs. A VB is a 1D array of structures, where the size of the structure as defined by the VB's **BufferPitch**. VBs are accessed either as *VERTEXDATA buffers* or *INSTANCEDATA buffers*, as defined by the **InstancingEnable** state in 3DSTATE_VF_INSTANCING. The VB's access type will determine whether the VF-computed VertexIndex or InstanceIndex is used to access data in the VB.

Given that the RANDOM access mode of the 3DPRIMITIVE command utilizes an IB (possibly provided by an application) to compute VB index values, VB definitions contain a **MaxIndex** value used to detect accesses beyond the end of the VBs. Any access outside the extent of a VB returns 0.

Register
3DSTATE_VERTEX_BUFFERS
VERTEX_BUFFER_STATE

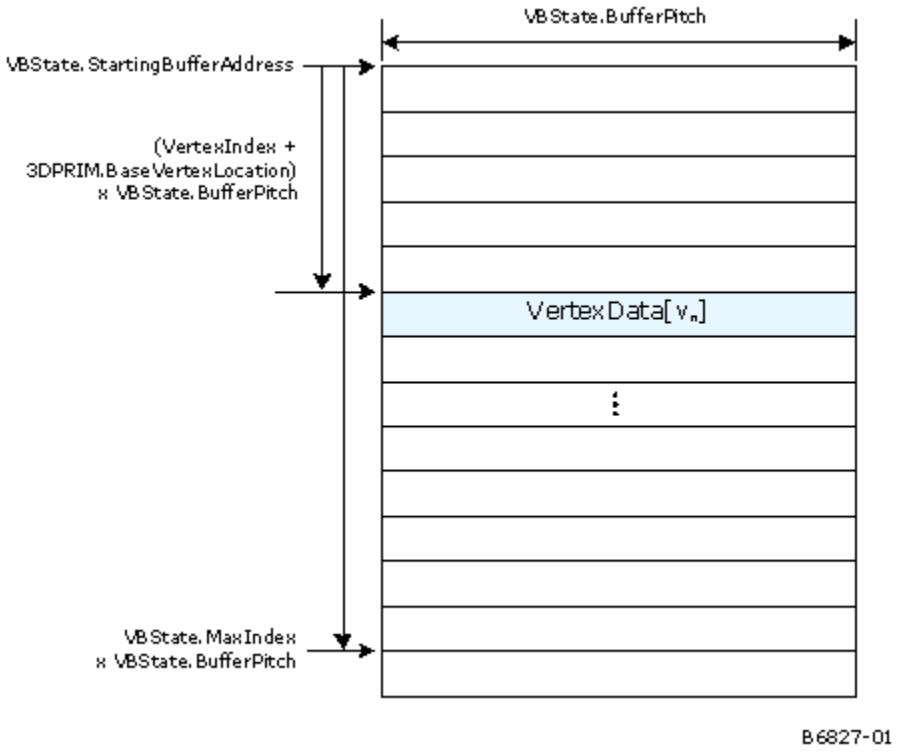
VERTEXDATA Buffers - SEQUENTIAL Access

Description
This section pertains to (a) 3DPRIMITIVE commands with VertexAccessType = SEQUENTIAL and (b) vertex elements with InstancingEnable set to DISABLED. Instead of "VBState.StartingBufferAddress + VBState.MaxIndex x VBState.BufferPitch", the address of the byte immediately beyond the last valid byte of the buffer is determined by "VBState.StartingBufferAddress + VBState.BufferSize".



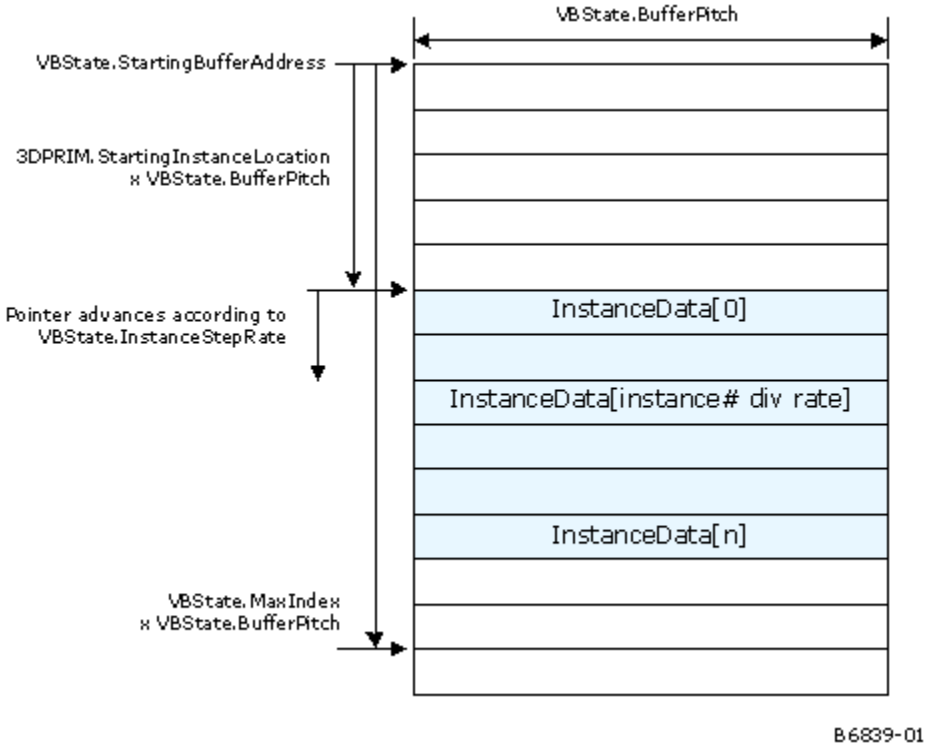
VERTEXDATA Buffers - RANDOM Access

Description
This section pertains to (a) 3DPRIMITIVE commands with VertexAccessType = RANDOM and (b) vertex elements with InstancingEnable set to DISABLED. Instead of "VBState.StartingBufferAddress + VBState.MaxIndex x VBState.BufferPitch", the address of the byte immediately beyond the last valid byte of the buffer is determined by "VBState.StartingBufferAddress + VBState.BufferSize".



INSTANCEDATA Buffers

Description
This section pertains to vertex elements with InstancingEnable set to ENABLED. Instead of "VBState.StartingBufferAddress + VBState.MaxIndex x VBState.BufferPitch", the address of the byte immediately beyond the last valid byte of the buffer is determined by "VBState.StartingBufferAddress + VBState.BufferSize".



Vertex Definition State

The following subsections define the state information for vertex data and describe some related processing.

Input Vertex Definition

The 3DSTATE_VERTEX_ELEMENTS command is used to define the source and format of input vertex data and the format of how it is stored in the destination VUE as part of 3DPRIMITIVE processing in the VF unit.

Two additional commands are added. 3DSTATE_VF_INSTANCING specifies the InstanceStepRate on a per-vertex-element basis. 3DSTATE_VF_SGVS specifies optional insertion of VertexID and/or InstanceID into the input vertex data (logically following the processing of the VERTEX_ELEMENT_STATE structures).

Refer to *3DPRIMITIVE Processing* below for the general flow of how input vertices are input and stored during processing of the 3DPRIMITIVE command.

Register
VERTEX_ELEMENT_STATE
3DSTATE_VERTEX_ELEMENTS
3D_Vertex_Component_Control
3DSTATE_VF_INSTANCING
3DSTATE_VF_SGVS
3DSTATE_VF_SGVS_2
3DSTATE_VF_COMPONENT_PACKING

3D Primitive Command

Following are 3D Primitive Commands:

3DPRIMITIVE

3D Primitive Topology Type Encoding

The following table defines the encoding of the Primitive Topology Type field. See *3D Pipeline* for details, programming restrictions, diagrams, and a discussion of the basic primitive types.

3D_Prim_Topo_Type

Functions

This section covers the various functions for Vertex Fetch.

Input Assembly

The VF's InputAssembly function includes (for each vertex generated):

- Generation of VertexIndex and InstanceIndex for each vertex, possibly via use of an Index Buffer.
- Lookup of the VertexIndex in the Vertex Cache (if enabled)
- If a cache miss is detected:
    - Use of computed indices to fetch data from memory-resident vertex buffers
    - Format conversion of the fetched vertex data
    - Assembly of the format conversion results (and possibly some internally generated data) to form the complete "input" (raw) vertex
    - Storing the input vertex data in a Vertex URB Entry (VUE) in the URB
    - Output of the VUE handle of the input vertex to the VS stage
- If a cache hit is detected, the VUE handle from the Vertex Cache is passed to the VS stage (marked as a cache hit to prevent any VS processing).

Vertex Assembly

The VF utilizes a number of VERTEX_ELEMENT state structures to define the contents and format of the vertex data to be stored in Vertex URB Entries (VUEs) in the URB. See below for a detailed description of the command used to define these structures (3DSTATE_VERTEX_ELEMENTS).

Each active VERTEX_ELEMENT structure defines up to 4 contiguous DWords of VUE data, where each DWord is considered a "component" of the vertex element. The starting destination DWord offset of the vertex element in the VUE is specified, and the VERTEX_ELEMENT structures must be defined with monotonically increasing VUE offsets. For each component, the source of the component is specified. The source may be a constant (0, 0x1, or 1.0f), a generated ID (VertexID, InstanceID or PrimitiveID), or a component of a structure in memory (e.g,. the Y component of an XYZW position in memory). In the case of a memory source, the Vertex Buffer sourcing the data, and the location and format of the source data with that VB are specified.

The VF's Vertex Assembly process can be envisioned as the VF unit stepping through the VERTEX_ELEMENT structures in order, fetching and format-converting the source information (if memory resident), and storing the results in the destination VUE.

The information supplied via the 3DSTATE_VF_SGVS command is also used to optionally insert VertexID and/or InstanceID into the input vertex data, after the VERTEX_ELEMENT structures are processed.

Vertex Cache

The VF stage communicates with the VS stage in order to implement a Vertex Cache function in the 3D pipeline. The Vertex Cache is strictly a performance-enhancing feature and has no impact on 3D pipeline results (other than a few statistics counters).

The Vertex Cache contains the VUE handles of VS-output (shaded) vertices if the VS function is enabled, and the VUE handles of VF-output (raw) vertices if the VS function is disabled. (Note that the actual vertex data is held in the URB, and only the handles of the vertices are stored in the cache). In either case, the contents of the cache (VUE handles) are tagged with the VertexIndex value used to fetch the input vertex data. The rationale for using the VertexIndex as the tag is that (assuming no other state or parameters change) a vertex with the same VertexIndex as a previous vertex will have the same input data, and therefore the same result from the VF+VS function.

Note that any change to the state controlling the InputAssembly function (e.g., vertex buffer definition), or any change to the state controlling the VS function (if enabled) (e.g., VS kernel), will result in the Vertex Cache being invalidated. In addition, any non-trivial use of instancing (i.e., more than one instance per 3DPRIMITIVE command and the inclusion of instance data in the input vertex) will effectively invalidate the cache between instances, as the InstanceIndex is not included in the cache tag. See Vertex Caching in *Vertex Shader* for more information on the Vertex Cache (e.g., when it is implicitly disabled, etc.)

The hardware interface to supply instancing state information is slightly different. Individual vertex elements (instead of buffers) are tagged as instanced or not.

Input Data: Push Model vs. Pull Model

Given the programmability of the pipeline, and the ability of shaders to input (load/sample) data from memory buffers in an arbitrary fashion, the decision arises in whether to push instance/vertex data into the front of the pipeline or defer the data access (pull) to the shaders that require it. Modern APIs directly support the latter model via *auto-generated IDs* in the Input Assembly function. An incrementing *VertexID*, *InstanceID*, and *PrimitiveID* are generated in the Input Assembly process, and these values can be declared as input to the "first enabled, relevant" shader. That shader can, for example, use the HW-generated ID as an index into a memory resource such as a constant buffer or vertex buffer. The 3D pipeline HW supports these IDs as required by the APIs.

There are tradeoffs involved in deciding between these models. For vertex data, it is probably always better to push the data into the pipeline, as the VF hardware attempts to cover the latency of the data fetch. The decision is less clear for instance data, as pushing instance data leads to larger Vertex URB entries which will be holding redundant data (as the instance data for vertices of an object are by definition the same). Regardless, the 3D pipeline supports both models.

Generated IDs

Note that the generated IDs are considered separate from any offset computations performed by the VF unit, and are therefore described separately here.

The VF generates InstanceID, VertexID, and PrimitiveID values as part of the InputAssembly process.

VertexID and InstanceID are only allowed to be inserted into the input vertex data as it is gathered and written into the URB as a VUE.

The definition/use of PrimitiveID is more complicated than the other auto-generated IDs. PrimitiveID is associated with an "object" and not a particular vertex.

It is only available to the GS and HS as a special non-vertex input and the PS as a constant-interpolated attribute. It is not seen by the VS or DS at all.

The PrimitiveID therefore is kept separate from the vertex data. Take for example a TRILIST primitive topology: It should be possible to share vertices between triangles in the list (i.e., reuse the VS output of a vertex), even though each triangle has a different PrimitiveID associated with it.

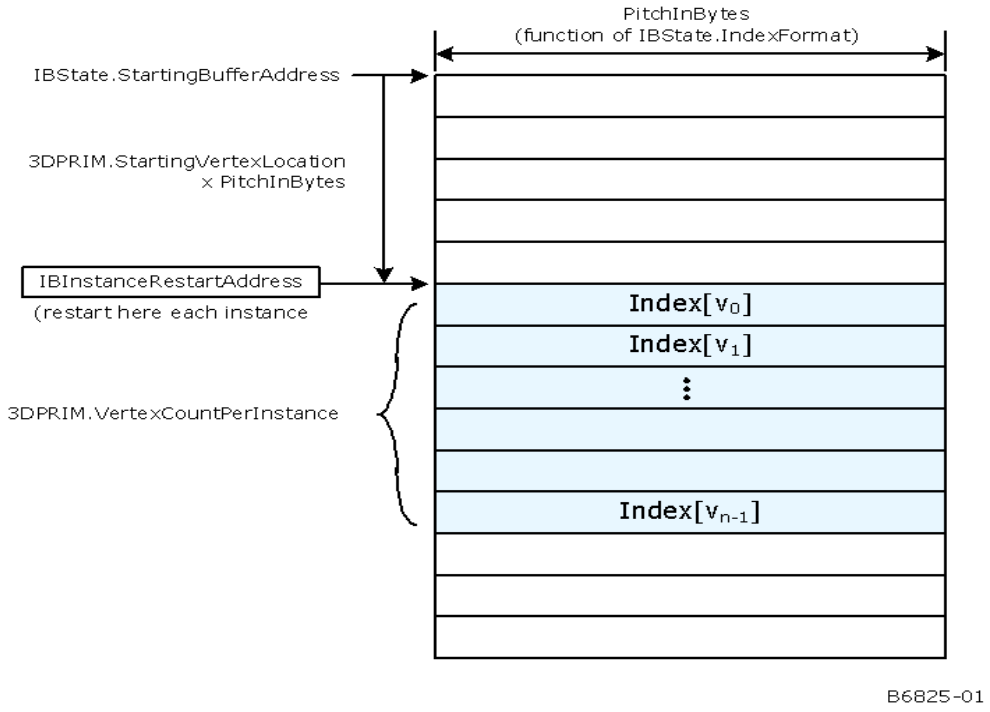
The optional insertion of VertexID and/or InstanceID into the input vertex data occurs as a separate step after the processing of VERTEX_ELEMENT structures and is controlled via the 3DSTATE_VF_SGVS command.

PrimitiveID is generated by hardware, plumbed down into the HS, GS and SF stages. It is passed along in HS/GS thread payloads. Software can also select PrimitiveID to be swizzled into vertex attribute data in the SF stage, though only if neither the HS nor GS stages are enabled.

3D Primitive Processing

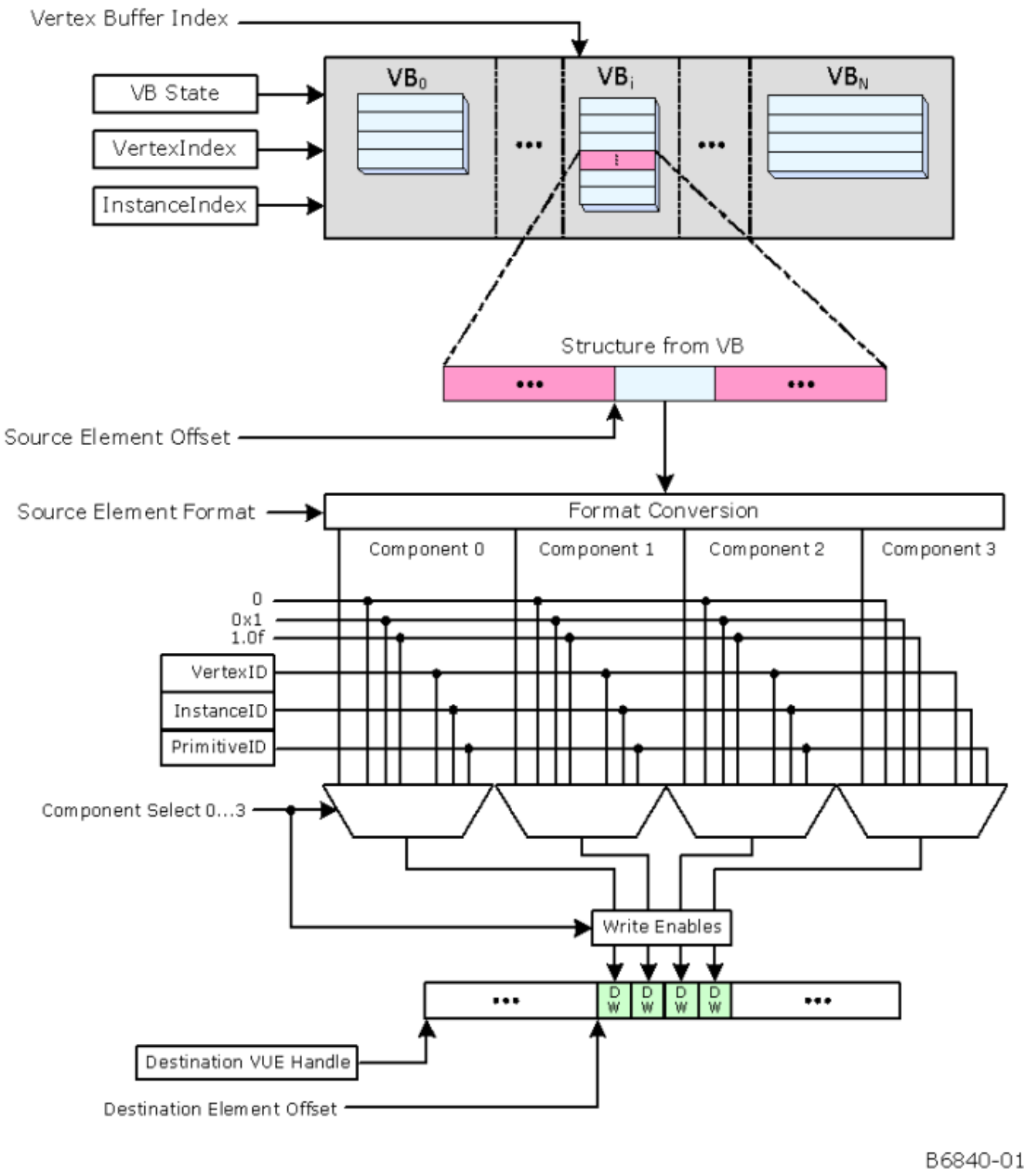
Index Buffer Access

The following figure illustrates how the Index Buffer is accessed.



Vertex Element Data Path

The following diagram shows the path by which a vertex element within the destination VUE is generated and how the fields of the VERTEX_ELEMENT_STATE structure is used to control the generation.


Vertex Buffer Index
VB State
VertexIndex
InstanceIndex
VB0
VBi
VBN
Structure from VB
Source Element Offset
Source Element Format
Format Conversion
Component 0
Component 1
Component 2
Component 3
0
0x1
1.0f
VertexID
InstanceID
PrimitiveID
Component Select 0...3
Write Enables
DW
DW
DW
DW
Destination VUE Handle
Destination Element Offset
B6840-01

FormatConversion

Once the VE source data has been fetched, it is subjected to format conversion. The output of format conversion is up to 4 32-bit components, each either integer or floating-point (as specified by the **Source Element Format**). See *Sampler* for conversion algorithms.

The following table lists the valid **Source Element Format** selections, along with the format and availability of the converted components (if a component is listed as -, it cannot be used as the source of a VUE component). **Note:** This table is a subset of the list of supported surface formats defined in the *Sampler* chapter. Please refer to that table as the "original list". This table is here only to identify the components available (per format) and their format.

Source Element Formats Supported in VF Unit

Source Element	Converted Component				
Surface Format Name	Format	0	1	2	3
R32G32B32A32_FLOAT	FLOAT	R	G	B	A
R32G32B32A32_SINT	SINT	R	G	B	A
R32G32B32A32_UINT	UINT	R	G	B	A
R32G32B32A32_UNORM	FLOAT	R	G	B	A
R32G32B32A32_SNORM	FLOAT	R	G	B	A
R64G64_FLOAT	FLOAT	R	G	-	-
R32G32B32A32_SSCALED	FLOAT	R	G	B	A
R32G32B32A32_USCALED	FLOAT	R	G	B	A
R32G32B32A32_SFIXED	FLOAT	R	G	B	A
R64G64_PASSTHRU	NONE	R	G	-	-
R32G32B32_FLOAT	FLOAT	R	G	B	-
R32G32B32_SINT	SINT	R	G	B	-
R32G32B32_UINT	UINT	R	G	B	-
R32G32B32_UNORM	FLOAT	R	G	B	-
R32G32B32_SNORM	FLOAT	R	G	B	-
R32G32B32_SSCALED	FLOAT	R	G	B	-
R32G32B32_USCALED	FLOAT	R	G	B	-
R32G32B32_SFIXED	FLOAT	R	G	B	-
R16G16B16A16_UNORM	FLOAT	R	G	B	A
R16G16B16A16_SNORM	FLOAT	R	G	B	A
R16G16B16A16_SINT	SINT	R	G	B	A
R16G16B16A16_UINT	UINT	R	G	B	A
R16G16B16A16_FLOAT	FLOAT	R	G	B	A
R32G32_FLOAT	FLOAT	R	G	-	-
R32G32_SINT	SINT	R	G	-	-

Source Element	Converted Component				
R32G32_UINT	UINT	R	G	-	-
R32G32_UNORM	FLOAT	R	G	-	-
R32G32_SNORM	FLOAT	R	G	-	-
R64_FLOAT	FLOAT	R	-	-	-
R16G16B16A16_SSCALED	FLOAT	R	G	B	A
R16G16B16A16_USCALED	FLOAT	R	G	B	A
R32G32_SSCALED	FLOAT	R	G	-	-
R32G32_USCALED	FLOAT	R	G	-	-
R32G32_SFIXED	FLOAT	R	G	-	-
R64_PASSTHRU	NONE	R	-	-	-
B8G8R8A8_UNORM	FLOAT	B	G	R	A
R10G10B10A2_UNORM	FLOAT	R	G	B	A
R10G10B10A2_UINT	UINT	R	G	B	A
R10G10B10_SNORM_A2_UNORM	FLOAT	R	G	B	A
R8G8B8A8_UNORM	FLOAT	R	G	B	A
R8G8B8A8_SNORM	FLOAT	R	G	B	A
R8G8B8A8_SINT	SINT	R	G	B	A
R8G8B8A8_UINT	UINT	R	G	B	A
R16G16_UNORM	FLOAT	R	G	-	-
R16G16_SNORM	FLOAT	R	G	-	-
R16G16_SINT	SINT	R	G	-	-
R16G16_UINT	UINT	R	G	-	-
R16G16_FLOAT	FLOAT	R	G	-	-
B10G10R10A2_UNORM	FLOAT	R	G	B	A
R11G11B10_FLOAT	FLOAT	R	G	B	-
R32_SINT	SINT	R	-	-	-
R32_UINT	UINT	R	-	-	-
R32_FLOAT	FLOAT	R	-	-	-
R32_UNORM	FLOAT	R	-	-	-
R32_SNORM	FLOAT	R	-	-	-
R10G10B10X2_USCALED	FLOAT	R	G	B	-
R8G8B8A8_SSCALED	FLOAT	R	G	B	A
R8G8B8A8_USCALED	FLOAT	R	G	B	A
R16G16_SSCALED	FLOAT	R	G	-	-
R16G16_USCALED	FLOAT	R	G	-	-
R32_SSCALED	FLOAT	R	-	-	-
R32_USCALED	FLOAT	R	-	-	-

Source Element	Converted Component				
R8G8_UNORM	FLOAT	R	G	-	-
R8G8_SNORM	FLOAT	R	G	-	-
R8G8_SINT	SINT	R	G	-	-
R8G8_UINT	UINT	R	G	-	-
R16_UNORM	FLOAT	R	-	-	-
R16_SNORM	FLOAT	R	-	-	-
R16_SINT	SINT	R	-	-	-
R16_UINT	UINT	R	-	-	-
R16_FLOAT	FLOAT	R	-	-	-
R8G8_SSCALED	FLOAT	R	G	-	-
R8G8_USCALED	FLOAT	R	G	-	-
R16_SSCALED	FLOAT	R	-	-	-
R16_USCALED	FLOAT	R	-	-	-
R8_UNORM	FLOAT	R	-	-	-
R8_SNORM	FLOAT	R	-	-	-
R8_SINT	SINT	R	-	-	-
R8_UINT	UINT	R	-	-	-
R8_SSCALED	FLOAT	R	-	-	-
R8_USCALED	FLOAT	R	-	-	-
R8G8B8_UNORM	FLOAT	R	G	B	-
R8G8B8_SNORM	FLOAT	R	G	B	-
R8G8B8_SSCALED	FLOAT	R	G	B	-
R8G8B8_USCALED	FLOAT	R	G	B	-
R8G8B8_SINT	SINT	R	G	B	-
R8G8B8_UINT	UINT	R	G	B	-
R8G8B8_UINT	UINT	R	G	B	-
R64G64B64A64_FLOAT	FLOAT	R	G	B	A
R64G64B64_FLOAT	FLOAT	R	G	B	A
R16G16B16_FLOAT	FLOAT	R	G	B	-
R16G16B16_UNORM	FLOAT	R	G	B	-
R16G16B16_SNORM	FLOAT	R	G	B	-
R16G16B16_SSCALED	FLOAT	R	G	B	-
R16G16B16_USCALED	FLOAT	R	G	B	-
R16G16B16_UINT	UINT	R	G	B	-
R16G16B16_SINT	SINT	R	G	B	-
R32_SFIXED	FLOAT	R	-	-	-
R10G10B10A2_SNORM	FLOAT	R	G	B	A

Source Element	Converted Component				
R10G10B10A2_USCALED	FLOAT	R	G	B	A
R10G10B10A2_SSCALED	FLOAT	R	G	B	A
R10G10B10A2_SINT	SINT	R	G	B	A
B10G10R10A2_SNORM	FLOAT	R	G	B	A
B10G10R10A2_USCALED	FLOAT	R	G	B	A
B10G10R10A2_SSCALED	FLOAT	R	G	B	A
B10G10R10A2_UINT	UINT	R	G	B	A
B10G10R10A2_SINT	SINT	R	G	B	A
R64G64B64A64_PASSTHRU	NONE	R	G	B	A
R64G64B64_PASSTHRU	NONE	R	G	B	-

DestinationFormatSelection

The **Component Select 0..3** bits are then used to select, on a per-component basis, which destination components will be written and with which value. The supported selections are the converted source component, VertexID, InstanceID, PrimitiveID, the constants 0 or 1.0f, or nothing (VFCOMP_NO_STORE). If a converted component is listed as '-' (not available) in the "Source Element Formats" table (above). It must not be selected (via VFCOMP_STORE_SRC), or an UNPREDICTABLE value will be stored in the destination component.

The selection process sequences from component 0 to 3. Once a **Component Select** of VFCOMP_NO_STORE is encountered, all higher-numbered **Component Select** settings must also be programmed as VFCOMP_NO_STORE. It is therefore not permitted to have 'holes' in the destination VE.

Dangling Vertex Removal

The last functional stage of processing of the 3DPRIMITIVE command is the removal of "dangling" vertices. This stage includes the discarding of primitive topologies without enough vertices for a single object (e.g., a TRISTRIP with only two vertices), as well as the discarding of trailing vertices that do not form a complete primitive (e.g., the last two vertices of a 5-vertex TRILIST). 3D APIs typically require these vertices to be (effectively) discarded before the VS stage.

Statistics Gathering

This function is best described as a filter operating on the vertex stream emitted from the processing of the 3DPRIMITIVE. The filter inputs the PrimType, PrimStart, and PrimEnd values associated with the generated vertices. The filter only outputs primitive topologies without dangling vertices. This requires the filter to (a) be able to buffer some number of vertices, and (b) be able to remove dangling vertices from the pipeline and dereference the associated VUE handles.

3DSTATE_VF_STATISTICS

Vertices Generated

VF will increment the IA_VERTICES_COUNT Register (see Memory Interface Registers in Volume Ia, *GPU*) for each vertex it fetches, even if that vertex comes from a cache rather than directly from a vertex buffer in memory. Any "dangling" vertices (fetched vertices that are part of an incomplete object) will not be included.

Objects Generated

VF will increment the IA_PRIMITIVES_COUNT Register (see Memory Interface Registers in vol1a System Overview) for each object (point, line, triangle, or quadrilateral) that it forwards down the pipeline.

Note
For LINELOOP, the last (closing) line object is counted.

Vertex Shader (VS) Stage

The Vertex Shader (VS) stage of the 3D Pipeline is used to perform processing ("shading") of vertices after they are assembled and written to the URB by the VF function. The primary function of the VS stage is to pass vertices that miss in the VS Cache to VS threads, and then pass the VS thread-generated vertices down the pipeline. Vertices that hit in the VS Cache have already been shaded and are therefore passed down the pipeline unmodified.

When the VS stage is disabled, vertices flow through the unit unmodified (i.e., as written by the VF unit).

State

Register
3DSTATE_VS
3DSTATE_CONSTANT_VS
3DSTATE_PUSH_CONSTANT_ALLOC_VS
3DSTATE_BINDING_TABLE_POINTERS_VS
3DSTATE_SAMPLER_STATE_POINTERS_VS
3DSTATE_URB_VS

Functions

Vertex Shader Cache (VS$)

Note: The VS$ should not be confused with input data caches used by the VF stage when fetching data from index or vertex buffers in memory.

The 3D Pipeline employs a Vertex Shader Cache (VS$) that is shared between the VF and VS stages. (See *Vertex Fetch* chapter for additional information). The vertex index generated by the VF stage is used as the cache tag. The cached data contains the URB handle of a VUE, which in turn typically contains the vertex data output from a previously-executed VS shader, though if the VS function is disabled the VUE will contain the input vertex data generated by the VF stage.

When the VF stage processes a vertex, it will first perform a lookup in the VS$. If the vertex hits in the VS$, the VS stage will return the hit VUE handle to the VF stage, and the VF stage will subsequently pass the returned VUE handle back down the FF pipeline to VS. If the vertex misses in the VS$ (or always, if the VS$ is disabled), the VS stage will allocate a VUE handle for the miss vertex and return this to the VF stage. The VF stage will then proceed to fetch/generate the input vertex data, store the results into the VUE, and then pass the VUE down to the VS stage. If the VS function is enabled, the VUE handle/data will be used as input to a VS shader thread, and that thread will overwrite the VUE with the shader results.

The VS$ may be explicitly DISABLED via the Vertex Cache Disable bit in 3DSTATE_VS. Even when explicitly ENABLED, the VS stage will (by default) implicitly disable the VS$ whenever it detects one of the following conditions:

Condition
Sequential indices are used in the 3DPRIMITIVE command (though this is effectively a don't care as there would not be any VS$ hits).

The implicit disable persists as long as one of these conditions persist, after which the VS$ is invalidated.

The VS$ is implicitly invalidated between 3DPRIMITIVE commands and between instances within a 3DPRIMITIVE command - therefore use of InstanceID in a Vertex Element is not a condition under which the cache is implicitly disabled.

The following table summarizes the modes of operation of the VS$.

VS$	VS Function Enable	Mode of Operation
DISABLED (implicitly or explicitly)	DISABLED	The VS$ is not used. VF stage assembles all vertices and writes them into the VUE supplied by the VS stage. VS stage subsequently passes references to these VUEs down the pipeline without spawning any VS threads. Usage Model: This is an exceptional condition, only required for when the VF-generated vertices contain PrimitiveID. Otherwise, the VS$ should be enabled.
	ENABLED	The VS$ is not used. VF stage assembles all vertices and writes them into the VUE supplied by the VS stage. VS stage subsequently spawns VS threads to process all vertices, overwriting the input data with the results. The VS stage pass references to these VUEs down the pipeline. Usage Model: This mode is only used when the VS function is required, but either (a) the VS kernel produces a side effect (e.g., writes to a memory buffer) which in turn requires every vertex to be processed by a VS thread, or (b) the input vertex contains PrimitiveID.
ENABLED	DISABLED	The VS$ is used to provide reuse of VF-generated vertices. The VF stage checks the cache and only processes (assembles/writes) vertices that miss in the VS$. In either case, the VS stage passes references to vertices (that hit or miss) down the pipeline without spawning any VS threads. Usage Model: Normal operation when the VS function is not required (e.g., SW has

VS$	VS Function Enable	Mode of Operation
		detected a VS shader that simply copies outputs to inputs).
	ENABLED	The VS$ is used to provide reuse of VS-processed vertices. The VF stage checks the cache and only processes (assembles/writes) vertices that miss in the VS$. The VS stage only processes (shades) the vertices that missed in the VS$. The VS stage sends references to hit or missed vertices down the pipeline in the correct order. Usage Model: Normal operation when the VS function is required and use of the VS$ is permissible.

VS Thread Dispatch Masks

The VS stage controls the initial value loaded into the EU's Dispatch Mask state register as part of thread dispatch.

SIMD8 Dispatch Mask

In SIMD8 dispatch mode, the EU Dispatch Mask is initialized as a function of the number of vertices included in the thread dispatch, as follows:

- 1 vertex: 0x00000001
- 2 vertices: 0x00000003
- 3 vertices: 0x00000007
- 4 vertices: 0x0000000F
- 5 vertices: 0x0000001F
- 6 vertices: 0x0000003F
- 7 vertices: 0x0000007F
- 8 vertices: 0x000000FF

Vertex Output

VS threads must always write the destination URB entries whose handles are passed in the thread payload. Refer to Vertex Data Overview for details on any required contents/formats.

Thread Termination

VS threads must signal thread termination, in all likelihood on the last message output to the URB shared function. Refer to the *ISA* doc for details on End-Of-Thread indication.

Primitive Output

The VS unit will produce an output vertex reference for every input vertex reference received from the VF unit, in the order received. The VS unit simply copies the PrimitiveType, StartPrim, and EndPrim information associated with input vertices to the output vertices, and does not use this information in any way. Neither does the VS unit perform any readback of URB data.

Statistics Gathering

The VS stage tracks a single pipeline statistic, the number of times a vertex shader is executed. A vertex shader is executed for each vertex that is fetched on behalf of a 3DPRIMITIVE command, unless the shaded results for that vertex are already available in the vertex cache. If the **Statistics Enable** bit in VS_STATE is set, the VS_INVOCATION_COUNT Register (see Memory Interface Registers in Volume Ia, *GPU*) will be incremented for *each vertex* that is dispatched to a VS thread.

Description
When **VS Function Enable** is DISABLED and **Statistics Enable** is ENABLED, VS_INVOCATION_COUNT increments by one for every vertex that passes through the VS stage, even though no VS threads are spawned.

Payloads

SIMD8 Payload

The following table describes the payload delivered to VS threads.

SIMD8 VS Thread Payload

DWord	Bits	Description
R0.7	31	
	30:0	Reserved
R0.6	31:24	Reserved
	23:0	**Thread ID.** This field uniquely identifies this thread within the threads spawned by this FF unit, over some period of time. Format: Reserved for HW Implementation Use.
R0.5	31:10	**Description** **Scratch Space Offset.** Specifies the offset of the scratch space allocated to the thread, specified as a 1KB-granular offset from the **General State Base Address**. See **Scratch Space Base Offset** description in VS_STATE. (See *3D Pipeline* for further description on scratch space allocation). Format = GeneralStateOffset[31:10]
R0.5	9:0	**Description** **FFTID:** This ID is assigned by the FF unit and used to identify the thread within the set of outstanding threads spawned by the FF unit. Reserved for HW Implementation Use. **Format:** U10 **Range:** 0-727

DWord	Bits	Description
R0.4	31:5	**Binding Table Pointer.** Specifies the 32-byte aligned pointer to the Binding Table. It is specified as an offset from the **Surface State Base Address**. Format = SurfaceStateOffset[31:5]
	4:0	Reserved
R0.3	31:5	**Sampler State Pointer.** Specifies the location of the Sampler State Table to be used by this thread, specified as a 32-byte granular offset from the **General State Base Address** or **Dynamic State Base Address**. Format = DynamicStateOffset[31:5]
	4	**Description** **Single Instance.** If set, all valid vertices included in the thread payload come from the same instance of a 3DPRIMITIVE command. Otherwise the vertices come from more than one instance. When **SIMD8SingleInstanceDispatchEnable** is ENABLED, this bit will (by definition) always be set.
	3:0	**Description** **Per Thread Scratch Space.** Specifies the amount of scratch space allowed to be used by this thread. The value specifies the power that two is raised to (over determine the amount of scratch space). (See *3D Pipeline* for further description.) Format = U4 power of two (in excess of 10) Range = [0,11] indicating [1K Bytes, 2M Bytes]
R0.2 : R0.0	31:0	Reserved: MBZ
R1.7	31:0	Vertex 7 URB Return Handle (see R1.0)
R1.6	31:0	Vertex 6 URB Return Handle (see R1.0)
R1.5	31:0	Vertex 5 URB Return Handle (see R1.0)
R1.4	31:0	Vertex 4 URB Return Handle (see R1.0)
R1.3	31:0	Vertex 3 URB Return Handle (see R1.0)
R1.2	31:0	Vertex 2 URB Return Handle (see R1.0)
R1.1	31:0	Vertex 1 URB Return Handle (see R1.0)
R1.0	31:16	Reserved
	15:0	**Vertex 0 URB Return Handle.** This is the offset within the URB where Vertex 0 is to be stored. Format: 64B-granular offset into the URB

DWord	Bits	Description
[Varies] optional	255:0	**Constant Data (optional):** Please refer to the Push Constants chapter in the General Programming of Thread-Generating Stages section for more details on size and source of constant data.
		Vertex Data: Input data for the 8 input vertices is located here. Vertex0 data is passed in DW0 of these GRFs, and Vertex 7 data is passed in DW7. The first GRF contains Element 0 Component 0 for all 8 vertices, followed by components 1-3 in the three subsequent GRFs. This is followed by GRFs containing Element 1, and so on, up to the number of elements specified by Vertex URB Read Length. Note that the maximum limit is 30 elements per vertex, though the practical limit imposed by the compiler is likely lower.
Rv.7	31:0	Vertex 7 Element 0 Component 0
Rv.6	31:0	Vertex 6 Element 0 Component 0
Rv.5	31:0	Vertex 5 Element 0 Component 0
Rv.4	31:0	Vertex 4 Element 0 Component 0
Rv.3	31:0	Vertex 3 Element 0 Component 0
Rv.2	31:0	Vertex 2 Element 0 Component 0
Rv.1	31:0	Vertex 1 Element 0 Component 0
Rv.0	31:0	Vertex 0 Element 0 Component 0
Rv+1.7	31:0	Vertex 7 Element 0 Component 1
Rv+1.6	31:0	Vertex 6 Element 0 Component 1
Rv+1.5	31:0	Vertex 5 Element 0 Component 1
Rv+1.4	31:0	Vertex 4 Element 0 Component 1
Rv+1.3	31:0	Vertex 3 Element 0 Component 1
Rv+1.2	31:0	Vertex 2 Element 0 Component 1
Rv+1.1	31:0	Vertex 1 Element 0 Component 1
Rv+1.0	31:0	Vertex 0 Element 0 Component 1
..		Vertex 0-7 Element 0 Component 2,3
..		Vertex 0-7 Element 1 Component 0-3
..		Vertex 0-7 Element 2-N Component 0-3

Hull Shader (HS) Stage

The Hull Shader (HS) stage of the pipeline is used to process patchlist (PATCHLIST_*n*) topologies in support of higher-order surface (HOS) tessellaton. If the HS stage is enabled, each incoming patch object is processed by a possible series of HS threads. The combined output of these threads is a Patch URB Entry ("patch record") written to the URB. This patch record is used by subsequent stages (TE, DS) to complete the HOS tessellation operations.

The vertices associated with patchlist primitives are also referred to as "Input Control Points" (ICPs) to contrast them with any "Output Control Points" the HS threads may write to the patch record. (The definition and use of OCPs are outside the scope of this document).

The HS stage also performs statistics counting. Incomplete topologies do not reach the HS stage.

The HS, TE, and DS stages must be enabled and disabled together. When these stages are disabled, all topologies (including patchlist topologies) simply pass through to the GS stage. When these stages are enabled, only patchlist topologies should be issued to the pipeline, otherwise behavior is UNDEFINED.

State

This section contains the state registers for the Hull Shader

Register
3DSTATE_HS
3DSTATE_PUSH_CONSTANT_ALLOC_HS
3DSTATE_CONSTANT_HS
3DSTATE_CONSTANT(Body)
3DSTATE_BINDING_TABLE_POINTERS_HS
3DSTATE_SAMPLER_STATE_POINTERS_HS
3DSTATE_URB_HS

Functions

Patch Object Staging

The HS unit accepts patchlist topologies as a stream of incoming vertices. Depending on the number of vertices per patch object (as specified by the PATCHLIST_*n* topology), the HS thread assembles each complete patch object and passes it (its vertices, PrimitiveID, etc.) to HS thread(s) as described below.

HS Thread Execution

Input to HS threads is comprised of:

- Input Control Points (incoming patch vertices), pushed into the payload and/or passed indirectly via URB handles.
- Push Constants (common to all threads)
- Patch Data handle
- Resources available via binding table entries (accessed through shared functions)
- Miscellaneous payload fields (Instance Number, etc.)

Typically the only output of the HS threads is the Patch URB Entry (patch record). All thread instances for an input patch are passed the same patch record handle. As the (possibly concurrent) threads can both read and write the patch record, it is up to the kernels to ensure deterministic results. One approach would be to use the thread's Instance Number as an index for URB write destinations.

HS Thread Dispatch Mask

The HS stage controls the initial value loaded into the EU's Dispatch Mask state register as part of thread dispatch.

SINGLE_PATCH Dispatch Mask

In SINGLE_PATCH mode, the EU Dispatch Mask is initialized at thread dispatch to 0x000000FF.

8_PATCH Dispatch Mask

In 8_PATCH mode, the EU Dispatch Mask is initialized as a function of the number of patches included in the thread dispatch, as follows:

- 1 patch: 0x00000001
- 2 patches 0x00000003
- 3 patches: 0x00000007
- 4 patches: 0x0000000F
- 5 patches: 0x0000001F
- 6 patches: 0x0000003F
- 7 patches: 0x0000007F
- 8 patches: 0x000000FF

Patch URB Entry (Patch Record) Output

For each patch, the HS thread(s) generate a single patch record, starting with a fixed 32B Patch Header.

When the final thread instance terminates, the patch record handle is passed down the pipeline to the Tessellation Engine (TE).

Patch Header DW0-7

The first eight DWords of the Patch URB Entry are defined as a "Patch Header". A Patch Header is written by an HS thread and read by the TE stage. It normally contains up to six **Tessellation Factors** (TFs) that determine how finely the TE stage needs to tessellate a domain (if at all).

The layout of the patch header depends on the **TE Domain** and **PatchHeaderLayout.**

The **PatchHeaderLayout** of REVERSED_TRI_INSIDE_SEPARATE can only be used with a **TE Domain** of TRI.

Patch Header: QUAD Domain / LEGACY Patch Header Layout

DWord	Bits	Description
7	31:0	**UEQ0 Tessellation Factor** Format: FLOAT32
6	31:0	**VEQ0 Tessellation Factor** Format: FLOAT32
5	31:0	**UEQ1 Tessellation Factor** Format: FLOAT32
4	31:0	**VEQ1 Tessellation Factor**

DWord	Bits	Description
		Format: FLOAT32
3	31:0	**Inside U Tessellation Factor** Format: FLOAT32
2	31:0	**Inside V Tessellation Factor** Format: FLOAT32
1-0	31:0	Reserved : MBZ

Patch Header: QUAD Domain / REVERSED Patch Header Layout

DWord	Bits	Description
7:6	31:0	Reserved : MBZ
5	31:0	**Inside V Tessellation Factor** Format: FLOAT32
4	31:0	**Inside U Tessellation Factor** Format: FLOAT32
3	31:0	**VEQ1 Tessellation Factor** Format: FLOAT32
2	31:0	**UEQ1 Tessellation Factor** Format: FLOAT32
1	31:0	**VEQ0 Tessellation Factor** Format: FLOAT32
0	31:0	**UEQ0 Tessellation Factor** Format: FLOAT32

Patch Header: TRI Domain / LEGACY Patch Header Layout

DWord	Bits	Description
7	31:0	**UEQ0 Tessellation Factor** Format: FLOAT32
6	31:0	**VEQ0 Tessellation Factor** Format: FLOAT32
5	31:0	**WEQ0 Tessellation Factor** Format: FLOAT32
4	31:0	**Inside Tessellation Factor** Format: FLOAT32
3-0	31:0	Reserved : MBZ

Patch Header: TRI Domain / REVERSED Patch Header Layout

DWord	Bits	Description
7-4	31:0	Reserved : MBZ
3	31:0	**Inside Tessellation Factor** Format: FLOAT32

DWord	Bits	Description
2	31:0	**WEQ0 Tessellation Factor** Format: FLOAT32
1	31:0	**VEQ0 Tessellation Factor** Format: FLOAT32
0	31:0	**UEQ0 Tessellation Factor** Format: FLOAT32

Patch Header: TRI Domain / REVERSED_TRI_INSIDE_SEPARATE Patch Header Layout

DWord	Bits	Description
7-5	31:0	Reserved : MBZ
4	31:0	**Inside Tessellation Factor** Format: FLOAT32
3	31:0	Reserved : MBZ
2	31:0	**WEQ0 Tessellation Factor** Format: FLOAT32
1	31:0	**VEQ0 Tessellation Factor** Format: FLOAT32
0	31:0	**UEQ0 Tessellation Factor** Format: FLOAT32

Patch Header: ISOLINE Domain / LEGACY Patch Header Layout

DWord	Bits	Description
7	31:0	**Line Detail Tessellation Factor** Format: FLOAT32
6	31:0	**Line Density Tessellation Factor** Format: FLOAT32
5-0	31:0	Reserved : MBZ

Patch Header: ISOLINE Domain / REVERSED Patch Header Layout

DWord	Bits	Description
7-2	31:0	Reserved : MBZ
1	31:0	**Line Density Tessellation Factor** Format: FLOAT32
0	31:0	**Line Detail Tessellation Factor** Format: FLOAT32

Statistics Gathering

HS Invocations

The HS unit controls the HS_INVOCATIONS counter, which counts the number of patches processed by the HS stage.

Payloads

8_PATCH Payload

The following table shows the layout of the payload delivered to HS threads. Refer to 3D Pipeline Stage Overview (*3D Pipeline*) for details on those fields that are common amongst the various pipeline stages.

Patch object vertex (ICP) data can be passed by value (data pushed in the payload) and/or by reference (URB handle pushed in the payload).

8_PATCH HS Thread Payload

GRF DWord	Bits	Description
R0.7	31	
	30:0	Reserved
R0.6	31	**Early Dereference Enable**: The enabling of early dereference. Reserved for Implementation Use
	30:24	Reserved
	23:0	**Thread ID.** This field uniquely identifies this thread within the threads spawned by this FF unit, over some period of time. Format: Reserved for HW Implementation Use.
R0.5	31:10	**Scratch Space Pointer.** Specifies the location of the scratch space allocated to this thread, specified as a 1KB-aligned offset from the General State Base Address. Format = GeneralStateOffset[31:10]
	9	Reserved
	8:0	**FFTID.** This ID is assigned by the fixed function unit and is a relative identifier for the thread. It is used to free up resources used by the thread upon thread completion. Format: Reserved for Implementation Use
R0.4	31:5	**Binding Table Pointer:** Specifies the 32-byte aligned pointer to the Binding Table. It is specified as an offset from the Surface State Base Address. Format = SurfaceStateOffset[31:5]

GRF DWord	Bits	Description
	4:0	**Reserved**
R0.3	31:5	**Sampler State Pointer.** Specifies the location of the Sampler State Table to be used by this thread, specified as a 32-byte granular offset from the Dynamic State Base Address. Format = DynamicStateOffset[31:5]
	4	Reserved
	3:0	**Per Thread Scratch Space.** Specifies the amount of scratch space allowed to be used by this thread. The value specifies the power that two will be raised to (over determine the amount of scratch space). Programming Notes:This amount is available to the kernel for information only. It will be passed verbatim (if not altered by the kernel) to the Data Port in any scratch space access messages, but the Data Port will ignore it. Format = U4 power of two (in excess of 10) Range = [0,11] indicating [1K Bytes, 2M Bytes]
R0.2	31	**Reserved**: MBZ.
	30:24	**BarrierID.** This field identifies which barrier was allocated for this thread. Format: U7 Range = [0,63]
	22:16	**Instance Number.** A patch-relative instance number between 0 and InstanceCount-1. Format = U7
	23	**Reserved**
	15:0	**Reserved**
R0.1-R0.0	31:0	Reserved
R1.7	31:0	**URB Return Handle for Patch 7** (See R1.0)
R1.6	31:0	**URB Return Handle for Patch 6** (See R1.0)
R1.5	31:0	**URB Return Handle for Patch 5** (See R1.0)
R1.4	31:0	**URB Return Handle for Patch 4** (See R1.0)
R1.3	31:0	**URB Return Handle for Patch 3** (See R1.0)
R1.2	31:0	**URB Return Handle for Patch 2** (See R1.0)
R1.1	31:0	**URB Return Handle for Patch 1** (See R1.0)
R1.0	31:16	Reserved
	15:0	**URB Return Handle 0:** This is the offset of the Patch 0's URB entry, where shading results are to be written. Format: U16 64B-aligned URB Offset

GRF DWord	Bits	Description
The following register is included only if Include PrimitiveID is enabled.		
R2.7	31:0	**Primitive ID 7.** This field contains the Primitive ID associated with Patch 7 Format: U32
R2.6	31:0	**Primitive ID 6.** This field contains the Primitive ID associated with Patch 6 Format: U32
R2.5	31:0	**Primitive ID 5.** This field contains the Primitive ID associated with Patch 5 Format: U32
R2.4	31:0	**Primitive ID 4.** This field contains the Primitive ID associated with Patch 4 Format: U32
R2.3	31:0	**Primitive ID 3.** This field contains the Primitive ID associated with Patch 3 Format: U32
R2.2	31:0	**Primitive ID 2.** This field contains the Primitive ID associated with Patch 2 Format: U32
R2.1	31:0	**Primitive ID 1.** This field contains the Primitive ID associated with Patch 1 Format: U32
R2.0	31:0	**Primitive ID 0.** This field contains the Primitive ID associated with Patch 0 Format: U32
The following registers are included only if Include Vertex Handles is enabled		
Rn.7	31:16	Reserved
	15:0	**Patch 7 ICP 0 Handle**
Rn.6	31:16	Reserved
	15:0	**Patch 6 ICP 0 Handle**
Rn.5	31:16	Reserved
	15:0	**Patch 5 ICP 0 Handle**
Rn.4	31:16	Reserved
	15:0	**Patch 4 ICP 0 Handle**
Rn.3	31:16	Reserved
	15:0	**Patch 3 ICP 0 Handle**
Rn.2	31:16	Reserved

GRF DWord	Bits	Description
	15:0	**Patch 2 ICP 0 Handle**
Rn.1	31:16	Reserved
	15:0	**Patch 1 ICP 0 Handle**
Rn.0	31:16	Reserved
	15:0	**Patch 0 ICP 0 Handle**
[Rn+1]	255:0	**ICP 1 Handle for Patches 0-7**
[Rn+2]	255:0	**ICP 2 Handle for Patches 0-7**
...		...
[Rn+31]	255:0	ICP 31 Handle for Patches 0-7
[Varies] optional	255:0	**Constant Data (optional):** Please refer to the Push Constants chapter in the General Programming of Thread-Generating Stages section for more details on size and source of constant data.
Varies		**Pushed Vertex Data (optional)** Input data for the 8 patches is located here. Patch 0 (starting with Vertex 0 of Patch 0) data is passed in DW0 of these GRFs, and Patch 7 data is passed in DW7. The first GRF contains Vertex 0 Element 0 Component 0 for all 8 patches, followed by components 1-3 in the three subsequent GRFs. This is followed by GRFs containing Vertex 0 Element 1 (if it exists), and so on, up to the number of Vertex 0 elements specified by Vertex URB Read Length. This is followed by the data for Vertex 1 for all patches (if it exists), and so on until all relevant vertices are passed. **Programming Note** **Context:** 8_PATCH Payload - Pushed Vertex Data The amount of data passed is limited by the number of GRFs supported by EUs. Software is responsible for comprehending this limit and resorting to the pull model as required.
Rv.7	31:0	**Patch 7 Vertex 0 Element 0 Component 0**
Rv.6	31:0	**Patch 6 Vertex 0 Element 0 Component 0**
Rv.5	31:0	**Patch 5 Vertex 0 Element 0 Component 0**
Rv.4	31:0	**Patch 4 Vertex 0 Element 0 Component 0**
Rv.3	31:0	**Patch 3 Vertex 0 Element 0 Component 0**
Rv.2	31:0	**Patch 2 Vertex 0 Element 0 Component 0**
Rv.1	31:0	**Patch 1 Vertex 0 Element 0 Component 0**
Rv.0	31:0	**Patch 0 Vertex 0 Element 0 Component 0**
Rv+1	31:0	**Patch 0-7 Vertex 0 Element 0 Component 1**
...		and so on...

Tessellation Engine (TE) Stage

When enabled, the Tessellation Engine (TE) stage performs tessellation of incoming patches (decomposition of patches into a set of smaller geometric objects, such as triangles or points). Patches are also subjected to a Patch Cull test prior to tessellation. Culled patches are immediately discarded. The TE stage is entirely fixed-function and does not spawn threads.

Patches are specified via URB handles output by the preceding Hull Shader stage. These handles reference Patch URB Entry data written into the URB by HS shaders. The tessellation process is controlled by TE state and Tessellation Factors (TFs) read from the Patch URB Entries.

The fixed-function tessellation algorithm is considered an implementation detail and is therefore beyond the scope of this document. That detail includes both the order of output topologies as well as the order of vertices (domain points) within the output topologies. Only a high-level overview is provided to describe how the (few) state variables can be used to control aspects of tessellation behavior. The implementation will generate deterministic results (given the same exact inputs it will produce exactly the same outputs).

Several domain types (QUAD, TRI, and ISOLINE) are supported. Depending on the domain type, the TE stage outputs the required point/line/triangle topologies including a domain point per vertex. These topologies will be output to the DS stage, where the domain points will be converted to 3D object vertices, resulting in 3D objects as typically input to the 3D pipeline when HOS tessellation is not used.

When tessellation is disabled, all topologies (including patchlist topologies) simply pass through to the GS stage. When tessellation is enabled, only patchlist topologies should be issued to the pipeline, else behavior is UNDEFINED. The MI_TOPOLOGY_FILTER command can be used to ensure this happens, i.e., it can be used to have the Command Stream ignore 3DPRIMITIVE commands that do not match a specific topology type.

Enabling tessellation is accomplished by enabling the HS/TE/DS stages in specific combinations. Those valid combinations are described in the table below.

Valid Tessellation Enabled Configurations
To enable tessellation, the HS, TE, and DS stages must be enabled and disabled together. Other configurations will result in behavior that is UNDEFINED.

State

This section contains the state registers for the Tessellation Engine.

3DSTATE_TE

Functions

Patch Culling

Normally, if any "outside" TF is <= 0.0 or NaN, the entire patch is culled at the TE stage.

Inside TFs are not used to cull patches.

Tessellation Factor Limits

After the Patch Culling test is performed, the TessFactors undergo a min() clamp to either the **MaxTessFactorOdd** (for FRACTIONAL_ODD partitioning) or **MaxTessFactorNotOdd** (for FRACTIONAL_EVEN or INTEGER partitioning). Exception: If the ISOLINE domain is specified, the LineDensity TessFactor will be clamped to the **MaxFactorNotOdd** value even if FRACTIONAL_ODD partitioning is specified).

Partitioning

The Partitioning state controls how the TFs are used to divide their corresponding edges.

Partitioning Mode	Definition
INTEGER	The edge is divided into an integral number of equal segments (given some fixed-point tolerance). After clamping, the TF is rounded up to an integer value. The edge is divided into that many equal segments.
EVEN_FRACTIONAL	The edge is divided into an *even* number of possibly-unequal segments. The total number of segments is determined by rounding up the post-clamped TF to an even number. More specifically, the edge is divided exactly in half. Like the endpoints of the edge, the midpoint of the edge is by definition a tessellation point. Each half contains some number of equal segments and possibly one smaller segment. The size of the smaller segment is determined by the position of the TF value within the range defined by the TF rounded down and up to even numbers. The closer the TF is to the smaller value, the smaller the segment size is. When the TF reaches the smaller even value, the smaller segment disappears. The closer the TF gets to the larger even value, the closer the smaller segment size approaches the size of the other segments. When the TF reaches the larger even value, all segments are equal. The position of the smaller segment along the half edge varies as a function of the TF value.
ODD_FRACTIONAL	The edge is divided into an *odd* number of possibly-unequal segments. The tessellation scheme is very similar to EVEN_FRACTIONAL partitioning, except that the edge midpoint is not included as a tessellation point. This, and the fact that the tessellation points are mirrored about the edge midpoint, causes an "odd" segment (which may or may not be the "smaller" segment) to straddle the edge midpoint, therefore resulting in the number of segments for the edge always being odd.
POW2	The edge is divided into an integral number of equal segments (given some fixed-point tolerance). After clamping, the TF is rounded up to an integer power of 2 value. The edge is divided into that many equal segments.

Domain Types and Output Topologies

The major (if only) task of the TE stage is to tessellate a 2D (u,v) domain region, as selected by the Domain state, into some number of 2D object topologies. (If the patch is culled, that number may be zero). The options for Domain state are:

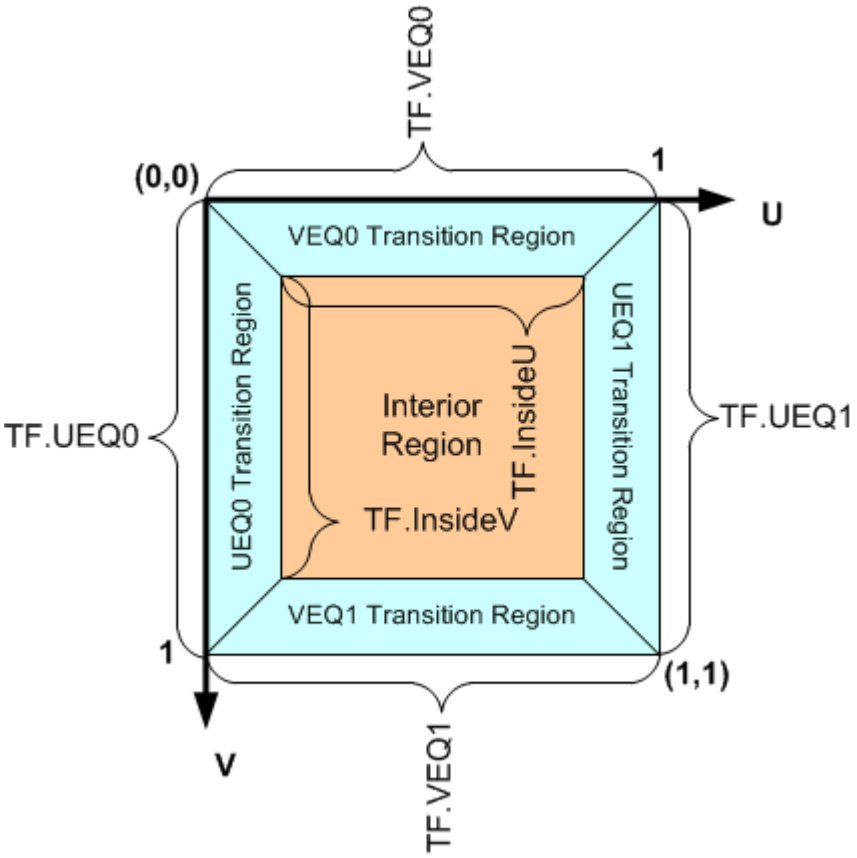
- QUAD: A square 2D region within a u,v Cartesian (rectangular) space. The region extends from the origin to u=1 and v=1. Within the region, tessellation domain locations are determined. The possible output topologies include points, clockwise triangles, and counterclockwise triangles.
- TRI: A triangular 2D region with u,v,w barycentric (areal) coordinates. The three edges correspond to u=0, v=0, and w=0 boundaries. In barycentric coordinates, w = 1 - u - v, therefore points within the region are fully defined as 2D (u,v) coordinates. Within the region, tessellation domain locations are determined. The possible output topologies include points, clockwise triangles, and counterclockwise triangles.
- ISOLINE: A series of points within a QUAD domain, where the points lie on lines parallel to the u axis and extending from [0,1) in the v direction. Either the segmented lines (linestrips) or individual point topologies can be output.

QUAD Domain Tessellation

The four "outside" TFs (TF.UEQ0, TF.VEQ0, TF.UEQ1, TF.VEQ1) are used to specify the level of tessellation along the four corresponding edges of the 2D quad domain. The two "inside" TFs (TF.InsideU, TF.InsideV) are used to determine the level of tessellation within a 2D "interior" region. Typically, the interior region appears as a "regularly-tessellated 2D grid", however under certain conditions the interior region may collapse in which case only the outside TFs are relevant.

In general, a transition region exists between each edge of the interior region and the corresponding outside edge. The topologies generated for these regions effectively "stitch together" locations along the outside and inside edges, as each of these edges can contain a different number of tessellated segments. In the case where all TFs in a given direction (e.g., TF.VEQ0, TF.InsideU, and TF.VEQ1) are the same value, it appears as if the regularly-tessellated interior region extends all the way to the outside edges. If this condition simultaneously exists for both u and v directions, the entire domain will appear to be tessellated into a regular grid, with no noticeable transition regions.

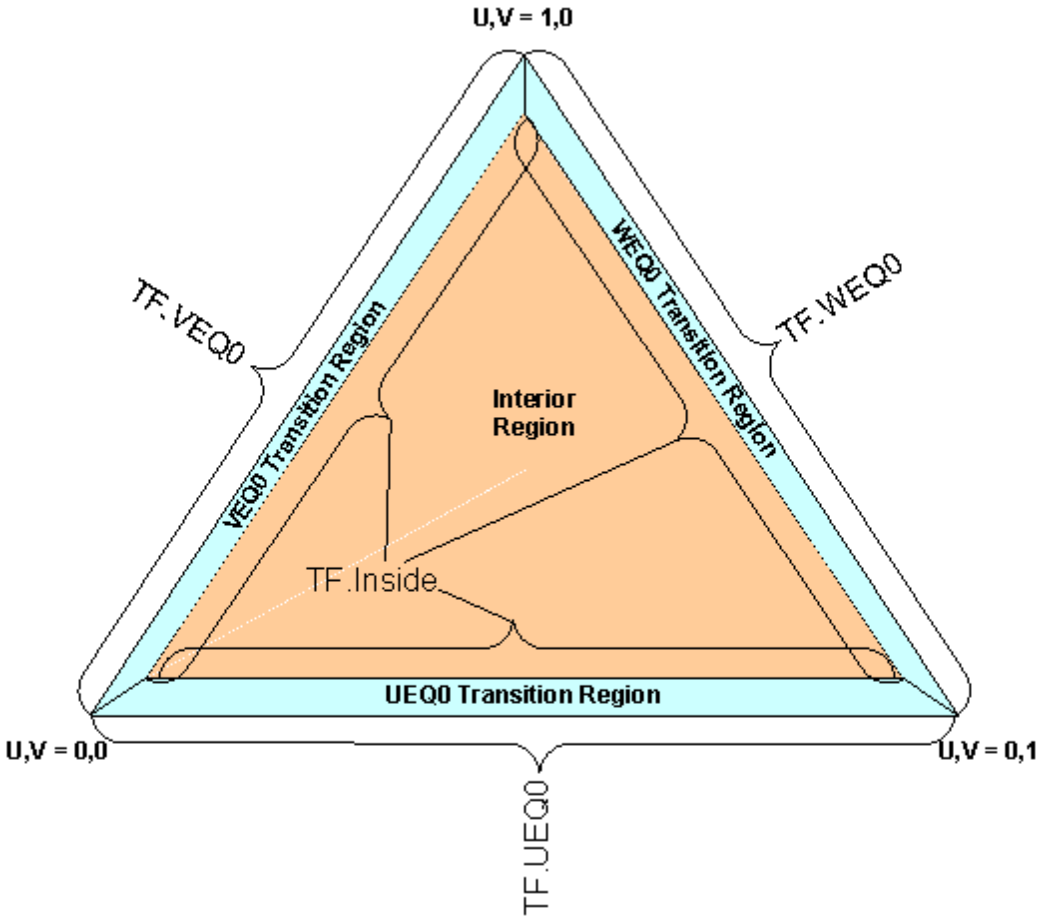
QUAD Domain



TRI Domain Tessellation

Tessellation of the TRI domain is similar to the QUAD domain, except only three outside edges/TFs are used, and the tessellation of the interior region is controlled by a single TF.

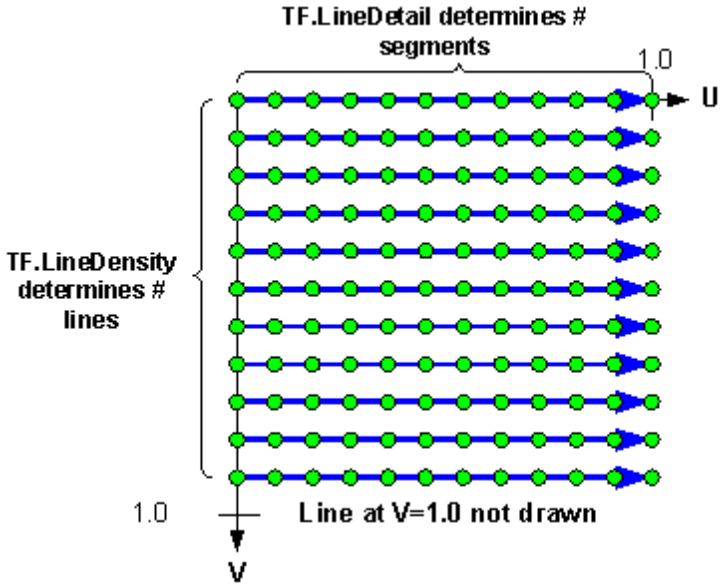
TRI Domain



ISOLINE Domain Tessellation

Tessellation of the ISOLINE domain is different but much simpler than QUAD and TRI domains. The TF.LineDetail TF controls how finely the U direction is tessellated, while the TF.LineDensity TF controls how finely the V direction is tessellated. When LINE output topology is selected, a series of segmented lines parallel to the U axis (constant V) are output. When POINT output topology is selected, only the line segment endpoints are output (as point objects). In either case there is no topology output for the V=1 edge, which avoids overlapping lines for adjacent patches.

ISOLINE Domain



Domain Shader (DS) Stage

The DS stage is very similar to the VS stage in that it is responsible for dispatching EU threads to shade vertices and maintaining a cache (with reference counts) of the shaded vertex outputs of these threads. Major differences are as follows:

- The DS receives topologies with "domain points" instead of vertices. The only data specific to a domain point are its U,V coordinates. These coordinates (plus a default or computed W coordinate) are passed directly in the DS thread payload. There is no other vertex-specific "input vertex data".
- The concatenation of the domain point U,V coordinates (vs. a vertex index) is used as the cache tag.
- The cache is invalidated between patches.

The DS stage accepts state information via the inline 3DSTATE_DS command.

State

This section contains the state registers for the Domain Shader.

Register
3DSTATE_DS
3DSTATE_PUSH_CONSTANT_ALLOC_DS
3DSTATE_CONSTANT_DS
3DSTATE_CONSTANT(Body)
3DSTATE_BINDING_TABLE_POINTERS_DS

Register
3DSTATE_SAMPLER_STATE_POINTERS_DS
3DSTATE_URB_DS

Functions

DUAL_PATCH Thread Execution

When Dispatch Mode is set to SIMD8_SINGLE_OR_DUAL_PATCH mode, both the KSP and DUAL_PATCH KSP kernels are enabled. The DS stage decides whether to spawn a SINGLE_PATCH (KSP) or DUAL_PATCH thread dynamically, based on the number of domain points associated with patches. (See Implementation Note below).

- The KSP kernel operates exactly like when SIMD8_SINGLE_PATCH mode is set. Up to 8 domain points for a single patch are processed by the DS thread, which operates in SIMD8 fashion.
- The DUAL KSP kernel uses a hybrid SIMD8 execution mode. The 8 execution channels are divided into 4 upper channels associated with Patch 1, and 4 lower channels associated with Patch 0. Patch data is passed in SIMD4x2 layout, with Patch 1 data (Primitive ID, pushed URB data) in the upper 4 channels, and Patch 0 data in the lower 4 channels. The kernel operates much like SIMD8_SINGLE_PATCH mode, though it needs to access the appropriate SIMD4 patch data.

Implementation Note: Kernel selection is as follows: If a patch requires more than 4 domain points to be shaded, SIMD8_SINGLE_PATCH threads are spawned until 4 or fewer domain points remain. These domain points (if any exist) are held pending until the next patch is received. Likewise, if a patch requires 4 or fewer total domain points, those domain points are held pending. In either case, if the subsequent patch requires 4 or fewer domain points to be shaded, a SIMD8_DUAL_PATCH thread is spawned to shade both sets of 4 or fewer domain points. If the subsequent patch requires more than 4 domain points, the (4 or fewer) buffered domain points of the previous patch are shaded via a SIMD8_SINGLE_PATCH thread, and the cycle continues.

Statistics Gathering

The DS stage maintains the DS_INVOCATIONS statistics counter, which counts the number of incoming domain points, irrespective of cache hit/miss. Note that this is different than VS_INVOCATIONS, which counts shader invocations and therefore doesn't count cache hits.

Payloads

SIMD8 Payload

The following table describes the payload delivered to DS threads.

DS Thread Payload (SIMD8)

DWord	Bits	Description
R0.7	31	
	30:0	Reserved
R0.6	31:24	Reserved
	23:0	**Thread ID.** This field uniquely identifies this thread within the threads spawned by this FF unit, over some period of time. Format: Reserved for HW Implementation Use.
R0.5	31:10	**Scratch Space Offset.** Specifies the offset of the scratch space allocated to the thread, specified as a 1KB-granular offset from the **General State Base Address**. See **Scratch Space Base Offset** description in VS_STATE. (See *3D Pipeline* for further description on scratch space allocation). Format = GeneralStateOffset[31:10]
	9:0	**FFTID.** This ID is assigned by the FF unit and used to identify the thread within the set of outstanding threads spawned by the FF unit. Format: Reserved for HW Implementation Use.
R0.4	31:5	**Binding Table Pointer.** Specifies the 32-byte aligned pointer to the Binding Table. It is specified as an offset from the Surface State Base Address. Format = SurfaceStateOffset[31:5]
	4:0	Reserved
R0.3	31:5	**Sampler State Pointer.** Specifies the location of the Sampler State Table to be used by this thread, specified as a 32-byte granular offset from the General State Base Address or Dynamic State Base Address. Format = DynamicStateOffset[31:5]
	4	Reserved
	3:0	**Per Thread Scratch Space.** Specifies the amount of scratch space allowed to be used by this thread. The value specifies the power that two will be raised to (over determine the amount of scratch space). Format = U4 power of two (in excess of 10) Range = [0,11] indicating [1K Bytes, 2M Bytes]
R0.2	31:0	Reserved: delivered as zeros (reserved for message header fields)
R0.1	31:0	**PrimitiveID.** This is the 32-bit PrimitiveID value associated with the patch. It is common to all output domain points resulting from the tessellation of the patch.

DWord	Bits	Description
		Format: U32 **Programming Note** The contents of this field is UNDEFINED if the 3DSTATE_DS::PrimitiveIDNotRequired state bit is set.
R0.0	31:27	Reserved
	26:16	**Description** Reserved
	15:0	**Patch URB Offset.** This is the offset within the URB where the patch data is stored. Format: U14 64B-granular offset into the URB
R1.7	31:0	**Domain Point 7 U Coordinate.** (See Domain Point 0 U Coordinate.)
R1.6	31:0	**Domain Point 6 U Coordinate.** (See Domain Point 0 U Coordinate.)
R1.5	31:0	**Domain Point 5 U Coordinate.** (See Domain Point 0 U Coordinate.)
R1.4	31:0	**Domain Point 4 U Coordinate.** (See Domain Point 0 U Coordinate.)
R1.3	31:0	**Domain Point 3 U Coordinate.** (See Domain Point 0 U Coordinate.)
R1.2	31:0	**Domain Point 2 U Coordinate.** (See Domain Point 0 U Coordinate.)
R1.1	31:0	**Domain Point 1 U Coordinate.** (See Domain Point 0 U Coordinate.)
R1.0	31:0	**Domain Point 0 U Coordinate.** U coordinate associated with Domain Point 0. Format: FLOAT32
R2.7	31:0	**Domain Point 7 V Coordinate.** (See Domain Point 0 V Coordinate.)
R2.6	31:0	**Domain Point 6 V Coordinate.** (See Domain Point 0 V Coordinate.)
R2.5	31:0	**Domain Point 5 V Coordinate.** (See Domain Point 0 V Coordinate.)
R2.4	31:0	**Domain Point 4 V Coordinate.** (See Domain Point 0 V Coordinate.)
R2.3	31:0	**Domain Point 3 V Coordinate.** (See Domain Point 0 V Coordinate.)
R2.2	31:0	**Domain Point 2 V Coordinate.** (See Domain Point 0 V Coordinate.)
R2.1	31:0	**Domain Point 1 V Coordinate.** (See Domain Point 0 V Coordinate.)
R2.0	31:0	**Domain Point 0 V Coordinate.** V coordinate associated with Domain Point 0. Format: FLOAT32
R3.7	31:0	**Domain Point 7 W Coordinate.** (See Domain Point 0 W Coordinate.)
R3.6	31:0	**Domain Point 6 W Coordinate.** (See Domain Point 0 W Coordinate.)
R3.5	31:0	**Domain Point 5 W Coordinate.** (See Domain Point 0 W Coordinate.)
R3.4	31:0	**Domain Point 4 W Coordinate.** (See Domain Point 0 W Coordinate.)
R3.3	31:0	**Domain Point 3 W Coordinate.** (See Domain Point 0 W Coordinate.)
R3.2	31:0	**Domain Point 2 W Coordinate.** (See Domain Point 0 W Coordinate.)

DWord	Bits	Description
R3.1	31:0	**Domain Point 1 W Coordinate.** (See Domain Point 0 W Coordinate.)
R3.0	31:0	**Domain Point 0 W Coordinate.** If Compute W Coordinate Enable is set, this field will receive the computed value (1 - U - V) for Domain Point 0. Otherwise it is passed as 0.0. Format: FLOAT32
R4.7	31:0	**Domain Point 7 URB Return Handle.** (See R4.0.)
R4.6	31:0	**Domain Point 6 URB Return Handle.** (See R4.0.)
R4.5	31:0	**Domain Point 5 URB Return Handle.** (See R4.0.)
R4.4	31:0	**Domain Point 4 URB Return Handle.** (See R4.0.)
R4.3	31:0	**Domain Point 3 URB Return Handle.** (See R4.0.)
R4.2	31:0	**Domain Point 2 URB Return Handle.** (See R4.0.)
R4.1	31:0	**Domain Point 1 URB Return Handle.** (See R4.0.)
R4.0	31:16	Reserved
	15:0	**Domain Point 0 URB Return Handle.** This is the offset within the URB where domain point 0 is to be stored. Format: U14 64B-granular offset into the URB
[Varies] optional	255:0	**Constant Data (optional):** Please refer to the Push Constants chapter in the General Programming of Thread-Generating Stages section for more details on size and source of constant data.
Varies [Optional]	255:0	**Patch URB Data (optional).** Some amount of Patch Data (possible none) can be extracted from the URB and passed to the thread in this location in the payload. The amount of data provided is defined by the Patch URB Entry Read Length state (3DSTATE_DS).

DUAL_PATCH Payload

The following table describes the payload delivered to DS threads.

DUAL_PATCH DS Thread Payload (SIMD8)

DWord	Bits	Description
R0.7	30:0	Reserved
R0.6	31:24	Reserved
	23:0	**Thread ID.** This field uniquely identifies this thread within the threads spawned by this FF unit, over some period of time. Format: Reserved for HW Implementation Use.
R0.5	31:10	**Scratch Space Offset.** Specifies the offset of the scratch space allocated to the thread,

DWord	Bits	Description
		specified as a 1KB-granular offset from the **General State Base Address**. See **Scratch Space Base Offset** description in VS_STATE. (See *3D Pipeline* for further description on scratch space allocation). Format = GeneralStateOffset[31:10]
	9:0	**FFTID.** This ID is assigned by the FF unit and used to identify the thread within the set of outstanding threads spawned by the FF unit. Format: Reserved for HW Implementation Use.
R0.4	31:5	**Binding Table Pointer.** Specifies the 32-byte aligned pointer to the Binding Table. It is specified as an offset from the Surface State Base Address. Format = SurfaceStateOffset[31:5]
	4:0	Reserved
R0.3	31:5	**Sampler State Pointer.** Specifies the location of the Sampler State Table to be used by this thread, specified as a 32-byte granular offset from the General State Base Address or Dynamic State Base Address. Format = DynamicStateOffset[31:5]
	4	Reserved
	3:0	**Per Thread Scratch Space.** Specifies the amount of scratch space allowed to be used by this thread. The value specifies the power that two will be raised to (over determine the amount of scratch space). Format = U4 power of two (in excess of 10) Range = [0,11] indicating [1K Bytes, 2M Bytes]
R0.2	31:0	Reserved: delivered as zeros (reserved for message header fields)
R0.1	31:27	Reserved
	26:16	**Description** Reserved
	15:0	**Patch 1 URB Offset.** This is the offset within the URB where the Patch 1 data is stored. Format: U16 64B-granular offset into the URB
R0.0	31:27	Reserved
	26:16	**Description** Reserved
	15:0	**Patch 0 URB Offset.** This is the offset within the URB where the Patch 0 data is stored. Format: U16 64B-granular offset into the URB

DWord	Bits	Description
		The following "R1" GRF is not included in the payload if 3DSTATE_DS::PrimitiveIDNotRequired is set.
R1.5-7	31:0	Reserved
R1.4	31:0	**Patch 1 PrimitiveID.** This is the 32-bit PrimitiveID value associated with Patch 1. Format: U32
R1.1-3	31:0	Reserved
R1.0	31:0	**Patch 0 PrimitiveID.** This is the 32-bit PrimitiveID value associated with Patch 0. Format: U32
R2.7	31:0	**Patch 1 Domain Point 3 U Coordinate.** (See R2.0.)
R2.6	31:0	**Patch 1 Domain Point 2 U Coordinate.** (See R2.0.)
R2.5	31:0	**Patch 1 Domain Point 1 U Coordinate.** (See R2.0.)
R2.4	31:0	**Patch 1 Domain Point 0 U Coordinate.** (See R2.0.)
R2.3	31:0	**Patch 0 Domain Point 3 U Coordinate.** (See R2.0.)
R2.2	31:0	**Patch 0 Domain Point 2 U Coordinate.** (See R2.0.)
R2.1	31:0	**Patch 0 Domain Point 1 U Coordinate.** (See R2.0.)
R2.0	31:0	**Patch 0 Domain Point 0 U Coordinate.** U coordinate associated with Domain Point 0 of Patch 0. Format: FLOAT32
R3.7	31:0	**Patch 1 Domain Point 3 V Coordinate.** (See R3.0.)
R3.6	31:0	**Patch 1 Domain Point 2 V Coordinate.** (See R3.0.)
R3.5	31:0	**Patch 1 Domain Point 1 V Coordinate.** (See R3.0.)
R3.4	31:0	**Patch 1 Domain Point 0 V Coordinate.** (See R3.0.)
R3.3	31:0	**Patch 0 Domain Point 3 V Coordinate.** (See R3.0.)
R3.2	31:0	**Patch 0 Domain Point 2 V Coordinate.** (See R3.0.)
R3.1	31:0	**Patch 0 Domain Point 1 V Coordinate.** (See R3.0.)
R3.0	31:0	**Patch 0 Domain Point 0 V Coordinate.** V coordinate associated with Domain Point 0. Format: FLOAT32
R4.7	31:0	**Patch 1 Domain Point 3 W Coordinate.** (See R4.0.)
R4.6	31:0	**Patch 1 Domain Point 2 W Coordinate.** (See R4.0.)
R4.5	31:0	**Patch 1 Domain Point 1 W Coordinate.** (See R4.0.)
R4.4	31:0	**Patch 1 Domain Point 0 W Coordinate.** (See R4.0.)
R4.3	31:0	**Patch 0 Domain Point 3 W Coordinate.** (See R4.0.)
R4.2	31:0	**Patch 0 Domain Point 2 W Coordinate.** (See R4.0.)
R4.1	31:0	**Patch 0 Domain Point 1 W Coordinate.** (See R4.0.)

DWord	Bits	Description
R4.0	31:0	**Patch 0 Domain Point 0 W Coordinate.** If Compute W Coordinate Enable is set, this field will receive the computed value (1 - U - V) for Domain Point 0. Otherwise it is passed as 0.0. Format: FLOAT32
R5.7	31:0	**Patch 1 Domain Point 3 URB Return Handle.** (See R5.0.)
R5.6	31:0	**Patch 1 Domain Point 2 URB Return Handle.** (See R5.0.)
R5.5	31:0	**Patch 1 Domain Point 1 URB Return Handle.** (See R5.0.)
R5.4	31:0	**Patch 1 Domain Point 0 URB Return Handle.** (See R5.0.)
R5.3	31:0	**Patch 0 Domain Point 3 URB Return Handle.** (See R5.0.)
R5.2	31:0	**Patch 0 Domain Point 2 URB Return Handle.** (See R5.0.)
R5.1	31:0	**Patch 0 Domain Point 1 URB Return Handle.** (See R5.0.)
R5.0	31:16	Reserved
	15:0	**Patch 0 Domain Point 0 URB Return Handle.** This is the offset within the URB where Patch 0 Domain Point 0 is to be stored. Format: U16 64B-granular offset into the URB
[Varies] optional	255:0	**Constant Data (optional):** Please refer to the Push Constants chapter in the General Programming of Thread-Generating Stages section for more details on size and source of constant data.
		Patch 0,1 URB Data follows (optional). This data is read from the URB and pushed in the payload. The amount of data provided for each patch (which may be 0) is defined by the Patch URB Entry Read Length state (3DSTATE_DS). The data is read from the URB starting at the Patch URB Entry Read Offset into each patch, so leading data with the Patch URB entries can be skipped over. Patch 1 data is passed in the upper 128 bits, while Patch 0 data is passed in the lower 128 bits. This is similar to how URB data is pushed into SIMD4x2 kernels (VS, GS, etc.).
[Varies] optional	255:128	**Patch 1 URB Data.**
	127:0	**Patch 0 URB Data.**

Geometry Shader (GS) Stage

GS Stage Overview

The GS stage of the 3D Pipeline converts objects within incoming primitives into new primitives through use of a spawned thread. When enabled, the GS unit buffers incoming vertices, assembles the vertices of each individual object within the primitives, and passes those object vertices (along with other data) to the graphics subsystem for processing by a GS thread.

When the GS stage is disabled, vertices flow through the unit unmodified.

Refer to the *Common 3D FF Unit Functions* subsection in the *3D Pipeline* chapter for a general description of a 3D Pipeline stage, as much of the GS stage operation and control falls under these "common" functions. I.e., most stage state variables and GS thread payload parameters are described in *3D Pipeline*, and although they are listed here for completeness, that chapter provides the detailed description of the associated functions.

Refer to this chapter for an overall description of the GS stage, and any exceptions the GS stage exhibits with respect to common FF unit functions.

State

This section contains the state registers for the Geometry Shader.

Registers
3DSTATE_GS (The state used by GS is defined with this inline state packet.)
3DSTATE_CONSTANT_GS
3DSTATE_CONSTANT(Body)
3DSTATE_PUSH_CONSTANT_ALLOC_GS
3DSTATE_BINDING_TABLE_POINTERS_GS
3DSTATE_SAMPLER_STATE_POINTERS_GS
3DSTATE_URB_GS

Functions

Object Staging

The GS unit's Object Staging Buffer (OSB) accepts primitive topologies as a stream of incoming vertices, and spawns a thread for each individual object within the topology.

Thread Request Generation

Object Vertex Ordering

The following table defines the number and order of object vertices passed in the Vertex Data portion of the GS thread payload, assuming an input topology with *N* vertices. The ObjectType passed to the thread is, by default, the incoming PrimTopologyType. Exceptions to this rule (for the TRISTRIP variants) are called out.

The following table also shows which vertex is selected to provide PrimitiveID (bold, underlined vertex number). In general, the vertex selected is the last vertex for non-adjacent prims, and the next-to-last vertex for adjacent prims. Note, however, that there are exceptions:

- reorder-enabled TRISTRIP[_REV], TRISTRIP_ADJ
- "odd-numbered" objects in TRISTRIP_ADJ

PrimTopologyType	Order of Vertices in Payload	GS Notes
<PRIMITIVE_TOPOLOGY> (N = # of vertices)	[<object#>] = (<vert#>,...); [{modified PrimType passed to thread}]	
POINTLIST	[0] = (0); [1] = (1); ...; [N-2] = (N-2);	
POINTLIST_BF	N/A	
LINELIST (N is multiple of 2)	[0] = (0,1); [1] = (2,3); ...; [(N/2)-1] = (N-2,N-1)	
LINELIST_ADJ (N is multiple of 4)	[0] = (0,1,2,3); [1] = (4,5,6,7); ...; [(N/4)-1)] = (N-4,N-3,N-2,N-1)	
LINESTRIP (N >= 2)	[0] = (0,1); [1] = (1,2); ...; [N-2] = (N-2,N-1)	
LINESTRIP_ADJ, LINESTRIP_ADJ_CONT (N >= 4)	[0] = (0,1,2,3); [1] = (1,2,3,4); ...; [N-4] = (N-4,N-3,N-2,N-1)	LINESTRIP_ADJ_CONT is generated by the Vertex Fetch unit on a restore of a mid-draw pre-empted 3DPRIMITIVE.
LINESTRIP_BF	N/A	
LINESTRIP_CONT	Same as LINESTRIP	Handled same as LINESTRIP
LINESTRIP_CONT_BF	N/A	
LINELOOP (N >= 2)	[0] = (0,1); [1] = (1,2); [N] = (N-1,0);	Not supported after GS.
TRILIST (N is multiple of 3)	[0] = (0,1,2); [1] = (3,4,5); ...; [(N/3)-1] = (N-3,N-2,N-1)	
RECTLIST, RECTLIST_SUBPIXEL	Same as TRILIST	Handled same as TRILIST

PrimTopologyType <PRIMITIVE_TOPOLOGY> (N = # of vertices)	Order of Vertices in Payload [<object#>] = (<vert#>,...); [{modified PrimType passed to thread}]	GS Notes
TRILIST_ADJ (N is multiple of 6)	[0] = (0,1,2,3,4,5); [1] = (6,7,8,9,10,11); ...; [(N/6)-1] = (N-6,N-5,N-4,N-3,N-2,N-1)	
TRISTRIP (Reorder Leading) (N >= 3)	[0] = (0,1,2); {TRISTRIP} [1] = (1,3,2); {TRISTRIP_REV} [k even] = (k,k+1,k+2) {TRISTRIP} [k odd] = (k,k+2,k+1) {TRISTRIP_REV} [N-3] = (see above)	"Odd" triangles have vertices reordered and identified as TRISTRIP to inform the thread.
TRISTRIP (Reorder Trailing) (N >= 3)	[0] = (0,1,2) {TRISTRIP} [1] = (2,1,3) {TRISTRIP_REV}; ... [k even] = (k,k+1,k+2) {TRISTRIP} [k odd] = (k+1,k,k+2) {TRISTRIP_REV} [N-3] = (see above)	"Odd" triangles have vertices reordered and identified as TRISTRIP_REV to inform the thread.
TRISTRIP_REV (Reorder Leading) (N >= 3)	[0] = (0,2,1) {TRISTRIP_REV}; [1] = (1,2,3) {TRISTRIP}; ...; [k even] = (k,k+2,k+1) {TRISTRIP_REV} [k odd] = (k,k+1,k+2) {TRISTRIP} [N-3] = (see above)	"Even" triangles have vertices reordered and identified as TRISTRIP to inform the thread.
TRISTRIP_REV (Reorder Trailing) (N >= 3)	[0] = (1,0,2) {TRISTRIP_REV} [1] = (1,2,3) {TRISTRIP}; ...; [k even] = (k+1,k,k+2,) {TRISTRIP_REV} [k odd] = (k,k+1,k+2) {TRISTRIP}	"Even" triangles have vertices reordered and identified as TRISTRIP_REV to inform the thread.

PrimTopologyType <PRIMITIVE_TOPOLOGY> (N = # of vertices)	Order of Vertices in Payload [<object#>] = (<vert#>,...); [{modified PrimType passed to thread}]	GS Notes
	[N-3] = (see above)	
TRISTRIP_ADJ (Reorder Leading) (N >= 6)	N = 6 or 7: [0] = (0,1,2,5,4,3) N = 8 or 9: [0] = (0,1,2,6,4,3); [1] = (2,5,6,7,4,0); ...; N >= 10: [0] = (0,1,2,6,4,3); [1] = (2,5,6,8,4,0); ...; [k>1, even] = (2k,2k-2, 2k+2, 2k+6,2k+4, 2k+3); [k>2, odd] = (2k, 2k+3, 2k+4, 2k+6, 2k+2, 2k-2);...; Trailing object: [(N/2)-3, even] = (N-6,N-8,N-4,N-1,N-2,N-3); [(N/2)-3, odd] = (N-6,N-3,N-2,N-1,N-4,N-8);	Objects have vertices reordered.
TRISTRIP_ADJ (Reorder Trailing) (N >= 6)	N = 6 or 7: [0] = (0,1,2,5,4,3) N = 8 or 9: [0] = (0,1,2,6,4,3); [1] = (4,0,2,5,6,7); ...; N >= 10: [0] = (0,1,2,6,4,3); [1] = (4,0,2,5,6,8); ...; [k>1, even] = (2k,2k-2, 2k+2, 2k+6,2k+4, 2k+3); [k>2, odd] = (2k+2, 2k-2, 2k, 2k+3, 2k+4, 2k+6);...; Trailing object: [(N/2)-3, even] = (N-6,N-	OpenGL ordering rules (last non-adjacent vertex is the last - aka provoking - vertex of the triangle). Even triangles have the same ordering as Leading Vertex, odd triangle ordering is different (rotated 2 vertices).

PrimTopologyType <PRIMITIVE_TOPOLOGY> (N = # of vertices)	Order of Vertices in Payload [<object#>] = (<vert#>,...); [{modified PrimType passed to thread}]	GS Notes
	8,N-4,N-1,N-2,N-3); [(N/2)-3, odd] = (N-4,N-8,N-6,N-3,N-2,N-1);	
TRIFAN (N > 2)	[0] = (0,1,2); [1] = (0,2,3); ...; [N-3] = (0, N-2, N-1);	
TRIFAN_NOSTIPPLE	Same as TRIFAN	
POLYGON, POLYGON_CONT	Same as TRIFAN	POLYGON_CONT is generated by the Vertex Fetch unit on a restore of a mid-draw pre-empted 3DPRIMITIVE.
QUADLIST	[0] = (0,1,2,3); [1] = (4,5,6,7); ...; [(N/4)-1] = (N-4,N-3,N-2,N-1);	Not supported after GS. QUADLIST primitives are converted into POLYGONS in VF, and therefore never reach the GS.
QUADSTRIP	[0] = (0,1,3,2); [1] = (2,3,5,4); ... ; [(N/2)-2] = (N-4,N-3,N-1,N-2);	Not supported after GS. QUADSTRIP primitives are converted into POLYGONS in VF, and therefore never reach the GS.

PrimTopologyType	Order of Vertices in Payload
PATCHLIST_1 PATCHLIST_2 PATCHLIST_3..32	[0] = (0); [1] = (1); ...; [N-2] = (N-2); [0] = (0,1); [1] = (2,3); ...; [(N/2)-1] = (N-2,N-1) similar to above

Thread Execution

A GS thread is capable of performing arbitrary algorithms given the thread payload (especially vertex) data and associated data structures (binding tables, sampler state, etc.) as input. Output can take the form of vertices output to the FF pipeline (at the GS unit) or data written to memory buffers (UAVs).

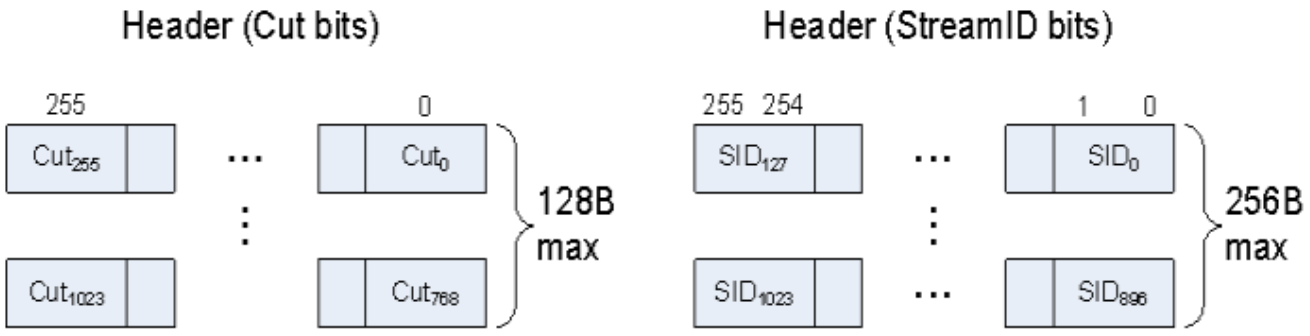
The primary usage models for GS threads include (possible combinations of):

- Compiled application-provided "GS shader" programs, specifying an algorithm to convert the vertices of an input object into some output primitives. For example, a GS shader may convert lines of a line strip into polygons representing a corresponding segment of a blade of grass centered on the line. Or it could use adjacency information to detect silhouette edges of triangles and output polygons extruding out from those edges. Or it could output absolutely nothing, effectively terminating the pipeline at the GS stage.
- Driver-generated instructions used to write pre-clipped vertices into memory buffers (see Stream Output below). This may be required whether or not an app-provided GS shader is enabled.
- Driver-generated instructions used to emulate API functions not supported by specialized hardware. These functions might include (but are not limited to):
    - Conversion of API-defined topologies into topologies that can be rendered (e.g., LINELOOP, LINESTRIP, POLYGON, TRIFAN, QUADs, TRIFAN, etc.)
    - Emulation of "Polygon Fill Mode", where incoming polygons can be converted to points, lines (wireframe), or solid objects.
    - Emulation of wide/sprite points.

Thread Execution

GS URB Entry

All outputs of a GS thread are stored in the single GS thread output URB entry. Cut (1 bit/vertex) or StreamID (2 bits/vertex) bits are packed into an optional 1-8 32B header. The **Control Data Format** and **Control Data Header Size** states specify the size and contents of the header data (if any).



Following the optional header is a variable number of 16B or 32B-aligned/granular vertices:

- When rendering is DISABLED, typically output vertices are 32B-aligned, with the exception of 16B-alignment for vertices <= 16B in length.
    - The absolute worst-case size comes from three DW scalars output per vertex. If these are, say, three ".x" outputs, you need to store each DW in a 128b (16B) element, plus another pad 16B to keep the 32B alignment. So you require 4*16B = 64B/vertex. You have to have room for 1024 scalars / 3 scalar/vtx = 341 vertices. 341*64B = 21,824B. Then add 96B to hold 2b/vtx streamID and you get 21,920B entries.
- When rendering is ENABLED, each output vertex is 32B-aligned. Here the vertex header and vertex 'position' are required and therefore the minimum size vertex is 32B.
    - Here the worst-case size isn't as bad as render-disabled, as you have to have a 4DW position output, plus any additional output. So, say you output 5 DW per vertex. You need 64B/vertex (16B vtx header, 16B position, 16B for the 2nd element, and 16B of pad). You have to have room for 1024 scalars / 5 = 204 vertices. 204*64 = 13,056B. Then add 64B to hold 2b/vtx streamID and you get 13,120B entries.

The size of the URB entry should be based on the declared maximum # of output vertices and the declared output vertex size (the union of per-stream vertex structures, if required).

GS URB Entry - Output Vertex Count

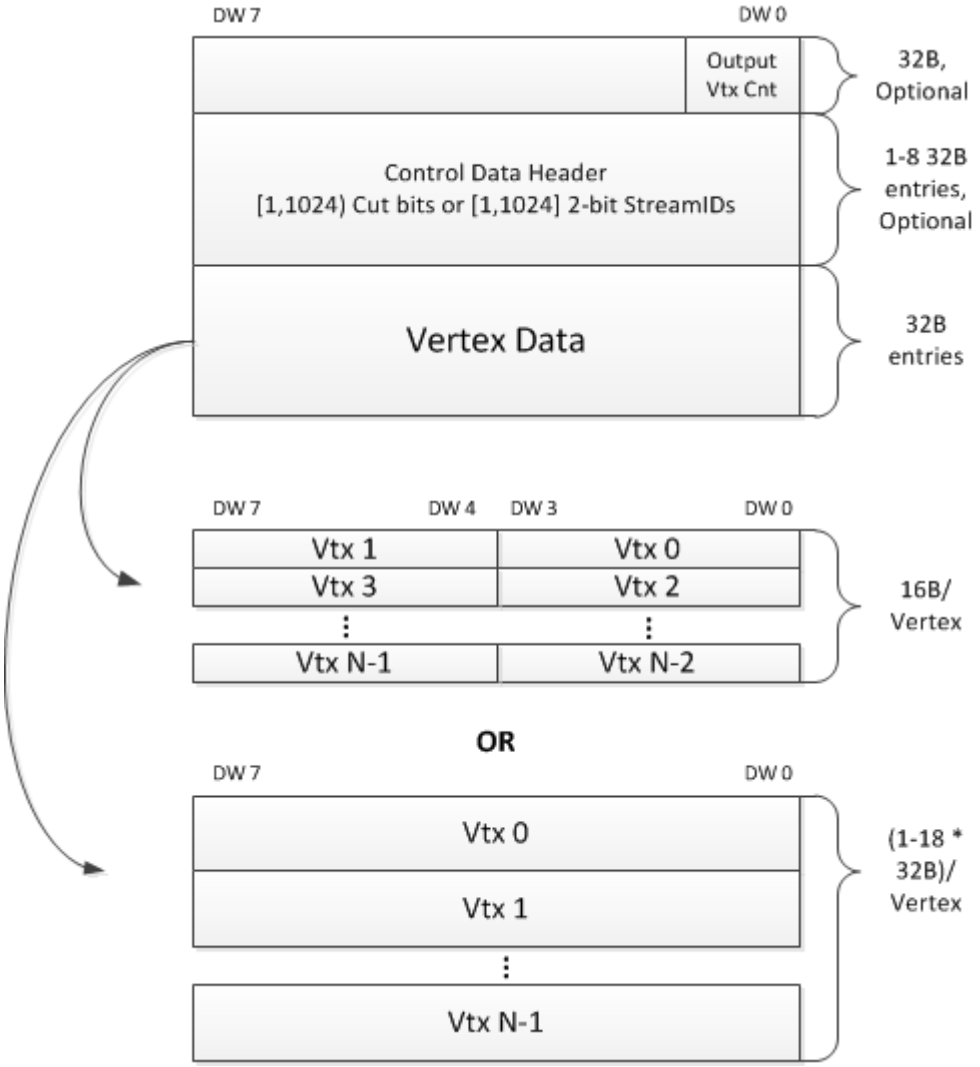
The GS URB entry is the same as in the two previous generations with the following exception: If **Static Output** (3DSTATE_GS) is clear, the URB entry starts with a 32B OUTPUT_VERTEX_COUNT structure as defined below. The control header (if present) immediately follows this structure. If **Static Output** is set, the control header (if present) appears at the very start of the URB entry (as described above).

GS OUTPUT_VERTEX_COUNT

DWord	Bit	Description
7:6	31:0	Reserved
0	31:16	Reserved
	15:0	**Output Vertex Count.** Indicates the number of vertices output from this GS shader invocation. Format = U16 Range: [0:1024]

This structure (if present) increases the maximum URB entry sizes (described above) by 32B.

The following diagram illustrates the possible layouts of a GS URB Entry:



GS Output Topologies

The following table lists which primitive topology types are valid for output by a GS thread.

PrimTopologyType	Supported for GS Thread Output?
LINELIST	Yes
LINELIST_ADJ	No
LINESTRIP	Yes
LINESTRIP_ADJ	No
LINESTRIP_BF	No
LINESTRIP_CONT	Yes
LINESTRIP_CONT_BF	No
LINELOOP	No
POINTLIST	Yes
POINTLIST_BF	No

PrimTopologyType	Supported for GS Thread Output?
POLYGON	Yes
QUADLIST	No
QUADSTRIP	No
RECTLIST	Yes
TRIFAN	Yes
TRIFAN_NOSTIPPLE	Yes
TRILIST	Yes
TRILIST_ADJ	No
TRISTRIP	Yes
TRISTRIP_ADJ	No
TRISTRIP_REV	Yes
PATCHLIST_xxx	Yes

GS Output StreamID

When the **GS Enable** is DISABLED, output vertices are assigned a StreamID = 0;

When the **GS Enable** is ENABLED, output vertices are assigned a StreamID = **Default StreamID** under the following conditions:

- **Control Data Header Size** = 0, or
- **Control Data Header Size** > 0 and **Control Data Format** = GSCTL_CUT

When the GS is enabled, **Control Data Header Size** > 0 and **Control Data Format** = GSCTL_SID, output vertices are assigned a StreamID as programmed in the Control Data output by the thread.

Primitive Output

(This section refers to output from the GS unit to the pipeline, not output from the GS thread)

The GS unit will output primitives (either passed-through or generated by a GS thread) in the proper order. This includes the buffering of a concurrent GS thread's output until the preceding GS thread terminates. Note that the requirement to buffer subsequent GS thread output until the preceding GS thread terminates has ramifications on determining the number of VUEs allocated to the GS unit and the number of concurrent GS threads allowed.

Statistics Gathering

There are a number of GS pipeline statistics counters associated with the GS stage and GS threads. This subsection describes these counters and controls depending on device, even in the cases where functions outside of the GS stage (e.g., DataPort) are involved in the statistics gathering.

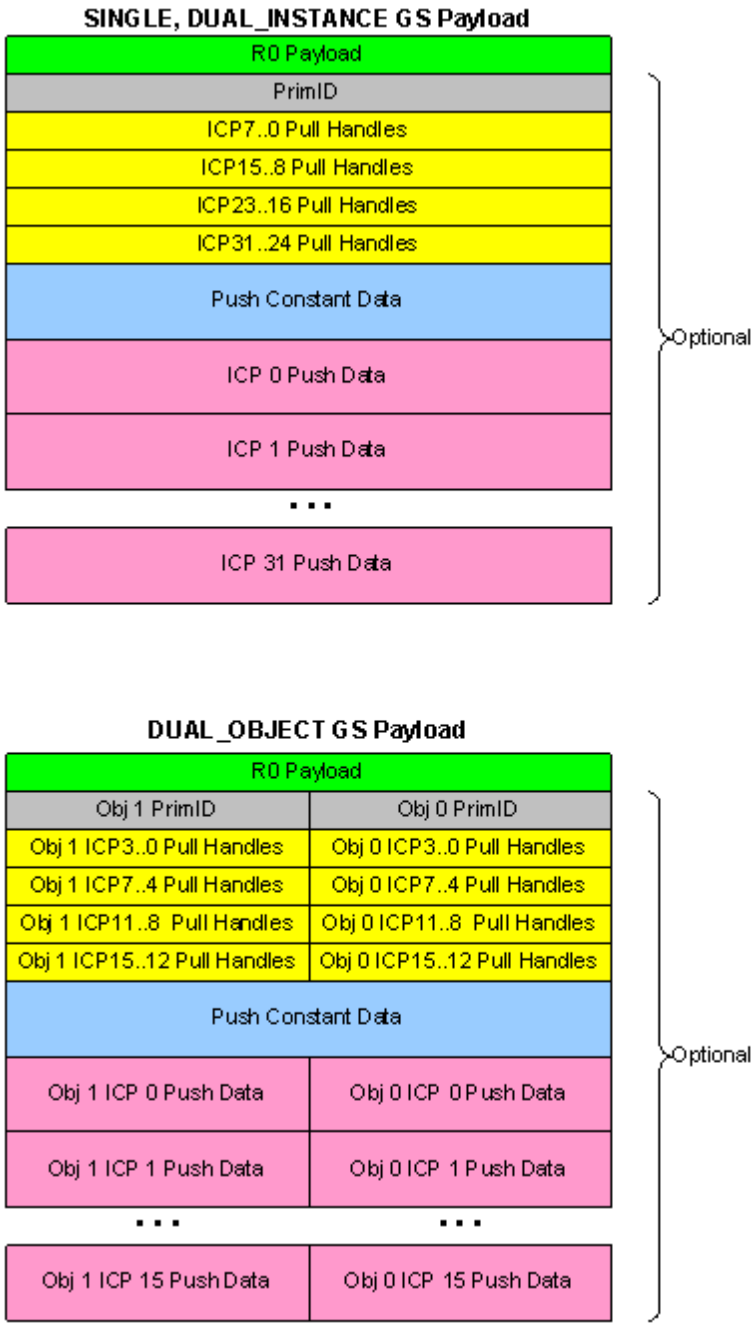
Refer to the *Statistics Gathering* summary provided earlier in this specification. Refer to the *Memory Interface Registers* chapter for details on these MMIO pipeline statistics counter registers, as well as the chapters corresponding to the other functions involved (e.g., DataPort, URB shared functions).

Payloads

Thread Payload High-Level Layout

GS Dispatch Layouts shows the high-level layout of the payload delivered to GS threads.

GS Dispatch Layouts



Subsequent sections provide detailed layouts for different processor projects.

SIMD8 Thread Payload

The table below shows the layout of the payload delivered to GS threads.

GRF DWord	Bits	Description
R0.7	31	
	30:0	Reserved.
R0.6	31	**Early Dereference Enable**: The enabe of early dereference. Reserved for Implementation Use
	30:24	Reserved.
	23:0	**Thread ID.** This field uniquely identifies this thread within the threads spawned by this FF unit, over some period of time. Format: Reserved for HW Implementation Use.
R0.5	31:10	**Scratch Space Pointer.** Specifies the location of the scratch space allocated to this thread, specified as a 1KB-aligned offset from the General State Base Address. Format = GeneralStateOffset[31:10]
	9:0	**FFTID.** This ID is assigned by the fixed function unit and is relative identifier for the thread. It is used to free up resources used by the thread upon thread completion. Format: Reserved for Implementation Use
R0.4	31:5	**Binding Table Pointer.** Specifies the 32-byte aligned pointer to the Binding Table. It is specified as an offset from the Surface State Base Address. Format = SurfaceStateOffset[31:5]
	4:0	Reserved.
R0.3	31:5	**Sampler State Pointer.** Specifies the location of the Sampler State Table to be used by this thread, specified as a 32-byte granular offset from the Dynamic State Base Address. Format = DynamicStateOffset[31:5]
	4	Reserved.
	3:0	**Per Thread Scratch Space.** Specifies the amount of scratch space allowed to be used by this thread. The value specifies the power that two is raised to (over determine the amount of scratch space). (See Generic Pipeline Stage for further description). **Programming Notes:** This amount is available to the kernel for information only. It is passed verbatim (if not altered by the kernel) to the Data Port in any scratch space access messages, but the Data Port ignores it.

GRF DWord	Bits	Description
		Format = U4 power of two (in excess of 10) Range = [0,11] indicating [1K Bytes, 2M Bytes]
R0.2	31:23	Reserved.
	22	**Hint:** This is a copy of the corresponding 3DSTATE_GS bit. Format: U1
	21:16	**Primitive Topology Type.** This field identifies the Primitive Topology Type associated with the primitive containing this object. It indirectly specifies the number of input vertices included in the thread payload. Note that the GS unit may toggle this value between TRISTRIP and TRISTRIP_REV. If the **Discard Adjacency** bit is set, the topology type passed in the payload is UNDEFINED. Format: See 3D Pipeline
	15:0	Reserved.
R0.1-R0.0	31:0	Reserved.
R1.7	31:0	GS Instance ID / URB Return Handle for Object 7 (See R1.0)
R1.6	31:0	GS Instance ID / URB Return Handle for Object 6 (See R1.0)
R1.5	31:0	GS Instance ID / URB Return Handle for Object 5 (See R1.0)
R1.4	31:0	GS Instance ID / URB Return Handle for Object 4 (See R1.0)
R1.3	31:0	GS Instance ID / URB Return Handle for Object 3 (See R1.0)
R1.2	31:0	GS Instance ID / URB Return Handle for Object 2 (See R1.0)
R1.1	31:0	GS Instance ID / URB Return Handle for Object 1 (See R1.0)
R1.0	31:27	**GS Instance ID 0.** For each input object, the GS unit can spawn multiple threads (instances). This field starts at zero for the first instance of an object and increments for subsequent instances. Format: U5
	26:16	Reserved.
	15:0	**URB Return Handle 0.** This is the URB offset where the EU's lower channels (DWords 3:0) results are to be stored. Format: U16 64B-aligned URB Offset
The following register is included only if **Include PrimitiveID** is enabled.		
R2.7	31:0	**Primitive ID 7.** This field contains the Primitive ID associated with input object 7 (or the single input object if InstanceCount>1) Format: U32
R2.6	31:0	**Primitive ID 6.** This field contains the Primitive ID associated with input object 6 (or the single input object if InstanceCount>1)

GRF DWord	Bits	Description
		Format: U32
R2.5	31:0	**Primitive ID 5.** This field contains the Primitive ID associated with input object 5 (or the single input object if InstanceCount>1) Format: U32
R2.4	31:0	**Primitive ID 4.** This field contains the Primitive ID associated with input object 4 (or the single input object if InstanceCount>1) Format: U32
R2.3	31:0	**Primitive ID 3.** This field contains the Primitive ID associated with input object 3 (or the single input object if InstanceCount>1) Format: U32
R2.2	31:0	**Primitive ID 2.** This field contains the Primitive ID associated with input object 2 (or the single input object if InstanceCount>1) Format: U32
R2.1	31:0	**Primitive ID 1.** This field contains the Primitive ID associated with input object 1 (or the single input object if InstanceCount>1) Format: U32
R2.0	31:0	**Primitive ID 0.** This field contains the Primitive ID associated with input object 0 (or the single input object if InstanceCount>1) Format: U32
The following registers are included only if **Include Vertex Handles** is enabled and **InstanceCount** == 1.		
Rn.7	31:16	Reserved.
	15:0	Object 7 ICP 0 Handle
Rn.6	31:16	Reserved.
	15:0	Object 6 ICP 0 Handle
Rn.5	31:16	Reserved.
	15:0	Object 5 ICP 0 Handle
Rn.4	31:16	Reserved.
	15:0	Object 4 ICP 0 Handle
Rn.3	31:16	Reserved.
	15:0	Object 3 ICP 0 Handle
Rn.2	31:16	Reserved.
	15:0	Object 2 ICP 0 Handle

GRF DWord	Bits	Description
Rn.1	31:16	Reserved.
	15:0	Object 1 ICP 0 Handle
Rn.0	31:16	Reserved.
	15:0	Object 0 ICP 0 Handle
[Rn+1]	255:0	ICP 1 Handle for Objects 0-7
[Rn+2]	255:0	ICP 2 Handle for Objects 0-7
...		...
[Rn+32]	255:0	ICP 32 Handle for Objects 0-7
The following registers are included only if **Include Vertex Handles** is enabled and **InstanceCount** > 1.		
Rn.7	31:16	Reserved.
	15:0	ICP 7 Handle (if required)
Rn.6	31:16	Reserved.
	15:0	ICP 6 Handle (if required)
Rn.5	31:16	Reserved.
	15:0	ICP 5 Handle (if required)
Rn.4	31:16	Reserved.
	15:0	ICP 4 Handle (if required)
Rn.3	31:16	Reserved.
	15:0	ICP 3 Handle (if required)
Rn.2	31:16	Reserved.
	15:0	ICP 2 Handle (if required)
Rn.1	31:16	Reserved.
	15:0	ICP 1 Handle (if required)
Rn.0	31:16	Reserved.
	15:0	ICP 0 Handle
[Rn+1]	255:0	ICP 8-15 Handle
[Rn+2]	255:0	ICP 16-23 Handle
[Rn+3]	255:0	ICP 24-31 Handle
[Varies] optional	255:0	**Constant Data (optional):** Please refer to the Push Constants chapter in the General Programming of Thread-Generating Stages section for more details on size and source of constant data.
Varies		**Pushed Vertex Data (InstanceCount == 1 Case): (optional)** Input data for the 8 input objects is located here. Object 0 (starting with Vertex 0 of Object 0) data is passed in DW0 of these GRFs, and Object 7 data is passed in DW7. The first GRF contains Vertex 0 Element 0 Component 0 for all 8 objects, followed by components 1-3 in the three subsequent GRFs. This is followed by GRFs containing Vertex 0 Element 1 (if it exists), and so on,

GRF DWord	Bits	Description
		up to the number of Vertex 0 elements specified by **Vertex URB Read Length**. This is followed by the data for Vertex 1 for all objects (if it exists), and so on until all relevant vertices are passed. Note that the amount of data passed is limited by the number of GRFs supported by EUs. Software is responsible for comprehending this limit and resorting to the pull model as required.
Rv.7	31:0	**Object 7 Vertex 0 Element 0 Component 0**
Rv.6	31:0	**Object 6 Vertex 0 Element 0 Component 0**
Rv.5	31:0	**Object 5 Vertex 0 Element 0 Component 0**
Rv.4	31:0	**Object 4 Vertex 0 Element 0 Component 0**
Rv.3	31:0	**Object 3 Vertex 0 Element 0 Component 0**
Rv.2	31:0	**Object 2 Vertex 0 Element 0 Component 0**
Rv.1	31:0	**Object 1 Vertex 0 Element 0 Component 0**
Rv.0	31:0	**Object 0 Vertex 0 Element 0 Component 0**
Rv+1	31:0	**Object 0-7 Vertex 0 Element 0 Component 1**
...		and so on...
Varies		**Pushed Vertex Data (InstanceCount > 1 Case): (optional)** Input data for the single input object (shared across all instances) is located here. The pushed data for Vertex 0 immediately follows any pushed constant data. The pushed data for Vertex 1 immediately follows Vertex 0, and so on. There is no upper/lower swizzling of data.

Stream Output Logic (SOL) Stage

The Stream Output Logic (SOL) stage receives 3D topologies originating in the VF, DS or GS stage. If enabled, the SOL stage uses programmed state information to copy portions of the vertex data associated with the incoming topologies across one or more Stream Output (SO) Buffers.

State

This section contains state commands and structures pertaining to the StreamOut Logic (SOL) stage of the 3D pipeline.

3DSTATE_STREAMOUT

The 3DSTATE_STREAMOUT command specifies control information for the SOL stage. Included are enables and sizes for input streams and enables for output buffers.

3DSTATE_STREAMOUT

3DSTATE_SO_DECL_LIST Command

The 3DSTATE_SO_DECL_LIST instruction defines a list of Stream Output (SO) declaration entries (SO_DECLs) and associated information for all specific SO streams in parallel.

3DSTATE_SO_DECL_LIST

SO_DECL

3DSTATE_SO_BUFFER

The 3DSTATE_SO_BUFFER command specifies the location and characteristics of an SO buffer in memory.

Command
3DSTATE_SO_BUFFER
3DSTATE_SO_BUFFER_INDEX_0
3DSTATE_SO_BUFFER_INDEX_1
3DSTATE_SO_BUFFER_INDEX_2
3DSTATE_SO_BUFFER_INDEX_3

Signal	Description	Formula

SOL_INT::Render_Enable	If clear, the SO stage will not forward any topologies down the pipeline.
	If set, the SO stage will forward topologies associated with Render Stream Select down the pipeline.
	This bit is used even if SO Function Enable is DISABLED.
		`= (3DSTATE_STREAMOUT::Force_Rending == Force_On) ||`
		`(`
		`(3DSTATE_STREAMOUT::Force_Rending != Force_Off) &&`
		`!(3DSTATE_GS::Enable && 3DSTATE_GS::Output Vertex Size == 0) &&`
		`!3DSTATE_STREAMOUT::API_Render_Disable &&`
		`(`
		`3DSTATE_DEPTH_STENCIL_STATE::Stencil_TestEnable ||`
		`3DSTATE_DEPTH_STENCIL_STATE::Depth_TestEnable ||`
		`3DSTATE_DEPTH_STENCIL_STATE::Depth_WriteEnable ||`
		`3DSTATE_PS_EXTRA::PS_Valid ||`
		`3DSTATE_WM::Legacy Depth_Buffer_Clear ||`
		`3DSTATE_WM::Legacy Depth_Buffer_Resolve_Enable ||`
		`3DSTATE_WM::Legacy Hierarchical_Depth_Buffer_Resolve_Enable`
		`)`
		`)`

DW1[21]	DW1[20]	Stream Offset	Action
Full legacy mode. HW doesn't LOAD or STORE, it simply updates the MMIO register during stream out. SW can the LOAD/STORE using MI_LOAD_REG/ MI_STORE_REG.			
0	0	not equal to 0xFFFFFFFF	SO_WRITE_OFFSET[x] = no action
0	0	equal to 0xFFFFFFFF	SO_WRITE_OFFSET[x] = no action
SW can cause the LOAD of the SO_OFFSET using MI_LOAD_REG, and HW performs the STORE.			
0	1	not equal to 0xFFFFFFFF	SO_WRITE_OFFSET[x] = No action, write SO_WRITE_OFFSET[x] to memory
0	1	equal to 0xFFFFFFFF	SO_WRITE_OFFSET[x] = No action, write SO_WRITE_OFFSET[x] to memory
HW performs the LOAD, and SW can cause the STOREs using MI_STORE_REG_MEM.			
1	0	not equal to 0xFFFFFFFF	SO_WRITE_OFFSET[x] = stream offset
1	0	equal to 0xFFFFFFFF	SO_WRITE_OFFSET[x] = load from memory
HW performs both the LOAD and STORE.			

DW1[21]	DW1[20]	Stream Offset	Action
1	1	not equal to 0xFFFFFFFF	SO_WRITE_OFFSET[x] = stream offset, write SO_WRITE_OFFSET[x] to memory
1	1	equal to 0xFFFFFFFF	SO_WRITE_OFFSET[x] = load from memory, write SO_WRITE_OFFSET[x] to memory

"SO_WRITE_OFFSET[x] =" occurs before the execution of the primitive, while write SO_WRITE_OFFSET[x] to memory occurs after the execution of the primitive.

Functions

Input Buffering

For the purposes of stream output, the SOL stage breaks incoming topologies into independent objects without adjacency information. In the process, any adjacent-only vertices are ignored. For example, it converts TRISTRIP_ADJ into independent 3-vertex triangles. However, if rendering is enabled, incoming topologies are passed to the Clip stage unmodified and therefore the Clip unit must be enabled if there is any possibility of "ADJ" topologies reaching it.

Note that the SOL unit will not see incomplete objects: the VF will remove incomplete input objects, the GS will remove GS-generated incomplete objects, and the DS does not output incomplete objects as only complete topologies are generated by the TE stage.

The OSB (Object Staging Buffer) reorders the vertices of odd-numbered triangles in TRISTRIP topologies to match API requirements.

Incoming topologies are tagged with a 2-bit StreamID. The StreamID is 0 for topologies originating from the VF stage (i.e., 3DPRIMITIVE_xxx). For topologies output from the GS stage, the StreamID is set by the GS shader. A Stream *n* Vertex Length is associated with each stream, and defines how much data is read from the URB for vertices in that stream.

The following table specifies how the SOL stage streams out object vertices for each incoming topology type.

PrimTopologyType <PRIMITIVE_TOPOLOGY> (N = # of vertices)	Order of Vertices Streamed Out [<object#>] = (<vert#>,...);	Any SOL Notes
POINTLIST POINTLIST_BF	[0] = (0); [1] = (1); ...; [N-2] = (N-2);	
LINELIST (N is multiple of 2)	[0] = (0,1); [1] = (2,3); ...; [(N/2)-1] = (N-2,N-1)	
LINELIST_ADJ (N is multiple of 4)	[0] = (1,2); [1] = (5,6); ...; [(N/4)-1)] = (N-3,N-2)	
LINESTRIP LINESTRIP_BF	[0] = (0,1); [1] = (1,2); ...;	

PrimTopologyType <PRIMITIVE_TOPOLOGY> (N = # of vertices)	Order of Vertices Streamed Out [<object#>] = (<vert#>,...);	Any SOL Notes
LINESTRIP_CONT LINESTRIP_CONT_BF (N >= 2)	[N-2] = (N-2,N-1)	
LINESTRIP_ADJ, LINESTRIP_ADJ_CONT (N >= 4)	[0] = (1,2); [1] = (2,3); ...; [N-4] = (N-3,N-2)	LINESTRIP_ADJ_CONT is generated by the Vertex Fetch unit on a restore of a mid-draw pre-empted 3DPRIMITIVE.
LINELOOP	N/A	Not supported after VF.
TRILIST (N is multiple of 3)	[0] = (0,1,2); [1] = (3,4,5); ...; [(N/3)-1] = (N-3,N-2,N-1)	
RECTLIST, RECTLIST_SUBPIXEL	Same as TRILIST	Handled same as TRILIST.
TRILIST_ADJ (N is multiple of 6)	[0] = (0,2,4); [1] = (6,8,10); ...; [(N/6)-1] = (N-6,N-4,N-2)	
TRISTRIP (N >= 3) REORDER_LEADING	[0] = (0,1,2); [1] = (1,3,2); [k even] = (k,k+1,k+2) [k odd] = (k,k+2,k+1) [N-3] = (see above)	"Odd" triangles have vertices reordered to yield increasing leading vertices starting with v0.
TRISTRIP (N >= 3) REORDER_TRAILING	[0] = (0,1,2); [1] = (2,1,3); [k even] = (k,k+1,k+2) [k odd] = (k+1,k,k+2) [N-3] = (see above)	"Odd" triangles have vertices reordered to yield increasing trailing vertices starting with v2.
TRISTRIP_REV (N >= 3) REORDER_LEADING	[0] = (0,2,1) [1] = (1,2,3);...; [k even] = (k,k+2,k+1) [k odd] = (k,k+1,k+2) [N-3] = (see above)	"Even" triangles have vertices reordered to yield increasing leading vertices starting with v0.
TRISTRIP_REV (N >= 3) REORDER_TRAILING	[0] = (1,0,2) [1] = (1,2,3);...; [k even] = (k+1,k,k+2) [k odd] = (k,k+1,k+2) [N-3] = (see above)	"Even" triangles have vertices reordered to yield increasing trailing vertices starting with v2.
TRISTRIP_ADJ (N even, N >= 6) REORDER_LEADING	N = 6 or 7: [0] = (0,2,4) N = 8 or 9: [0] = (0,2,4); [1] = (2,6,4); ...; N > 10: [0] = (0,2,4); [1] = (2,6,4); ...; [k>1, even] = (2k, 2k+2, 2k+4);	"Odd" objects have vertices reordered to yield increasing-by-2 leading vertices starting with v0.

PrimTopologyType <PRIMITIVE_TOPOLOGY> (N = # of vertices)	Order of Vertices Streamed Out [<object#>] = (<vert#>,...);	Any SOL Notes
	[k>2, odd] = (2k, 2k+4, 2k+2);...; Trailing object: [(N/2)-3, even] = (N-6,N-4,N-2); [(N/2)-3, odd] = (N-6,N-2,N-4);	
TRISTRIP_ADJ (N even, N >= 6) REORDER_TRAILING	N = 6 or 7: [0] = (0,2,4) N = 8 or 9: [0] = (0,2,4); [1] = (4,2,6); ...; N > 10: [0] = (0,2,4); [1] = (4,2,6); ...; [k>1, even] = (2k, 2k+2, 2k+4); [k>2, odd] = (2k+2,2k, 2k+4,);...; Trailing object: [(N/2)-3, even] = (N-6,N-4,N-2); [(N/2)-3, odd] = (N-4,N-6,N-2);	"Odd" objects have vertices reordered to yield increasing-by-2 trailing vertices starting with v4.
TRIFAN (N > 2)	[0] = (0,1,2); [1] = (0,2,3); ...; [N-3] = (0, N-2, N-1);	
TRIFAN_NOSTIPPLE	Same as TRIFAN	
POLYGON, POLYGON_CONT	Same as TRIFAN	POLYGON_CONT is generated by the Vertex Fetch unit on a restore of a mid-draw pre-empted 3DPRIMITIVE.
QUADLIST QUADSTRIP	N/A	Not supported after VF.
PATCHLIST_1	[0] = (0); [1] = (1); ...; [N-2] = (N-2);	
PATCHLIST_2	[0] = (0,1); [1] = (2,3); ...; [(N/2)-1] = (N-2,N-1)	
PATCHLIST_3..32	similar to above	

Stream Output Function

As previously mentioned, incoming 3D topologies are targeted at one of the four streams. The SOL stage contains state information specific to each of the four streams.

A stream's list of SO declarations (SO_DECL structures) is used to perform the SO function for objects targeted to that particular stream. The 3DSTATE_SO_DECL_LIST command is used to specify the list of SO_DECL structures for all four streams in parallel. Software is required to scan the SO_DECL lists of streams to determine which SO buffers are targeted. The Stream To Buffer Selects bits in 3DSTATE_SO_DECL_LIST must be programmed accordingly (if the buffer is targeted, the select bit must set, else it must be cleared).

If a stream has no SO_DECL state defined (NumEntries is 0), incoming objects targeting that stream are effectively ignored. As there is no attempt to perform stream output, overflow detection is neither required nor performed.

Otherwise, an overflow check is performed. First any attempt to output to a disabled buffer is detected. This occurs when the stream has a Stream To Buffer Selects bit set but the corresponding SO Buffer Enable is clear. Assuming all targeted buffers are enabled, an additional check is made to ensure that there is enough room in each targeted buffer to hold the number of vertices which be output to it (for the input object). Here the buffer's current end address is compared to what the write offset would be if the output was performed. The latter value is computed as (write_offset + vertex_count * buffer_pitch). If this value is greater than the end address, an overflow is signaled. This check is performed for each buffer included in Stream To Buffer Selects.

If an overflow is not signaled, the SO function is performed. The SO_DECL list for the targeted stream is traversed independently for each object vertex, and the operation specified by the SO_DECL structure is performed (typically causing data to be appended to an SO buffer). In the process, SO buffer Write Offsets are incremented.

Stream Output Buffers

Up to four SO buffers are supported. The SO buffer parameters (start/end address, etc.) are specified by the 3DSTATE_SO_BUFFER_INDEX_# commands.

The 3DSTATE_STREAMOUT command specifies an SO Buffer Enable bit for each of the buffers. If a buffer is disabled, its state is ignored, and no output will be attempted for that buffer. Any attempt to output to that buffer will immediately signal an overflow condition.

The SOL stage maintains a current Write Offset register value for each SO buffer. These registers can be written via MI_LOAD_REGISTER_MEM or MI_LOAD_REGISTER_IMM commands. The SOL stage will increment the Write Offsets as a part of the SO function. Software can cause a Write Offset register to be written to memory via an MI_STORE_REGISTER_MEM command, though a preceding flush operation may be required to ensure that any previous SO functions have completed.

Surface Format Name
R32G32B32A32_FLOAT
R32G32B32A32_SINT
R32G32B32A32_UINT
R32G32B32_FLOAT
R32G32B32_SINT
R32G32B32_UINT

Surface Format Name
R32G32_FLOAT
R32G32_SINT
R32G32_UINT
R32_SINT
R32_UINT
R32_FLOAT

Rendering Disable

Independent of SOL function enable, if rendering (i.e, 3D pipeline functions past the SOL stage) is enabled (via clearing the Rendering Disable bit), the SOL stage will pass topologies for a specific input stream (as selected by Render Stream Select) down the pipeline, with the exception of PATCHLIST_n topologies which are never passed downstream. Software must ensure that the vertices exiting the SOL stage include a vertex header and position value so that the topologies can be correctly processed by subsequent pipeline stages. Specifically, rendering must be disabled whenever 128-bit vertices are output from a GS thread.

If Rendering Disable is set, the SOL stage will prevent any topologies from exiting the SOL stage.

Statistics

The SOL stage controls the incrementing of two 64-bit statistics counter registers for each of the four output buffer slots, SO_NUM_PRIMS_WRITTEN[] and SO_PRIM_STORAGE_NEEDED[].

3D Pipeline Rasterization

Common Rasterization State

This section contains rasterization state pointers.

Pointers
3DSTATE_VIEWPORT_STATE_POINTERS_CC
3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP
3DSTATE_SCISSOR_STATE_POINTERS
3DSTATE_RASTER

3D Pipeline - CLIP Stage Overview

The CLIP stage of the 3D Pipeline is similar to the GS stage in that it can be used to perform general processing on incoming 3D objects via spawned threads. However, the CLIP stage also includes specialized logic to perform a *ClipTest* function on incoming objects. These two usage models of the CLIP stage are outlined below.

Refer to the *Common 3D FF Unit Functions* subsection in the *3D Overview* chapter for a general description of a 3D Pipeline stage, as much of the CLIP stage operation and control falls under these

"common" functions. I.e., many of the CLIP stage state variables and CLIP thread payload parameters are described in *3D Overview*, and although they are listed here for completeness, that chapter provides the detailed description of the associated functions.

Refer to this chapter for an overall description of the CLIP stage, details on the ClipTest function, and any exceptions the CLIP stage exhibits with respect to common FF unit functions.

Clip Stage - 3D Clipping

The ClipTest fixed function is provided to optimize the CLIP stage for support of generalized *3D Clipping*. The CLIP FF unit examines the position of incoming vertices, performs a fixed function *VertexClipTest* on these positions, and then examines the results for the vertices of each independent object in *ClipDetermination*.

The results of ClipDetermination indicate whether an object is to be clipped (MustClip), discarded (TrivialReject) or passed down the pipeline unmodified (TrivialAccept). In the MustClip case, the fixed function clipping hardware is responsible for performing the actual 3D Clipping algorithm. The CLIP hardware is passed the source object vertex data and is able to output a new, arbitrary 3D primitive (e.g., the clipped primitive), or no output at all. Note that the output primitive is comprised of newly generated vertex positions, barycentric attributes and shares vertices with the source primitive for rest of the attributes. The CLIP unit maintains the proper ordering of CLIP-generated primitives and any surrounding trivially accepted primitives and processes all the primitives in order.

The outgoing primitive stream is sent down the pipeline to the Strip/Fan (SF) FF stage (now including the read back VUE Vertex Header data such as Vertex Position (NDC or screen space), RTAIndex, VPIndex, PointWidth) and control information (PrimType, PrimStart, PrimEnd) while the remainder of the vertex data remains in the VUE in the URB.

Fixed Function Clipper

The GPU supports Fixed Function Clipping.

Note: In an earlier generation, clipping was done in the EU. However, the clipper thread latency was high and caused a bottleneck in the pipeline. Hence the motivation for a fixed function clipper.

Concepts

This section provides an overview of 3D clip-testing and clipping concepts, as defined by the Direct3D* and OpenGL* APIs. It is provided as background material. Some of the concepts impact HW functionality while others impact CLIP kernel functionality.

* Other names and brands may be claimed as the property of others.

CLIP Stage Input

As a stage of the 3D pipeline, the CLIP stage receives inputs from the previous (GS) stage. Refer to *3D Overview* for an overview of the various types of input to a 3D Pipeline stage. The remainder of this subsection describes the inputs specific to the CLIP stage.

State

This section contains state clips for the Clip Stage. For each processor generation, the state used by the clip stage is defined by the appropriate inline state packet, linked below.

3DSTATE_CLIP

3D_STATE_CLIP

The Clip unit will unconditionally reject incoming PATCHLIST topologies, if not already discarded by SOL. So there is no need for SW to explicitly set the CLIP_mode to reject PATCHLIST topologies.

Clip Unit also receives 3DSTATE_RASTER. It also receives 3DSTATE_INT which is transparent to SW. 3DSTATE_INT provides 3DSTATE_VS, 3DSTATE_DS and 3DSTATE_GS fields.

Signal	Description	Formula
CLIP_INT::Front_Winding	Determines whether a triangle object is considered "front facing" if the screen space vertex positions, when traversed in the order, result in a clockwise (CW) or counter-clockwise (CCW) winding order. Does not apply to points or lines.	= 3DSTATE_RASTER::FrontWinding
CLIP_INT::CullMode	Controls removal (culling) of triangle objects based on orientation. The cull mode only applies to triangle objects and does not apply to lines, points, or rectangles.	= 3DSTATE_RASTER::CullMode
CLIP_INT::Viewport Z ClipTest Enable	This field is used to control whether the Viewport Z extents (near, far) are considered in VertexClipTest.	= 3DSTATE_RASTER::Viewport Z ClipTest Enable
CLIP_INT::User Clip Distance Cull Test Enable Bitmask	This 8-bit mask field selects which of the 8 user clip distances against which trivial reject/trivial accept determination needs to be made (does not cause a must clip). DX10 allows simultaneous use of ClipDistance and Cull Distance test of up to 8 distances. This mask must be mutually exclusive to final CLIP_INT::User	= (3DSTATE_CLIP::ForceUser Clip Distance Cull Test Enable Bitmask == Force) ? 3DSTATE_CLIP::User Clip Distance Cull Test Enable Bitmask : 3DSTATE_GS::GS_Enable ? 3DSTATE_GS:: GS User Clip Distance Cull Test Enable Bitmask : 3DSTATE_DS:DS_Enable ? 3DSTATE_DS:: DS User Clip Distance Cull Test Enable Bitmask : 3DSTATE_INT:VS_Enable ? 3DSTATE_VS:: VS User Clip Distance Cull Test Enable Bitmask : 0

Signal	Description	Formula
	Clip Distance Clip Test Enable Bitmask. Same mask bit can't be set for both.	
CLIP_INT::User Clip Distance Clip Test Enable Bitmask	This 8-bit mask field selects which of the 8 user clip distances against which trivial reject/trivial accept determination needs to be made (does not cause a must clip). DX10 allows simultaneous use of ClipDistance and Clip Distance test of up to 8 distances. This mask must be mutually exclusive to final CLIP_INT::User Clip Distance Cull Test Enable Bitmask. Same mask bit can't be set for both.	= (3DSTATE_CLIP::ForceUser Clip Distance Clip Test Enable Bitmask == Force) ? 3DSTATE_CLIP::User Clip Distance Clip Test Enable Bitmask : 3DSTATE_GS:GS_Enable ? 3DSTATE_GS::GS User Clip Distance Clip Test Enable Bitmask : 3DSTATE_DS::DS_Enable ? 3DSTATE_DS::DS User Clip Distance Clip Test Enable Bitmask : 3DSTATE_VS::VS_Enable ? 3DSTATE_VS::VS User Clip Distance Clip Test Enable Bitmask : 0

VUE Readback

Starting with the CLIP stage, the 3D pipeline requires vertex information in addition to the VUE handle. For example, the CLIP unit's VertexClipTest function needs the vertex position, as does the SF unit's functions. This information is obtained by the 3D pipeline reading a portion of each vertex's VUE data directly from the URB. This readback (effectively) occurs immediately before the CLIP VertexClipTest function, and immediately after a CLIP thread completes the output of a destination VUE.

The Vertex Header (first 256 bits) of the VUE data is read back. (See the previous *VUE Formats* subsection (above) for details on the content and format of the Vertex Header.) Additional Clip/Cull data (located immediately past the Vertex Header) may be read prior to clipping.

This readback occurs automatically and is not under software control. The only software implication is that the Vertex Header must be valid at the readback points, and therefore must have been previously loaded or written by a thread.

VertexClipTest Function

The VertexClipTest function compares each incoming vertex position (x,y,z,w) with various viewport and guardband parameters (either hard-coded values or specified by state variables).

The RHW component of the incoming vertex position is tested for NaN value, and if a NaN is detected, the vertex is marked as "BAD" by setting the outcode[BAD]. If a NaN is detected in any vertex

homogeneous x,y,z,w component or an enabled ClipDistance value, the vertex is marked as "BAD" by setting the outcode[BAD].

In general, any object containing a BAD vertex will be discarded, as how to clip/render such objects is undefined.

However, in the case of CLIP_ALL mode, a CLIP thread will be spawned even for objects with "BAD" vertices. The CLIP kernel is required to handle this case, and can examine the **Object Outcode [BAD]** payload bit to detect the condition. (Note that the VP and GB Object Outcodes are UNDEFINED when BAD is set.)

If the incoming RHW coordinate is negative (including negative 0) the NEGW outcode is set. Also, this condition is used to select the proper comparison functions for the VP and GB outcode tests (below).

Next, the VPXY and GB outcodes are computed, depending on the corresponding enable SV bits. If one of VPXY or GB is disabled, the enabled set of outcodes are copied to the disabled set of outcodes. This effectively defines the disabled boundaries to coincide with the enabled boundaries (i.e., disabling the GB is just like setting it to the VPXY values, and vice versa).

The VPZ outcode is computed as required by the API mode SV.

Separate VP Z Near (ZMin) and Z Far (ZMax) ClipTest Enable state bits are provided in 3DSTATE_RASTER.

Finally, the incoming UserClipFlags are masked and copied to corresponding outcodes.

The following algorithm is used by VertexClipTest:

```
//
// Vertex ClipTest
//
// On input:
// if (CLIP.PreMapped)
//   x,y are viewport mapped
//   z is NDC ([0,1] is visible)
// else
//   x,y,z are NDC (post-perspective divide)
//   w is always 1/w
//
// Initialize outCodes to "inside"
//
outCode[*] = 0
//
// Check if w is NaN
// Any object containing one of these "bad" vertices will likely be discarded
//
if (ISNAN(homogeneous x,y,z,w or enabled ClipDistance value)
{
   outCode[BAD] = 1
}
//
// If 1/w is negative, w is negative and therefore outside of the w=0 plane
//
//
rhw_neg = ISNEG(rhw)
if (rhw_neg)
{
   outCode[NEGW] = 1
}
//
```

```
// View Volume Clip Test
// If Premapped, the 2D viewport is defined in screen space
// otherwise the canonical NDC viewvolume applies ([-1,1])
//
if (CLIP_STATE.PreMapped)
{
   vp_XMIN = CLIP_STATE.VP_XMIN
   vp_XMAX = CLIP_STATE.VP_XMAX
   vp_YMIN = CLIP_STATE.VP_YMIN
   vp_YMAX = CLIP_STATE.VP_YMAX
} else {
   vp_XMIN = -1.0f
   vp_XMAX = +1.0f
   vp_YMIN = -1.0f
   vp_YMAX = +1.0f
}

if (CLIP_STATE.ViewportXYClipTestEnable) {
   outCode[VP_XMIN] = (x < vp_XMIN)
   outCode[VP_XMAX] = (x > vp_XMAX)
   outCode[VP_YMIN] = (y < vp_YMIN)
   outCode[VP_YMAX] = (y > vp_YMAX)

   #ifdef (BW-E0)
   if (rhw_neg) {
      outCode[VP_XMIN] = (x >= vp_XMIN)
      outCode[VP_XMAX] = (x <= vp_XMAX)
      outCode[VP_YMIN] = (y >= vp_XMIN)
      outCode[VP_YMAX] = (y <= vp_XMAX)
   }
   #endif
   if (rhw_neg) {
      outCode[VP_XMIN] = (x > vp_XMIN)
      outCode[VP_XMAX] = (x < vp_XMAX)
      outCode[VP_YMIN] = (y > vp_XMIN)
      outCode[VP_YMAX] = (y < vp_XMAX)
   }
}

if (CLIP_STATE.ViewportZClipTestEnable) {
   if (CLIP_STATE.APIMode == APIMODE_D3D) {
      vp_ZMIN = 0.0f
      vp_ZMAX = 1.0f
   } else { // OGL
      vp_ZMIN = -1.0f
      vp_ZMAX = 1.0f
   }
   outCode[VP_ZMIN] = (z < vp_ZMIN)
   outCode[VP_ZMAX] = (z > vp_ZMAX)

   #ifdef (BW-E0)
   if (rhw_neg) {
      outCode[VP_ZMIN] = (z >= vp_ZMIN)
      outCode[VP_ZMAX] = (z <= vp_ZMAX)
   }
   #endif
   if (rhw_neg) {
      outCode[VP_ZMIN] = (z > vp_ZMIN)
      outCode[VP_ZMAX] = (z < vp_ZMAX)
   }
}
//
// Guardband Clip Test
//
```

```
if {CLIP_STATE.GuardbandClipTestEnable) {
   gb_XMIN = CLIP_STATE.Viewport[vpindex].GB_XMIN
   gb_XMAX = CLIP_STATE.Viewport[vpindex].GB_XMAX
   gb_YMIN = CLIP_STATE.Viewport[vpindex].GB_YMIN
   gb_YMAX = CLIP_STATE.Viewport[vpindex].GB_YMAX
   outCode[GB_XMIN] = (x < gb_XMIN)
   outCode[GB_XMAX] = (x > gb_XMAX)
   outCode[GB_YMIN] = (y < gb_YMIN)
   outCode[GB_YMAX] = (y > gb_YMAX)

   #ifdef (BW-E0)
   if (rhw_neg) {
      outCode[GB_XMIN] = (x >= gb_XMIN)
      outCode[GB_XMAX] = (x <= gb_XMAX)
      outCode[GB_YMIN] = (y >= gb_YMIN)
      outCode[GB_YMAX] = (y <= gb_YMAX)
   }
   #endif
   if (rhw_neg) {
      outCode[GB_XMIN] = (x > gb_XMIN)
      outCode[GB_XMAX] = (x < gb_XMAX)
      outCode[GB_YMIN] = (y > gb_YMIN)
      outCode[GB_YMAX] = (y < gb_YMAX)
   }
}
//
// Handle case where either VP or GB disabled (but not both)
// In this case, the disabled set take on the outcodes of the enabled set
//
if (CLIP_STATE.ViewportXYClipTestEnable && !CLIP_STATE.GuardbandClipTestEnable) {
   outCode[GB_XMIN] = outCode[VP_XMIN]
   outCode[GB_XMAX] = outCode[VP_XMAX]
   outCode[GB_YMIN] = outCode[VP_YMIN]
   outCode[GB_YMAX] = outCode[VP_YMAX]
} else if (!CLIP_STATE.ViewportXYClipTestEnable && CLIP_STATE.GuardbandClipTestEnable) {
   outCode[VP_XMIN] = outCode[GB_XMIN]
   outCode[VP_XMAX] = outCode[GB_XMAX]
   outCode[VP_YMIN] = outCode[GB_YMIN]
   outCode[VP_YMAX] = outCode[GB_YMAX]
}
//
// X/Y/Z NaN Handling
//
xyorgben = (CLIP_STATE.ViewportXYClipTestEnable || CLIP_STATE.GuardbandClipTestEnable)
if (isNAN(x)) {
   outCode[GB_XMIN] = xyorgben
   outCode[GB_XMAX] = xyorgben
   outCode[VP_XMIN] = xyorgben
   outCode[VP_XMAX] = xyorgben
}
if (isNAN(y)) {
   outCode[GB_YMIN] = xyorgben
   outCode[GB_YMAX] = xyorgben
   outCode[VP_YMIN] = xyorgben
   outCode[VP_YMAX] = xyorgben
}
if (isNaN) {
   outCode[VP_ZMIN] = CLIP_STATE.ViewportZClipTestEnable
   outCode[VP_ZMAX] = CLIP_STATE.ViewportZClipTestEnable
}
//
// UserClipFlags
//
ExamineUCFs
```

```
for (i=0; i<7; i++)
{
   outCode[UC0+i] = userClipFlag[i] & CLIP_STATE.UserClipFlagsClipTestEnableBitmask[i]
}
outCode[UC7] = userClipFlag[i] & CLIP_STATE.UserClipFlagsClipTestEnableBitmask[7]
```

Object Staging

The CLIP unit's Object Staging Buffer (OSB) accepts streams of input vertex information packets, along with each vertex's VertexClipTest result (outCode). This information is buffered until a complete object or the last vertex of the primitive topology is received. The OSB then performs the ClipDetermination function on the object vertices, and takes the actions required by the results of that function.

Partial Object Removal

The OSB is responsible for removing incomplete LINESTRIP and TRISTRIP objects that it may receive from the preceding stage (GS). Partial object removal is not supported for other primitive types due to either (a) the GS is not permitted to output those primitive types (e.g., primitives with adjacency info), and the VF unit will have removed the partial objects as part of 3DPRIMITIVE processing, or (b) although the GS thread is allowed to output the primitive type (e.g., LINELIST), it is assumed that the GS kernel will be correctly implemented to avoid outputting partial objects (or pipeline behavior is UNDEFINED).

An object is considered 'partial' if the last vertex of the primitive topology is encountered (i.e., PrimEnd is set) before a complete set of vertices for that object have been received. Given that only LINESTRIP and TRISTRIP primitive types are subject to CLIP unit partial object removal, the only supported cases of partial objects are 1-vertex LINESTRIPs and 1 or 2-vertex TRISTRIPs.

ClipDetermination Function

In ClipDetermination, the vertex outcodes of the primitive are combined in order to determine the clip status of the object (TR: trivially reject; TA: trivial accept; MC: must clip; BAD: invalid coordinate). Only those vertices included in the object are examined (3 vertices for a triangle, 2 for a line, and 1 for a point). The outcode bit arrays for the vertices are separately ANDed (intersection) and ORed (union) together (across vertices, not within the array) to yield objANDCode and objORCode bit arrays.

TR/TA against interesting boundary subsets are then computed. The TR status is computed as the logical OR of the appropriate objANDCode bits, as the vertices need only be outside of one common boundary to be trivially rejected. The TA status is computed as the logical NOR of the appropriate objORCode bits, as any vertex being outside of any of the boundaries prevents the object from being trivially accepted.

If any vertex contains a BAD coordinate, the object is considered BAD and any computed TR/TA results will effectively be ignored in the final action determination.

Next, the boundary subset TR/TA results are combined to determine an overall status of the object. If the object is TR against any viewport or enabled UC plane, the object is considered TR. Note that, by definition, being TR against a VPXY boundary implies that the vertices will be TR agains the corresponding GB boundary, so computing TR_GB is unnecessary.

The treatment of the UCF outcodes is conditional on the UserClipFlags MustClip Enable state. If DISABLED, an object that is not TR against the UCFs is considered TA against them. Put another way,

objects will only be culled (not clipped) with respect to the UCFs. If ENABLED, the UCF outcodes are treated like the other outcodes, in that they are used to determine TR, TA or MC status, and an object can be passed to a CLIP thread simply based on it straddling a UCF.

Finally, the object is considered MC if it is neither TR or TA.

The following logic is used to compute the final TR, TA, and MC status.

```
//
// ClipDetermination
//
// Compute objANDCode and objORCode
//
switch (object type) {
   case POINT:
      {
         objANDCode[...] = v0.outCode[...]
         objORCode[...]  = v0.outCode[...]
      } break
   case LINE:
      {
         objANDCode[...] = v0.outCode[...] & v1.outCode[...]
         objORCode[...]  = v0.outCode[...] | v1.outCode[...]
      } break
   case TRIANGLE:
      {
         objANDCode[...] = v0.outCode[...] & v1.outCode[...] & v2.outCode[...]
         objORCode[...]  = v0.outCode[...] | v1.outCode[...] | v2.outCode[...]
      } break
}
//
// Determine TR/TA against interesting boundary subsets
//
TR_VPXY = (objANDCode[VP_L] | objANDCode[VP_R] | objANDCode[VP_T] | objANDCode[VP_B])
TR_GB   = (objANDCode[GB_L] | objANDCode[GB_R] | objANDCode[GB_T] | objANDCode[GB_B])
TA_GB   = !(objORCode[GB_L] | objORCode[GB_R]  | objORCode[GB_T]  | objORCode[GB_B])
TA_VPZ  = !(objORCode[VP_N] | objORCode[VP_Z])
TR_VPZ  = (objANDCode[VP_N] | objANDCode[VP_Z])
TA_UC   = !(objORCode[UC0]  | objORCode[UC1]   | ... | objORCode[UC7])
TR_UC   = (objANDCode[UC0]  | objANDCode[UC1]  | ... | objANDCode[UC7])
BAD     = objORCode[BAD]
TA_NEGW = !objORCode[NEGW]
TR_NEGW = objANDCode[NEGW]

//
// Trivial Reject
//
// An object is considered TR if all vertices are TR against any common boundary
// Note that this allows the case of the VPXY being outside the GB
//
TR = TR_GB || TR_VPXY || TR_VPZ || TR_UC || TR_NEGW
#else
TR = TR_GB || TR_VPXY || TR_VPZ || TR_UC

//
// Trivial Accept
//
// For an object to be TA, it must be TA against the VPZ and GB, not TR,
// and considered TA against the UC planes and NEGW
// If the UCMC mode is disabled, an object is considered TA against the UC
// as long as it isn't TR against the UC.
// If the UCMC mode is enabled, then the object really has to be TA against the UC
// to be considered TA
```

```
// In this way, enabling the UCMC mode will force clipping if the object is neither
// TA or TR against the UC
//
TA   = !TR && TA_GB && TA_VPZ && TA_NEGW
UCMC = CLIP_STATE.UserClipFlagsMustClipEnable
TA   = TA && ( (UCMC && TA_UC) || (!UCMC && !TR_UC) )

//
// MustClip
// This is simply defined as not TA or TR
// Note that exactly one of TA, TR and MC will be set
//
MC = !(TA || TR)
```

ClipMode State

The ClipMode state determines what action the CLIP unit takes given the results of ClipDetermination. The possible actions are:

- **PASSTHRU:** Pass the object directly down the pipeline. A CLIP thread is not spawned.
- **DISCARD:** Remove the object from the pipeline and dereference object vertices as required (that is, dereferencing will not occur if the vertices are shared with other objects).
- **SPAWN:** Pass the object to a CLIP thread. In the process of initiating the thread, the object vertices may be dereferenced.

The following logic is used to determine what to do with the object (PASSTHRU or DISCARD or SPAWN).

```
//
// Use the ClipMode to determine the action to take
//
switch (CLIP_STATE.ClipMode) {
   case NORMAL:
      {
         PASSTHRU = TA && !BAD
         DISCARD = TR || BAD
         SPAWN   = MC && !BAD
      }
   case CLIP_ALL:
      {
         PASSTHRU = 0
         DISCARD  = 0
         SPAWN    = 1
      }
   case CLIP_NOT_REJECT:
      {
         PASSTHRU = 0
         DISCARD  = TR   || BAD
         SPAWN    = !(TR || BAD)
      }
   case REJECT_ALL:
      {
         PASSTHRU = 0
         DISCARD  = 1
         SPAWN    = 0
      }
   case ACCEPT_ALL:
      {
         PASSTHRU = !BAD
         DISCARD  = BAD
         SPAWN    = 0
```

```
        }
} endswitch
```

NORMAL ClipMode

In NORMAL mode, objects will be discarded if TR or BAD, passed through if TA, and passed to a CLIP thread if MC. Those mode is typically used when the CLIP kernel is only used to perform 3D Clipping (the expected usage model).

CLIP_ALL ClipMode

In CLIP_ALL mode, all objects (regardless of classification) will be passed to CLIP threads. Note that this includes BAD objects. This mode can be used to perform arbitrary processing in the CLIP thread, or as a backup if for some reason the CLIP unit fixed functions (VertexClipTest, ClipDetermination) are not sufficient for controlling 3D Clipping.

CLIP_NON_REJECT ClipMode

This mode is similar to CLIP_ALL mode, but TR and BAD objects are discarded an all other (TA, MC) objects are passed to CLIP threads. Usage of this mode assumes that the CLIP unit fixed functions (VertexClipTest, ClipDetermination) are sufficient at least in respect to determining trivial reject.

REJECT_ALL ClipMode

In REJECT_ALL mode, all objects (regardless of classification) are discarded. This mode effectively clips out all objects.

ACCEPT_ALL ClipMode

In ACCEPT_ALL mode, all non-BAD objects are passed directly down the pipeline. This mode partially disables the CLIP stage. BAD objects will still be discarded, and incomplete primitives (generated by a GS thread) will be discarded.

Primitive topologies with adjacency are also handled, in that the adjacent-only vertices are dereferenced and only non-adjacent objects are passed down the pipeline. This condition can arise when primitive topologies with adjacency are generated but the GS stage is disabled. If this condition is allowed, the CLIP stage must not be completely disabled - as this would allow adjacent vertices to pass through the CLIP stage and lead to unpredictable results as the rest of the pipeline does not comprehend adjacency.

Object Pass-Through

Depending on ClipMode, objects may be passed directly down the pipeline. The PrimTopologyType associated with the output objects may differ from the input PrimTopologyType, as shown in the table below.

Programming Note: The CLIP unit does *not* tolerate primitives with adjacency that have "dangling vertices". This should not be an issue under normal conditions, as the VF unit does not generate these sorts of primitives and the GS thread is restricted (though by specification only) to not output these sorts of primitives.

Input PrimTopologyType	Pass-Through Output PrimTopologyType	Notes
POINTLIST	POINTLIST	
POINTLIST_BF	POINTLIST_BF	
LINELIST	LINELIST	
LINELIST_ADJ	LINELIST	Adjacent vertices removed.
LINESTRIP	LINESTRIP	
LINESTRIP_ADJ, LINESTRIP_ADJ_CONT	LINESTRIP	Adjacent vertices removed. LINESTRIP_ADJ_CONT is generated by the Vertex Fetch unit on a restore of a mid-draw pre-empted 3DPRIMITIVE.
LINESTRIP_BF	LINESTRIP_BF	
LINESTRIP_CONT	LINESTRIP_CONT	
LINESTRIP_CONT_BF	LINESTRIP_CONT_BF	
LINELOOP	N/A	Not supported after GS.
TRILIST	TRILIST	
RECTLIST	RECTLIST	
TRILIST_ADJ	TRILIST	Adjacent vertices removed.
TRISTRIP	TRISTRIP or TRISTRIP_REV	Depends on where the incoming strip is broken (if at all) by discarded or clipped objects See Tristrip Clipping subsection.
TRISTRIP_REV	TRISTRIP or TRISTRIP_REV	Depends on where the incoming strip is broken (if at all) by discarded or clipped objects. See Tristrip Clipping subsection.
TRISTRIP_ADJ	TRISTRIP or TRISTRIP_REV	Depends on where the incoming strip is broken (if at all) by discarded or clipped objects. Adjacent vertices removed. See Tristrip Clipping subsection.
TRIFAN	TRIFAN	
TRIFAN_NOSTIPPLE	TRIFAN_NOSTIPPLE	
POLYGON, POLYGON_CONT	POLYGON	POLYGON_CONT is generated by the Vertex Fetch unit on a restore of a mid-draw pre-empted 3DPRIMITIVE.

Input PrimTopologyType	Pass-Through Output PrimTopologyType	Notes
QUADLIST	N/A	Not supported after GS.
QUADSTRIP	N/A	Not supported after GS.

Primitive Output

(This section refers to output from the CLIP unit to the pipeline, not output from the CLIP thread)

The CLIP unit will output primitives (either passed-through or generated by a CLIP thread) in the proper order. This includes the buffering of a concurrent CLIP thread's output until the preceding CLIP thread terminates. Note that the requirement to buffer subsequent CLIP thread output until the preceding CLIP thread terminates has ramifications on determining the number of VUEs allocated to the CLIP unit and the number of concurrent CLIP threads allowed.

Other Functionality

Statistics Gathering

Software is responsible for controlling (enabling) these counters in order to provide the required statistics at the DDI level. For example, software might need to disable statistics gathering before submitting non-API-visible objects (e.g., RECTLISTs) for processing.

The CLIP unit must be ENABLED (via the CLIP Enable bit of PIPELINED_STATE_POINTERS) for it to affect the statistics counters. This might lead to a pathological case where the CLIP unit needs to be ENABLED simply to provide statistics gathering. If no clipping functionality is desired, Clip Mode can be set to ACCEPT_ALL to effectively inhibit clipping while leaving the CLIP stage ENABLED.

The statistic the CLIP unit affects (if enabled) is CL_INVOCATION_COUNT, incremented for every object received from the GS stage.

CL_INVOCATION_COUNT

If the **Statistics Enable** bit (CLIP_STATE) is set, the CLIP unit increments the CL_INVOCATION_COUNT register for every complete object received from the GS stage.

To maintain a count of application-generated objects, software must clear the CLIP unit's **Statistic Enable** whenever driver-generated objects are rendered.

3D Pipeline - Strips and Fans (SF) Stage

The Strips and Fan (SF) stage of the 3D pipeline is responsible for performing "setup" operations required to rasterize 3D objects.

This functionality is handled completely in hardware, and the SF unit no longer has the ability to spawn threads.

Inputs from CLIP

The following table describes the per-vertex inputs passed to the SF unit from the previous (CLIP) stage of the pipeline.

SF's Vertex Pipeline Inputs

Variable	Type	Description
primType	enum	Type of primitive topology the vertex belongs to. SF-Supported Primitive Types & Vertex Count Restrictions for a list of primitive types supported by the SF unit. See *3D Pipeline* for descriptions of these topologies. Notes: The CLIP unit will convert any primitive with adjacency (3DPRIMxxx_ADJ) it receives from the pipeline into the corresponding primitive without adjacency (3DPRIMxxx). QUADLIST, QUADSTRIP, LINELOOP primitives are not supported by the SF unit. Software must use a GS thread to convert these to some other (supported) primitive type. If an object is clipped by the hardware clipper, the CLunit would force this field to "3DPRIM_POLYGON". SFunit would process this incoming object just as it would any other "3DPRIM_POLYGON". SFunit selects vertex 0 as the provoking vertex.
primStart,primEnd	boolean	Indicate vertex's position within the primitive topology
vInX[]	float	Vertex X position (screen space or NDC space)
vInY[]	float	Vertex Y position (screen space or NDC space)
vInZ[]	float	Vertex Z position (screen space or NDC space)
vInInvW[]	float	Reciprocal of Vertex homogeneous (clip space) W
hVUE[]	URB address	Points to the vertex's data stored in the URB (one VUE handle per vertex)
renderTargetArrayIndex	uint	Index of the render target (array element or 3D slice), clamped to 0 by the GS unit if the max value was exceeded. If this vertex is the leading vertex of an object within the primitive topology, this value will be associated with that object in subsequent processing.
viewPortIndex	uint	Index of a viewport transform matrix within the SF_VIEWPORT structure used to perform Viewport Transformation on object vertices and scissor operations on an object. If this vertex is the leading vertex of an object within the primitive topology, this value will be associated with that object in the Viewport Transform and Scissor subfunctions, otherwise the value is ignored. Note that for primitive topologies with vertices shared between objects, this means a shared vertex may be subject to multiple Viewport Transformation operations if the viewPortIndex varies

Variable	Type	Description
		within the topology.
pointSize	uint	If this vertex is within a POINTLIST[_BF] primitive topology, this value specifies the screen space size (width,height) of the square point to be rasterized about the vertex position. Otherwise the value is ignored.

Attribute Setup/Interpolation Process

The following sections describe the Attribute Setup/Interpolation Process.

Attribute Setup/Interpolation Process

Hardware computes all needed parameters, as there is no setup thread.

Outputs to WM

The outputs from the SF stage to the WM stage are mostly comprised of implementation-specific information required for the rasterization of objects. The types of information is summarized below, but as the interface is not exposed to software a detailed discussion is not relevant to this specification.

- PrimType of the object
- VPIndex, RTAIndex associated with the object
- Coefficients for Z, 1/W, perspective and non-perspective b1 and b2 per vertex, and attribute vertex deltas a0, a1, and a2 per attribute.
- Information regarding the X,Y extent of the object (e.g., bounding box, etc.).
- Edge or line interpolation information (e.g., edge equation coefficients, etc.).
- Information on where the WM is to start rasterization of the object.
- Object orientation (front/back-facing).
- Last Pixel indication (for line drawing).

Primitive Assembly

The first subfunction within the SF unit is *Primitive Assembly*. Here 3D primitive vertex information is buffered and, when a sufficient number of vertices are received, converted into basic 3D objects which are then passed to the Viewport Transformation subfunction.

The number of vertices passed with each primitive is constrained by the primitive type. Passing any other number of vertices results in UNDEFINED behavior. Note that this restriction only applies to primitive output by GS threads (which is under control of the GS kernel). See the Vertex Fetch chapter for details on how the VF unit automatically removes incomplete objects resulting from processing a 3DPRIMITIVE command.

SF-Supported Primitive Types & Vertex Count Restrictions

primType	VertexCount Restriction
3DPRIM_TRILIST	nonzero multiple of 3
3DPRIM_TRISTRIP 3DPRIM_TRISTRIP_REVERSE	>=3
3DPRIM_TRIFAN 3DPRIM_TRIFAN_NOSTIPPLE 3DPRIM_POLYGON	>=3
3DPRIM_LINELIST	nonzero multiple of 2
3DPRIM_LINESTRIP 3DPRIM_LINESTRIP_CONT 3DPRIM_LINESTRIP_BF 3DPRIM_LINESTRIP_CONT_BF	>=2
3DPRIM_RECTLIST	nonzero multiple of 3
3DPRIM_POINTLIST 3DPRIM_POINTLIST_BF	nonzero

3D Object Types for a list of the 3D object types.

3D Object Types

objectType	generated by primType	Vertices/Object
3DOBJ_POINT	3DPRIM_POINTLIST 3DPRIM_POINTLIST_BF	1
3DOBJ_LINE	3DPRIM_LINELIST 3DPRIM_LINESTRIP 3DPRIM_LINESTRIP_CONT 3DPRIM_LINESTRIP_BF 3DPRIM_LINESTRIP_CONT_BF	2
3DOBJ_TRIANGLE	3DPRIM_TRILIST 3DPRIM_TRISTRIP 3DPRIM_TRISTRIP_REVERSE 3DPRIM_TRIFAN 3DPRIM_TRIFAN_NOSTIPPLE 3DPRIM_POLYGON	3

objectType	generated by primType	Vertices/Object
3DOBJ_RECTANGLE	3DPRIM_RECTLIST	3 (expanded to 4 in RectangleCompletion)

Primitive Decomposition Outputs for the outputs of Primitive Decomposition.

Primitive Decomposition Outputs

Variable	Type	Description
objectType	enum	Type of object. 3D Object Types
nV	uint	The number of object vertices passed to Object Setup. 3D Object Types
v[0..nV-1]*	various	Data arrays associated with object vertices. Data in the array consists of X, Y, Z, invW and a pointer to the other vertex attributes. These additional attributes are not used by directly by the 3D fixed functions but are made available to the SF thread. The number of valid vertices depends on the object type. *Primitive Assembly*
invertOrientation	enum	Indicates whether the orientation (CW or CCW winding order) of the vertices of a triangle object should be inverted. Ignored for non-triangle objects.
backFacing	enum	Valid only for points and line objects, indicates a back facing object. This is used later for culling.
provokingVtx	uint	Specifies the index (into the *v[]* arrays) of the vertex considered the "provoking" vertex (for flat shading). The selection of the provoking vertex is programmable via SF_STATE (**xxx Provoking Vertex Select** state variables.)
polyStippleEnable	boolean	TRUE if Polygon Stippling is enabled. FALSE for TRIFAN_NOSTIPPLE. Ignored for non-triangle objects.
continueStipple	boolean	Only applies to line objects. TRUE if Line Stippling should be continued (i.e., not reset) from where the previous line left off. If FALSE, Line Stippling is reset for each line object.
renderTargetIndex	uint	Index of the render target (array element or 3D slice), clamped to 0 by the GS unit if the max value was exceeded. This value is simply passed in SF thread payloads and not used within the SF unit.
viewPortIndex	uint	Index of a viewport transform matrix within the SF_VIEWPORT structure used to perform Viewport Transformation on object vertices and scissor operations on an object.
pointSize	unit	For point objects, this value specifies the screen space size (width,height) of the square point to be rasterized about the vertex position. Otherwise, the value is ignored.

The following table defines, for each primitive topology type, which vertex's VPIndex/RTAIndex applies to the objects within the topology.

VPIndex/RTAIndex Selection

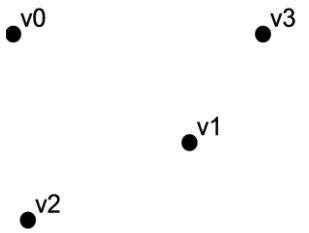
PrimTopologyType	Viewport Index Usage
POINTLIST POINTLIST_BF	Each vertex supplies the VPIndex for the corresponding point object
LINELIST	The leading vertex of each line supplies the VPIndex for the corresponding line object. V0.VPIndex-> Line(V0,V1) V2.VPIndex-> Line(V2,V3) ...
LINESTRIP LINESTRIP_BF LINESTRIP_CONT LINESTRIP_CONT_BF	The leading vertex of each line segment supplies the VPIndex for the corresponding line object. V0.VPIndex-> Line(V0,V1) V1.VPIndex-> Line(V1,V2) ... NOTE: If the VPIndex changes within the topology, shared vertices will be processed (mapped) multiple times.
TRILIST RECTLIST	The leading vertex of each triangle/rect supplies the VPIndex for the corresponding triangle/rect objects. V0.VPIndex-> Tri(V0,V1,V2) V3.VPIndex-> Tri(V3,V4,V5) ... NOTE: For Autostrips multiple viewport index is not supported. APIs defines the multiple viewport index to be output from Geometry shader that only generates Tristrips. If any other shader outputing multiple viewport indices for other topologies either autostrip needs to be disable or clipper guardband test needs to be disabled.
TRISTRIP TRISTRIP_REVERSE	The leading vertex of each triangle supplies the VPIndex for the corresponding triangle object. V0.VPIndex-> Tri(V0,V1,V2) V1.VPIndex-> Tri(V1,V2,V3) ... NOTE: If the VPIndex changes within the primitive, shared vertices will be processed (mapped) multiple times.

PrimTopologyType	Viewport Index Usage
TRIFAN TRIFAN_NOSTIPPLE POLYGON	The first vertex (V0) supplies the VPIndex for all triangle objects.

Point List Decomposition

The 3DPRIM_POINTLIST and 3DPRIM_POINTLIST_BACKFACING primitives specify a list of independent points.

3DPRIM_POINTLIST Primitive



The decomposition process divides the list into a series of basic 3DOBJ_POINT objects that are then passed individually and in order to the Object Setup subfunction. The *provokingVertex* of each object is, by definition, v[0].

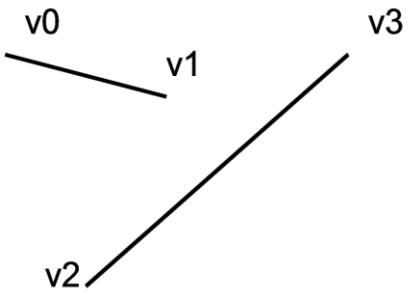
Points have no winding order, so the primitive command is used to explicitly state whether they are back-facing or front-facing points. Primitives of type 3DPRIM_POINTLIST_BACKFACING are decomposed exactly the same way as 3DPRIM_POINTLIST primitives, but the *backFacing* variable is set for resulting point objects being passed on to object setup.

```
PointListDecomposition()
{
   objectType = 3DOBJ_POINT
   nV = 1
   provokingVtx = 0
   if (primType == 3DPRIM_POINTLIST)
      {
         backFacing = FALSE
      }
   else // primType == 3DPRIM_POINTLIST_BACKFACING
      {
         backFacing = TRUE
      }

   for each (vertex in [0..vertexCount-1])
      {
         v[0] <- vIn[i] // copy all arrays
                        // (for example, v[]X, v[]Y, and so on)
         ObjectSetup()
      }
}
```

Line List Decomposition

The 3DPRIM_LINELIST primitive specifies a list of independent lines.

3DPRIM_LINELIST Primitive



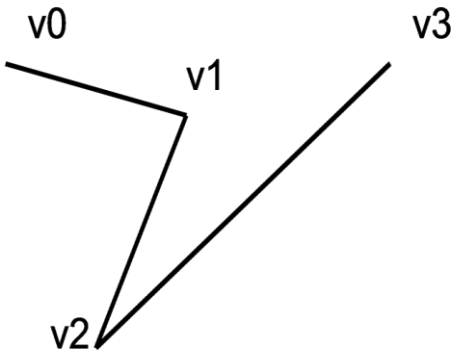
The decomposition process divides the list into a series of basic 3DOBJ_LINE objects that are then passed individually and in order to the Object Setup stage. The lines are generated with the following object vertex order: v0, v1; v2, v3; and so on. The *provokingVertex* of each object is taken from the **Line List/Strip Provoking Vertex Select** state variable, as programmed via SF_STATE.

```
LineListDecomposition()
   {
      objectType = 3DOBJ_LINE
      nV = 2
      provokingVtx = Line List/Strip Provoking Vertex Select       continueStipple = FALSE
      for each (vertex I in [0..vertexCount-2] by 2)
         {
             v[0] arrays <- vIn[i] arrays
             v[1] arrays <- vIn[i+1] arrays
             ObjectSetup()
         }
   }
```

Line Strip Decomposition

The 3DPRIM_LINESTRIP, 3DPRIM_LINESTRIP_CONT, 3DPRIM_LINESTRIP_BF, and 3DPRIM_LINESTRIP_CONT_BF primitives specify a list of connected lines.

3DPRIM_LINESTRIP_xxx Primitive



The decomposition process divides the strip into a series of basic 3DOBJ_LINE objects that are then passed individually and in order to the Object Setup stage. The lines are generated with the following object vertex order: v0,v1; v1,v2; and so on. The *provokingVertex* of each object is taken from the **Line List/Strip Provoking Vertex Select** state variable, as programmed via SF_STATE.

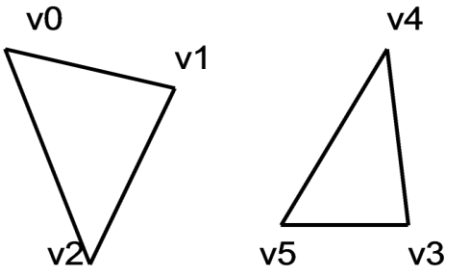
Lines have no winding order, so the primitive command is used to explicitly state whether they are back-facing or front-facing lines. Primitives of type 3DPRIM_LINESTRIP[_CONT]_BF are decomposed exactly the same way as 3DPRIM_LINESTRIP[_CONT] primitives, but the *backFacing* variable is set for the resulting line objects being passed on to object setup. Likewise, 3DPRIM_LINESTRIP_CONT[_BF] primitives are decomposed identically to basic line strips, but the *continueStipple* variable is set to true so that the line stipple pattern will pick up from where it left off with the last line primitive, rather than being reset.

```
LineStripDecomposition()
   {
      objectType = 3DOBJ_LINE
      nV = 2
      provokingVtx = Line List/Strip Provoking Vertex Select
      if (primType == 3DPRIM_LINESTRIP)
         {
             backFacing = FALSE
             continueStipple = FALSE
         } else if (primType == 3DPRIM_LINESTRIP_BF)
            {
               backFacing = TRUE
               continueStipple = FALSE
            } else if (primType == 3DPRIM_LINESTRIP_CONT)
               {
                  backFacing = FALSE
                  continueStipple = TRUE
               } else if (primType == 3DPRIM_LINESTRIP_CONT_BF)
                  {
                     backFacing = TRUE
                     continueStipple = TRUE
                  }
      for each (vertex I in [0..vertexCount-1])
         {
             v[0] arrays <- vIn[i] arrays
             v[1] arrays <- vIn[i+1] arrays
             ObjectSetup()
             continueStipple = TRUE
         }
   }
```

Triangle List Decomposition

The 3DPRIM_TRILIST primitive specifies a list of independent triangles.

3DPRIM_TRILIST Primitive



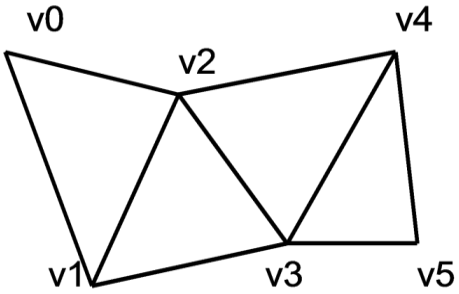
The decomposition process divides the list into a series of basic 3DOBJ_TRIANGLE objects that are then passed individually and in order to the Object Setup stage. The triangles are generated with the following object vertex order: v0,v1,v2; v3,v4,v5; and so on. The *provokingVertex* of each object is taken from the **Triangle List/Strip Provoking Vertex Select** state variable, as programmed via SF_STATE.

```
TriangleListDecomposition() {
  objectType = 3DOBJ_TRIANGLE
      nV = 3
  invertOrientation = FALSE
  provokingVtx = Triangle List/Strip Provoking Vertex Select
  polyStippleEnable = TRUE
  for each (vertex I in [0..vertexCount-3] by 3)
      {
          v[0] arrays <- vIn[i] arrays
          v[1] arrays <- vIn[i+1] arrays
          v[2] arrays <- vIn[i+2] arrays
          ObjectSetup()
      }
}
```

Triangle Strip Decomposition

The 3DPRIM_TRISTRIP and 3DPRIM_TRISTRIP_REVERSE primitives specify a series of triangles arranged in a strip, as illustrated below.

3DPRIM_TRISTRIP[_REVERSE] Primitive



The decomposition process divides the strip into a series of basic 3DOBJ_TRIANGLE objects that are then passed individually and in order to the Object Setup stage. The triangles are generated with the following object vertex order: v0,v1,v2; v1,v2,v3; v2,v3,v4; and so on. Note that the *winding order* of the vertices alternates between CW (clockwise), CCW (counter-clockwise), CW, etc. The *provokingVertex* of each object is taken from the **Triangle List/Strip Provoking Vertex Select** state variable, as programmed via SF_STATE.

The 3D pipeline uses the winding order of the vertices to distinguish between front-facing and back-facing triangles (Triangle Orientation (Face) Culling below). Therefore, the 3D pipeline must account for the alternation of winding order in strip triangles. The *invertOrientation* variable is generated and used for this purpose.

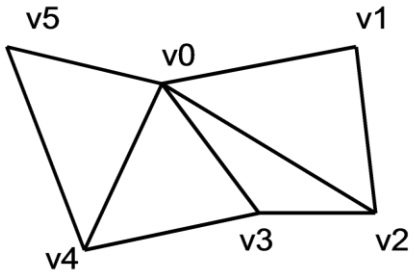
To accommodate the situation where the driver is forced to break an input strip primitive into multiple tristrip primitive commands (for example, due to ring or batch buffer size restrictions), two tristrip primitive types are supported. 3DPRIM_TRISTRIP is used for the initial section of a strip, and wherever a continuation of a strip starts with a triangle with a CW winding order. 3DPRIM_TRISTRIP_REVERSE is used for a continuation of a strip that starts with a triangle with a CCW winding order.

```
TriangleStripDecomposition()
   {
      objectType = 3DOBJ_TRIANGLE
      nV = 3
      provokingVtx = Triangle List/Strip Provoking Vertex Select
      if (primType == 3DPRIM_TRISTRIP)
         invertOrientation = FALSE
         else // primType == 3DPRIM_TRISTRIP_REVERSE
            invertOrientation = TRUE
      polyStippleEnable = TRUE
      for each (vertex I in [0..vertexCount-3])
        {
            v[0] arrays <- vIn[i] arrays
            v[1] arrays <- vIn[i+1] arrays
            v[2] arrays <- vIn[i+2] arrays
            ObjectSetup()
            invertOrientation = ! invertOrientation
        }
   }
```

Triangle Fan Decomposition

The 3DPRIM_TRIFAN and 3DPRIM_TRIFAN_NOSTIPPLE primitives specify a series of triangles arranged in a fan, as illustrated below.

3DPRIM_TRIFAN Primitive



The decomposition process divides the fan into a series of basic 3DOBJ_TRIANGLE objects that are then passed individually and in order to the Object Setup stage. The triangles are generated with the following object vertex order: v0,v1,v2; v0,v2,v3; v0,v3,v4; and so on. As there is no alternation in the vertex winding order, the *invertOrientation* variable is output as FALSE unconditionally. The *provokingVertex* of each object is taken from the **Triangle Fan Provoking Vertex** state variable, as programmed via SF_STATE.

Primitives of type 3DPRIM_TRIFAN_NOSTIPPLE are decomposed exactly the same way, except the *polyStippleEnable* variable is FALSE for the resulting objects being passed on to object setup. This will inhibit polygon stipple for these triangle objects.

```
TriangleFanDecomposition()
   {
       objectType = 3DOBJ_TRIANGLE
       nV = 3
       invertOrientation = FALSE
       provokingVtx = Triangle Fan Provoking Vertex Select
       if (primType == 3DPRIM_TRIFAN)
           polyStippleEnable = TRUE
           else // primType == 3DPRIM_TRIFAN_NOSTIPPLE
              polyStippleEnable = FALSE

       v[0] arrays <- vIn[0] arrays
         // the 1st vertex is common

       for each (vertex I in [1..vertexCount-2])
          {
              v[1] arrays <- vIn[i] arrays
              v[2] arrays <- vIn[i+1] arrays
              ObjectSetup()
          }
   }
```

Polygon Decomposition

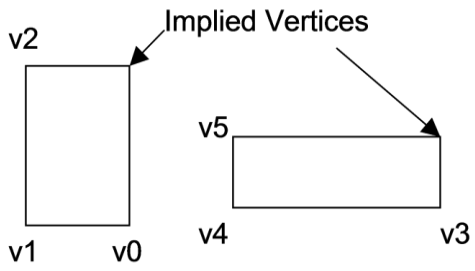
The 3DPRIM_POLYGON primitive is identical to the 3DPRIM_TRIFAN primitive with the exception that the *provokingVtx* is overridden with 0. This support has been added specifically for OpenGL support, avoiding

the need for the driver to change the provoking vertex selection when switching between trifan and polygon primitives.

Rectangle List Decomposition

The 3DPRIM_RECTLIST primitive command specifies a list of independent, axis-aligned rectangles. Only the lower right, lower left, and upper left vertices (in that order) are included in the command - the upper right vertex is derived from the other vertices (in Object Setup).

3DPRIM_RECTLIST Primitive



The decomposition of the 3DPRIM_RECTLIST primitive is identical to the 3DPRIM_TRILIST decomposition, with the exception of the *objectType* variable.

```
RectangleListDecomposition()
   {
       objectType = 3DOBJ_RECTANGLE
       nV = 3
       invertOrientation = FALSE
       provokingVtx = 0
       for each (vertex I in [0..vertexCount-3] by 3)
          {
              v[0] arrays <-  vIn[i] arrays
              v[1] arrays <-  vIn[i+1] arrays
              v[2] arrays <-  vIn[i+2] arrays
              ObjectSetup()
          }
   }
```

Object Setup

The Object Setup subfunction of the SF stage takes the post-viewport-transform data associated with each vertex of a basic object and computes various parameters required for scan conversion. This includes generation of implied vertices, translations and adjustments on vertex positions, and culling (removal) of certain classes of objects. The final object information is passed to the Windower/Masker (WM) stage where the object is rasterized into pixels.

Invalid Position Culling (Pre/Post-Transform)

At input the the SF stage, any objects containing a floating-point NaN value for Position X, Y, Z, or RHW will be unconditionally discarded. Note that this occurs on an object (not primitive) basis.

If Viewport Transformation is enabled, any objects containing a floating-point NaN value for post-transform Position X, Y or Z will be unconditionally discarded.

Viewport Transformation

If the **Viewport Transform Enable** bit of SF_STATE is ENABLED, a viewport transformation is applied to each vertex of the object.

The VPIndex associated with the leading vertex of the object is used to obtain the **Viewport Matrix Element** data from the corresponding element of the SF_VIEWPORT structure in memory. For each object vertex, the following scale and translate transformation is applied to the position coordinates:

x' = **m00** * x + **m30**

y' = **m11** * y + **m31**

z' = **m22** * z + **m32**

Software is responsible for computing the matrix elements from the viewport information provided to it from the API.

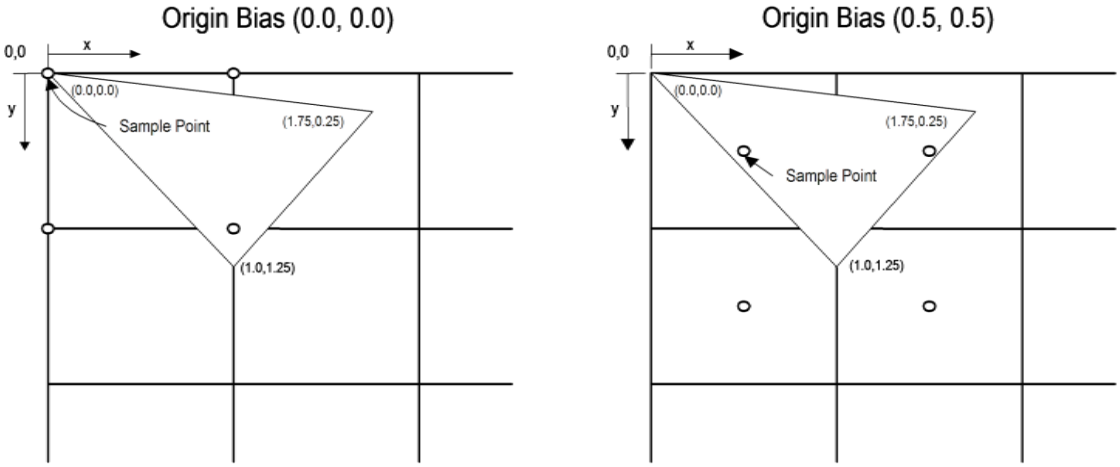
Destination Origin Bias

The positioning of the pixel sampling grid is programmable and is controlled by the **Destination Origin Horizontal/Vertical Bias** state variables (set via SF_STATE). If these bias values are both 0, pixels are sampled on an integer grid. Pixel (0,0) will be considered inside the object if the sample point at XY coordinate (0,0) falls within the primitive.

If the bias values are both 0.5, pixels are sampled on a "half" integer grid (i.e., X.5, Y.5). Pixel (0,0) will be considered inside the object if the sample point at XY coordinate (0.5,0.5) falls within the primitive. This positioning of the sample grid corresponds with the OpenGL rasterization rules, where "fragment centers" lay on a half-integer grid. It also corresponds with the Intel740 rasterizer (though that device did not employ "top left" rules).

Note that subsequent descriptions of rasterization rules for the various objects will be with reference to the pixel sampling grid.

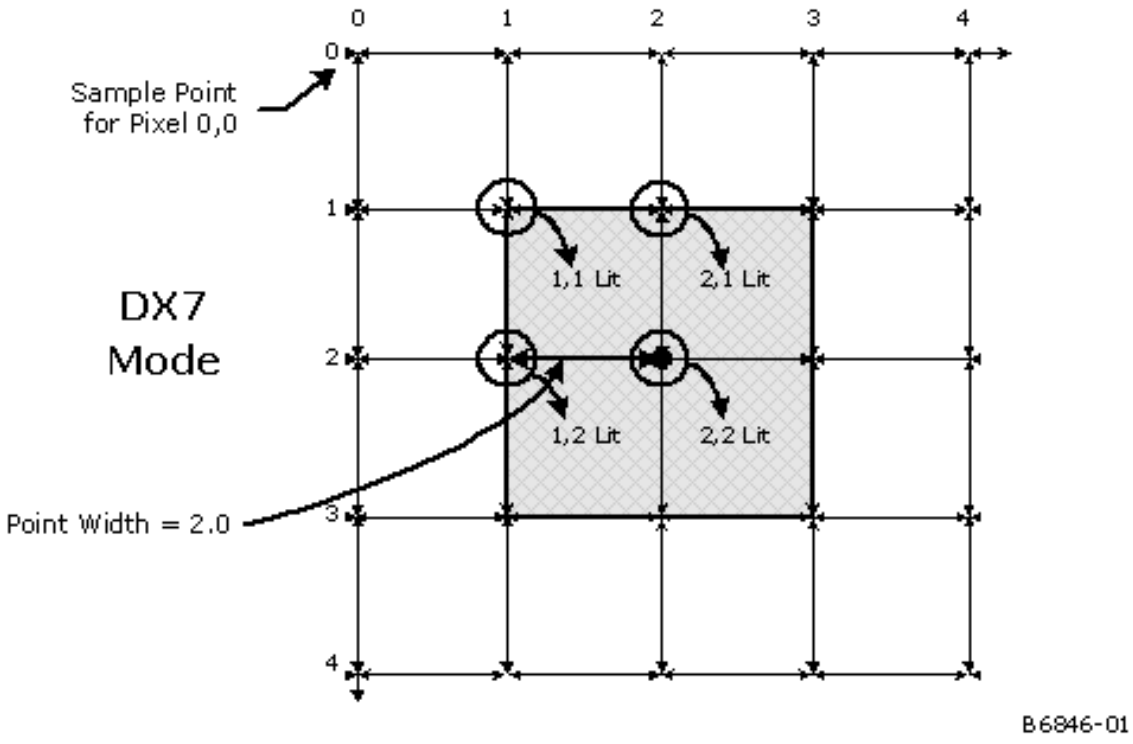
Destination Origin Bias



Point Rasterization Rule Adjustment

POINT objects are rasterized as square RECTANGLEs, with one exception: The **Point Rasterization Rule** state variable (in SF_STATE) controls the rendering of point object edges that fall directly on pixel sample points, as the treatment of these edge pixels varies between APIs.

RASTRULE_UPPER_LEFT

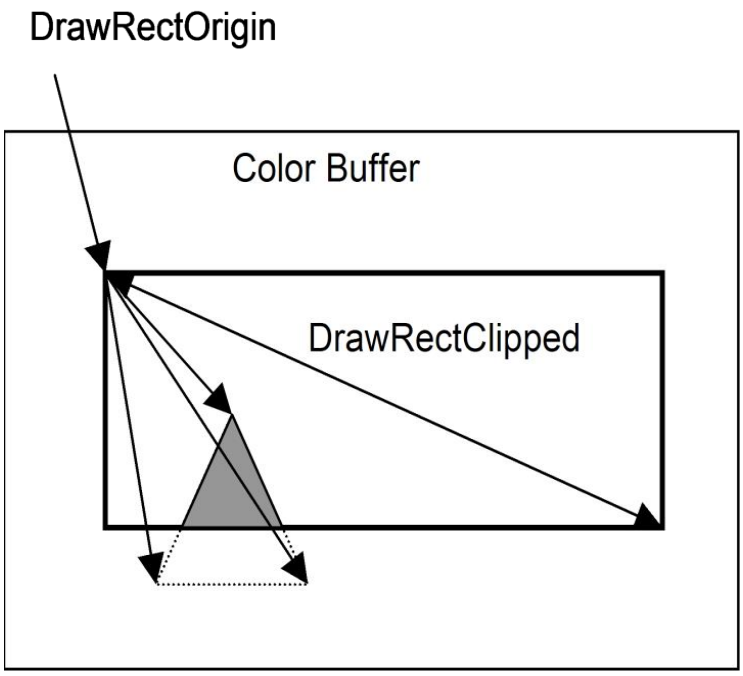


Drawing Rectangle Offset Application

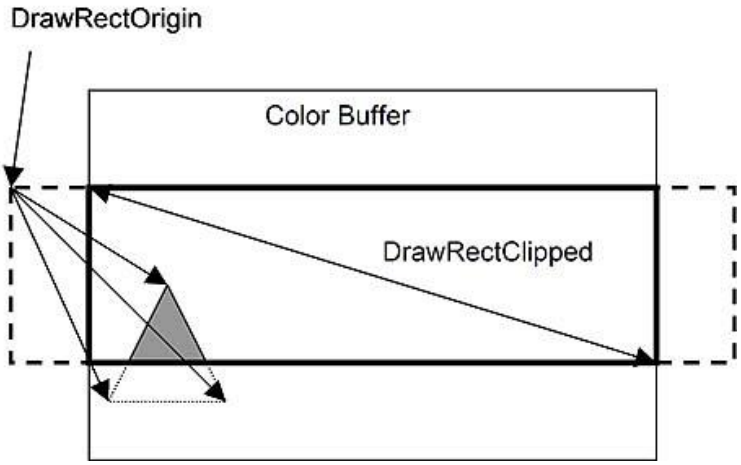
The Drawing Rectangle Offset subfunction offsets the object's vertex X,Y positions by the pixel-exact, unclipped drawing rectangle origin (as programmed via the **Drawing Rectangle Origin X,Y** values in the 3DSTATE_DRAWING_RECTANGLE command). The Drawing Rectangle Offset subfunction (at least with respect to Color Buffer access) is unconditional, and therefore to (effectively) turn off the offset function the origin would need to be set to (0,0). A non-zero offset is typically specified when window-relative or viewport-relative screen coordinates are input to the device. Here the drawing rectangle origin would be loaded with the absolute screen coordinates of the window's or viewport's upper-left corner.

Clipping of objects which extend outside of the Drawing Rectangle occurs later in the pipeline. Note that this clipping is based on the "clipped" draw rectangle (as programmed via the **Clipped Drawing Rectangle** values in the 3DSTATE_DRAWING_RECTANGLE command), which must be clamped by software to the rendertarget boundaries. The unclipped drawing rectangle origin, however, can extend outside the screen limits in order to support windows whose origins are moved off-screen. This is illustrated in the following diagrams.

Onscreen Draw Rectangle



Partially-offscreen Draw Rectangle



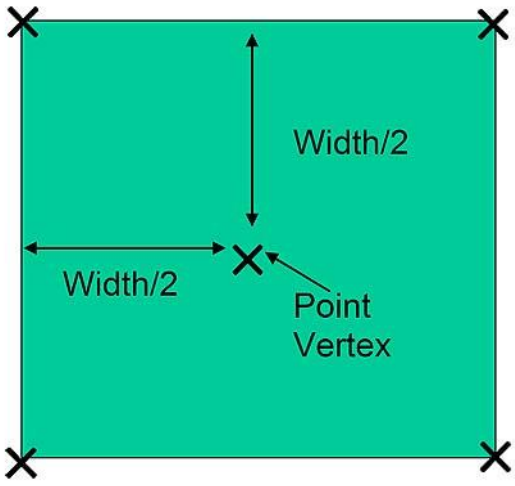
3DSTATE_DRAWING_RECTANGLE

Point Width Application

This stage of the pipeline applies only to 3DOBJ_POINT objects. Here the point object is converted from a single vertex to four vertices located at the corners of a square centered at the point's X,Y position. The width and height of the square are specified by a *point width* parameter. The **Point Width Source** value in SF_STATE determines the source of the point width parameter: the point width is either taken from the **Point Width** value programmed in SF_STATE or the PointWidth specified with the vertex (as read back from the vertex VUE earlier in the pipeline).

The corner vertices are computed by adding and subtracting one half of the point width.

Point Width Application

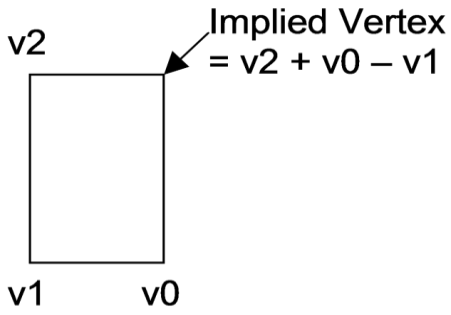


Z and W vertex attributes are copied from the single point center vertex to each of the four corner vertices.

Rectangle Completion

This stage of the pipeline applies only to 3DOBJ_RECTANGLE objects. Here the X,Y coordinates of the 4th (upper right) vertex of the rectangle object is computed from the first 3 vertices as shown in the following diagram. The other vertex attributes assigned to the implied vertex (v[3]) are UNDEFINED as they are not used. The Object Setup subfunction will use the values at only the first 3 vertices to compute attribute interpolants used across the entire rectangle.

Rectangle Completion



Vertex XY Clamping and Quantization

At this stage of the pipeline, vertex X and Y positions are in continuous screen (pixel) coordinates. These positions are quantized to subpixel precision by rounding the incoming values to the nearest subpixel (using round-to-nearest-or-even rules matching the DirectX reference device). The device supports rasterization with either 4 or 8 fractional (subpixel) position bits, as specified by the **Vertex SubPixel Precision Select** bit of SF_STATE.

The vertex X and Y screenspace coordinates are also *clamped* to the fixed-point "guardband" range supported by the rasterization hardware, as listed in the following table:

Per-Device Guardband Extents

Supported X,Y ScreenSpace "Guardband" Extent	Maximum Post-Clamp Delta (X or Y)
[-32K,32K-1]	N/A

Note that this clamping occurs after the Drawing Rectangle Origin has been applied and objects have been expanded (i.e., points have been expanded to squares, etc.). In almost all circumstances, if an object's vertices are actually modified by this clamping (i.e., had X or Y coordinates outside of the guardband extent the rendered object will not match the intended result. Therefore, software should take steps to ensure that this does not happen - e.g., by clipping objects such that they do not exceed these limits after the Drawing Rectangle is applied.

In addition, in order to be correctly rendered, objects must have a screenspace bounding box not exceeding 8K in the X or Y direction. This additional restriction must also be comprehended by software, i.e., enforced by use of clipping.

Degenerate Object Culling

At this stage of the pipeline, "degenerate" objects are discarded. This operation is automatic and cannot be disabled. (The object rasterization rules would by definition cause these objects to be "invisible" - this culling operation is mentioned here to reinforce that the device implementation optimizes these degeneracies as early as possible).

Degenerate Objects for definitions of degenerate objects.

Degenerate Objects

objType	Degenerate Object Definition
3DOBJ_POINT	Two or more corner vertices are coincident (i.e., the radius quantized to zero)
3DOBJ_LINE	The endpoints are coincident
3DOBJ_TRIANGLE	All three vertices are collinear or any two vertices are coincident and SOLID fill mode applies to the triangle
3DOBJ_RECTANGLE	Two or more corner vertices are coincident

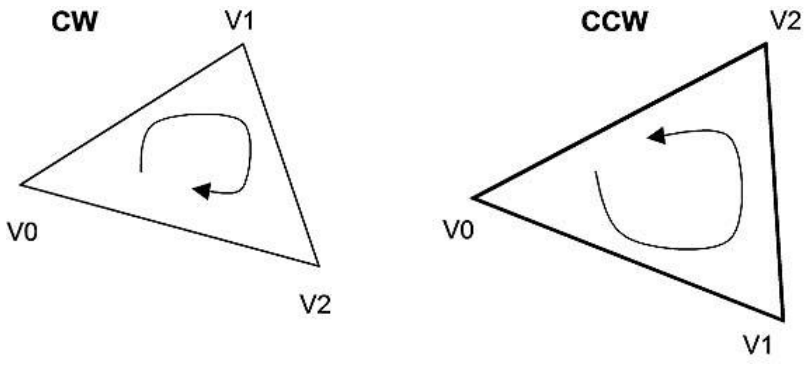
Triangle Orientation (Face) Culling

At this stage of the pipeline, 3DOBJ_TRIANGLE objects can be optionally discarded based on the "face orientation" of the object. This culling operation does not apply to the other object types.

This operation is typically called "back face culling", though front facing objects (or all 3DOBJ_TRIANGLE objects) can be selected to be discarded as well. Face culling is typically used to eliminate triangles facing away from the viewer, thus reducing rendering time.

The "winding order" of a triangle is defined by the the triangle vertex's 2D (X,Y) screen space position when traversed from v0 to v1 to v2. That traversal proceeds in either a clockwise (CW) or counter-clockwise (CCW) direction. The "winding order" of a triangle is defined by the the triangle vertex's 2D (X,Y) screen space position when traversed from v0 to v1 to v2. That traversal will proceed in either a clockwise (CW) or counter-clockwise (CCW) direction. A degenerate triangle is considered "backfacing", regardless of the FrontWinding state.

Triangle Winding Order



The **Front Winding** state variable in SF_STATE controls whether CW or CCW triangles are considered as having a "front-facing" orientation (at which point non-front-facing triangles are considered "back-

facing"). The internal variable *invertOrientation* associated with the triangle object is then used to determine whether the orientation of a that triangle should be inverted. Recall that this variable is set in the Primitive Decomposition stage to account for the alternating orientations of triangles in strip primitives resulting form the ordering of the vertices used to process them.

The **Cull Mode** state variable in SF_STATE specifies how triangles are discarded according to their resultant orientation. See Degenerate Objects.

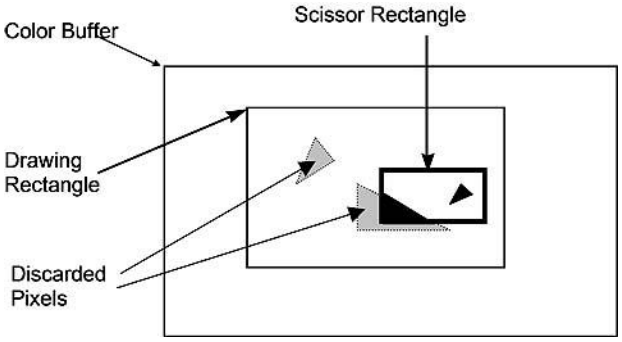
Cull Mode

CullMode	Definition
CULLMODE_NONE	The face culling operation is disabled.
CULLMODE_FRONT	Triangles with "front facing" orientation are discarded.
CULLMODE_BACK	Triangles with "back facing" orientation are discarded.
CULLMODE_BOTH	All triangles are discarded.

Scissor Rectangle Clipping

A *scissor* operation can be used to restrict the extent of rendered pixels to a screen-space aligned rectangle. If the scissor operation is enabled, portions of objects falling outside of the intersection of the scissor rectangle and the clipped draw rectangle are clipped (pixels discarded).

The scissor operation is enabled by the **Scissor Rectangle Enable** state variable in SF_STATE. If enabled, the VPIndex associated with the leading vertex of the object is used to select the corresponding SF_VIEWPORT structure. Up to 16 structures are supported. The **Scissor Rectangle X,Y Min,Max** fields of the SF_VIEWPORT structure defines a scissor rectangle as a rectangle in integer pixel coordinates relative to the (unclipped) origin of the Drawing Rectangle. The scissor rectangle is defined relative to the Drawing Rectangle to better support the OpenGL API. (OpenGL specifies the "Scissor Box" in window-relative coordinates). This allows instruction buffers with embedded Scissor Rectangle definitions to remain valid even after the destination window (drawing rectangle) moves.



Specifying either scissor rectangle xmin > xmax or ymin > ymax will cause all polygons to be discarded for a given viewport (effectively a null scissor rectangle).

Viewport Extents Test

Viewport extents test can be used to restrict the extent of rendered pixels to the viewport extents. If this operation is enabled, portion of the objects falling outside of the intersection of the scissor rectangle (if enabled) and the clipped draw rectangle and viewport extents are clipped (pixels discarded). This operation similar to the scissor test except both have different enables and the viewport extents can be programmed to the fractional float values.

This operation is enabled by the **View Transform Enable** state variable in **SF_STATE**. If enabled, the VPIndex associated with the leading vertex of the object is used to select the corresponding **SF_CLIP_VIEWPORT** structure. Up to 16 structures are supported. The X/Y **Min/Max ViewPort** fields of the **SF_CLIP_VIEWPORT** structure defines viewport extents as a rectangle in float screen pixel coordinates relative to the (unclipped) origin of the Drawing Rectangle. Please note that these co-ordinates can be fractional values and hardware will do appropriate rounding and convert it to integer pixel co-ordinates (floor rouding used). This **View Transform Enable** state also controls the viewport transform so appropriate the viewport transform coefficients need to be populated in the SF_CLIP_VIEPWORT structure along with the viewport extents.

Final clip rectangle used to define the rendering area will now depend on three rectangles namely drawing rectangle, Scissor rectangle, Viewport Extents. If both **Scissor Rectangle Enable** and **View transform enable** are set then intersection of all rectangles (Viewport extents, Scissor rectangle, Draw rectangle) becomes final clip rectangle, while If only **Scissor Rectangle Enable** is enabled then the intersection of (Scissor rectangle, Draw rectangle) becomes final clip rectangle. If only **View transform enable** is enabled then intersection of (Viewport extents, Draw rectangle) become the final clip rectangle, while If none is enabled then (Draw rectangle) is the final clip rectangle.

Specifying either viewport extents xmin > xmax or ymin > ymax will cause all polygons to be discarded for a given viewport (effectively a null viewport).

Line Rasterization

The device supports three styles of line rendering: *zero-width* (*cosmetic*) lines, *non-antialiased* lines, and *antialiased* lines.

Non-antialiased lines are rendered as a polygon having a specified width as measured parallel to the major axis of the line. Antialiased lines are rendered as a rectangle having a specified width measured perpendicular to the line connecting the vertices.

The functions required to render lines are split between the SF and WM units. The SF unit is responsible for computing the overall geometry of the object to be rendered, including the pixel-exact bounding box, edge equations, etc., and therefore is provided with the screen-geometry-related state variables. The WM unit performs the actual scan conversion, determining the exact pixels included/excluded and coverage values for anti-aliased lines.

Zero-Width (Cosmetic) Line Rasterization

Note: The specification of zero-width line rasterization would be more correctly included in the WM Unit chapter, though is being included here to keep it with the rasterization details of the other line types.

When the **Line Width** is set to zero, the device will use special rules to rasterize zero-width ("cosmetic") lines. The **Anti-Aliasing Enable** state variable is ignored when **Line Width** is zero.

When the **LineWidth** is set to zero, the device will use special rules to rasterize "cosmetic" lines.

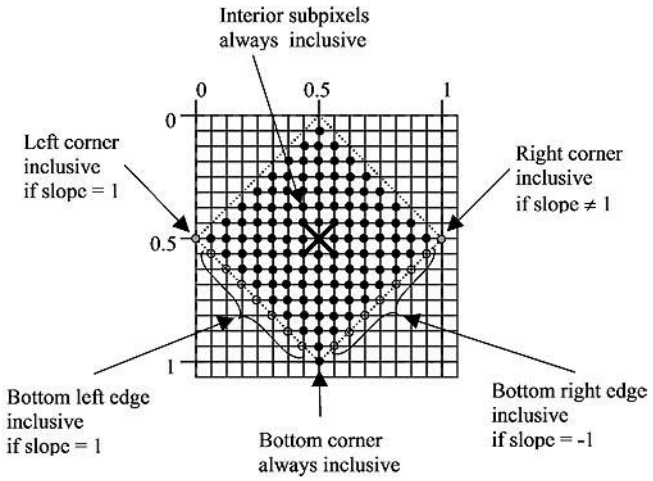
The rasterization rules also comply with the OpenGL conformance requirements (for 1-pixel wide non-smooth lines). Refer to the appropriate API specifications for details on these requirements.

The GIQ rules basically intersect the directed, ideal line connecting two endpoints with an array of diamond-shaped areas surrounding pixel sample points. Wherever the line exits a diamond (including passing through a diamond), the corresponding pixel is lit. Special rules are used to define the subpixel locations that are considered interior to the diamonds, as a function of the slope of the line. When a line ends in a diamond (and therefore does not exit that diamond), the corresponding pixel is not drawn. When a line starts in a diamond and exits that diamond, the corresponding pixel is drawn.

GIQ (Diamond) Sampling Rules - Legacy Mode

When the **Legacy Line Rasterization Enable** bit in WM_STATE is ENABLED, zero-width lines are rasterized according to the algorithm presented in this subsection. Also note that the **Last Pixel Enable** bit of SF_STATE controls whether the last pixel of the last line in a LINESTRIP_xxx primitive or the last pixel of each line in a LINELIST_xxx primitive is rendered.

Refer to the following figure, which shows the neighborhood of subpixels around a given pixel sample point. Note that the device divides a pixel into a 16x16 array of subpixels, referenced by their upper left corners.



The solid-colored subpixels are considered "interior" to the diamond centered on the pixel sample point. Here the Manhattan distance to the pixel sample point (center) is less than 1/2.

The subpixels falling on the edges of the diamond (Manhattan distance = 1/2) are exclusive, with the following exceptions:

1. **The bottom corner subpixel is always inclusive.** This is to ensure that lines with slopes in the open range (-1,1) touch a diamond even when they cross exactly between pixel diamonds.

2. **The right corner subpixel is inclusive as long as the line slope is not exactly one, in which case the left corner subpixel is inclusive.** Including the right corner subpixel ensures that lines with slopes in the range (1, +infinity] or [-infinity, -1) touch a diamond even when they cross exactly between pixel diamonds. Including the left corner on slope=1 lines is required for proper handling of slope=1 lines (see (3) below) - where if the right corner was inclusive, a slope=1 line falling exactly between pixel centers would wind up lighting pixel on both sides of the line (not desired).
3. **The subpixels along the bottom left edge are inclusive only if the line slope = 1.** This is to correctly handle the case where a slope=1 line falls enters the diamond through a left or bottom corner and ends on the bottom left edge. One does not consider this "passing through" the diamond (where the normal rules would have us light the pixel). This is to avoid the following case: One slope=1 line segment enters through one corner and ends on the edge, and another (continuation) line segments starts at that point on the edge and exits through the other corner. If simply passing through a corner caused the pixel to be lit, this case would case the pixel to be lit twice - breaking the rule that connected line segments should not cause double-hits or missing pixels. So, by considering the entire bottom left edge as "inside" for slope=1 lines, we will only light the pixel when a line passes through the entire edge, or starts on the edge (or the left or bottom corner) and exits the diamond.
4. **The subpixels along the bottom right edge are inclusive only if the line slope = -1.** Similar case as (3), except slope=-1 lines require the bottom right edge to be considered inclusive.

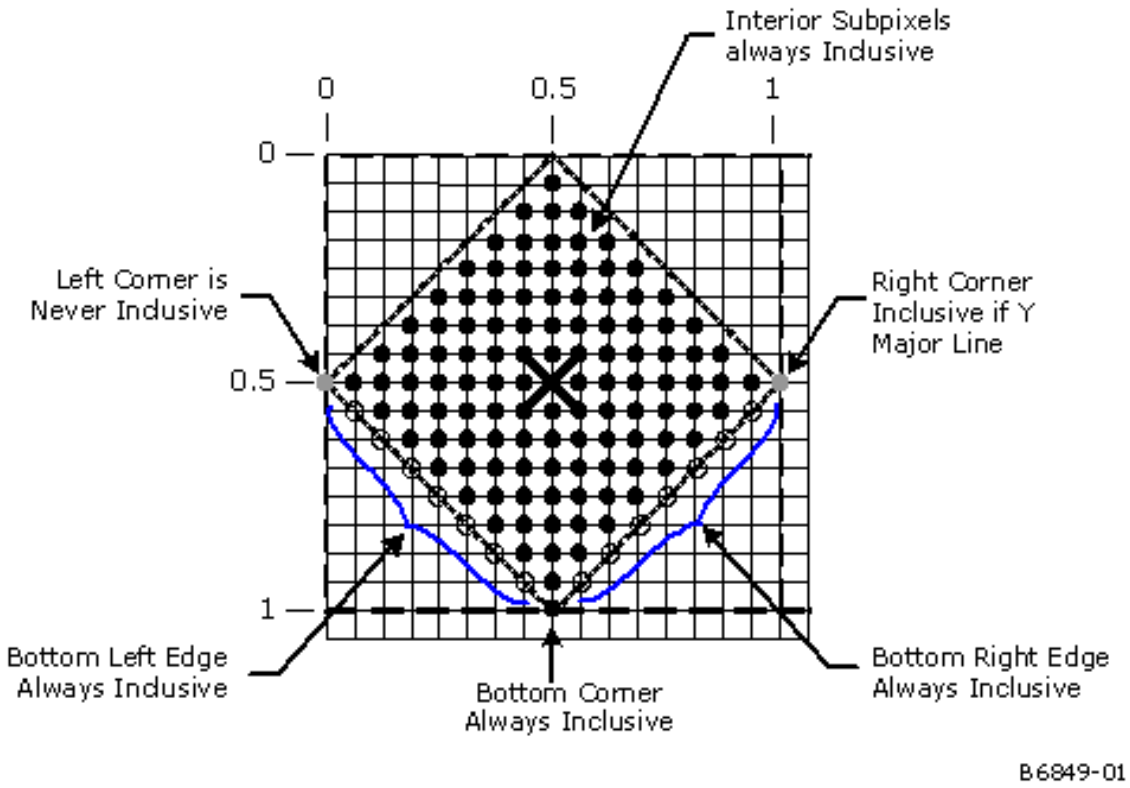
The following equation determines whether a point (point.x, point.y) is inside the diamond of the pixel sample point (sample.x, sample.y), given additional information about the slope (slopePosOne, slopeNegOne).

```
      delta_x                          = point.x - sample.x
      delta_y                          = point.y - sample.y
      distance                   = abs(delta_x) + abs(delta_y)
      interior                   = (distance < 0.5)
      bottom_corner              = (delta_x == 0.0) && (delta_y == 0.5)
      left_corner                = (delta_x == -0.5) && (delta_y == 0.0)
      right_corner               = (delta_x == 0.5) && (delta_y == 0.0)
      bottom_left_edge           = (distance == 0.5) && (delta_x < 0) && (delta_y > 0)
      bottom_right_edge          = (distance == 0.5) && (delta_x > 0) && (delta_y > 0)

      inside = interior || bottom_corner || (slopePosOne ? left_corner : right_corner) ||
(slopePosOne && left_edge) || (slopeNegOne && right_edge)
```

GIQ (Diamond) Sampling Rules - DX10 Mode

When the **Legacy Line Rasterization Enable** bit in WM_STATE is DISABLED, zero-width lines are rasterized according to the algorithm presented in this subsection. Also note that the **Last Pixel Enable** bit of SF_STATE controls whether the last pixel of the last line in a LINESTRIP_xxx primitive or the last pixel of each line in a LINELIST_xxx primitive is rendered.

Refer to the following figure, which shows the neighborhood of subpixels around a given pixel sample point. Note that the device divides a pixel into a 16x16 array of subpixels, referenced by their upper left corners.



The solid-colored subpixels are considered "interior" to the diamond centered on the pixel sample point. Here the Manhattan distance to the pixel sample point (center) is less than 1/2.

The subpixels falling on the edges of the diamond (Manhattan distance = 1/2) are exclusive, with the following exceptions:

1. **The bottom corner subpixel is always inclusive.** This is to ensure that lines with slopes in the open range (-1,1) touch a diamond even when they cross exactly between pixel diamonds.
2. **The right corner subpixel is inclusive as long as the line is not X Major (X Major is defined as -1 <= slope <= 1).** Including the right corner subpixel ensures that lines with slopes in the range (>1, +infinity] or [-infinity, <-1) touch a diamond even when they cross exactly between pixel diamonds.
3. **The left corner subpixel is never inclusive.** For Y Major lines, having the right corner subpixel as always inclusive requires that the left corner subpixel should never be inclusive, since a line falling exactly between pixel centers would wind up lighting pixel on both sides of the line (not desired).
4. **The subpixels along the bottom left edge are always inclusive.** This is to correctly handle the case where a line enters the diamond through a left or bottom corner and ends on the bottom left edge. One does not consider this "passing through" the diamond (where the normal rules would have us light the pixel). This is to avoid the following case: One line segment enters through one corner and ends on the edge, and another (continuation) line segments starts at that point on the edge and exits through the other corner. If simply passing through a corner caused the pixel to be lit, this case would cause the pixel to be lit twice - breaking the rule that connected line segments should not cause double-hits or missing pixels. So, by considering the entire bottom left edge as

"inside", the pixel is only lit when a line passes through the entire edge, or starts on the edge (or the left or bottom corner) and exits the diamond.

5. **The subpixels along the bottom right edge are always inclusive.** Same as case as (4), except slope=-1 lines require the bottom right edge to be considered inclusive.

The following equation determines whether a point (point.x, point.y) is inside the diamond of the pixel sample point (sample.x, sample.y), given additional information about the slope (XMajor).

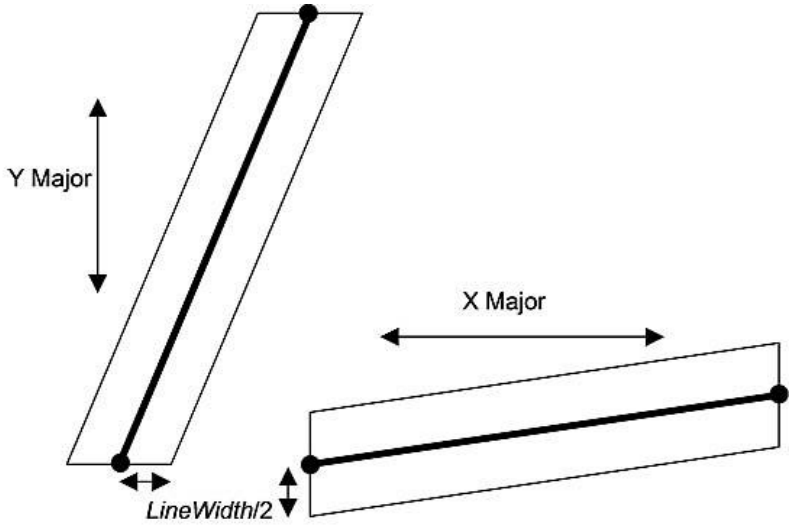
```
delta_x                         = point.x - sample.x
delta_y                         = point.y - sample.y
distance                        = abs(delta_x) + abs(delta_y)
interior                        = (distance < 0.5)
bottom_corner                           = (delta_x == 0.0)  && (delta_y == 0.5)
left_corner                     = (delta_x == -0.5) && (delta_y == 0.0)
right_corner                    = (delta_x == 0.5)  && (delta_y == 0.0)
bottom_left_edge                = (distance == 0.5) && (delta_x < 0) && (delta_y > 0)
bottom_right_edge               = (distance == 0.5) && (delta_x > 0) && (delta_y > 0)

inside = interior || bottom_corner || (!XMajor && right_corner) || ( bottom_left_edge)
|| ( bottom_right_edge)
```

Non-Antialiased Wide Line Rasterization

Non-anti-aliased, non-zero-width lines are rendered as parallelograms that are centered on, and aligned to, the line joining the endpoint vertices. Pixels sampled interior to the parallelogram are rendered; pixels sampled exactly on the parallelogram edges are rendered according to the polygon "top left" rules.

The parallelogram is formed by first determining the major axis of the line (diagonal lines are considered x-major). The corners of the parallelogram are computed by translating the line endpoints by +/-(**Line Width** / 2) in the direction of the minor axis, as shown in the following diagram.

Non-Antialiased Line Rasterization

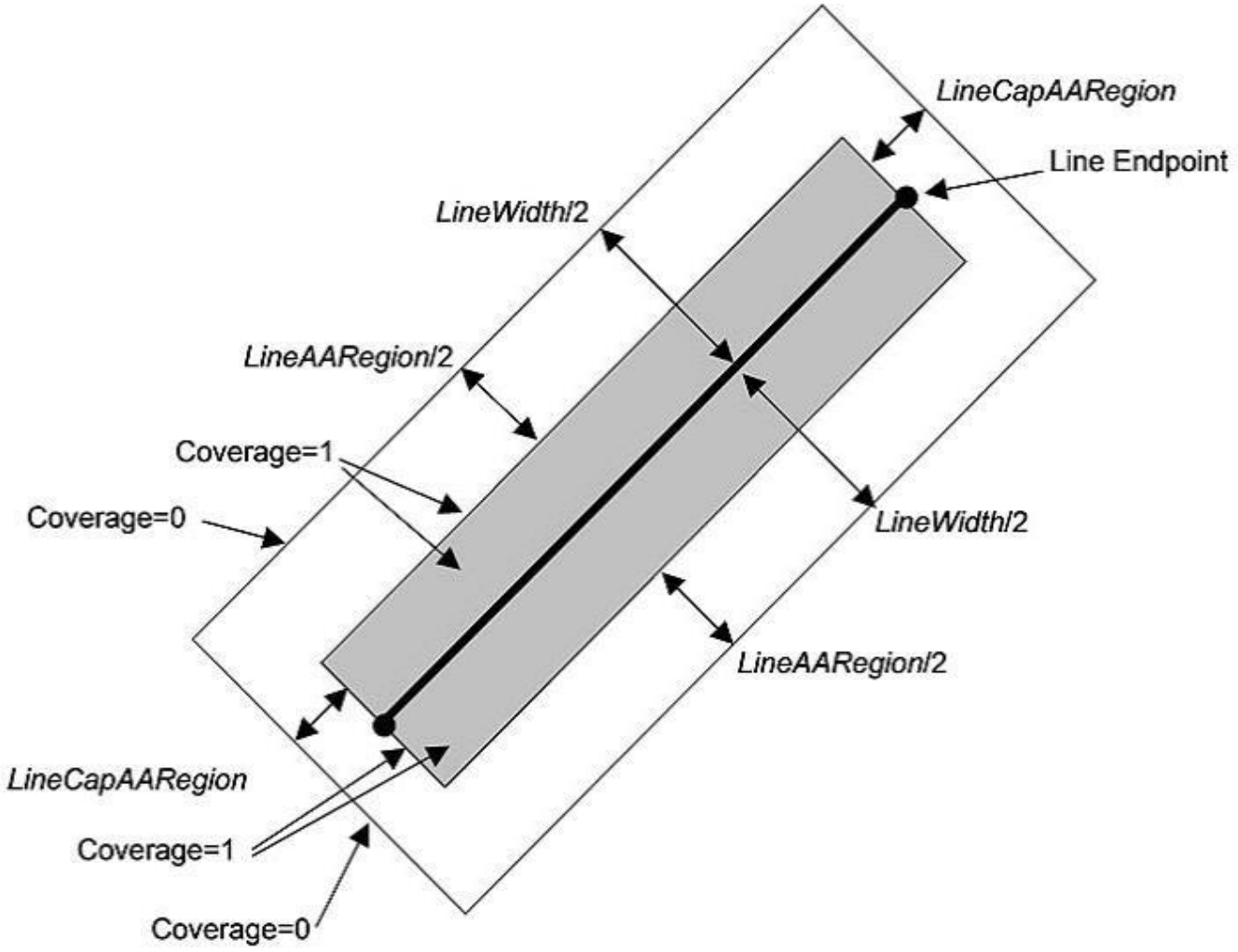


Anti-Aliased Line Rasterization

Anti-aliased lines are rendered as rectangles that are centered on, and aligned to, the line joining the endpoint vertices. For each pixel in the rectangle, a fractional coverage value (referred to as Antialias Alpha) is computed - this coverage value is normally used to attenuate the pixel's alpha in the pixel shader thread. The resultant alpha value is therefore available for use in those downstream pixel pipeline stages to generate the desired effect (e.g., use the attenuated alpha value to modulate the pixel's color, and add the result to the destination color, etc.). Note that software is required to explicitly program the pixel shader and pixel pipeline to obtain the desired anti-aliasing effect - the device simply makes the coverage-attenuated pixel alpha values available for use in the pixel shader.

The dimensions of the rendered rectangle, and the parameters controlling the coverage value computation, are programmed via the **Line Width**, **Line AA Region**, and **Line Cap AA Region** state variables, as shown below. The edges parallel to the line are located at the distance (*LineWidth*/2) from the line (measured in screen pixel units perpendicular to the line). The end-cap edges are perpendicular to the line and located at the distance (*LineCapAARegion*) from the endpoints.

Anti-aliased Line Rasterization



Along the parallel edges, the coverage values ramp from the value 0 at the very edges of the rectangle to the value 1 at the perpendicular distance (*LineAARegion*/2) from a given edge (in the direction of the line). A pixel's coverage value is computed with respect to the closest edge. In the cases where (*LineAARegion*/2) < (*LineWidth*/2), this results in a region of fractional coverage values near the edges of the rectangle, and a region of "fully-covered" coverage values (i.e., the value 1) at the interior of the line. When (*LineAARegion*/2) == (*LineWidth*/2), only pixel sample points falling exactly on the line can generate fully-covered coverage values. If (*LineAARegion*/2) > (*LineWidth*/2), no pixels can be fully-covered (it is expected that this case is not typically desired).

Along the end cap edges, the coverage values ramp from the value 1 at the line endpoint to the value 0 at the cap edge - itself at a perpendicular distance (*LineCapAARegion*) from the endpoint. Note that, unlike the line-parallel edges, there is only a single parameter (*LineCapAARegion*) controlling the extension of the line at the end caps and the associated coverage ramp.

The regions near the corners of the rectangle have coverage values influenced by distances from both the line-parallel and end cap edges - here the two coverage values are multiplied together to provide a composite coverage value.

The computed coverage value for each pixel is passed through the Windower Thread Dispatch payload. The Pixel Shader kernel should be passed (unmodified) by the shader to the Render Cache as part of it's output message.

SF Pipeline State Summary

3DSTATE_RASTER

3DSTATE_RASTER

Signal	SF_INT::Multisample Rasterization Mode
Description	This field determines whether multisample rasterization is enabled and how pixel sample points are defined.
Formula	See Table: WM_INT::Multisample Rasterization Mode in 3D Pipeline Windower > Windower Pipelined State > 3DSTATE_WM > 3DSTATE_WM

Signal	SF_INT::Global Depth Offset Enable Solid
Description	This field determines when Global Depth bias gets enabled.
Formula	= (3DSTATE_RASTER:: Global Depth Offset Enable Solid? ((3DSTATE_RASTER::Global Depth Offset Constant != IEEE_FP_ZERO) || (3DSTATE_RASTER::Global Depth Offset Scale != IEEE_FP_ZERO)): Disable

Signal	SF_INT:: Global Depth Offset Enable Wireframe
Description	This field determines when Global Depth bias gets enabled.
Formula	= (3DSTATE_RASTER:: Global Depth Offset Enable Wireframe? ((3DSTATE_RASTER::Global Depth Offset Constant != IEEE_FP_ZERO) || (3DSTATE_RASTER::Global Depth Offset Scale != IEEE_FP_ZERO)): Disable

Signal	SF_INT:: Global Depth Offset Enable Point
Description	This field determines when Global Depth bias gets enabled.
Formula	= (3DSTATE_RASTER:: Global Depth Offset Enable Point? ((3DSTATE_RASTER::Global Depth Offset Constant != IEEE_FP_ZERO) || (3DSTATE_RASTER::Global Depth Offset Scale != IEEE_FP_ZERO)): Disable

3DSTATE_SF

The state used by the SF stage is defined by this inline state packet.

3DSTATE_SF

The SF Unit also receives 3DSTATE_RASTER. It also receives 3DSTATE_INT which is transparent to SW. 3DSTATE_INT provides 3DSTATE_WM, 3DSTATE_WM_HZ_OP, 3DSTATE_DETPH_BUFFER, and 3DSTATE_MULTISAMPLE fields.

Signal	SF_INT::Number of Multisamples
Description	Set the number of multisamples.
Formula	= (WM_INT::WM_HZ_OP) ? 3DSTATE_ WM_HZ_OP::Number of Multisamples : 3DSTATE_MULTISAMPLE::Number of Multisamples

Signal	SF_INT::Pixel Position Offset Enable
Description	Enables the device to offset pixel positions by 0.5 both in horizontal and vertical directions.
Formula	= (WM_INT::WM_HZ_OP) ? 3DSTATE_ WM_HZ_OP:: Pixel Position Offset Enable: 3DSTATE_MULTISAMPLE:: Pixel Position Offset Enable

Signal	SF_INT::Pixel Position Offset
Description	Causes the device to offset pixel positions by 0.5 both in horizontal and vertical directions. It is to be noted this is done to adjust the pixel co-ordinate system to DX9 like, so any screen space rectangles (eg: HiZ Clear, Resolve etc) generated internally by driver in this mode needs to be aware of this offset adjustment and send the rectangles according to alignment restriction taking this offset adjustment into consideration.

Formula	= (SF_INT::Number of Multisamples >1) && (3DSTATE_MULTISAMPLE:: Pixel Location == PIXLOC_UL_CORNER) && SF_INT::Pixel Position Offset Enable

Signal	SF_INT: Global Depth Offset Enable Solid
Description	Set the number of multisamples
Formula	= (3DSTATE_RASTER:: Global Depth Offset Enable Solid ? ((3DSTATE_RASTER::Global Depth Offset Constant != IEEE_FP_ZERO) || (3DSTATE_RASTER::Global Depth Offset Scale != IEEE_FP_ZERO)): Disable

Signal	SF_INT: Global Depth Offset Enable Wireframe
Description	Set the number of multisamples
Fromula	= (3DSTATE_RASTER:: Global Depth Offset Enable Wireframe ? ((3DSTATE_RASTER::Global Depth Offset Constant != IEEE_FP_ZERO) || (3DSTATE_RASTER::Global Depth Offset Scale != IEEE_FP_ZERO)): Disable

Signal	SF_INT: Global Depth Offset Enable Point
Description	Set the number of multisamples.
Formula	= (3DSTATE_RASTER:: Global Depth Offset Enable Point ? ((3DSTATE_RASTER::Global Depth Offset Constant != IEEE_FP_ZERO) || (3DSTATE_RASTER::Global Depth Offset Scale != IEEE_FP_ZERO)): Disable

Signal	SF_INT::FrontFace Fill Mode
Description	This state controls how front-facing triangle and rectangle objects are rendered.
Formula	= 3DSTATE_INT::WM_HZ_OP ? SOLID : 3DSTATE_RASTER:: FrontFace Fill Mode

Signal	SF_INT::BackFace Fill Mode
Description	This state controls how Back-facing triangle and rectangle objects are rendered.
Formula	= 3DSTATE_INT::WM_HZ_OP ? SOLID : 3DSTATE_RASTER:: BackFace Fill Mode

Signal	SF_INT::FrontWinding
Description	Determines whether a triangle object is considered "front facing" if the screen space vertex positions, when traversed in the order, result in a clockwise (CW) or counter-clockwise (CCW) winding order. Does not apply to points or lines.
Formula	= 3DSTATE_INT::WM_HZ_OP ? FRONTWINDING_CW : 3DSTATE_RASTER::FrontWinding

Signal	SF_INT::Cull Mode
Description	Controls removal (culling) of triangle objects based on orientation. The cull mode only applies to triangle objects and does not apply to lines, points or rectangles.
Formula	= SF_INT::WM_HZ_OP ? CULLMODE_BACK : 3DSTATE_RASTER:: Cull Mode

Signal	SF_INT::Scissor Rectangle Enable
Description	This field is used to control whether the Viewport Z extents (near, far) are considered in VertexClipTest.
Formula	= SF_INT::WM_HZ_OP ?

	3DSTATE_WM_HZ_OP:: Scissor Rectangle Enable : 3DSTATE_RASTER::Scissor Rectangle Enable

Signal	SF_INT::Anti-aliasing Enable
Description	This field enables "alpha-based" line antialiasing.
Formula	= = SF_INT::WM_HZ_OP ? 3DSTATE_WM_HZ_OP:: Scissor Rectangle Enable : 3DSTATE_RASTER::Anti-aliasing Enable

Signal	SF_INT::Global Depth Offset Constant
Description	Specifies the constant term in the Global Depth Offset function.
Formula	= 3DSTATE_RASTER::Global Depth Offset Constant

Signal	SF_INT::Global Depth Offset Scale
Description	Specifies the constant term in the Global Depth Offset function.
Formula	= 3DSTATE_RASTER::Global Depth Offset Scale

Signal	SF_INT::Global Depth Offset Clamp
Description	Specifies the clamp term used in the Global Depth Offset function.
Formula	= 3DSTATE_RASTER::Global Depth Offset Clamp

Signal	SF_INT::Line Stipple Enable
Description	Specifies the clamp term used in the Global Depth Offset function.
Formula	= 3DSTATE_WM::Line Stipple Enable

Signal	SF_INT::RT Independent Rasterization Enable
Description	Enables RT Independent Rasterization.
Formula	= 3DSTATE_INT::WM_HZ_OP ? Disable : 3DSTATE_RASTER::ForcedSampleCount != NUMRASTSAMPLES_0

Signal	SF_INT::WM_HZ_OP
Description	Enables WM_HZ_OP.
Formula	= (3DSTATE_WM_HZ_OP::DepthBufferClear || 3DSTATE_WM_HZ_OP::DepthBufferResolve || 3DSTATE_WM_HZ_OP::Hierarchical Depth Buffer Resolve Enable || 3DSTATE_WM_HZ_OP::StencilBufferClear || 3DSTATE_WM_HZ_OP::StencilBufferResolve) ? Enable : Disable

Signal	SF_INT:: View Transform Enable
Description	Enables View Transform
Formula	= SF_INT::WM_HZ_OP ? Disable : 3DSTATE_SF::View Transform Enable

Signal	SF_INT::Render Target Array index
Description	Render Target Array index being render to
Formula	= 3DSTATE_INT::WM_HZ_OP ? 0 : Render Target Array index pipelined from clipper

Signal	SF_INT::Depth Buffer Surface Format
Description	Depth format being used
Formula	= 3DSTATE_INT:: Depth Buffer Surface Format

Signal	SF_INT::Viewport index
Description	Viewport being used.
Formula	= SF_INT::WM_HZ_OP ? 0 : Viewport index pipelined from clipper

SF_CLIP_VIEWPORT

The viewport-specific state used by both the SF and CL units (SF_CLIP_VIEWPORT) is stored as an array of up to 16 elements, each of which contains the DWords described below. The start of each element is spaced 16 DWords apart. The location of the first element of the array, as specified by both **Pointer to SF_VIEWPORT** and **Pointer to CLIP_VIEWPORT**, is aligned to a 64-byte boundary.

SCISSOR_RECT

Attribute Interpolation Setup

With the attribute interpolation setup function being implemented in hardware, a number of state fields in 3DSTATE_SF are utilized to control interpolation setup.

Number of SF Output Attributes sets the number of attributes that will be output from the SF stage, not including position. This can be used to specify up to 32, and may differ from the number of input attributes. The number of input attributes is derived from the **Vertex URB Entry Read Length** field. Note that this field is also used to specify whether swizzling is to be performed on Attributes 0-15 or Attributes 16-32. See the state field definition for details.

Attribute Swizzling

The first or last set of 16 attributes can be swizzled according to certain state fields. **Attribute Swizzle Enable** enables the swizzling for all 16 of these attributes, and each of the attributes has a 2-bit **Swizzle Select** field that controls swizzling with the following settings:

- INPUTATTR - This attribute is sourced from AttrInputReg[SourceAttribute].
- INPUTATTR_FACING - This attribute is sourced from AttrInputReg[SourceAttribute] if the object is front-facing, otherwise it is sourced from AttrInputReg[SourceAttribute+1].
- INPUTATTR_W - This attribute is sourced from AttrInputReg[SourceAttribute]. WYZW (the W component of the source is copied to the X component of the destination).
- INPUTATTR_FACING - If the object is front-facing, this attribute is sourced from AttrInputReg[SourceAttribute]. WYZW (the W component of the source is copied to the X component of the destination). If the object is front-facing, this attribute is sourced from AttrInputReg[SourceAttribute+1]. WYZW.

Each of the first or last set of 16 attributes also has a 5-bit **Source Attribute** field which specify, per output attribute (not component), which input attribute sources the output attribute when INPUTATTR is selected for **Swizzle Select**. A **Source Attribute** value of 0 corresponds to the 128-bit attribute immediately following the vertex 4D position. If INPUTATTR_FACING is selected, this specifies the first of two consecutive (front,back) input attributes, where the SourceAttribute value can be an odd or even number (just not 31, as that would place the back-face input attribute past the end of the input max complement of input attributes).

Constant overriding is also available for the first or last set of 16 attributes. Each attribute has a **Constant Source** field which specifies the constant values per swizzled attribute, with the following settings available:

- XYZW = 0000
- XYZW = 0001
- XYZW = 1111

Each channel of each attribute has a **Component Override** field to control whether the corresponding channel is overridden with the constant value defined in **Constant Source**.

Interpolation Modes

All 32 attributes have a **Constant Interpolation Enable** state field bit to specify whether all components of the post-swizzled attribute are to be interpolated as constant values (not varying over the pixels of the object). If set, the attribute at the provoking vertex is copied to a0, and a1 and a2 are set to zero - this results in a constant interpolation of the provoking vertex value. If clear, the attribute is linearly interpolated. Attributes 0-15 are further subjected to Wrap Shortest processing on a per-component basis, via the **Attribute WrapShortest Enables** state bitfields. WrapShortest processing modifies the a1 and/or a2 values depending on attribute deltas. All

The table below indicates the output values of a0, a1, and a2 depending on interpolation mode settings.

	a0	a1	a2
Constant	A0	0.0	0.0
Linear	A0	A1-A0	A2-A0
Wrap Shortest	A0	(A1-A0)+1 (A1-A0) <= -0.5 (A1-A0)-1 (A1-A0) >= 0.5 (A1-A0) otherwise	(A2-A0)+1 (A2-A0) <= -0.5 (A2-A0)-1 (A2-A0) >= 0.5 (A2-A0) otherwise

Point Sprites

Normally all vertex attributes (including texture coordinates) other than position are simply replicated from the incoming point center vertex to the generated point object (corner) vertices. However, both DX9 and OGL support "sprite points", where some/all texture coordinates are replaced with full-scale 2D texture coordinates.

A 32-bit **PointSprite TextureCoordinate Enable** bit mask controls whether the corresponding vertex attribute is to be replaced by a sprite point texture coordinate. The global (not per-attribute) **Point Sprite TextureCoordinate Origin** field controls how the point object vertex (top/bottom, left/right) texture coordinates are generated:

UPPERLEFT	Left	Right
Top	(0,0,0,1)	(1,0,0,1)
Bottom	(0,1,0,1)	(1,1,0,1)

LOWERLEFT	Left	Right
Top	(0,1,0,1)	(1,1,0,1)
Bottom	(0,0,0,1)	(1,0,0,1)

The state used by "setup backend" is defined by the following inline state packet.

3DSTATE_SBE

The state used by "setup backend" is defined by the following inline state packet.

3DSTATE_SBE_SWIZ

SBE Unit also receives 3DSTATE_INT which is transparent to SW. 3DSTATE_INT provides 3DSTATE_VS, 3DSTATE_DS, and 3DSTATE_GS fields.

Signal	SBD_INT::Vertex URB Entry Read Lenth
Description	Specifies the amount of URB data read for each Vertex URB entry, in 256-bit register increments.
Formula	= (3DSTATE_SBE::Force Vertex URB Entry Read Length == Force) ? 3DSTATE_SBE::Vertex URB Entry Read Length : 3DSTATE_GS::GS_Enable ? 3DSTATE_GS::Vertex URB Entry Output Length : 3DSTATE_DS::DS_Enable ? 3DSTATE_DS::Vertex URB Entry Output Length : 3DSTATE_VS::Vertex URB Entry Output Length

Signal	SBE_INT::Vertex URB Entry Read Offset
Description	Specifies the offset (in 256-bit units) at which Vertex URB data is to be read from the URB
Formula	= (3DSTATE_SBE::Force Vertex URB entry Offset == Force) ? 3DSTATE_SBE::Vertex URB Entry Read Offset: 3DSTATE_GS::GS_Enable ? 3DSTATE_GS:: Vertex URB Entry Output Read Offset: 3DSTATE_DS:DS_Enable ? 3DSTATE_DS:: Vertex URB Entry Output Read Offset : 3DSTATE_VS:: Vertex URB Entry Output Read Offset

Signal	SBE_INT::PrimId_override
Description	When true indicates that SBE provides the Primitive ID.
Formula	= 3DSTATE_GS::GS_Enable ? false : 3DSTATE_SBE::Primitive ID Override Component Select !=0

Barycentric Attribute Interpolation

Given hardware clipper and setup, some of the previous flexibility in the algorithm used to interpolate attributes is no longer available. Hardware uses barycentric parameters to aid in attribute interpolation, and these parameters are computed in hardware per-pixel (or per-sample) and delivered in the thread payload to the pixel shader. Also delivered in the payload are a set of vertex deltas (a0, a1, and a2) per channel of each attribute.

There are six different barycentric parameters that can be enabled for delivery in the pixel shader payload. These are enabled via the **Barycentric Interpolation Mode** bits in 3DSTATE_WM.

In the pixel shader kernel, the following computation is done for each attribute channel of each pixel/sample given the corresponding attribute channel a0/a1/a2 and the pixel/sample's b1/b2 barycentric parameters, where A is the value of the attribute channel at that pixel/sample:

A = a0 + (a1 * b1) + (a2 * b2)

Depth Offset

The state for depth offset in 3DSTATE_SF controls the depth offset function. Since this function was previously contained in the Windower stage, refer to the "Depth Offset" section in the Windower chapter for more details on this function.

Other SF Functions

The only other SF-related function is statistics gathering.

Statistics Gathering

The SF stage itself does not have any associated pipeline statistics; however, it counts the number of objects being output by the clipper on the clipper's behalf, since it is less feasible to have the CLIP unit figure out how many objects have been output by a clip thread. It is easy for the SF unit to count the number of objects it receives from the CLIP stage since it is decomposing the output primitive topologies into objects anyway.

If the **Statistics Enable** bit is set in SF_STATE, then SF will increment the CL_PRIMITIVES_COUNT Register (see Memory Interface Registers in Volume Ia, *GPU*) once for each object in each primitive topology it receives from the CLIP stage. This bit should always be set if clipping is enabled and pipeline statistics are desired.

Software should always clear the **Statistics Enable** bit in SF_STATE if the clipper is disabled since objects SF receives are not considered "primitives output by the clipper" unless the clipper is enabled. Note that the clipper can be disabled either using bypass mode via a PIPELINE_STATE_POINTERS command with **Clip Enable** clear *or* by setting **Clip Mode** in CLIP_STATE to CLIPMODE_ACCEPT_ALL.

Windower (WM) Stage

Overview

As mentioned in the *SF Unit* chapter, the SF stage prepares an object for scan conversion by the Window/Masker (WM) unit Refer to the *SF Unit* chapter for details on the screen-space geometry of objects to be rendered The WM unit uses the parameters provided by the SF unit in the object-specific rasterization algorithms.

The WM stage of the 3D pipeline performs the following operations (at a high level)

- Pre-scan-conversion modification of some primitive attributes, including
    - Application of Depth Offset to the position Z attribute
- Scan-conversion of the various primitive types, including
    - 2D clipping to the scissor/draw rectangle intersection
- Spawning of Pixel Shader (PS) threads to process the pixels resulting from scan-conversion

The spawned Pixel Shader (PS) threads are responsible for the following (high-level) operations

- interpolation of vertex attributes (other than X,Y,Z) to the pixel location
- performing any "Pixel Shader" operations dictated by the API PS program
    - Using the Sampler shared function to sample data from "texture" surfaces
    - Using the DataPort to perform general memory I/O
- Submitting the shaded pixel results to the DataPort for any subsequent "blending" (aka Output Merger) operation and write to the RenderCache.

The WM unit keeps a scoreboard of pixels being processed in outstanding PS threads in order to guarantee in-order rasterization results This allows the WM unit to overlap processing of several objects.

Inputs from SF to WM

The outputs from the SF stage to the WM stage are mostly comprised of implementation-specific information required for the rasterization of objects The types of information is summarized below, but as the interface is not exposed to software a detailed discussion is not relevant to this specification.

- PrimType of the object
- VPIndex, RTAIndex associated with the object
- Handle of the Primitive URB Entry (PUE) that was written by the SF (Setup) thread This handle will be passed to all WM (PS) threads spawned from the WM's rasterization process.
- Information regarding the X,Y extent of the object (e.g., bounding box, etc.)
- Edge or line interpolation information (e.g., edge equation coefficients, etc.)
- Information on where the WM is to start rasterization of the object
- Object orientation (front/back-facing)

- Last Pixel indication (for line drawing)

Windower Pipelined State

3DSTATE_WM

The following inline state packets define the state used by the windower stage for different generations.

State Packets
3DSTATE_WM

Programming Note	
Context:	Windower Pipelined State
Note: WM Unit also receives 3DSTATE_WM_HZ_OP, 3DSTATE_RASTER, 3DSTATE_MULTISAMPLE, 3DSTATE_WM_CHROMAKEY, 3DSTATE_PS_BLEND, and 3DSTATE_PS_EXTRA	

Signal	WM_INT:: Pixel Shader Computed Stencil
Description	This field specifies the computed stencil mode for the pixel shader.
Formula	= (WM_INT::WM_HZ_OP) ? 0 : 3DSTATE_PS_EXTRA::Computed Stencil

Signal	WM_INT::ThreadDispatchEnable
Description	This bit, if set, indicates that it is possible for a PS thread to modify a render target.
Formula	= (3DSTATE_WM::ForceThreadDispatch == ON) || ((3DSTATE_WM::ForceThreadDispatch != OFF) && ! WM_INT::WM_HZ_OP && 3DSTATE_PS_EXTRA::PixelShaderValid && ((!3DSTATE_PS_EXTRA::PixelShaderDoesNotWriteRT && 3DSTATE_PS_BLEND::HasWriteableRT) || (3DSTATE_PS_EXTRA::PixelShaderHasUAV) || WM_INT:: Pixel Shader Kill Pixel ||

	(WM_INT::Pixel Shader Computed Depth Mode != PSCDEPTH_OFF && (WM_INT::Depth Test Enable || WM_INT::Depth Write Enable)) || (3DSTATE_PS_EXTRA::Computed Stencil && WM_INT::Stencil Test Enable) || (3DSTATE_WM::EDSC_Mode == 1 && (WM_INT::Depth Test Enable || WM_INT::Depth Write Enable || WM_INT::Stencil Test Enable)) || (WM_INT::RT Independent Rasterization Enable)))

Signal	WM_INT::Pixel Shader Computed Depth Mode
Description	This field specifies the computed depth mode for the pixel shader.
Formula	= (3DSTATE_PS_EXTRA::ForceComputedDepth == Force) ? 3DSTATE_PS_EXTRA::Pixel Shader Computed Depth Mode : (WM_INT::WM_HZ_OP || WM_INT::RT Independent Rasterization Enable) ? PSCDEPTH_OFF: 3DSTATE_PS_EXTRA::Pixel Shader Computed Depth Mode

Signal	WM_INT::Pixel Shader Uses Source Depth
Description	This bit, if ENABLED, indicates that the PS kernel requires the source depth value (vPos.z) to be passed in the payload.
Formula	= 3DSTATE_PS_EXTRA::Pixel Shader Uses Source Depth

Signal	WM_INT::Pixel Shader Uses Source W
Description	This bit, if ENABLED, indicates that the PS kernel requires the interpolated source W value (vPos.w) to be passed in the payload
Formula	= 3DSTATE_PS_EXTRA::Pixel Shader Uses Source W

Signal	WM_INT::Pixel Shader Uses Input Coverage Mask
Description	This bit, if ENABLED, indicates that the PS kernel requires the input coverage mask to be passed in the payload.
Formula	= 3DSTATE_PS_EXTRA::Pixel Shader Uses Input Coverage Mask

Signal	WM_INT::Multisample Rasterization Mode
Description	This field determines whether multisample rasterization is enabled and how pixel sample points are defined.
Formula	See Table below: WM_INT::Multisample Rasterization Mode

WM_INT::Multisample Rasterization Mode

3DSTATE_RASTER:: Force Multisampling	Force	Force	Force	Force	Normal	Normal	Normal
3DSTATE_RASTER:: DX Multisample Rasterization Mode	MSRASTMODE_ OFF_PIXEL	MSRASTMODE_ OFF_PATTERN	MSRASTMODE_ ON_PIXEL	MSRASTMODE_ ON_PATTERN	*	*	*
WM_INT::WM_HZ_OP	*	*	*	*	True	True	False
3DSTATE_WM_HZ_OP:: Number of Multisamples	*	*	*	*	> NUMSAMPLES_1	NUMSAMPLES_1	*
WM_INT::Multisample Rasterization Mode	OFF_PIXEL	OFF_PATTERN	ON_PIXEL	ON_PATTERN	ON_PATTERN	ON_PIXEL	Determined from Table 1 in 3D Pipeline Windower > Multisampling > Multisample Modes/State)

Note: OFF_PIXEL, OFF_PATTERN, ON_PIXEL, ON_PATTERN modes are described in 3D Pipeline Windower > Multisampling > Multisample Modes/State.

Signal	WM_INT::Multisample Dispatch Mode
Description	This bit, determines how PS threads are dispatched
Formula	= (WM_INT::RT Independent Rasterization Enable)? PerPixel: (3DSTATE_PS_EXTRA::PixelShaderIsPerSample) ? PerSample : PerPixel

Signal	WM_INT::Pixel Shader Kill Pixel
Description	This bit, if ENABLED, indicates that the PS kernel or color calculator has the ability to kill (discard) pixels or samples, *other than due to depth or stencil testing*.
Formula	= (3DSTATE_WM::ForceKillPixel == ON) || ((3DSTATE_WM::ForceKillPixel != Off) && ! WM_INT::WM_HZ_OP && ! 3DSTATE_WM::EDSC_Mode == 2 && (WM_INT::Depth Write Enable || WM_INT::Stencil Write Enable) && (3DSTATE_PS_EXTRA::PixelShaderKillsPixels || 3DSTATE_PS_EXTRA:: **oMask Present to RenderTarget** || 3DSTATE_PS_BLEND::AlphaToCoverageEnable || 3DSTATE_PS_BLEND::AlphaTestEnable || 3DSTATE_WM_CHROMAKEY::ChromaKeyKillEnable))

Signal	WM_INT::Early Depth/Stencil Control
Description	This field specifies the behavior of early depth/stencil test.
Formula	= (WM_INT::WM_HZ_OP) ? EDSC_NORMAL :

	WM_INT::RT Independent Rasterization Enable ? EDSC_PSEXEC : 3DSTATE_WM::Early Depth/Stencil Control

Signal	WM_INT::RT Independent Rasterization Enable
Description	Enables Render Target Independent Rasterization.
Formula	= (WM_INT::WM_HZ_OP ? Disable : (3DSTATE_RASTER::ForceSampleCount != NUMRASTSAMPLES_0) ? Enable : Disable

Signal	WM_INT::Statistics Enable
Description	Enables Statistics
Formula	= (WM_INT::WM_HZ_OP) ? Disable : 3DSTATE_WM:: Statistics Enable

Signal	WM_INT::Polygon Stipple Enable
Description	Enables Poly Stipple
Formula	= (WM_INT::WM_HZ_OP) ? Disable : 3DSTATE_WM::Polygon Stipple Enable

Signal	WM_INT::WM_HZ_OP
Description	Enables WM_HZ_OP
Formula	= (3DSTATE_WM_HZ_OP::DepthBufferClear || 3DSTATE_WM_HZ_OP::DepthBufferResolve || 3DSTATE_WM_HZ_OP::Hierarchical Depth Buffer Resolve Enable ||

	3DSTATE_WM_HZ_OP::StencilBufferResolve || 3DSTATE_WM_HZ_OP::StencilBufferClear || 3DSTATE_WM_HZ_OP::DepthBufferPartialResolve) ? Enable : Disable

Signal	WM_INT:: Pixel Location
Description	Sets the input pixel location to Center if UL and doing multisampling
Formula	(3DSTATE_MULTISAMPLE::Pixel Location && 3DSTATE_MULTISAMPLE::Pixel Position Offset Enable && WM_MULTISAMPLE_INT::Number of Multisamples > 0) ? 0 : 3DSTATE_MULTISAMPLE::Pixel Location

3DSTATE_SAMPLE_MASK

The following inline state packets define the sample mask state used by the windower stage for different generations.

State Packets
3DSTATE_SAMPLE_MASK

Signal	WM_INT:: Sample Mask Enable
Description	Sets Sample Mask used in rasterization

Formula	Switch(WM_MULTISAMPLE_INT::Number of Multisamples
	{
	Case NUMSAMPLES_1: WM_INT:: Sample Mask Enable = 0x0001; break;
	Case NUMSAMPLES_2: WM_INT:: Sample Mask Enable = 0x0003; break;
	Case NUMSAMPLES_4: WM_INT:: Sample Mask Enable = 0x000F; break;
	Case NUMSAMPLES_8: WM_INT:: Sample Mask Enable = 0x00FF; break;
	Add this additional case:
	Case NUMSAMPLES_16: WM_INT:: Sample Mask Enable = 0xFFFF; break;
	}

Signal	WM_INT:: Sample Mask
Description	Sets Sample Mask used in rasterization
Formula	= WM_INT:: Sample Mask Enable & (WM_INT::WM_HZ_OP) ? 3DSTATE_WM_HZ_OP::Sample Mask: 3DSTATE_SAMPLE_MASK::Sample Mask)

3DSTATE_WM_CHROMAKEY

3DSTATE_WM_HZ_OP

State Restrictions

State	Restriction
3DSTATE_PS::Render Target Fast Clear Enable	Must be disabled
3DSTATE_PS:: Render Target Resolve Enable	Must be disabled
3DSTATE_WM:: Legacy Depth Buffer Clear	Must be disabled
3DSTATE_WM:: Legacy Depth Buffer Resolve	Must be disabled
3DSTATE_WM:: Legacy Hierarchical Depth Buffer Resolve Enable	Must be disabled
3DSTATE_MULTISAMPLE::Pixel Location	Must be set according to the API being used.

State Overrides

State	Stencil buffer Clear	Depth buffer clear	Depth Buffer Resolve Enable (full or partial)	Hierarchical Depth Buffer Resolve Enable
SF_INT:: Statistics Enable	Disable	Disable	Disable	Disable
SF_INT:: View Transform Enable	Disable	Disable	Disable	Disable
SF_INT::Multisample Rasterization Mode	(3DSTATE_WM_HZ_OP::NumberOfSamples > 1) ? ON_PATTERN : ON_PIXEL	(3DSTATE_WM_HZ_OP::NumberOfSamples > 1) ? ON_PATTERN : ON_PIXEL	(3DSTATE_WM_HZ_OP::NumberOfSamples > 1) ? ON_PATTERN : ON_PIXEL	(3DSTATE_WM_HZ_OP::NumberOfSamples > 1) ? ON_PATTERN : ON_PIXEL
SF_INT::Cull Mode	CULLMODE_BACK	CULLMODE_BACK	CULLMODE_BACK	CULLMODE_BACK
SF_INT::Scissor Rectangle Enable	3DSTATE_WM_HZ_OP:: Scissor Rectangle Enable	3DSTATE_WM_HZ_OP:: Scissor Rectangle Enable	3DSTATE_WM_HZ_OP:: Scissor Rectangle Enable	3DSTATE_WM_HZ_OP:: Scissor Rectangle Enable
SF_INT::RT Independent Rasterization Enable	Disable	Disable	Disable	Disable
SF_INT::FrontFace Fill Mode	SOLID	SOLID	SOLID	SOLID

State	Stencil buffer Clear	Depth buffer clear	Depth Buffer Resolve Enable (full or partial)	Hierarchical Depth Buffer Resolve Enable
SF_INT::FrontWinding	FRONTWINDING_CW	FRONTWINDING_CW	FRONTWINDING_CW	FRONTWINDING_CW
SF_INT::Render Target Array index	0	0	0	0
SF_INT::Viewport index	0	0	0	0
SF_INT:: Geometry Hashing Disable	Disable	Disable	Disable	Disable
WM_INT::StencilTest Enable	Enable	Stencil buffer Clear ? Enable : Disable	Disable	Disable
WM_INT::StencilWriteEnable	Enable	Stencil buffer Clear ? Enable : Disable	Disable	Disable
WM_INT::DepthTest Enable	Disable	Disable	Enable	Disable
WM_INT::DepthWriteEnable	Depth buffer Clear ? Enable : Disable	Enable	Enable	Enable
WM_INT::DepthTest Function	NEVER	NEVER	NEVER	NEVER
WM_INT::StencilTest Function	ALWAYS	Stencil buffer Clear ? ALWAYS: No Override	No Override	No Override
WM_INT::StencilPassDepthPassOp	REPLACE	Stencil buffer Clear ? REPLACE: No Override	No Override	No Override
WM_INT:: Statistics Enable	Disable	Disable	Disable	Disable
WM_INT::ThreadDispatchEnable	Disable	Disable	Disable	Disable
WM_INT:: Pixel Shader Kill Pixel	Disable	Disable	Disable	Disable
WM_INT:: Pixel Shader Computed Depth Mode	PSCDEPTH_OFF	PSCDEPTH_OFF	PSCDEPTH_OFF	PSCDEPTH_OFF
WM_INT: Computed	Disable	Disable	Disable	Disable

State	Stencil buffer Clear	Depth buffer clear	Depth Buffer Resolve Enable (full or partial)	Hierarchical Depth Buffer Resolve Enable
Stencil Enable				
WM_INT::RT Independent Rasterization Enable	Disable	Disable	Disable	Disable
WM_INT::Polygon Stipple Enable	Disable	Disable	Disable	Disable
WM_INT::Multisample Rasterization Mode	NumberOfSamples > 0 ? ON_PATTERN : ON_PIXEL)	NumberOfSamples > 0 ? ON_PATTERN : ON_PIXEL)	NumberOfSamples > 0 ? ON_PATTERN : ON_PIXEL)	NumberOfSamples > 0 ? ON_PATTERN : ON_PIXEL)
MULTISAMPLE_INT:: Number of Multisamples	3DSTATE_WM_HZ_OP:: Number of Multisamples	3DSTATE_WM_HZ_OP:: Number of Multisamples	3DSTATE_WM_HZ_OP:: Number of Multisamples	3DSTATE_WM_HZ_OP:: Number of Multisamples
WM_INT::Sample Mask	3DSTATE_WM_HZ_OP::Sample Mask	3DSTATE_WM_HZ_OP::Sample Mask	3DSTATE_WM_HZ_OP::Sample Mask	3DSTATE_WM_HZ_OP::Sample Mask
WM_INT::Early Depth/Stencil Control	EDSC_NORMAL	EDSC_NORMAL	EDSC_NORMAL	EDSC_NORMAL
WM_INT:: Full Surface Depth Clear	Depth buffer clear ? WM_HZ_OP:: Full Surface Depth Clear : Disable	WM_HZ_OP:: Full Surface Depth Clear	Disable	Disable
WM_INT:: Full Surface Depth Clear	Depth buffer clear ? WM_HZ_OP:: Full Surface Depth Clear : Disable	WM_HZ_OP:: Full Surface Depth Clear	Disable	Disable

3DSTATE_WM_DEPTH_STENCIL

Signal	WM_INT::DepthTestEnable
Description	Enables Depth Test
Formula	= (3DSTATE_DEPTH_BUFFER::SURFACE_TYPE != NULL) && (WM_INT::WM_HZ_OP ? Use the WM_INT::DepthTestEnable from WM_HZ_OP table : ((3DSTATE_WM_DEPTH_STENCIL::DepthTestEnable || WM_INT::DBT_ENABLE) && ! WM_INT::RT Independent Rasterization Enable))

Signal	WM_INT::DepthTestFunction
Description	Depth Test Function
Formula	WM_INT::WM_HZ_OP ? Use the WM_INT::DepthTestFunction from WM_HZ_OP table : (((!3DSTATE_WM_DEPTH_STENCIL::DepthTestEnable && WM_INT::DBT_enable) ? 3D_Compare_Function:: COMPAREFUNCTION_ALWAYS : 3DSTATE_WM_DEPTH_STENCIL::DepthTestFunction)

Signal	WM_INT::DBT_enable
Description	Depth Bounds Test enable
Formula	(WM_INT::WM_HZ_OP) ? 0 : 3DSTATE_DEPTH_BOUNDS::DBT_enable && (3DSTATE_DEPTH_BUFFER::SURFACE_TYPE != NULL)

Signal	WM_INT::StencilWriteEnable
Description	Enables writes to the Stencil Buffer
Formula	= 3DSTATE_STENCIL_BUFFER::SURFACE_TYPE != NULL && 3DSTATE_STENCIL_BUFFER::STENCIL_WRITE_ENABLE && ((WM_INT::WM_HZ_OP ? Use the WM_INT::StencilWriteEnable from WM_HZ_OP table : WM_INT::StencilTestEnable && 3DSTATE_WM_DEPTH_STENCIL::StencilBufferWriteEnable))

Signal	WM_INT::StencilWriteEnable
Description	Enables writes to the Stencil Buffer
Formula	= 3DSTATE_STENCIL_BUFFER::SURFACE_TYPE != NULL && 3DSTATE_STENCIL_BUFFER::STENCIL_WRITE_ENABLE && (

	(WM_INT::WM_HZ_OP ? Use the WM_INT::StencilWriteEnable from WM_HZ_OP table : WM_INT::StencilTestEnable && 3DSTATE_WM_DEPTH_STENCIL::StencilBufferWriteEnable))

Signal	WM_INT::DepthWriteEnable
Description	Enables Depth Write
Formula	= (3DSTATE_DEPTH_BUFFER::SURFACE_TYPE != NULL) && 3DSTATE_DEPTH_BUFFER::DEPTH_WRITE_ENABLE && (WM_INT::WM_HZ_OP ? Use the WM_INT::DepthWriteEnable from WM_HZ_OP table : 3DSTATE_WM_DEPTH_STENCIL::DepthWriteEnable)

Signal	WM_INT::StencilTestFunction
Description	Stencil Test Function
Formula	WM_INT::WM_HZ_OP ? Use the WM_INT::StencilTestFunction from WM_HZ_OP table : 3DSTATE_WM_DEPTH_STENCIL::StencilTestFunction

Signal	WM_INT::StencilPassDepthPassOp
Description	StencilPassDepthPassOp
Formula	WM_INT::WM_HZ_OP ? Use the WM_INT::StencilPassDepthPassOp from WM_HZ_OP table : 3DSTATE_WM_DEPTH_STENCIL::StencilPassDepthPassOp

Signal	WM_INT::Stencil Test Mask
Description	Stencil test Mask
Formula	= 3DSTATE_WM_HZ_OP::StencilClear ? 0xFF : 3DSTATE_WM_DEPTH_STENCIL::Stencil Test Mask

Signal	WM_INT::Stencil Write Mask
Description	Stencil Write Mask
Formula	= 3DSTATE_WM_HZ_OP::StencilClear ? 0xFF : 3DSTATE_WM_DEPTH_STENCIL::Stencil Write Mask

Signal	WM_INT::BackFace Stencil Test Mask
Description	Stencil test Mask
Formula	= 3DSTATE_WM_HZ_OP::StencilClear ? 0xFF : 3DSTATE_WM_DEPTH_STENCIL:: Backface Stencil Test Mask

Signal	WM_INT:: BackFace Stencil Write Mask
Description	Stencil Write Mask
Formula	= 3DSTATE_WM_HZ_OP::StencilClear ? 0xFF : 3DSTATE_WM_DEPTH_STENCIL::Backface Stencil Write Mask

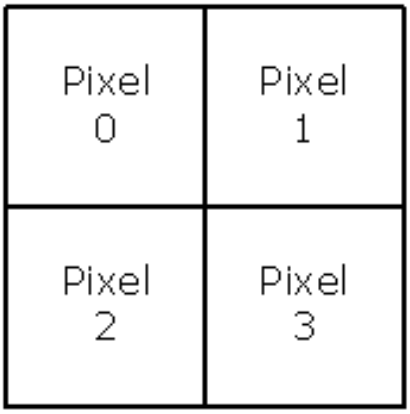
Rasterization

The WM unit uses the setup computations performed by the SF unit to rasterize objects into the corresponding set of pixels Most of the controls regarding the screen-space geometry of rendered objects are programmed via the SF unit.

The rasterization process generates pixels in 2x2 groups of pixels called *subspans* (see *Pixels with a SubSpan* below) which, after being subjected to various inclusion/discard tests, are grouped and passed to spawned Pixel Shader (PS) threads for subsequent processing Once these PS threads are spawned, the

WM unit provides only bookkeeping functions on the pixels Note that the WM unit can proceed on to rasterize subsequent objects while PS threads from previous objects are still executing.

Pixels with a SubSpan



B6850-01

Drawing Rectangle Clipping

The Drawing Rectangle defines the maximum extent of pixels which can be rendered Portions of objects falling outside the Drawing Rectangle will be clipped (pixels discarded) Implementations will typically discard objects falling completely outside of the Drawing Rectangle as early in the pipeline as possible There is no control to turn off Drawing Rectangle clipping - it is unconditional.

For the purposes of clipping, the Drawing Rectangle must itself be clipped to the destination buffer extents (The Drawing Rectangle Origin, used to offset relative X,Y coordinates earlier in the pipeline, is permitted to lie offscreen). The **Clipped Drawing Rectangle X,Y Min,Max** state variables (programmed via 3DSTATE_DRAWING_RECTANGLE - See *SF Unit*) defines the intersection of the Drawing Rectangle and the Color Buffer It is specified with non-negative integer pixel coordinates relative to the Destination Buffer upper-left origin.

Pixels with coordinates outside of the Drawing Rectangle cannot be rendered (i.e., the rectangle is inclusive) For example, to render to a full-screen 1280x1024 buffer, the following values would be required: Xmin=0, Ymin=0, Xmax=1279 and Ymax=1023

For "full screen" rendering, the Drawing Rectangle coincides with the screen-sized buffer For "front-buffer windowed" rendering it coincides with the destination "window".

Line Rasterization

See *SF Unit* chapter for details on the screen-space geometry of the various line types.

Coverage Values for Anti-Aliased Lines

The WM unit is provided with both the **Line Anti-Aliasing Region Width** and **Line End Cap Anti-aliasing Region Width** state variables (in WM_STATE) in order to compute the coverage values for anti-aliased lines.

3DSTATE_AA_LINE_PARAMS

3DSTATE_AA_LINE_PARAMETERS

The slope and bias values should be computed to closely match the reference rasterizer results Based on empirical data, the following recommendations are offered:

The final alpha for the center of the line needs to be 148 to match the reference rasterizer In this case, the Lo to edge 0 and edge 3 will be the same Since the alpha for each edge is multiplied together, we get:

edge0alpha * edge1alpha = 148/255 = 0.580392157

Since edge0alpha = edge3alpha we get:

$(edge0alpha)^2 = 0.580392157$

edge0alpha = sqrt(0.580392157) = 0.761834731 at the center pixel

The desired alpha for pixel 1 = 54/255 = 0.211764706

The slope is (0.761834731 - 0.211764706) = 0.550070025

Since we are using 8 bit precision, the slope becomes

AA Coverage [EndCap] Slope = 0.55078125

The alpha value for Lo = 0 (second pixel from center) determines the bias term and is equal to

(0.211764706 - 0.550070025) = -0.338305319

With 8 bits of precision the programmed bias value

Line Stipple

Line stipple, controlled via the **Line Stipple Enable** state variable in WM_STATE, discards certain pixels that are produced by non-AA line rasterization.

The line stipple rule is specified via the following state variables programmed via 3DSTATE_LINE_STIPPLE: the 16-bit **Line Stipple Pattern** (p), **Line Stipple Repeat Count** l, and **Line Stipple Inverse Repeat Count.** Software must compute **Line Stipple Inverse Repeat Count** as 1.0f / **Line Stipple Repeat Count** and then converted from float to the required fixed point encoding (see 3STATE_LINE_STIPPLE).

The WM unit maintains an internal Line Stipple Counter state variable (s) The initial value of s is zero; s is incremented after production of each pixel of a line segment (pixels are produced in order, beginning at the starting point and working towards the ending point). S is reset to 0 whenever a new primitive is processed (unless the primitive type is LINESTRIP_CONT or LINESTRIP_CONT_BF), and before every line segment in a group of independent segments (LINELIST primitive).

During the rasterization of lines, the WM unit computes:

$$b = \lfloor s/r \rfloor \bmod 16,$$

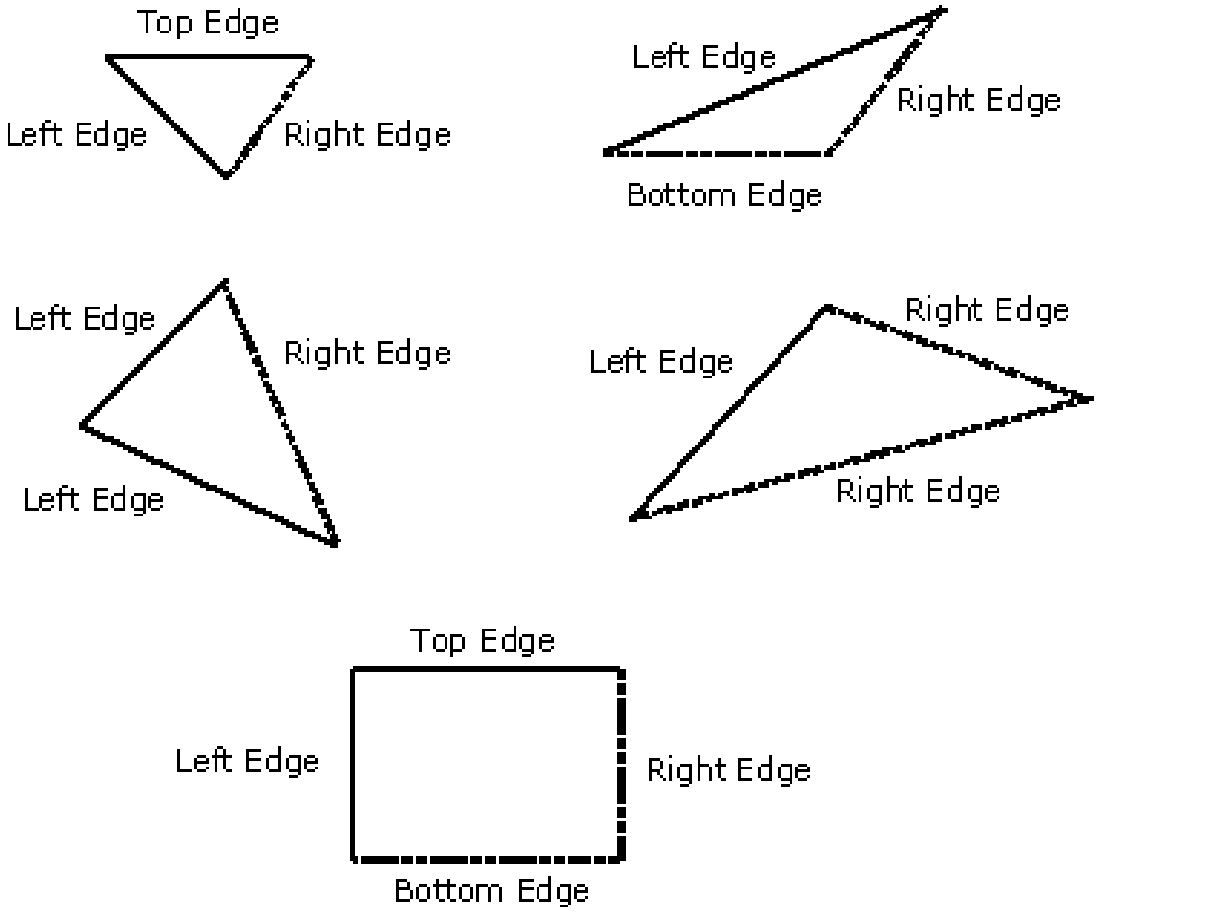
A pixel is rendered if the bth bit of p is 1, otherwise it is discarded. The bits of p are numbered with 0 being the least significant and 15 being the most significant.

Polygon (Triangle and Rectangle) Rasterization

The rasterization of LINE, TRIANGLE, and RECTANGLE objects into pixels requires a "pixel sampling grid" to be defined This grid is defined as an axis-aligned array of pixel sample points spaced exactly 1 pixel unit apart If a sample point falls within one of these objects, the pixel associated with the sample point is considered "inside" the object, and information for that pixel is generated and passed down the pipeline

For TRIANGLE and RECTANGLE objects, if a sample point intersects an edge of the object, the associated pixel is considered "inside" the object if the intersecting edge is a "left" or "top" edge (or, more exactly, the intersected edge is not a "right" or "bottom" edge) Note that "top" and "bottom" edges are by definition exactly horizontal. See *TRIANGLE and RECTANGLE Edge Types* below for the edge types for representative TRIANGLE and RECTANGLE objects (solid edges are inclusive, dashed edges are exclusive).

TRIANGLE and RECTANGLE Edge Types



Polygon Stipple

The *Polygon Stipple* function, controlled via the **Polygon Stipple Enable** state variable in WM_STATE, allows only selected pixels of a repeated 32x32 pixel pattern to be rendered Polygon stipple is applied only to the following primitive types:

3DPRIM_POLYGON
3DPRIM_TRIFAN
3DPRIM_TRILIST
3DPRIM_TRISTRIP
3DPRIM_TRISTRIP_REVERSE

Note that the 3DPRIM_TRIFAN_NOSTIPPLE object is never subject to polygon stipple.

The stipple pattern is defined as a 32x32 bit pixel mask via the 3DSTATE_POLY_STIPPLE_PATTERN command. This is a non-pipelined command which incurs an implicit pipeline flush when executed.

The origin of the pattern is specified via **Polygon Stipple X,Y Offset** state variables programmed via the 3DSTATE_POLY_STIPPLE_OFFSET command The offsets are pixel offsets from the Color Buffer origin to the upper left corner of the stipple pattern. This is a non-pipelined command which incurs an implicit pipeline flush when executed.

3DSTATE_POLY_STIPPLE_OFFSET

3DSTATE_POLY_STIPPLE_PATTERN

Multisampling

The multisampling function has two components:

- **Multisample Rasterization**: multisample rasterization occurs at a subpixel level, wherein each pixel consists of a number of "samples" at state-defined positions within the pixel footprint. Coverage of the primitive as well as color calculator operations (stencil test, depth test, color buffer blending, etc.) are done at the sample level. In addition the pixel shader itself can optionally run at the sample level depending on a separate state field.
- **Multisample Render Targets (MSRT)**: The render targets, as well as the depth and stencil buffers, now have the ability to store per-sample values. When combined with multisample rasterization, color calculator operations such as stencil test, depth test, and color buffer blending are done with the destination surface containing potentially different values per sample.

3DSTATE_MULTISAMPLE

Signal	WM_MULTISAMPLE_INT::Number of Multisamples
Description	Set the number of multisamples
Formula	= (WM_INT::WM_HZ_OP) ? 3DSTATE_ WM_HZ_OP::Number of Multisamples : 3DSTATE_MULTISAMPLE::Number of Multisamples

3DSTATE_SAMPLE_PATTERN

Multisample ModesState

A number of state variables control the operation of the multisampling function. The following table indicates the states and their location. Refer to the state definition for more details.

State Element	Source	Description
Multisample Rasterization Mode	WM_INT::Multisample Rasterization Mode	Controls whether rasterization of non-lines is performed on a pixel or sample basis (PIXEL vs. PATTERN), and whether multisample rasterization of lines is enabled (OFF vs. ON). From this generation forward, this state element becomes an internal signal computed by other state variables (also listed here) unless certain modes are set, which can be seen in the WM_INT equation for the signal.
Multisample Dispatch Mode	WM_INT::Multisample Dispatch Mode	Controls whether the pixel shader is executed per pixel or per sample.
Number of Multisamples	3DSTATE_MULTISAMPLE and SURFACE_STATE	Indicates the number of samples per pixel contained on the surface. This field in 3DSTATE_MULTISAMPLE must match the corresponding field in SURFACE_STATE for each render target. The depth, hierarchical depth, and stencil buffers inherit this field from 3DSTATE_MULTISAMPLE.
RTIR Enabled	3DSTATE_RASTER::ForcedSampleCount != NUMRASTSAMPLES_0	Enable Render Target Independent Rasterization.
Pixel Location	3DSTATE_MULTISAMPLE	Indicates the subpixel location where values specified as "pixel" are sampled. This is either the upper left corner or the center.
MSAA Sample Offsets	3DSTATE_SAMPLE_PATTERN	For each of the N samples, specifies the subpixel location of each sample.
RTIR Sample Offsets	3DSTATE_SAMPLE_PATTERN	For each of the N samples, specifies the subpixel location of each sample.
API Mode	3DSTATE_RASTER	One of the deciding factors of what the Multisample Rasterization Mode should be according to WM_INT::Multisample Rasterization Mode. Software sets this field according to the API's version.
DX Multisample Rasterization Enable	3DSTATE_RASTER	Controls ON/OFF part of Multisample Rasterization Mode, depending on the API Mode according to WM_INT::Multisample Rasterization Mode.

This table does not apply if (3DSTATE_RASTER::ForceMultisampleRasterMode == Force) or (WM_INT::WM_HZ_OP == true).

Table 1: Multisample Rasterization Modes

Number of Multisamples	NUMSAMPLES_1	NUMSAMPLES_1	> NUMSAMPLES_1	> NUMSAMPLES_1	Any	Any	Any	Any
DX Multisample Rasterization Enable	0	1	0	1	0	1	0	1
Rast Number of Samples	Disabled	Disabled	Disabled	Disabled	NUMRAST SAMPLES_1	NUMRAST SAMPLES_1	> NUMRAST SAMPLES_1	> NUMRAST SAMPLES_1
API Mode == DX9.0/OGL	OFF_PIXEL	OFF_PIXEL	OFF_PIXEL	ON_PATTERN	Invalid	Invalid	Invalid	Invalid
API Mode == DX10.0	OFF_PIXEL	ON_PIXEL	OFF_PIXEL	ON_PATTERN	OFF_PIXEL	Invalid	Invalid	Invalid
API Mode == DX10.1+/Vulkan	OFF_PIXEL	ON_PIXEL	OFF_PATTERN	ON_PATTERN	OFF_PIXEL	ON_PIXEL	OFF_PATTERN	ON_PATTERN

Definitions for lines terms used in Table 2 through Table 4:

- **Legacy Lines:** Way of drawing lines that allows Diamond Lines (SF_STATE::Line Width == 0.0), Non-anti-aliased Wide Lines (SF_STATE::Line Width != 0.0), and Line Stippling (3DSTATE_WM:: Line Stipple Enable == 1).
- **AA Lines:** Way of drawing lines that allows Anti-aliased line. These are lines rendered as rectangles that are centered on, and aligned to, the line joining the endpoint vertices with coverage value (referred to as Anti-alias Alpha) computed per pixel.

AA Line Support Requirement
SF_INT::Anti-aliasing Enable == 1

- **MSAA Lines:** Way of drawing lines that allows Multisample Anti-aliased lines. These are lines rendered as rectangles that are centered on, and aligned to, the line joining the endpoint vertices, but no Anti alias alpha coverage is computed.

- **Table 2: Type of Line Algorithm Given an Arrangement of State Variables**

Multisample Rasterization Mode	Anti-Aliasing Enable	SF_STATE::Line Width	Line Algorithm
OFF_*	0	Non-Zero	Non-Anti-aliased Wide Lines
OFF_*	0	0.0	Diamond Lines
OFF_*	1	Non-Zero	See Note A below.
OFF_*	1	0.0	Diamond Lines
ON_*	*	*	MSAA Lines

- **Note A: Anti-Aliasing Details for Table 2**

Anti-Aliasing Details
Anti-Aliased Lines with Alpha Coverage

- **Table 3: Multisample Modes with RTIR Disabled**

Number of Multisamples	MS RAST MODE	MS DISP MODE	HW Mode
NUMSAMPLES_1	OFF_PIXEL	PERSAMPLE	**Legacy Non-MSAA Mode** 1X rasterization, using Pixel Location Legacy lines or AA-line rasterization 1X PS, sample at Pixel Location 1X output merge, eval Depth at Pixel Location
	ON_PIXEL	PERSAMPLE	**1X Multisampling Mode** 1X rasterization, using Pixel Location MSAA lines only, using Pixel Location 1X PS, sample at Pixel Location 1X output merge, eval Depth at Pixel Location
	-	PERPIXEL	Treated the same as PERSAMPLE
	ON_PATTERN	-	Invalid
	OFF_PATTERN	-	Invalid
n where n > NUMSAMPLES_1	OFF_PIXEL	PERPIXEL	**MSRT Only, PerPixel PS** 1X rasterization, using Pixel Location See Note B below. 1X PS, sample at Pixel Location 4X output merge, eval Depth at Pixel Location
		PERSAMPLE	**MSRT Only, PerSample PS** 1X rasterization, using Pixel Location See Note B below. nX PS, all samples at Pixel Location nX output merge, eval Depth at Pixel Location
	ON_PIXEL	PERPIXEL	**Multibuffering MSAA, PerPixel PS** 1X rasterization, using Pixel Location MSAA lines only 1X PS, sample at Pixel Location

Number of Multisamples	MS RAST MODE	MS DISP MODE	HW Mode
			4X output merge, eval Depth at Pixel Location
		PERSAMPLE	**Multibuffering MSAA, PerSample PS** 1X rasterization, using Pixel Location MSAA lines only nX PS, all samples at Pixel Location nX output merge, eval Depth at Pixel Location
	OFF_PATTERN	PERPIXEL	**Mixed Mode, PerPixel PS** See Note B below. Non-Lines: nX rasterization, using Sample Offsets 1X PS, sample at Pixel Location nX output merge, eval depth at Sample Offsets
		PERSAMPLE	**Mixed Mode, PerSample PS** See Note B below. Non-Lines: nX rasterization, using Sample Offsets nX PS, sample at Pixel Location or Sample Offsets nX output merge, eval depth at Sample Offsets
	ON_PATTERN	PERPIXEL	**Pattern MSAA, PerPixel PS** nX rasterization, using Sample Offsets MSAA lines only 1X PS, sample at Pixel Location nX output merge, eval depth at Sample Offsets
		PERSAMPLE	**Pattern MSAA, PerSample PS** nX rasterization, using Sample Offsets MSAA lines only nX PS, sample at Pixel Location or Sample Offsets nX output merge, eval depth at Sample Offsets

Note B: Line Details for Table 3 and Table 4

Line Details
Legacy lines or AA-line rasterization. For PERPIXEL or PERSAMPLE in Table 3 use pixel location. For OFF_PATTERN in Table 4 use pixel location.

Table 4: Multisample Modes with RTIR Enabled

Rast Number of Samples	MS RAST MODE	HW Mode
NUMRASTSAMPLES_1	OFF_PIXEL	**Legacy Non-MSAA Mode** 1X rasterization, using Pixel Location Legacy lines or AA-line rasterization 1X PS, sample at Pixel Location 1X output merge, eval Depth at Pixel Location
	ON_PIXEL	**1X Multisampling Mode** 1X rasterization, using Pixel Location MSAA lines only, using Pixel Location 1X PS, sample at Pixel Location 1X output merge, eval Depth at Pixel Location
	ON_PATTERN	Invalid
	OFF_PATTERN	Invalid
n where n > NUMRASTSAMPLES_1	OFF_PIXEL	**Invalid**
	ON_PIXEL	**Invalid**
	OFF_PATTERN	**Mixed Mode, PerPixel PS** See Note B above. Non-Lines: nX rasterization, using Sample Offsets 1X PS, sample at Pixel Location 1X output merge, eval depth atPixel Location
	ON_PATTERN	**Pattern RTIR, PerPixel PS** nX rasterization, using Sample Offsets MSAA lines only 1X PS, sample at Pixel Location 1X output merge, eval Depth at Pixel Location

Note: Multisample Dispatch Mode is not taken into account in Table 4 given that with RTIR:

Details
The value of PERSAMPLE is converted to PERPIXEL internally.

Other WM Functions

The only other WM function is Statistics Gathering.

Statistics Gathering

If **Statistics Enable** is set in WM_STATE or 3DSTATE_WM, the Windower increments the PS_INVOCATIONS_COUNT register once for each unmasked pixel (or sample) that is *dispatched* to a Pixel Shader thread.

If **Early Depth Test Enable** is set it is possible for pixels or samples to be discarded before reaching the Pixel Shader due to failing the depth or stencil test. PS_INVOCATIONS_COUNT will still be incremented for these pixels or samples since the depth test occurs after the pixel shader from the point of view of SW.

Pixel

This section contains the following subsections:

- **Depth and Stencil**, which covers the Depth and Stencil test functions
- **Pixel Dispatch**, which covers pixel shader state, pixel grouping, multisampling effects on pixel shader dispatch, and pixel shader thread payload
- **Pixel Backend**, which covers backend processing

Texel Shading

Traditional rendering involving computed/procedural textures tend to render the entire texture in the pass before getting used. Often times, not all texels get used in the subsequent pass. Significant shading work can be reduced if texels are "demand shaded", i.e. shade the ones that are actually required to be sampled in the subsequent pass. By moving expensive shader computations from the pixel shader (screen space) to a texel shader (texture space), large performance savings are possible. This rendering method is referred to as "*texture-space shading*" or "*texture-space lighting*". This class of rendering technique is referred to as Adaptive Multi Frequency Shading (AMFS).

The following figure shows the 3D pipeline augmented with the new Texel Shader (TS) stage, which writes to a new Procedural Texture (PT) resource type. See definitions below. The geometry-related stages (tessellation, hull, domain, geometry shaders, and stream out) are not shown here for brevity.

Texel Shaders are similar to Pixel Shaders, with the following restrictions:

1) Texel Shaders are executed for 2x2 texel quads, similar as Pixel Shaders are executed for 2x2 pixel quads.

2) Texel Shader input argments are limited to TexelPosition (float and or integer), there is no attribute interpolation as in pixel shaders.

3) Texel Shader output argments are limited to single texel color value (float or unorm, depending on surface type).

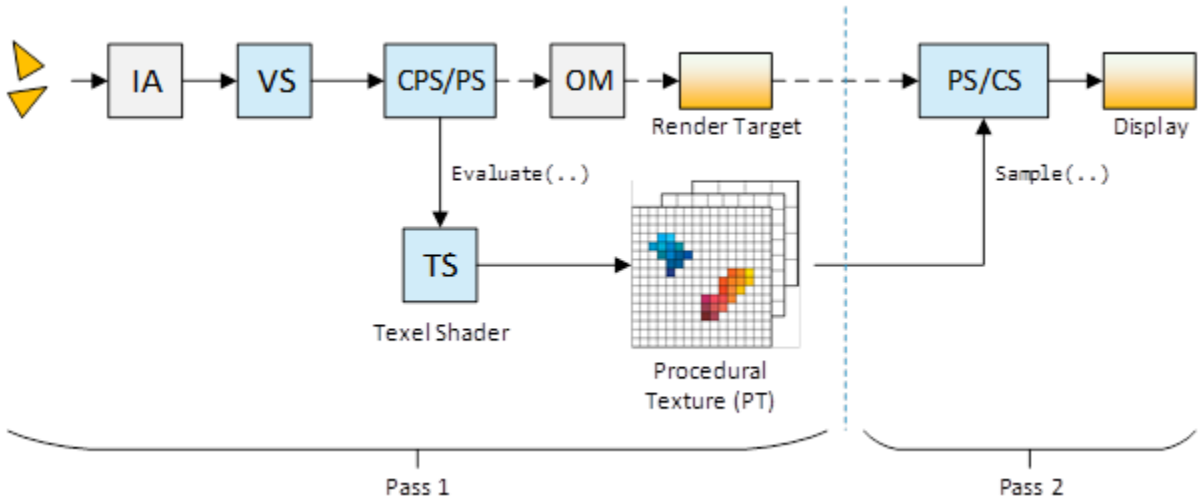
4) Texel Shader instructions are same as in Pixel Shader, except for:

a) no access to evaluate messages - Texel Shaders cannot recursively evaluate other procedural textures

b) no kill pixel, no access to Pixel Interpolator messages, no access to RT read/write messages

c) explicit derivatives are computed in texture-space UV, not in pixel-space XY.

Following diagram depicts the two pass approach to the Texel Shading:



Pixel Hashing

A block of pixel is hashed to slice and subslice based on screenspace X,Y. **Cross Slice Hashing Mode** and **Subslice Hashing Mode** indicate the size and shape of the pixel block used for the hash. Subslice hashing is calculated independently from slice hashing, but together determine how pixel workload is distributed to all the subslices.

3DSTATE_3D_MODE

X,Y shown below is the pixel block address.

PixelPipe Hashing

Wherever hashing to slice is mentioned, it refers to PixelPipe. For example, if a configuration has two PixelPipes and 2 DS for each PixelPipe, the mapping is described below for the purpose of the pixel distribution.

- PixelPipe0 = LogicalSlice0 = DSS0 & DSS1

PixelPipe1 = LogicalSlice1 = DSS2 & DSS3

SubSlice Hashing

2-way Hashing

subslice_id = X[0] ^ Y[0]

3-way Hashing

If X+Y is divisible by 3, then subslice_id=0, else subslice_id = 1 + X[2]^Y[0]

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	1	0	2	2	0	2	1	0	1	1	0	2	2	0
1	2	2	0	2	1	0	1	1	0	2	2	0	1	1	0	1
2	1	0	1	1	0	2	2	0	1	1	0	1	2	0	2	2
3	0	2	2	0	1	1	0	1	2	0	2	2	0	1	1	0
4	1	1	0	1	2	0	2	2	0	1	1	0	2	2	0	2
5	2	0	2	2	0	1	1	0	2	2	0	2	1	0	1	1
6	0	1	1	0	2	2	0	2	1	0	1	1	0	2	2	0
7	2	2	0	2	1	0	1	1	0	2	2	0	1	1	0	1
8	1	0	1	1	0	2	2	0	1	1	0	1	2	0	2	2
9	0	2	2	0	1	1	0	1	2	0	2	2	0	1	1	0
10	1	1	0	1	2	0	2	2	0	1	1	0	2	2	0	2
11	2	0	2	2	0	1	1	0	2	2	0	2	1	0	1	1
12	0	1	1	0	2	2	0	2	1	0	1	1	0	2	2	0
13	2	2	0	2	1	0	1	1	0	2	2	0	1	1	0	1
14	1	0	1	1	0	2	2	0	1	1	0	1	2	0	2	2
15	0	2	2	0	1	1	0	1	2	0	2	2	0	1	1	0

Programmable DualSubSlice Hashing

In addition to the above calculated dualsubslice_id, when enabled via **Subslice Hashing Table Enable**, dualsubslice_id is indicated by the entry in the dualsubslice_id hashing tables. X,Y address used to index the table is either 8x8 or 16x16 pixel block.

DUALSUBSLICE_HASH_TABLE_8x8

DUALSUBSLICE_HASH_TABLE_16x8

Slice Hashing

Table based Hashing

Slice_id hash is a lookup into a 256 entry slice_hash_table. The lowest 4 bits of the pixel block X,Y is used to index into the 16x16 table.

slice_id = slice_hash_table[Y[3:0]][X[3:0]]

Default ROM Table

When **Slice Hashing Table Enable** is set to false, slice_hash_table defaults to ROM tables based on the current number of active slices, and the following effective slice_id.

slice_id = (X[3:0] + Y[3:0]) % active_number_slices

Programmable Hashing

When **Slice Hashing Table Enable** is set to true, slice_hash_table is fetched via indirect state. An array of slice_hash_tables is stored at memory pointed to by **Slice Hash Table State Pointer**. First entry in the array is slices==2. No slice_hash_table is fetched when there is only one active slice.

3DSTATE_SLICE_TABLE_STATE_POINTERS

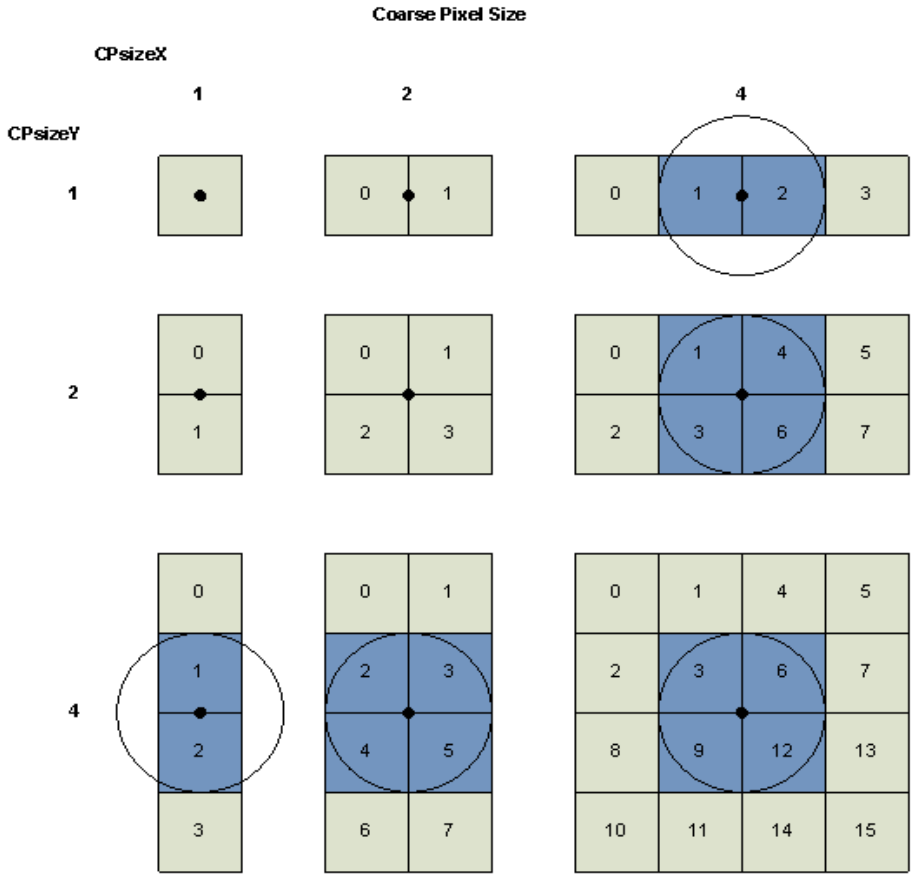
SLICE_HASH_TABLE - SLICE_HASH_TABLE

Coarse Pixel Shading

In Prior products, Pixel Shader can be invoked at either pixel frequency or at sample frequency. In general, finer grain shading creates more BW and compute demands in the graphics sub-system. In certain use cases, it is possible to do coarser grain shading than pixel without noticeable change to the image quality. This observation leads to HW support for coarser than pixel grain shading rate. It is called Coarse Pixel Shading (CPS).

Coarse Pixels

Similar to how pixels consist of multiple samples under MSAA, a coarse pixel consists of several pixels. Coarse pixel size is defined in terms of pixels by an ordered pair, e.g. (2,4) means CP has 2 pixels in X-axis and 4 pixels in Y-axis. Allowable CP sizes are from the set {1,2,4} X {1,2,4}. When CPS is enabled, HW computes CPsizes and gathers visible pixels to form CPs.



CPS Modes

Coarse Pixel Shading requires per primitive CP sizes to be determined. CPsize is fixed for a block of 8X8 aligned pixel block. There are two modes for determining the CPsize:

Constant: Entire RTV has the flat CPsizes as defined by the 3DSTATE_CPS. This mode can be used to under-shade uniformly.

Radial: Since camera focal plane tends to carry more detailed information in the rendered image, this mode provides increasing CPsizes as distance from the focal point (defined in 3DSTATE_PS) increases. Therefore, shading rate decreases as pixels fall farther from the focal point. In this mode, CPsize is a function of the fragment's position.XY. Detailed computation with respect to parameters from 3DSTATE_CPS is described below:

First define the following variables:

- Let (Px, Py) be the (x,y) pixel position of the pixel for which the *requested coarse pixel size* is being computed, in pixel coordinates
- Let (Cx, Cy) be the center position, in pixel coordinates, specified as CoarsePixelSizeState.CenterX and CoarsePixelSizeState.CenterY. These values must be in [0.0f, MAX_RENDER_TARGET_LINEAR_RESOLUTION].
- Let (S_x^{min}, S_y^{min}) be the maximum coarse pixel size specified as (CoarsePixelSizeState.MinSizeX, CoarsePixelSizeState.MinSizeY). These values must be in [1.0f, 4.0f] (the minimum and maximum allowed coarse pixel sizes).
- Let (S_x^{max}, S_y^{max}) be the minimum coarse pixel size specified as (CoarsePixelSizeState.MaxSizeX, CoarsePixelSizeState.MaxSizeY). These values must be in [1.0f, 4.0f] (the minimum and maximum allowed coarse pixel sizes).
- Let R_{min} be the minimum radius specified as CoarsePixelSizeState.RadiusMinSize. These values must be in [0.0f, MAX_RENDER_TARGET_LINEAR_RESOLUTION * sqrt(2.0f)] (the minimum and maximum radii in pixel coordinates).
- Let R_{max} be the maximum radius specified as CoarsePixelSizeState.RadiusMaxSize. These values must be in [0.0f, MAX_RENDER_TARGET_LINEAR_RESOLUTION * sqrt(2.0ff)] (the minimum and maximum radii in pixel coordinates).
- Let (M_x, M_y) be $\left(\frac{S_x^{max} - S_x^{min}}{R_{max} - R_{min}}, \frac{S_y^{max} - S_y^{min}}{R_{max} - R_{min}}\right)$
- Let A_{ratio} be the radial function aspect ratio defined as the length in pixels of the ellipse X axis over the Y axis.

  Then determine SV_CoarsePixelSize as follows:

  DeltaX = (C_x - P_x)

  DeltaY = (C_y - P_y)

  *If (A_{ratio} <= 1) DeltaY = A_{ratio} * DeltaY*

*Else DeltaX = (1/A_{ratio})*DeltaX*

D= Distance(DeltaX,DeltaY)

$CoarsePixelSize_x$= M_x (D - R_{min}) +* S_x^{min}

$CoarsePixelSize_y$= M_y(D- R_{min})+* S_y^{min}

$CoarsePixelSize_x$= clamp($CoarsePixelSize_x$, S_x^{min} , S_x^{max} *)*

$CoarsePixelSize_y$= clamp($CoarsePixelSize_y$, S_y^{min} , S_y^{max} *)*

RequestedCoarsePixelSize = ($CoarsePixelSize_x$, Coars, e$PixelSize_y$)

Where *Distance()* is the Pythagorean distance function between two points defined by the following function:

$$Distance(DeltaX, DeltaY) = \sqrt{DeltaX^2 + DeltaY^2}$$

$$Distance(DeltaX, DeltaY) = \sqrt{DeltaX^2 + DeltaY^2}$$

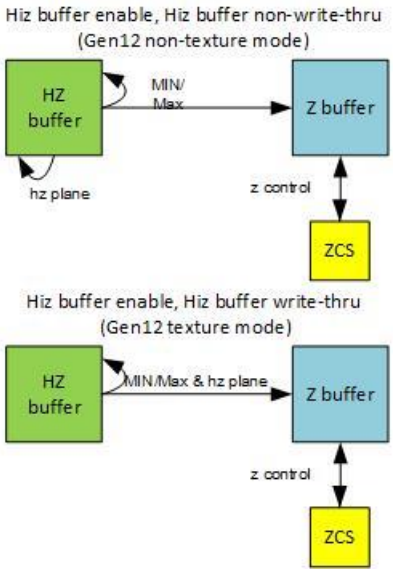
However, because this function may be expensive to compute exactly, the distance function may be approximated by HW.

Early Depth/Stencil Processing

The Windower/IZ unit provides the Early Depth Test function, a major performance-optimization feature where an attempt is made to remove pixels that fail the Depth and Stencil Tests prior to pixel shading. This requires the WM unit to perform the interpolation of pixel ("source") depth values, read the current ("destination") depth values from the cached depth buffer, and perform the Depth and Stencil Tests As the WM unit has per-pixel source and destination Z values, these values are passed in the PS thread payload, if required.

Compressed Depth Buffer

The depth buffer can be compressed. 2 modes of operation are possible by using the 3DSTATE_DEPTH_BUFFER::Depth Buffer Compression Enable & the 3DSTATE_HIER_DEPTH_BUFFER:: Hierarchical Depth Buffer Write Thru Disable.



The operations of these 2 modes are given in the tables below.

	Non-texture performant Hiz Write thru == disabled Depth buffer compression == enabled			Texture performant Hiz Write thru == enabled Depth buffer compression == enabled		
	HiZ	ZCS	Z	HiZ	ZCS	Z
Min/Max cases	Update	Update on cache evict	Update Compress on cache evict	Update	Update on depth cache evict	Update Compress on cache evict
Plane cases	Update	ZCS buffer unchanged	Z buffer unchanged	Update	Update on depth cache evict	Update Compress on cache evict
Clears	Update	ZCS buffer unchanged	Z buffer unchanged	Update	Update with clear at either 16x8 or 8x4 granularity, based on fs_clr or otherwise	Update if entire cache line is not cleared Compress on cache evict

When the 3DSTATE_DEPTH_BUFFER::Depth Buffer Compression Enable is set, a compression control buffer must be allocated and the 3DSTATE_DEPTH_CNTL_BUFFER command is used to specify the buffer.

Depth Offset

Note: The depth offset function is contained in SF unit, thus the state to control it is also contained in SF unit.

There are occasions where the Z position of some objects need to be slightly offset to reduce artifacts due to coplanar or near-coplanar primitives. A typical example is drawing the edges of triangles as wireframes - the lines need to be drawn slightly closer to the viewer to ensure they will not be occluded by the underlying polygon. Another example is drawing objects on a wall - without a bias on the z positions, they might be fully or partially occluded by the wall.

The device supports *global* depth offset, applied only to triangles, that bases the offset on the object's z slope Note that there is no clamping applied at this stage after the Z position is offset - clamping to [0,1] can be performed later after the Z position is interpolated to the pixel. This is preferable to clamping prior to interpolation, as the clamping would change the Z slope of the entire object.

The Global Depth Offset function is controlled by the **Global Depth Offset Enable** state variable in WM_STATE Global Depth Offset is only applied to 3DOBJ_TRIANGLE objects.

When Global Depth Offset Enable is ENABLED, the pipeline will compute:

```
    MaxDepthSlope = max(abs(dZ/dX),abs(dz/dy)) // approximation of max depth slope for
polygon
```

When UNORM Depth Buffer is at Output Merger (or no Depth Buffer):

```
    Bias = GlobalDepthOffsetConstant * r + GlobalDepthOffsetScale * MaxDepthSlope
```

Where r is the minimum representable value > 0 in the depth buffer format, converted to float32 (note: If state bit **Legacy Global Depth Bias Enable** is set, the r term will be forced to 1.0)

When Floating Point Depth Buffer at Output Merger:

```
    Bias = GlobalDepthOffsetConstant * 2^(exponent(max z in primitive) - r) +
GlobalDepthOffsetScale * MaxDepthSlope
```

Where r is the # of mantissa bits in the floating point representation (excluding the hidden bit), e.g. 23 for float32 (note: If state bit Legacy Global Depth Bias Enable is set, no scaling is applied to the GobalDepthOffsetConstant).

Adding Bias to z:

```
    if (GlobalDepthOffsetClamp > 0)
        Bias = min(DepthBiasClamp, Bias)
    else if(GlobalDepthOffsetClamp < 0)
        Bias = max(DepthBiasClamp, Bias)
    // else if GlobalDepthOffsetClamp == 0, no clamping occurs
    z = z + Bias
```

Biasing is constant for a given primitive. The biasing formulas are performed with float32 arithmetic Global Depth Bias is not applied to any point or line primitives.

Early Depth Test/Stencil Test/Write

When **Early Depth Test Enable** is ENABLED, the WM unit will attempt to discard depth-occluded pixels during scan conversion (before processing them in the Pixel Shader). Pixels are only discarded when the

WM unit can ensure that they would have no impact to the ColorBuffer or DepthBuffer. This function is therefore only a performance feature.

Note: The **Early Depth Test Enable** bit is no longer present. This function is always enabled.

If some pixels within a subspan are discarded, only the pixel mask is affected indicating that the discarded pixels are not active. If all pixels within a subspan are discarded, that subspan will not even be dispatched.

Software-Provided PS Kernel Info

For the WM unit to properly perform Early Depth Test and supply the proper information in the PS thread payload (and even determine if a PS thread needs to be dispatched), it requires information regarding the PS kernel operation This information is provided by a number of state bits in WM_STATE, as summarized in the following table.

State Bit	Description
Pixel Shader Kill Pixel	This must be set when there is a chance that valid pixels passed to a PS thread may be discarded. This includes the discard of pixels by the PS thread resulting from a "killpixel" or "alphatest" function or as dictated by the results of the sampling of a "chroma-keyed" texture The WM unit needs this information to prevent early depth/stencil writes for pixels which might be killed by the PS thread, etc. See WM_STATE/3DSTATE_WM for more information.
Pixel Shader Computed Depth	This must be set when the PS thread computes the "source" depth value (i.e., from the API POV, writes to the "oDepth" output) In this case the WM unit can't make any decisions based on the WM-interpolated depth value. See WM_STATE/3DSTATE_WM for more information.
Pixel Shader Uses Source Depth	Must be set if the PS thread requires the WM-interpolated source depth value. This forces the source depth to be passed in the thread payload where otherwise the WM unit would not have seen it as required. See WM_STATE/3DSTATE_WM for more information.

Hierarchical Depth Buffer

A hierarchical depth buffer is supported to reduce memory traffic due to depth buffer accesses. This buffer is supported only in Tile Y memory.

The **Surface Type**, **Height**, **Width**, **Depth**, **Minimum Array Element**, **Render Target View Extent**, and **Depth Coordinate Offset X/Y** of the hierarchical depth buffer are inherited from the depth buffer. The height and width of the hierarchical depth buffer that must be allocated are computed by the following formulas, where HZ is the hierarchical depth buffer and Z is the depth buffer. The Z_Height, Z_Width, and Z_Depth values given in these formulas are those present in 3DSTATE_DEPTH_BUFFER incremented by one.

The Z_Height and Z_Width values must equal those present in 3DSTATE_DEPTH_BUFFER incremented by one.

Surface Type	HZ_Width (Bytes)	HZ_Height (Rows)	HZ_Qpitch (Rows)
SURFTYPE_1D			not applicable
SURFTYPE_2D	ceiling(Z_Width / 16) * 16	ceiling((HZ_QPitch/2)/16) *16 * Z_Depth	see below
SURFTYPE_3D			not applicable
SURFTYPE_CUBE	ceiling(Z_Width / 16) * 16	ceiling((HZ_QPitch/2)/16) *16 * 6 * Z_Depth	see below

To compute the minimum QPitch for the HZ surface, the height of each LOD in pixels is determined using the equations for hL in the GPU Overview volume, using a vertical alignment j=16. The following equation gives the minimum HZ_QPitch based on largest LOD m defined in the surface:

$$HZ\_QPitch = h_0 + max\left(h_1, \sum_{i=2}^{m} \mathrm{h_i}\right)$$

If m is less than 2, treat all h_L with L > m as zero and use the above equation.

The format of the data in the hierarchical depth buffer is not documented here, as this surface needs only to be allocated by software. Hardware will read and write this surface during operation and its contents are discarded once the last primitive is rendered that uses the hierarchical depth buffer.

The hierarchical depth buffer can be enabled whenever a depth buffer is defined, with its effect being invisible other than generally higher performance. The only exception is the hierarchical depth buffer must be disabled when using software tiled rendering.

If HiZ is enabled, you must initialize the clear value by either:

1. Perform a depth clear pass to initialize the clear value.
2. Send a 3dstate_clear_params packet with valid = 1.

Without one of these events, context switching will fail, as it will try to save off a clear value even though no valid clear value has been set. When context restore happens, HW will restore an uninitialized clear value.

Depth Buffer Clear

With the hierarchical depth buffer enabled, performance is generally improved by using the special clear mechanism described here to clear the hierarchical depth buffer and the depth buffer. This is enabled though the **Depth Buffer Clear** field in WM_STATE or 3DSTATE_WM or using the 3DSTATE_WM_HZ_OP. This bit can be used to clear the depth buffer in the following situations:

- Complete depth buffer clear.
- Partial depth buffer clear with the clear value the same as the one used on the previous clear.
- Partial depth buffer clear with the clear value different than the one used on the previous clear can use this mechanism if a depth buffer resolve is performed first.

The following is required when performing a depth buffer clear using any of the above clearing methods (WM_STATE, 3DSTATE_WM or 3DSTATE_WM_HZ_OP).

- The hierarchical depth buffer enable must be set in the 3DSTATE_DEPTH_BUFFER.
- The fields in 3DSTATE_CLEAR_PARAMS are set to indicate the source of the clear value and (if source is in this command) the clear value itself.
- The clear value must be between the min and max depth values (inclusive) defined in the CC_VIEWPORT. If the depth buffer.
    - The following alignment restrictions need to be met while doing the fast-clear:

Alignment Restriction
The minimum granularity of clear is one pixel, but all samples of the pixel must be cleared. Clearing partial samples of a pixel is not supported. If a newly allocated depth buffer is not padded to an integer multiple of 8x4 pixels, and if the first operation on the depth buffer does not clear the entire width and height of the surface, then first a HiZ ambiguate must be done on the portions of the depth buffer that are not cleared. If the depth buffer clear operation does clear the entire width and height of the surface, then the "full surface clear" bit in 3DSTATE_WM_OP must be set to 1.
The minimum granularity of clear is one pixel, but all samples of the pixel must be cleared. Clearing partial samples of a pixel is not supported. If the first operation of a newly allocated depth buffer does not clear the entire width and height of the surface, then first a HiZ ambiguate must be done on the portions of the depth buffer that are not cleared. If the depth buffer clear operation does clear the entire width and height of the surface, then the "full surface clear" bit in 3DSTATE_WM_OP must be set to 1.

The following is required when performing a depth buffer clear with using the WM_STATE or 3DSTATE_WM:

- If other rendering operations have preceded this clear, a PIPE_CONTROL with depth cache flush enabled, Depth Stall bit enabled must be issued before the rectangle primitive used for the depth buffer clear operation.
- **Depth Test Enable** must be disabled and **Depth Buffer Write Enable** must be enabled (if depth is being cleared).
- Stencil buffer clear can be performed at the same time by enabling Stencil Buffer Write Enable. Stencil Test Enable must be enabled and Stencil Pass Depth Pass Op set to REPLACE, and the clear value that is placed in the stencil buffer is the **Stencil Reference Value** from COLOR_CALC_STATE.
- Note also that stencil buffer clear can be performed without depth buffer clear. For stencil only clear, **Depth Test Enable** and **Depth Buffer Write Enable** must be disabled.

In some cases **Depth Buffer Clear** cannot be enabled and the legacy method of clearing must be used:

- If the depth buffer format is D32_FLOAT_S8X24_UINT or D24_UNORM_S8_UINT.
- If stencil test is enabled but the separate stencil buffer is disabled.

Depth buffer clear pass using any of the methods (WM_STATE, 3DSTATE_WM or 3DSTATE_WM_HZ_OP) must be followed by a PIPE_CONTROL command with DEPTH_STALL bit and Depth FLUSH bits "**set**" before starting to render.

Note: If using the optimized depth buffer clear, this pipecontrol should be done after the resetting of the clear/resolve bits in the 3DSTATE_WM_HZ_OP (step #8).

- Since the fast clear cycles to CCS are not cached in TileCache, any previous depth buffer writes to overlapping pixels must be flushed out of TileCache before a succeeding Depth Buffer Clear. This restriction only applies to Depth Buffer with write-thru enabled, since fast clears to CCS only occur for write-thru mode.

Depth Buffer Resolve

If the hierarchical depth buffer is enabled, the depth buffer may contain incorrect results after rendering is complete If the depth buffer is retained and used for another purpose (i.e as input to the sampling engine as a shadow map), it must first be "resolved" This is done by setting the **Depth Buffer Resolve Enable** field in WM_STATE or 3DSTATE_WM and rendering a full render target sized rectangle. Once this is complete, the depth buffer will contain the same contents as it would have had the rendering been performed with the hierarchical depth buffer disabled. In a typical usage model, depth buffer needs to be resolved after rendering on it and before using a depth buffer as a source for any consecutive operation. Depth buffer can be used as a source in three different cases: using it as a texture for the nest rendering sequence, honoring a lock on the depth buffer to the host OR using the depth buffer as a blit source.

The following is required when performing a depth buffer resolve:

- The surface must have been initialized with a Depth Buffer Clear after its allocation to initialize the Depth Clear Value.
- A rectangle primitive of the same size as the previous depth buffer clear operation must be delivered, and depth buffer state cannot have changed since the previous depth buffer clear operation.
- **Depth Test Enable** must be enabled with the **Depth Test Function** set to NEVER. **Depth Buffer Write Enable** must be enabled. **Stencil Test Enable** and **Stencil Buffer Write Enable** must be disabled.
- **Pixel Shader Dispatch, Alpha Test, Pixel Shader Kill Pixel** and **Pixel Shader Computed Depth** must all be disabled.

Programming Note	
Context:	HTML
HW uses the clear value from the 3DSTATE_CLEAR_PARAM. If you change the value in the 3DSTATE_CLEAR_PARAMS before resolve, it will flush the depth caches and have the new-clear value in its register. When doing the resolve pass, it is driver's responsibility to make sure that the clear-value for the depth buffer is the same one as the clear-pass.	

Hierarchical Depth Buffer Resolve

If the hierarchical depth buffer is enabled, the hierarchical depth buffer may contain incorrect results if the depth buffer is written to outside of the 3D rendering operation If this occurs, the hierarchical depth buffer must be *resolved* to avoid incorrect device behavior. This is done by setting the Hierarchical Depth Buffer Resolve Enable field in WM_STATE or 3DSTATE_WM and rendering a full render target sized

rectangle. Once this is complete, the hierarchical depth buffer will contain contents such that rendering will give the same results as it would have had the rendering been performed with the hierarchical depth buffer disabled.

The following is required when performing a hierarchical depth buffer resolve:

- A rectangle primitive covering the full render target must be delivered.
- **Depth Test Enable** must be disabled. **Depth Buffer Write Enable** must be enabled. **Stencil Test Enable** and **Stencil Buffer Write Enable** must be disabled.
- **Pixel Shader Dispatch, Alpha Test, Pixel Shader Kill Pixel**, and **Pixel Shader Computed Depth** must all be disabled.

Optimized Depth Buffer Clear and/or Stencil Buffer Clear

With the hierarchical depth buffer enabled, performance is generally improved by using the special clear mechanism described here to clear the hierarchical depth buffer and the depth buffer This is enabled though the **Depth Buffer Clear** field in 3DSTATE_WM_HZ_OP This bit can be used to clear the depth buffer in the following situations:

- All 3D units before SF will be bypassed by WM_HZ_OP and states for those units need not be set/restored for these rectangles.
- Complete depth buffer clear
- Partial depth buffer clear with the clear value the same as the one used on the previous clear
- Partial depth buffer clear with the clear value different than the one used on the previous clear can use this mechanism if a depth buffer resolve is performed first.
- The minimum granularity of clear is one pixel, but all samples of the pixel must be cleared. Clearing partial samples of a pixel is not supported

Stencil Buffer Clears can be alone or at the same time as depth buffer clears by using the Stencil Buffer Clear bit in 3DSTATE_WM_HZ_OP.

Note for SURFACE_TYPE CUBE: To clear / resolve a CUBE_SURFACE using WM_HZ_OP, the surface_type must be changed to 2D and the depth is calculated for that.

As there are 6 faces of the cube, the depth is multiplied by 6 to get the number of slices in the cube. The min_array_index is one of the slices.

Hence, in order to clear / resolve, go through each slice & multiply depth by 6 and then using the min-array-index, point to the respective slice for clear/resolve.

The proper sequence of commands is as follows:

1. Setup 3DSTATE_DEPTH_BUFFER (as needed). Render Target Array index will be internally force to zero. SW must set 3DSTATE_DEPTH_BUFFER::**MinimumArrayElement** to render to the array to be cleared.
2. Setup 3DSTATE_HIER_DEPTH_BUFFER (as needed)
3. Setup 3DSTATE_STENCIL_BUFFER (as needed)

4. Setup 3DSTATE_CLEAR_PARMS (as needed).
5. Setup 3DSTATE_DRAWING_RECTANGLE (as needed and only if it is different from already existing drawing rectangle)
6. 3DSTATE_WM_HZ_OP w/ 1 of the clear/resolve bits set

   // This overrides existing state and forces them to what is needed for the clear

   // This also carries the vertex info for doing the clear
7. PIPE_CONTROL w/ all bits clear except for "Post-Sync Operation" must set to "Write Immediate Data" enabled.

   // This causes 3DSTATE_WM_HZ_OP state to be committed to SF and WM as a pipeline state. Once state is committed to SF, causes to spawn a rectangle to be drawn
8. 3DSTATE_WM_HZ_OP w/ none of the clear/resolve bits set

   // This clears the overrides
9. Restore 3DSTATE_DEPTH_BUFFER (as needed).
10. Restore 3DSTATE_HIER_DEPTH_BUFFER (as needed)
11. Restore 3DSTATE_STENCIL_BUFFER (as needed)

Arbitrary size rectangles are supported using the Top Left X, Top Left Y, Bottom Right X, Bottom Right Y fields in the 3DSTATE_WM_HZ_OP.

Optimized Depth Buffer Resolve

If the hierarchical depth buffer is enabled, the depth buffer may contain incorrect results after rendering is complete If the depth buffer is retained and used for another purpose (i.e locked by the app), it must first be "resolved" This is done by setting the **Depth Buffer Resolve Enable** field in 3DSTATE_WM_HZ_OP. The depth buffer resolve uses the same sequence as the optimized Depth buffer clear (see above) except the **Depth Buffer Resolve Enable** bit is set. Once this is complete, the depth buffer will contain the same contents as it would have had the rendering been performed with the hierarchical depth buffer disabled. In a typical usage model, depth buffer needs to be resolved after rendering on it and before using a depth buffer as a source for any consecutive operation. Depth buffer can be used as a source in three different cases: using it as a texture for the nest rendering sequence, honoring a lock on the depth buffer to the host OR using the depth buffer as a blit source.

Doing a resolve operation requires that a preceding Depth Buffer Clear operation is required to have initialized the Depth Clear Value.

Optimized Hierarchical Depth Buffer Resolve

If the hierarchical depth buffer is enabled, the hierarchical depth buffer may contain incorrect results if the depth buffer is written to outside of the 3D rendering operation. If this occurs, the hierarchical depth buffer must be "resolved" to avoid incorrect device behavior. This is done by setting the **Hierarchical Depth Buffer Resolve Enable** field in 3DSTATE_WM_HZ_OP and specifying a full render target sized rectangle. The depth buffer resolve uses the same sequence as the optimized Depth buffer clear (see above) except the **Hierarchical Depth Buffer Resolve Enable** bit is set. Once this is complete, the

hierarchical depth buffer will contain contents such that rendering will give the same results as it would have had the rendering been performed with the hierarchical depth buffer disabled.

The following is required when performing a hierarchical depth buffer resolve:

- A rectangle primitive covering the full render target must be programmed on Xmin, Ymin, Xmax, and Ymax in the 3DSTATE_WM_HZ_OP command.
- The rectangle primitive size must be aligned to 8x4 pixels.

Separate Stencil Buffer

The following tables describe the separate stencil buffer for different generations.

The stencil buffer has a format of R8_UNIT,. The **Surface Type**, **Height**, **Width**, and **Depth**, **Minimum Array Element**, **Render Target View Extent**, **Depth Coordinate Offset X/Y**, **LOD**, and **Stencil Buffer Object Control State** are defined in the 3DSTATE_STENCIL_BUFFER

DepthStencil Buffer State

This section contains the state registers for the Depth/Stencil Buffers.

Register
3DSTATE_DEPTH_BUFFER
3DSTATE_STENCIL_BUFFER
3DSTATE_HIER_DEPTH_BUFFER
3DSTATE_CLEAR_PARAMS

Pixel Shader Thread Generation

After a group of object fragments have been rasterized, the Pixel Shader (PSD) function is invoked to further compute output information and cause results to be written to output surfaces (like color, depth, stencil, UAvs etc). Fragments can be P or S.

Fragments can also be CP.

For each fragment, the Pixel Shader calculates the values of the various vertex attributes that are to be interpolated across the object using the interpolation coefficients. It then executes an API-supplied Pixel Shader Program. Instructions in this program permit the accessing of texture map data, where Texture Samplers are employed to sample and filter texture maps (see the Shared Functions chapter). Arithmetic operations can be performed on the texture data, input fragment information, and Pixel Shader Constants to compute the resultant fragment's output. The Pixel Shader program also allows the pixel to be discarded from further processing.

3DSTATE_PS

This command is used to set state used by the pixel shader dispatch stage.

Command
3DSTATE_PS

Programming Note	
Context:	Pixel Shader Thread Generation
Note: The PS Unit also receives 3DSTATE_PS_BLEND, 3DSTATE_SAMPLEMASK, 3DSTATE_MULTISAMPLE, and 3DSTATE_PS_EXTRA.	

Signal	PS_INT::oMask Present to RenderTarget
Description	This bit is inserted in the PS payload header and made available to the DataPort (either via the message header or via header bypass) to indicate that oMask data (one or two phases) is included in Render Target Write messages. If present, the oMask data is used to mask off samples.
Formula	= 3DSTATE_PS_EXTRA::oMask Present to RenderTarget

Signal	PS_INT::Dual Source Blend Enable
Description	This field is set if dual source blend is enabled. If this bit is disabled, the data port dual source message reverts to a single source message using source 0.
Formula	= 3DSTATE_PS_BLEND::ColorBufferBlendEnable && (PS_INT::UsesSrc1BlendFactor || (PS_INT::IndependentAlphaUsesSrc1BlendFactors && 3DSTATE_PS_BLEND::Independent Alpha Blend Enable))

Signal	PS_INT::UsesSrc1BlendFactor
Description	
Formula	= (3DSTATE_PS_BLEND::SourceBlendFactor == BLENDFACTOR_SRC1_COLOR) || (3DSTATE_PS_BLEND::SourceBlendFactor == BLENDFACTOR_SRC1_ALPHA) || (3DSTATE_PS_BLEND::SourceBlendFactor == BLENDFACTOR_INV_SRC1_COLOR) || (3DSTATE_PS_BLEND::SourceBlendFactor == BLENDFACTOR_INV_SRC1_ALPHA) || (3DSTATE_PS_BLEND::DestinationBlendFactor == BLENDFACTOR_SRC1_COLOR) ||

	(3DSTATE_PS_BLEND::DestinationBlendFactor == BLENDFACTOR_SRC1_ALPHA) || (3DSTATE_PS_BLEND::DestinationBlendFactor == BLENDFACTOR_INV_SRC1_COLOR) || (3DSTATE_PS_BLEND::DestinationBlendFactor == BLENDFACTOR_INV_SRC1_ALPHA)

Signal	PS_INT::IndependentAlphaUsesSrc1BlendFactors
Description	
Formula	= (3DSTATE_PS_BLEND::SourceAlphaBlendFactor == BLENDFACTOR_SRC1_COLOR) || (3DSTATE_PS_BLEND::SourceAlphaBlendFactor == BLENDFACTOR_SRC1_ALPHA) || (3DSTATE_PS_BLEND::SourceAlphaBlendFactor == BLENDFACTOR_INV_SRC1_COLOR) || (3DSTATE_PS_BLEND::SourceAlphaBlendFactor == BLENDFACTOR_INV_SRC1_ALPHA) || (3DSTATE_PS_BLEND::DestinationAlphaBlendFactor == BLENDFACTOR_SRC1_COLOR) || (3DSTATE_PS_BLEND::DestinationAlphaBlendFactor == BLENDFACTOR_SRC1_ALPHA) || (3DSTATE_PS_BLEND::DestinationAlphaBlendFactor == BLENDFACTOR_INV_SRC1_COLOR) || (3DSTATE_PS_BLEND::DestinationAlphaBlendFactor == BLENDFACTOR_INV_SRC1_ALPHA)

Signal	PS_INT::PS UAV-only
Description	Pixel Shader UAV-only render target
Formula	= 3DSTATE_PS_EXTRA::Pixel Shader Has UAV && !3DSTATE_PS_EXTRA:: Pixel Shader Does not write to RT

Command
3DSTATE_PS_EXTRA This command is used to set state used by the pixel shader dispatch stage
3DSTATE_PS_BLEND This command is used to set state used by the pixel shader dispatch stage
3DSTATE_CONSTANT_PS
3DSTATE_BINDING_TABLE_POINTERS_PS
3DSTATE_PUSH_CONSTANT_ALLOC_PS
3DSTATE_SAMPLER_STATE_POINTERS_PS

Pixel Grouping (Dispatch Size) Control

The WM unit can pass a grouping of 2 subspans (8 pixels), 4 subspans (16 pixels), or 8 subspans (32 pixels) to a Pixel Shader thread. Software should take into account the following considerations when determining which groupings to support/enable during operation. This determination involves a tradeoff

of these likely conflicting issues. Note that the size of the dispatch has significant impact on the kernel program. (It is certainly not transparent to the kernel.) Also note that there is no implied spatial relationship between the subspans passed to a PS thread, other than the fact that they come from the same object.

- **Thread Efficiency**: In general, there is some amount of overhead involved with PS thread dispatch, and if this can be amortized over a larger number of pixels, efficiency will likely increase. This is especially true for very short PS kernels, as may be used for desktop composition, etc.
- **GRF Consumption**: Processing more pixels per thread requires a larger thread payload and likely more temporary register usage, both of which translate into a requirement for a larger GRF register allocation for the threads. This increased GRF usage could lead to increased use of scratch space (for spill/fill, etc.) and possibly less efficient use of the EUs (as it would be less likely to find an EU with enough free physical GRF registers to service the thread).
- **Object Size**: If the number of very small objects (e.g., covering 2 subspans or fewer) is expected to comprise a significant portion of the workload, supporting the 8-pixel dispatch mode may be advantageous. Otherwise there could be a large number of 16-pixel dispatches with only 1 or 2 valid subspans, resulting in low efficiency for those threads.
- **Intangibles**: Kernel footprint & Instruction Cache impact; Complexity; ....

The groupings of subspans that the WM unit is allowed to include in a PS thread payload is controlled by the **32,16,8 Pixel Dispatch Enable** state variables programmed in WM_STATE. Using these state variables, the WM unit attempts to dispatch the largest allowed grouping of subspans. The following table lists the possible combinations of these state variables.

Please note that, the valid column in the table indicates which products supports the combination dispatch. Combinations that are not listed in the table are not available on any product.

The letter codes A, B, D, and E used in the Variable Pixel Dispatch table below are valid for all projects with some specific mode restrictions for specific projects for B, D, and E as indicated in the next few tables.

D is like B with an added general restriction, that it cannot be used in non-1x PERSAMPLE mode.

E cannot be used in PERSAMPLE mode with number of multisamples >= 2.

See SIMD32 enable for additional programming restrictions.

F is valid combination.

Variable Pixel Dispatch

Contiguous 64 Pixel Dispatch Enable	Contiguous 32 Pixel Dispatch Enable	32 Pixel Dispatch Enable	16 Pixel Dispatch Enable	8 Pixel Dispatch Enable	Valid	IP for n-pixel Contiguous Dispatch		IP for n-pixel Dispatch (KSP offsets are in 128-bit instruction units)		
						n=64	n=32	n=32	n=16	n=8
0	0	0	0	1	A					KSP[0]
0	0	0	1	0	B				KSP[0]	
0	0	0	1	1	D				KSP[2]	KSP[0]
0	0	1	0	0	B			KSP[0]		
0	0	1	1	0	E			KSP[1]	KSP[2]	
0	0	1	1	1	D			KSP[1]	KSP[2]	KSP[0]
0	0	1	0	1	F			KSP[1]		KSP[0]
0	1	1	1	0	D		KSP[2]	KSP[1]	KSP[0]	
1	0	1	1	0	D	KSP[2]		KSP[1]	KSP[0]	

Each of the three KSP values is separately specified. In addition, each kernel has a separately-specified GRF register count.

Depending on the subspan grouping selected, the WM unit will modify the starting PS Instruction Pointer (derived from the Kernel Start Pointer in WM_STATE) as a means to inform the PS kernel of the number of subspans included in the payload. The modified IP is a function of the enabled modes and the dispatch size, as shown in the table below.

HW will pick the right Kernel start pointer according to the dispatch size. (Note that the pointer from WM_STATE is 64-byte aligned which corresponds to four 128-bit instructions.)

If only one dispatch mode is enabled, the Jitter should not include any jump table entries at the beginning of the PS kernel. If multiple dispatch modes are enabled, a two entry jump table should always be inserted, regardless of which modes are enabled (jump table entry for 8 pixel dispatch, followed by jump table entry for 32 pixel dispatch).

Note that for SIMD32 dispatch, pixel shader dispatch function increments GRF Start Register for URB Data state by 2 to account for the additional SIMD16 payload. The Pixel Shader kernel needs to comprehend this modification for SIMD32.

```
if ( 32PixelDispatchEnable && n > 7 )
    Dispatch 32 Pixels
else if ( 16PixelDispatchEnable && ( n > 2 || ! 8PixelDispatchEnable) )
    Dispatch 16 Pixels
else
    Dispatch 8 Pixels
end if
```

Multisampling Effects on Pixel Shader Dispatch

The pixel shader payloads are defined in terms of subspans and pixels. The slots in the pixel shader thread previously mapped 1:1 with pixels. With multisampling, a slot could contain a pixel or may just contain a single sample, depending on the mode. Payload definitions now refer to *slot* to make the definition independent of multisampling mode.

MSDISPMODE_PERPIXEL Thread Dispatch

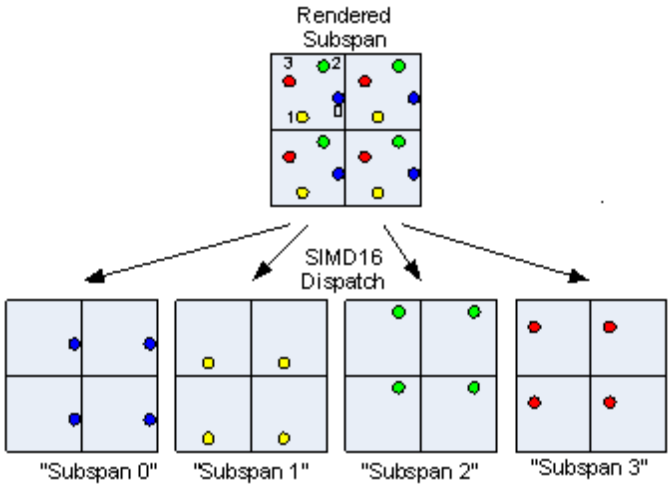
In PERPIXEL mode, the pixel shader kernel still works on 2/4/8 separate subspans, depending on dispatch mode. The fact that rasterization and the depth/stencil tests are being performed on a per-sample (not per-pixel) basis is transparent to the pixel shader kernel.

Programming Note	
Context:	MSDISPMODE_PERPIXEL Thread Dispatch
When NUM_MULTISAMPLES == 16 (i.e. 16x MSAA) and PS_DISPATCH_MODE is PER_PIXEL, SIMD32 pixel shader is not supported.	

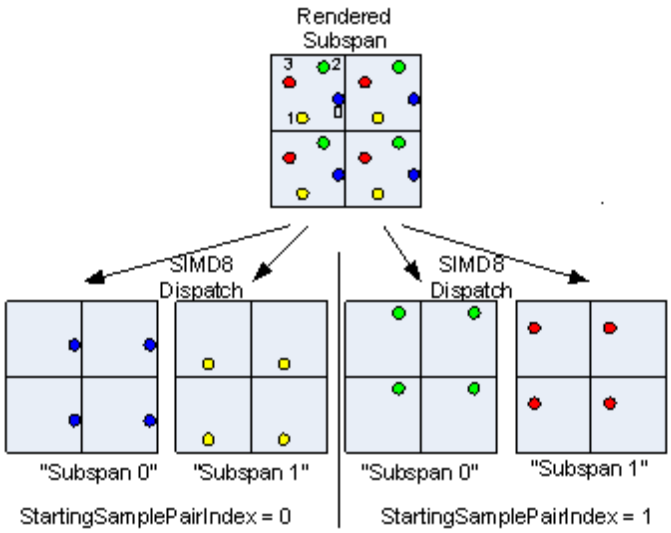
MSDISPMODE_PERSAMPLE Thread Dispatch

In PERSAMPLE mode, the pixel shader needs to operate on a sample vs. pixel basis (although this collapses in NUMSAMPLES_1 mode) Instead of processing strictly different subspans in parallel , the PS kernel processes different sample indices of one or more subspans in parallel For example, a SIMD16 dispatch in PERSAMPLE/NUMSAMPLES_4 mode would operate on a single subspan, with the usual "4 Subspan0 pixel slots" used for the "4 Sample0 locations of the (single) subspan" Subspan1 slots would be used for the Sample1 locations, and so on This layout allows the pixel shader to compute derivatives/LOD based on deltas between corresponding sample locations in the subspan in the same fashion as LEGACY pixel shader execution, and as required by DX10.1.

Depending on the dispatch mode (8/16/32 pixels) and multisampling mode (1X/4X), there are different mappings of subspans/samples onto dispatches and slots-within-dispatch In some cases, more than one subspan may be included in a dispatch, while in other cases multiple dispatches are be required to process all samples for a single subspan In the latter case, the **StartingSamplePairIndex** value is included in the payload header so the Render Target Write message will access the correct samples with each message.



PERSAMPLE SIMD16 4X Dispatch



PERSAMPLE Dispatch

The following table provides the complete dispatch/slot mappings for all the MS/Dispatch combinations.

Dispatch Size	Num Samples	Slot Mapping (SSPI = Starting Sample Pair Index)
SIMD32	*1X*	*Slot[3:0] = Subspan[0].Pixel[3:0].Sample[0]* *Slot[7:4] = Subspan[1].Pixel[3:0].Sample[0]* *Slot[11:8] = Subspan[2].Pixel[3:0].Sample[0]* *Slot[15:12] = Subspan[3].Pixel[3:0].Sample[0]*

Dispatch Size	Num Samples	Slot Mapping (SSPI = Starting Sample Pair Index)
		Slot[19:16] = Subspan[4].Pixel[3:0].Sample[0] *Slot[23:20] = Subspan[5].Pixel[3:0].Sample[0]* *Slot[27:24] = Subspan[6].Pixel[3:0].Sample[0]* *Slot[31:28] = Subspan[7].Pixel[3:0].Sample[0]*
	2X	*Slot[3:0] = Subspan[0].Pixel[3:0].Sample[0]* *Slot[7:4] = Subspan[0].Pixel[3:0].Sample[1]* *Slot[11:8] = Subspan[1].Pixel[3:0].Sample[0]* *Slot[15:12] = Subspan[1].Pixel[3:0].Sample[1]* *Slot[19:16] = Subspan[2].Pixel[3:0].Sample[0]* *Slot[23:20] = Subspan[2].Pixel[3:0].Sample[1]* *Slot[27:24] = Subspan[3].Pixel[3:0].Sample[0]* *Slot[31:28] = Subspan[3].Pixel[3:0].Sample[1]*
	4X	*Slot[3:0] = Subspan[0].Pixel[3:0].Sample[0]* *Slot[7:4] = Subspan[0].Pixel[3:0].Sample[1]* *Slot[11:8] = Subspan[0].Pixel[3:0].Sample[2]* *Slot[15:12] = Subspan[0].Pixel[3:0].Sample[3]* *Slot[19:16] = Subspan[1].Pixel[3:0].Sample[0]* *Slot[23:20] = Subspan[1].Pixel[3:0].Sample[1]* *Slot[27:24] = Subspan[1].Pixel[3:0].Sample[2]* *Slot[31:28] = Subspan[1].Pixel[3:0].Sample[3]*
	8X	*Slot[3:0] = Subspan[0].Pixel[3:0].Sample[0]* *Slot[7:4] = Subspan[0].Pixel[3:0].Sample[1]* *Slot[11:8] = Subspan[0].Pixel[3:0].Sample[2]* *Slot[15:12] = Subspan[0].Pixel[3:0].Sample[3]* *Slot[19:16] = Subspan[0].Pixel[3:0].Sample[4]* *Slot[23:20] = Subspan[0].Pixel[3:0].Sample[5]* *Slot[27:24] = Subspan[0].Pixel[3:0].Sample[6]* *Slot[31:28] = Subspan[0].Pixel[3:0].Sample[7]*
	16x	*Dispatch[i]: (i=0, 4)*

Dispatch Size	Num Samples	Slot Mapping (SSPI = Starting Sample Pair Index)
		SSPI = i *Slot[3:0] = Subspan[0].Pixel[3:0].Sample[SSPI*2+0]* *Slot[7:4] = Subspan[0].Pixel[3:0].Sample[SSPI*2+1]* *Slot[11:8] = Subspan[0].Pixel[3:0].Sample[SSPI*2+2]* *Slot[15:12] = Subspan[0].Pixel[3:0].Sample[SSPI*2+3]* *Slot[19:16] = Subspan[0].Pixel[3:0].Sample[SSPI*2+4]* *Slot[23:20] = Subspan[0].Pixel[3:0].Sample[SSPI*2+5]* *Slot[27:24] = Subspan[0].Pixel[3:0].Sample[SSPI*2+6]* *Slot[31:28] = Subspan[0].Pixel[3:0].Sample[SSPI*2+7]*
SIMD16	*1X*	*Slot[3:0] = Subspan[0].Pixel[3:0].Sample[0]* *Slot[7:4] = Subspan[1].Pixel[3:0].Sample[0]* *Slot[11:8] = Subspan[2].Pixel[3:0].Sample[0]* *Slot[15:12] = Subspan[3].Pixel[3:0].Sample[0]*
	2X	*Slot[3:0] = Subspan[0].Pixel[3:0].Sample[0]* *Slot[7:4] = Subspan[0].Pixel[3:0].Sample[1]* *Slot[11:8] = Subspan[1].Pixel[3:0].Sample[0]* *Slot[15:12] = Subspan[1].Pixel[3:0].Sample[1]*
	4X	*Slot[3:0] = Subspan[0].Pixel[3:0].Sample[0]* *Slot[7:4] = Subspan[0].Pixel[3:0].Sample[1]* *Slot[11:8] = Subspan[0].Pixel[3:0].Sample[2]* *Slot[15:12] = Subspan[0].Pixel[3:0].Sample[3]*
	8X	*Dispatch[i]: (i=0, 2)* *SSPI = i* *Slot[3:0] = Subspan[0].Pixel[3:0].Sample[SSPI*2+0]* *Slot[7:4] = Subspan[0].Pixel[3:0].Sample[SSPI*2+1]* *Slot[11:8] = Subspan[0].Pixel[3:0].Sample[SSPI*2+2]* *Slot[15:12] = Subspan[0].Pixel[3:0].Sample[SSPI*2+3]*
	16x	*Dispatch[i]: (i=0, 2, 4, 6)* *SSPI = i*

Dispatch Size	Num Samples	Slot Mapping (SSPI = Starting Sample Pair Index)
		*Slot[3:0] = Subspan[0].Pixel[3:0].Sample[SSPI*2+0]* *Slot[7:4] = Subspan[0].Pixel[3:0].Sample[SSPI*2+1]* *Slot[11:8] = Subspan[0].Pixel[3:0].Sample[SSPI*2+2]* *Slot[15:12] = Subspan[0].Pixel[3:0].Sample[SSPI*2+3]*
SIMD8	*1X*	*Slot[3:0] = Subspan[0].Pixel[3:0].Sample[0]* *Slot[7:4] = Subspan[1].Pixel[3:0].Sample[0]*
	2X	*Slot[3:0] = Subspan[0].Pixel[3:0].Sample[0]* *Slot[7:4] = Subspan[0].Pixel[3:0].Sample[1]*
	4X	*Dispatch[i]: (i=0..1)* *SSPI = i* *Slot[3:0] = Subspan[0].Pixel[3:0].Sample[SSPI*2+0]* *Slot[7:4] = Subspan[0].Pixel[3:0].Sample[SSPI*2+1]*
	8X	*Dispatch[i]: (i=0, 1, 2, 3)* *SSPI = i* *Slot[3:0] = Subspan[0].Pixel[3:0].Sample[SSPI*2+0]* *Slot[7:4] = Subspan[0].Pixel[3:0].Sample[SSPI*2+1]*
	16x	*Dispatch[i]: (i=0, 1, 2, 3, 4, 5, 6, 7)* *SSPI = i* *Slot[3:0] = Subspan[0].Pixel[3:0].Sample[SSPI*2+0]* *Slot[7:4] = Subspan[0].Pixel[3:0].Sample[SSPI*2+1]*

PS Thread Payload for Normal Dispatch

The following table lists all possible contents included in a PS thread payload, in the order they are provided. Certain portions of the payload are optional, in which case the corresponding phase is skipped.

PS Thread Payload for Normal Dispatch

All registers are numbered starting at 0, but many registers are skipped depending on configuration. This causes all registers below to be renumbered to fill in the skipped locations. The only case where actual registers may be skipped is immediately before the constant data and again before the setup data.

PS Thread Payload for Normal Dispatch

DWord	Bits	Description
R0.7	31	Reserved.
	30:24	Reserved
	23:0	**Primitive Thread ID:** This field contains the primitive thread count passed to the Windower from the Strips Fans Unit. Format: Reserved for HW Implementation Use.
R0.6	31:24	Reserved
	23:0	**Thread ID:** This field contains the fused thread count which is incremented by the Windower for every fused thread that is dispatched. Format: Reserved for HW Implementation Use.
R0.5	31:10	**Scratch Space Pointer:** Specifies the 1K-byte aligned pointer to the scratch space available for this PS fused thread. This is specified as an offset to the **General State Base Address**. Format = GeneralStateOffset[31:10]
	9:8	Reserved
	7:0	**FFTID:** This ID is assigned by the WM unit and is an identifier for the fused thread. It is used to free up resources used by the thread upon thread completion. Format: Reserved for HW Implementation Use.
R0.4	31:5	**Binding Table Pointer:** Specifies the 32-byte aligned pointer to the Binding Table. It is specified as an offset from the **Surface State Base Address**. Format = SurfaceStateOffset[31:5]
	4:0	Reserved
R0.3	31:5	**Sampler State Pointer:** Specifies the 32-byte aligned pointer to the Sampler State table. It is specified as an offset from the **Dynamic State Base Address**. Format = DynamicStateOffset[31:5]
	4	Reserved
	3:0	**Per Thread Scratch Space:** Specifies the amount of scratch space allowed to be used by this fused thread. **Programming Notes:** This amount is available to the kernel for information only. It will be passed verbatim (if not altered by the kernel) to the Data Port in any scratch space access messages, but the Data Port will ignore it. Format = U4 Range = [0,11] indicating [1k bytes, 2M bytes] in powers of two

DWord	Bits	Description
R0.2	31:0	Reserved: Delivered as zeros (reserved for message header fields).
R0.1	31:6	**Color Calculator State Pointer:** Specifies the 64-byte aligned pointer to the Color Calculator state (COLOR_CALC_STATE structure in memory). It is specified as an offset from the **Dynamic State Base Address**. This value is eventually passed to the ColorCalc function in the DataPort and is used to fetch the corresponding CC_STATE data. Format = DynamicStateOffset[31:5]
	5:0	Reserved
R0.0	31:14	Reserved
	13	**Source Depth to Render Target:** Indicates that source depth will be sent to the render target.
	12	**oMask to Render Target:** Indicates that oMask will be sent to the render target.
	11:5	Reserved
	4:0	**Primitive Topology Type:** This field identifies the Primitive Topology Type associated with the primitive spawning this object. The WM unit does not modify this value (e.g., objects within POINTLIST topologies see POINTLIST). Format: (See 3DPRIMITIVE command in *3D Pipeline*.)
R1.7	31:16	**Pixel/Sample Mask (SubSpan[3:0]):** Indicates which pixels within the four subspans are lit. **Note:** This is not a duplicate of the Dispatch Mask that is delivered to the thread. The dispatch mask has all pixels within a subspan as active if any of them are lit to enable LOD calculations to occur correctly. This field must not be modified by the Pixel Shader kernel.
	15:0	**Pixel/Sample Mask Copy (SubSpan[3:0]):** This is a duplicate copy of the pixel mask. This copy can be modified as the pixel shader thread executes in order to turn off pixels based on kill instructions.
R1.6	31	Reserved
	30:27	**Viewport Index 1:** Specifies the index of the viewport currently being used for upper SIMD8 of dual-SIMD8 thread. Format = U4 Range = [0,15]
	26:16	**Render Target Array Index 1:** Specifies the array index to be used for upper SIMD8 of dual-SIMD8 thread Following surface types: SURFTYPE_1D: specifies the array index Range = [0,2047]

<table>
<tr><th>DWord</th><th>Bits</th><th>Description</th></tr>
<tr><td></td><td></td><td>SURFTYPE_2D: specifies the array index Range = [0,2047]
SURFTYPE_3D: specifies the "r" coordinate Range = [0,2047]
SURFTYPE_CUBE: specifies the face identifier Range = [0,5]
<table><tr><th>Face</th><th>Render Target Array Index</th></tr><tr><td>+x</td><td>0</td></tr><tr><td>-x</td><td>1</td></tr><tr><td>+y</td><td>2</td></tr><tr><td>-y</td><td>3</td></tr><tr><td>+z</td><td>4</td></tr><tr><td>-z</td><td>5</td></tr></table>Format = U11</td></tr>
<tr><td></td><td>15</td><td>Front/Back Facing Polygon 1: Determines whether the polygon 1 is front or back facing.
Used by the render cache to determine which stencil test state to use.
0: Front Facing
1: Back Facing</td></tr>
<tr><td></td><td>14:9</td><td>Reserved</td></tr>
<tr><td></td><td>8</td><td>Reserved for expansion of Starting Sample Pair Index 1</td></tr>
<tr><td></td><td>7:6</td><td>Starting Sample Pair Index 1: Indicates the index of the second sample pair of dual-SIMD8 dispatch.
Format = U2
Range = [0,3]</td></tr>
<tr><td></td><td>5:0</td><td>Reserved</td></tr>
<tr><td>R1.5</td><td>31:16</td><td>Y3: Y coordinate (screen space) for upper-left pixel of subspan 3 (slot 12).
Format = U16</td></tr>
<tr><td></td><td>15:0</td><td>X3: X coordinate (screen space) for upper-left pixel of subspan 3 (slot 12).
Format = U16</td></tr>
<tr><td>R1.4</td><td>31:16</td><td>Y2: Y coordinate (screen space) for upper-left pixel of subspan 2 (slot 8).
Format = U16</td></tr>
<tr><td></td><td>15:0</td><td>X2: X coordinate (screen space) for upper-left pixel of subspan 2 (slot 8).
Format = U16</td></tr>
</table>

<table>
<tr><th>DWord</th><th>Bits</th><th>Description</th></tr>
<tr><td rowspan="2">R1.3</td><td>31:16</td><td>Y1: Y coordinate (screen space) for upper-left pixel of subspan 1 (slot 4).
Format = U16</td></tr>
<tr><td>15:0</td><td>X1: X coordinate (screen space) for upper-left pixel of subspan 1 (slot 4).
Format = U16</td></tr>
<tr><td rowspan="2">R1.2</td><td>31:16</td><td>Y0: Y coordinate (screen space) for upper-left pixel of subspan 0 (slot 0).
Format = U16</td></tr>
<tr><td>15:0</td><td>X0: X coordinate (screen space) for upper-left pixel of subspan 0 (slot 0).
Format = U16</td></tr>
<tr><td rowspan="4">R1.1</td><td>31</td><td>Reserved</td></tr>
<tr><td>30:27</td><td>Viewport Index 0: Specifies the index of the viewport currently being used.
Format = U4
Range = [0,15]</td></tr>
<tr><td>26:16</td><td>Render Target Array Index 0: Specifies the array index to be used for the following surface types:
SURFTYPE_1D: specifies the array index Range = [0,2047]
SURFTYPE_2D: specifies the array index Range = [0,2047]
SURFTYPE_3D: specifies the "r" coordinate Range = [0,2047]
SURFTYPE_CUBE: specifies the face identifier Range = [0,5]
<table>
<tr><th>Face</th><th>Render Target Array Index</th></tr>
<tr><td>+x</td><td>0</td></tr>
<tr><td>-x</td><td>1</td></tr>
<tr><td>+y</td><td>2</td></tr>
<tr><td>-y</td><td>3</td></tr>
<tr><td>+z</td><td>4</td></tr>
<tr><td>-z</td><td>5</td></tr>
</table>
Format = U11</td></tr>
<tr><td>15</td><td>Front/Back Facing Polygon 0: Determines whether the polygon 0 is front or back facing. Used by the render cache to determine which stencil test state to use.
0: Front Facing</td></tr>
</table>

DWord	Bits	Description
		1: Back Facing
	14:9	Reserved
	8	Reserved for expansion of **Starting Sample Pair Index 0**
	7:6	**Starting Sample Pair Index 0:** Indicates the index of the first sample pair of the dispatch. Format = U2 Range = [0,3]
	5:0	Reserved
R1.0	31:20	Reserved
	19:16	**MSAA rate** (multisample count) Format: U4 [1..16] This field specifies MSAA sampling rate (required for PS+S monolithic shader).
	15:8	**ActualCoarsePixelShadingSize.Y** if coarse pixel dispatch Format: U8 This field specifies size (in pixels) of coarse pixel shading rate in Y dimension. Valid values are 1, 2, and 4. Note: coarse shading rate is constant for all coarse pixels in same thread dispatch.
	15:12	Slot 3 SampleID (if pixel or sample dispatch) Format = U4 1X MSAA range: [0] 2X MSAA range [0,1] 4X MSAA range [0..3] 8X MSAA range [0..7]
	11:8	Slot 2 SampleID (if pixel or sample dispatch) Format = U4 1X MSAA range: [0] 2X MSAA range [0,1] 4X MSAA range [0..3] 8X MSAA range [0..7]
	7:0	**ActualCoarsePixelShadingSize.X** if coarse pixel dispatch Format: U8

DWord	Bits	Description
		This field specifies size (in pixels) of coarse pixel shading rate in X dimension. Valid values are 1, 2, and 4. Note: coarse shading rate is constant for all coarse pixels in same thread dispatch.
	7:4	Slot 1 SampleID (if pixel or sample dispatch) Format = U4 1X MSAA range: [0] 2X MSAA range [0,1] 4X MSAA range [0..3] 8X MSAA range [0..7]
	3:0	Slot 0 SampleID (if pixel or sample dispatch) Format = U4 1X MSAA range: [0] 2X MSAA range [0,1] 4X MSAA range [0..3] 8X MSAA range [0..7]
		R2: Delivered only if this is a *32-pixel dispatch*.
R2.7	31:16	**Pixel/Sample Mask (SubSpan[7:4]):** Indicates which pixels within the upper four subspans are lit. This field is valid only when the 32 pixel dispatch state is enabled. This field must not be modified by the pixel shader thread. Note: This is not a duplicate of the dispatch mask that is delivered to the thread. The dispatch mask has all pixels within a subspan as active if any of them are lit to enable LOD calculations to occur correctly. This field must not be modified by the Pixel Shader kernel.
	15:0	**Pixel/Sample Mask Copy (SubSpan[7:4]):** This is a duplicate copy of pixel mask for the upper 16 pixels. This copy will be modified as the pixel shader thread executes to turn off pixels based on kill instructions.
R2.6	31:0	Reserved
R2.5	31:16	**Y7:** Y coordinate (screen space) for upper-left pixel of subspan 7 (slot 28) Format = U16
	15:0	**X7:** X coordinate (screen space) for upper-left pixel of subspan 7 (slot 28) Format = U16

DWord	Bits	Description
R2.4	31:16	Y6
	15:0	X6
R2.3	31:16	Y5
	15:0	X5
R2.2	31:16	Y4
	15:0	X4
R2.1	31:0	Reserved
R2.0	31:16	Reserved
	15:12	Slot 7 SampleID Format = U4 1X MSAA range: [0] 2X MSAA range [0,1] 4X MSAA range [0..3] 8X MSAA range [0..7] 16X MSAA range [0..15]
	11:8	Slot 6 SampleID Format = U4 1X MSAA range: [0] 2X MSAA range [0,1] 4X MSAA range [0..3] 8X MSAA range [0..7] 16X MSAA range [0..15]
	7:4	Slot 5 SampleID Format = U4 1X MSAA range: [0] 2X MSAA range [0,1] 4X MSAA range [0..3] 8X MSAA range [0..7] 16X MSAA range [0..15]
	3:0	Slot 4 SampleID Format = U4

DWord	Bits	Description
		1X MSAA range: [0] 2X MSAA range [0,1] 4X MSAA range [0..3] 8X MSAA range [0..7] 16X MSAA range [0..15]
		R3-R26: Delivered only if the corresponding **Barycentric Interpolation Mode** bit is set. Register phases containing Slot 8-15 data are not delivered in *8-pixel dispatch* mode.
R3.7	31:0	Perspective Pixel Location Barycentric[1] for Slot 7 This and the next register phase is only included if the corresponding enable bit in **Barycentric Interpolation Mode** is set. Format = IEEE_Float
R3.6	31:0	Perspective Pixel Location Barycentric[1] for Slot 6
R3.5	31:0	Perspective Pixel Location Barycentric[1] for Slot 5
R3.4	31:0	Perspective Pixel Location Barycentric[1] for Slot 4
R3.3	31:0	Perspective Pixel Location Barycentric[1] for Slot 3
R3.2	31:0	Perspective Pixel Location Barycentric[1] for Slot 2
R3.1	31:0	Perspective Pixel Location Barycentric[1] for Slot 1
R3.0	31:0	Perspective Pixel Location Barycentric[1] for Slot 0
R4		Perspective Pixel Location Barycentric[2] for Slots 7:0
R5.7	31:0	Perspective Pixel Location Barycentric[1] for Slot 15
R5.6	31:0	Perspective Pixel Location Barycentric[1] for Slot 14
R5.5	31:0	Perspective Pixel Location Barycentric[1] for Slot 13
R5.4	31:0	Perspective Pixel Location Barycentric[1] for Slot 12
R5.3	31:0	Perspective Pixel Location Barycentric[1] for Slot 11
R5.2	31:0	Perspective Pixel Location Barycentric[1] for Slot 10
R5.1	31:0	Perspective Pixel Location Barycentric[1] for Slot 9
R5.0	31:0	Perspective Pixel Location Barycentric[1] for Slot 8
R6		Perspective Pixel Location Barycentric[2] for Slots 15:8
R7:10		Perspective Centroid Barycentric
R11:14		Perspective Sample Barycentric
R15:18		Linear Pixel Location Barycentric
R19:22		Linear Centroid Barycentric
R23:26		Linear Sample Barycentric
		R27: Delivered only if **Pixel Shader Uses Source Depth** is set.

DWord	Bits	Description
R27.7	31:0	Interpolated Depth for Slot 7 Format = IEEE_Float This and the next register phase is only included if **Pixel Shader Uses Source Depth** (WM_STATE) is set.
R27.6	31:0	Interpolated Depth for Slot 6
R27.5	31:0	Interpolated Depth for Slot 5
R27.4	31:0	Interpolated Depth for Slot 4
R27.3	31:0	Interpolated Depth for Slot 3
R27.2	31:0	Interpolated Depth for Slot 2
R27.1	31:0	Interpolated Depth for Slot 1
R27.0	31:0	Interpolated Depth for Slot 0
		R28: Delivered only if **Pixel Shader Uses Source Depth** is set and this is not an *8-pixel dispatch*.
R28.7	31:0	Interpolated Depth for Slot 15
R28.6	31:0	Interpolated Depth for Slot 14
R28.5	31:0	Interpolated Depth for Slot 13
R28.4	31:0	Interpolated Depth for Slot 12
R28.3	31:0	Interpolated Depth for Slot 11
R28.2	31:0	Interpolated Depth for Slot 10
R28.1	31:0	Interpolated Depth for Slot 9
R28.0	31:0	Interpolated Depth for Slot 8
		R29: Delivered only if **Pixel Shader Uses Source W** is set.
R29.7	31:0	Interpolated W for Slot 7 Format = IEEE_Float This and the next register phase are only included if **Pixel Shader Uses Source W** (WM_STATE) is set.
R29.6	31:0	Interpolated W for Slot 6
R29.5	31:0	Interpolated W for Slot 5
R29.4	31:0	Interpolated W for Slot 4
R29.3	31:0	Interpolated W for Slot 3
R29.2	31:0	Interpolated W for Slot 2
R29.1	31:0	Interpolated W for Slot 1
R29.0	31:0	Interpolated W for Slot 0

DWord	Bits	Description
		R30: Delivered only if **Pixel Shader Uses Source W** is set and this is not an *8-pixel dispatch*.
R30.7	31:0	Interpolated W for Slot 15
R30.6	31:0	Interpolated W for Slot 14
R30.5	31:0	Interpolated W for Slot 13
R30.4	31:0	Interpolated W for Slot 12
R30.3	31:0	Interpolated W for Slot 11
R30.2	31:0	Interpolated W for Slot 10
R30.1	31:0	Interpolated W for Slot 9
R30.0	31:0	Interpolated W for Slot 8
		R31: Delivered only if **Position XY Offset Select** is either POSOFFSET_CENTROID or POSOFFSET_SAMPLE.
R31.7	31:24	Position Offset Y for Slot 15 This field contains either the CENTROID or SAMPLE position offset for Y, depending on the state of **Position XY Offset Select**. Format = U4.4 For non-CP rate dispatch: Range = [0.0,1.0) For CP rate dispatch: Range = [0.0,4.0)
	23:16	Position Offset X for Slot 15 This field contains either the CENTROID or SAMPLE position offset for X, depending on the state of **Position XY Offset Select**. Format = U4.4 For non-CP rate dispatch: Range = [0.0,1.0) For CP rate dispatch: Range = [0.0,4.0)
	15:8	Position Offset Y for Slot 14
	7:0	Position Offset X for Slot 14
R31.6	31:24	Position Offset Y for Slot 13
	23:16	Position Offset X for Slot 13
	15:8	Position Offset Y for Slot 12
	7:0	Position Offset X for Slot 12
R31.5:4		Position Offset X/Y for Slot[11:8]
R31.3:2		Position Offset X/Y for Slot[7:4]
R31.1:0		Position Offset X/Y for Slot[3:0]
		R32: Delivered only if **Pixel Shader Uses Input Coverage Mask** is set.

DWord	Bits	Description
R32.7	31:0	Input Coverage Mask for Slot 7 Format = U32 This and the next register phase is only included if **Pixel Shader Uses Input Coverage Mask** (3DSTATE_PS) is set. Fields encode sample Coverage Mask for kernels dispatched at pixel-rate or pixel Coverage Mask for kernels dispatched at coarse-rate.
R32.6	31:0	Input Coverage Mask for Slot 6 Fields encode sample Coverage Mask for kernels dispatched at pixel-rate or pixel Coverage Mask for kernels dispatched at coarse-rate.
R32.5	31:0	Input Coverage Mask for Slot 5 Fields encode sample Coverage Mask for kernels dispatched at pixel-rate or pixel Coverage Mask for kernels dispatched at coarse-rate.
R32.4	31:0	Input Coverage Mask for Slot 4 Fields encode sample Coverage Mask for kernels dispatched at pixel-rate or pixel Coverage Mask for kernels dispatched at coarse-rate.
R32.3	31:0	Input Coverage Mask for Slot 3 Fields encode sample Coverage Mask for kernels dispatched at pixel-rate or pixel Coverage Mask for kernels dispatched at coarse-rate.
R32.2	31:0	Input Coverage Mask for Slot 2 Fields encode sample Coverage Mask for kernels dispatched at pixel-rate or pixel Coverage Mask for kernels dispatched at coarse-rate.
R32.1	31:0	Input Coverage Mask for Slot 1 Fields encode sample Coverage Mask for kernels dispatched at pixel-rate or pixel Coverage Mask for kernels dispatched at coarse-rate.
R32.0	31:0	Input Coverage Mask for Slot 0 Fields encode sample Coverage Mask for kernels dispatched at pixel-rate or pixel Coverage Mask for kernels dispatched at coarse-rate.
		R33: Delivered only if **Pixel Shader Uses Input Coverage Mask** is set and this is not an *8-pixel dispatch*.
R33.7	31:0	Input Coverage Mask for Slot 15 Fields encode sample Coverage Mask for kernels dispatched at pixel-rate or pixel Coverage

DWord	Bits	Description
		Mask for kernels dispatched at coarse-rate.
R33.6	31:0	Input Coverage Mask for Slot 14 Fields encode sample Coverage Mask for kernels dispatched at pixel-rate or pixel Coverage Mask for kernels dispatched at coarse-rate.
R33.5	31:0	Input Coverage Mask for Slot 13 Fields encode sample Coverage Mask for kernels dispatched at pixel-rate or pixel Coverage Mask for kernels dispatched at coarse-rate.
R33.4	31:0	Input Coverage Mask for Slot 12 Fields encode sample Coverage Mask for kernels dispatched at pixel-rate or pixel Coverage Mask for kernels dispatched at coarse-rate.
R33.3	31:0	Input Coverage Mask for Slot 11 Fields encode sample Coverage Mask for kernels dispatched at pixel-rate or pixel Coverage Mask for kernels dispatched at coarse-rate.
R33.2	31:0	Input Coverage Mask for Slot 10 Fields encode sample Coverage Mask for kernels dispatched at pixel-rate or pixel Coverage Mask for kernels dispatched at coarse-rate.
R33.1	31:0	Input Coverage Mask for Slot 9 Fields encode sample Coverage Mask for kernels dispatched at pixel-rate or pixel Coverage Mask for kernels dispatched at coarse-rate.
R33.0	31:0	Input Coverage Mask for Slot 8 Fields encode sample Coverage Mask for kernels dispatched at pixel-rate or pixel Coverage Mask for kernels dispatched at coarse-rate.
		R34-R57: Delivered only if the corresponding **Barycentric Interpolation Mode** bit is set and this is a *32-pixel dispatch*.
R34.7	31:0	Perspective Pixel Location Barycentric[1] for Slot 23 This and the next register phase is only included if the corresponding enable bit in **Barycentric Interpolation Mode** is set. Format = IEEE_Float
R34.6	31:0	Perspective Pixel Location Barycentric[1] for Slot 22
R34.5	31:0	Perspective Pixel Location Barycentric[1] for Slot 21
R34.4	31:0	Perspective Pixel Location Barycentric[1] for Slot 20

DWord	Bits	Description
R34.3	31:0	Perspective Pixel Location Barycentric[1] for Slot 19
R34.2	31:0	Perspective Pixel Location Barycentric[1] for Slot 18
R34.1	31:0	Perspective Pixel Location Barycentric[1] for Slot 17
R34.0	31:0	Perspective Pixel Location Barycentric[1] for Slot 16
R35		Perspective Pixel Location Barycentric[2] for Slots 23:16
R36.7	31:0	Perspective Pixel Location Barycentric[1] for Slot 31
R36.6	31:0	Perspective Pixel Location Barycentric[1] for Slot 30
R36.5	31:0	Perspective Pixel Location Barycentric[1] for Slot 29
R36.4	31:0	Perspective Pixel Location Barycentric[1] for Slot 28
R36.3	31:0	Perspective Pixel Location Barycentric[1] for Slot 27
R36.2	31:0	Perspective Pixel Location Barycentric[1] for Slot 26
R36.1	31:0	Perspective Pixel Location Barycentric[1] for Slot 25
R36.0	31:0	Perspective Pixel Location Barycentric[1] for Slot 24
R37		Perspective Pixel Location Barycentric[2] for Slots 31:24
R38:41		Perspective Centroid Barycentric
R42:45		Perspective Sample Barycentric
R46:49		Linear Pixel Location Barycentric
R50:53		Linear Centroid Barycentric
R54:57		Linear Sample Barycentric
		R58-R59: Delivered only if **Pixel Shader Uses Source Depth** is set and this is a *32-pixel dispatch*.
R58.7	31:0	Interpolated Depth for Slot 23 Format = IEEE_Float This and the next register phase is only included if **Pixel Shader Uses Source Depth** (WM_STATE) bit is set.
R58.6	31:0	Interpolated Depth for Slot 22
R58.5	31:0	Interpolated Depth for Slot 21
R58.4	31:0	Interpolated Depth for Slot 20
R58.3	31:0	Interpolated Depth for Slot 19
R58.2	31:0	Interpolated Depth for Slot 18
R58.1	31:0	Interpolated Depth for Slot 17
R58.0	31:0	Interpolated Depth for Slot 16
R59.7	31:0	Interpolated Depth for Slot 31
R59.6	31:0	Interpolated Depth for Slot 30
R59.5	31:0	Interpolated Depth for Slot 29

DWord	Bits	Description
R59.4	31:0	Interpolated Depth for Slot 28
R59.3	31:0	Interpolated Depth for Slot 27
R59.2	31:0	Interpolated Depth for Slot 26
R59.1	31:0	Interpolated Depth for Slot 25
R59.0	31:0	Interpolated Depth for Slot 24
		R60-R61: Delivered only if **Pixel Shader Uses Source W** is set and this is a *32-pixel dispatch*.
R60.7	31:0	Interpolated W for Slot 23 Format = IEEE_Float This and the next register phase are only included if **Pixel Shader Uses Source W** (WM_STATE) bit is set.
R60.6	31:0	Interpolated W for Slot 22
R60.5	31:0	Interpolated W for Slot 21
R60.4	31:0	Interpolated W for Slot 20
R60.3	31:0	Interpolated W for Slot 19
R60.2	31:0	Interpolated W for Slot 18
R60.1	31:0	Interpolated W for Slot 17
R60.0	31:0	Interpolated W for Slot 16
R61.7	31:0	Interpolated W for Slot 31
R61.6	31:0	Interpolated W for Slot 30
R61.5	31:0	Interpolated W for Slot 29
R61.4	31:0	Interpolated W for Slot 28
R61.3	31:0	Interpolated W for Slot 27
R61.2	31:0	Interpolated W for Slot 26
R61.1	31:0	Interpolated W for Slot 25
R61.0	31:0	Interpolated W for Slot 24
		R62: Delivered only if **Position XY Offset Select** is either POSOFFSET_CENTROID or POSOFFSET_SAMPLE and this is a *32-pixel dispatch*.
R62.7	31:24	Position Offset Y for Slot 31 This field contains either the CENTROID or SAMPLE position offset for Y, depending on the state of **Position XY Offset Select**. Format = U4.4 Range = [0.0,1.0)
	23:16	Position Offset X for Slot 31 This field contains either the CENTROID or SAMPLE position offset for X, depending on the

DWord	Bits	Description
		state of **Position XY Offset Select**. Format = U4.4 Range = [0.0,1.0)
	15:8	Position Offset Y for Slot 30
	7:0	Position Offset X for Slot 30
R62.6	31:24	Position Offset Y for Slot 29
	23:16	Position Offset X for Slot 29
	15:8	Position Offset Y for Slot 28
	7:0	Position Offset X for Slot 28
R62.5:4		Position Offset X/Y for Slot[27:24]
R62.3:2		Position Offset X/Y for Slot[23:20]
R62.1:0		Position Offset X/Y for Slot[19:16]
		R63-R64: Delivered only if **Pixel Shader Uses Input Coverage Mask** is set and this is a *32-pixel dispatch*.
R63.7	31:0	Input Coverage Mask for Slot 23 Format = U32 This and the next register phase are only included if **Pixel Shader Uses Input Coverage Mask** (3DSTATE_PS) is set.
R63.6	31:0	Input Coverage Mask for Slot 22
R63.5	31:0	Input Coverage Mask for Slot 21
R63.4	31:0	Input Coverage Mask for Slot 20
R63.3	31:0	Input Coverage Mask for Slot 19
R63.2	31:0	Input Coverage Mask for Slot 18
R63.1	31:0	Input Coverage Mask for Slot 17
R63.0	31:0	Input Coverage Mask for Slot 16
R64.7	31:0	Input Coverage Mask for Slot 31
R64.6	31:0	Input Coverage Mask for Slot 30
R64.5	31:0	Input Coverage Mask for Slot 29
R64.4	31:0	Input Coverage Mask for Slot 28
R64.3	31:0	Input Coverage Mask for Slot 27
R64.2	31:0	Input Coverage Mask for Slot 26
R64.1	31:0	Input Coverage Mask for Slot 25
R64.0	31:0	Input Coverage Mask for Slot 24
		R65 delivered ONLY if **Pixel Shader Requires RequiredCoarsePixelShadingSize** is set. Value is undefined for SIMD32 thread payload.

DWord	Bits	Description
R65.7	31:0	**RequestedCoarsePixelShadingRate.Y** for subspan 3 (slot 12) This is post-clamp value, expected range [1.0f,4.0f] or inner range if min/max configured. Format: IEEE_Float
R65.6	31:0	**RequestedCoarsePixelShadingRate.Y** for subspan 2 (slot 8) Format: IEEE_Float
R65.5	31:0	**RequestedCoarsePixelShadingRate.Y** for subspan 1 (slot 4) Format: IEEE_Float
R65.4	31:0	**RequestedCoarsePixelShadingRate.Y** for subspan 0 (slot 0) Format: IEEE_Float
R65.3	31:0	**RequestedCoarsePixelShadingRate.X** for subspan 3 (slot 12) Format: IEEE_Float
R65.2	31:0	**RequestedCoarsePixelShadingRate.X** for subspan 2 (slot 8) Format: IEEE_Float
R65.1	31:0	**RequestedCoarsePixelShadingRate.X** for subspan 1 (slot 4) Format: IEEE_Float
R65.0	31:0	**RequestedCoarsePixelShadingRate.X** for subspan 0 (slot 0) Format: IEEE_Float
	R66: delivered only if **Pixel Shader Requires Source Depth and/or W Attribute Vertex Deltas** is set.	
R66.7	31:0	**rhw_c0** – Co for 1/w plane Format = IEEE_Float
R66.6	31:0	**YStart** coordinate (screen space) for upper-left vertex of a triangle being rasterized. Format = float32
R66.5	31:0	**rhw_cx** – Cx for 1/w plane Format = IEEE_Float
R66.4	31:0	**rhw_cy** - Cy for 1/w plane

DWord	Bits	Description
		Format = IEEE_Float
R66.3	31:0	**z_c0** – Co for z plane Format = IEEE_Float
R66.2	31:0	**XStart** coordinate (screen space) for upper-left vertex of a triangle being rasterized. Format = float32
R66.1	31:0	**z_cx** – Cx for z plane Format = IEEE_Float
R66.0	31:0	**z_cy** - Cy for z plane Format = IEEE_Float
	R67: delivered only if **Pixel Shader Requires Source Depth and/or W Attribute Vertex Deltas** and **Dual 8-pixel dispatch** are set.	
R67		Defines RHW and Z coefficients for poly 1.
	R68: delivered only if **Pixel Shader Requires Perspective Bary Planes** is set.	
R68.7	31:0	**bary2_c0** – Co for bary2/w plane Format = IEEE_Float
R68.6	31:0	**YStart** coordinate (screen space) for upper-left vertex of a triangle being rasterized. Format = float32
R68.5	31:0	**bary2_cx** – Cx for bary2/w plane Format = IEEE_Float
R68.4	31:0	**bary2_cy** - Cy for bary2/w plane Format = IEEE_Float
R68.3	31:0	**bary1_c0** – Co for bary1/w plane Format = IEEE_Float
R68.2	31:0	**XStart** coordinate (screen space) for upper-left vertex of a triangle being rasterized. Format = float32

DWord	Bits	Description
R68.1	31:0	**bary1_cx** – Cx for bary1/w plane Format = IEEE_Float
R68.0	31:0	**bary1_cy** - Cy for bary1/w plane Format = IEEE_Float
	R69: delivered only if **Pixel Shader Requires Perspective Bary Planes** and **Dual 8-pixel dispatch** are set.	
R69		Defines perspective bary1/bary2 coefficients and vertex StartX/StartY for poly 1.
	R70: delivered only if **Pixel Shader Requires Non-Perspective Bary Planes** is set.	
R70.7	31:0	**npc_bary2_c0** – Co for npc_bary2 plane Format = IEEE_Float
R70.5	31:0	**YStart** coordinate (screen space) for upper-left vertex of a triangle being rasterized. Format = float32
R70.5	31:0	**npc_bary2_cx** – Cx for npc_bary2 plane Format = IEEE_Float
R70.4	31:0	**npc_bary2_cy** - Cy for npc_bary2 plane Format = IEEE_Float
R70.3	31:0	**npc_bary1_c0** – Co for npc_bary1 plane Format = IEEE_Float
R70.2	31:0	**XStart** coordinate (screen space) for upper-left vertex of a triangle being rasterized. Format = float32
R70.1	31:0	**npc_bary1_cx** – Cx for npc_bary1 plane Format = IEEE_Float
R70.0	31:0	**npc_bary1_cy** - Cy for npc_bary1 plane Format = IEEE_Float
	R71: delivered only if **Pixel Shader Requires Non-Perspective Bary Planes** and **Dual 8-pixel dispatch** are set.	
R71		Defines non-perspective bary1/bary2 coefficients and vertex StartX/StartY for poly 1.

DWord	Bits	Description
	R72: delivered only if **Pixel Shader Requires sample offsets** is set.	
R72.7	31:28	Reserved – MBZ
	29:24	Sub-sample Y offset for sample 15 Format: U0.4 (Upper two bits MBZ.) Subpixel Y offset of Sample 15 relative to the UL pixel origin Range: [0,0.9375] (0/16 - 15/16) The number of valid sub-sample offsets is the number of MSAA samples
	23:22	Reserved – MBZ
	21:16	Sub-sample Y offset for sample 14
	15:14	Reserved – MBZ
	13:8	Sub-sample Y offset for sample 13
	7:6	Reserved – MBZ
	5:0	Sub-sample Y offset for sample 12
R72.6	31:30	Reserved – MBZ
	29:24	Sub-sample Y offset for sample 11
	23:22	Reserved – MBZ
	21:16	Sub-sample Y offset for sample 10
	15:14	Reserved – MBZ
	13:8	Sub-sample Y offset for sample 9
	7:6	Reserved – MBZ
	5:0	Sub-sample Y offset for sample 8
R72.5	31:30	Reserved – MBZ
	29:24	Sub-sample Y offset for sample 7
	23:22	Reserved – MBZ
	21:16	Sub-sample Y offset for sample 6
	15:14	Reserved – MBZ
	13:8	Sub-sample Y offset for sample 5
	7:6	Reserved – MBZ
	5:0	Sub-sample Y offset for sample 4
R72.4	31:30	Reserved – MBZ
	29:24	Sub-sample Y offset for sample 3
	23:22	Reserved – MBZ
	21:16	Sub-sample Y offset for sample 2
	15:14	Reserved – MBZ
	13:8	Sub-sample Y offset for sample 1

DWord	Bits	Description
	7:6	Reserved – MBZ
	5:0	Sub-sample Y offset for sample 0
R72.3	31:30	Reserved – MBZ
	29:24	Sub-sample X offset for sample 15 Format: U0.4 (Upper two bits MBZ.) Subpixel X offset of Sample 15 relative to the UL pixel origin Range: [0,0.9375] (0/16 - 15/16) The number of valid sub-sample offsets is the number of MSAA samples
	23:22	Reserved – MBZ
	21:16	Sub-sample X offset for sample 14
	15:14	Reserved – MBZ
	13:8	Sub-sample X offset for sample 13
	7:6	Reserved – MBZ
	5:0	Sub-sample X offset for sample 12
R72.2	31:30	Reserved – MBZ
	29:24	Sub-sample X offset for sample 11
	23:22	Reserved – MBZ
	21:16	Sub-sample X offset for sample 10
	15:14	Reserved – MBZ
	13:8	Sub-sample X offset for sample 9
	7:6	Reserved – MBZ
	5:0	Sub-sample X offset for sample 8
R72.1	31:30	Reserved – MBZ
	29:24	Sub-sample X offset for sample 7
	23:22	Reserved – MBZ
	21:16	Sub-sample X offset for sample 6
	15:14	Reserved – MBZ
	13:8	Sub-sample X offset for sample 5
	7:6	Reserved – MBZ
	5:0	Sub-sample X offset for sample 4
R72.0	31:30	Reserved – MBZ
	29:24	Sub-sample X offset for sample 3
	23:22	Reserved – MBZ
	21:16	Sub-sample X offset for sample 2
	15:14	Reserved – MBZ

DWord	Bits	Description
	13:8	Sub-sample X offset for sample 1
	7:6	Reserved – MBZ
	5:0	Sub-sample X offset for sample 0
R73.7	31:0	Primitive **Full Pixel Coverage** bitmask Bit *i*, when enabled, indicates an input pixel *i* is fully covered by the Xmin/Xmax/Ymin/Ymax bounding box. For example, with SIMD16 pixel shader dispatch mode, a value of 0x00FF indicates all 16 input pixels have their area fully covered by the input primitive bounding box. A pixel shader kernel can use this bitmask as fast check of full coverage condition. Format: U32
R73.4:6	31:0	Reserved - MBZ
R73.3	31:24	Reserved - MBZ
	23:0	Primitive **Ymax -** maximum value of sub-pixel aligned position.Y attribute in screen-space, computed from vertices forming an input primitive. Format: U16.8
R73.2	31:24	Reserved - MBZ
	23:0	Primitive **Ymin -** minumum value of sub-pixel aligned position.Y attribute in screen-space, computed from vertices forming an input primitive Format: U16.8
R73.1	31:24	Reserved - MBZ
	23:0	Primitive **Xmax -** maximum value of sub-pixel aligned position.X attribute in screen-space, computed from vertices forming an input primitive Format: U16.8
R73.0	31:24	Reserved - MBZ
	23:0	Primitive **Xmin -** minimum value of sub-pixel aligned position.X attribute in screen-space, computed from vertices forming an input primitive Format: U16.8
R74		Defines sub-pixel aligned quad and full pixel coverage mask for poly 1.
		Optional Padding before the Start of Constant/Setup Data The locations between the end of the Optional Payload Header and the location programmed via Dispatch GRF Start Register for Constant/Setup Data are considered “padding” and Reserved (see below).
Optional, multiple of 8 DWs	31:0	Reserved
		The **Dispatch GRF Start Register for Constant/Setup Data** state variable in 3DSTATE_WM

DWord	Bits	Description
		is used to define the starting location of the constant and setup data within the PS thread payload. This control is provided to allow this data to be located at a fixed location within thread payloads, regardless of the amount of data in the Optional Payload Header. This permits the kernel to use direct GRF addressing to access the constant/setup data, regardless of the optional parameters being passed (as these are determined on-the-fly by the WM unit).
[Varies] optional	255:0	**Constant Data (optional):** For more details about the size and source of constant data, please refer to **General Programming of Thread-Generating Stages** in the Push Constants chapter.
		Setup Data (Attribute Vertex Deltas) for dispatches other than dual 8-pixel dispatch. Output data from the SF stage is delivered in the PS thread payload. The amount of data is determined by the **Number of Output Attributes** field. Each register contains two channels of one attribute. Thus, the total number of registers sent is equal to 2 * Number of Output Attributes.
Rp.7	31:0	**a0[0].y** – a0 vertex data for Attribute0.y
Rp.6	31:0	Reserved
Rp.5	31:0	**a2[0].y** – a2-a0 vertex delta for Attribute0.y
Rp.4	31:0	**a1[0].y** – a1-a0 vertex delta for Attribute0.y
Rp.3	31:0	**a0[0].x** – a0 vertex data for Attribute0.x
Rp.2	31:0	Reserved
Rp.1	31:0	**a2[0].x** – a2-a0 vertex delta for Attribute0.x
Rp.0	31:0	**a1[0].x** – a1-a0 vertex delta for Attribute0.x
R(p+1).7	31:0	**a0[0].w** – a0 vertex data for Attribute0.w
R(p+1).6	31:0	Reserved
R(p+1).5	31:0	**a2[0].w** – a2-a0 vertex delta for Attribute0.w
R(p+1).4	31:0	**a1[0].w** – a1-a0 vertex delta for Attribute0.w
R(p+1).3	31:0	**a0[0].z** – a0 vertex data for Attribute0.z
R(p+1).2	31:0	Reserved
R(p+1).1	31:0	**a2[0].z** – a2-a0 vertex delta for Attribute0.z
R(p+1).0	31:0	**a1[0].z** – a1-a0 vertex delta for Attribute0.z
R(p+2*n)		**xy Vertex Deltas for Attributes n** See definition of Rp for format.
R(p+2*n+1)		**zw Vertex Deltas for Attribute n** See definition of R(p+1) for format.
		Setup Data (Attribute Vertex Deltas) for dual 8-pixel dispatch.

DWord	Bits	Description
		Output data from the SF stage is delivered in the PS thread payload. The amount of data is determined by the **Number of Output Attributes** field. Each register contains two channels of one attribute and same layout is used as for dispatches other than dual 8-pixel dispatch. However, two registers used - one for object 0 and another for object 1. Thus, the total number of registers sent is equal to 4 * Number of Output Attributes.
R(p+4*n)		**xy Vertex Deltas for Attribute n from object 0** See definition of Rp for format.
R(p+4*n+1)		**xy Vertex Deltas for Attribute n from object 1** See definition of Rp for format.
R(p+4*n+2)		**zw Vertex Deltas for Attribute n from object 0** See definition of R(p+1) for format.
R(p+4*n+3)		**zw Vertex Deltas for Attribute n from object 1** See definition of R(p+1) for format.

Pixel Backend

This section contains the following subsections:

- MCS Buffer for Render Target(s)
- Render Target Fast Clear
- Render TargetResolve

Color Calculator (Output Merger)

Overview

Note: The Color Calculator logic resides in the Render Cache backing Data Port (DAP) shared function. It is described in this chapter as the Color Calc functions are naturally an extension of the 3D pipeline past the WM stage. See the DataPort chapter for details on the messages used by the Pixel Shader to invoke Color Calculator functionality.

The *Color Calculator* (referred to as "Output Merger in the DX Spec) function within the Data Port shared function completes the processing of rasterized pixels after the pixel color and depth have been computed by the Pixel Shader. This processing is initiated when the pixel shader thread sends a Render Target Write message (see *Shared Functions*) to the Render Cache. (Note that a single pixel shader thread may send multiple Render Target Write messages, with the result that multiple render targets get updated.) The pixel variables pass through a pipeline of fixed (yet programmable) functions, and the results are conditionally written into the appropriate buffers.

The word "pixel" used in this section is effectively replaced with the word "sample" if multisample rasterization is enabled.

Pipeline Stage	Description
Alpha Coverage	It generates coverage masks using AlphaToCoverage AND/OR AlphaToOne functions based on src0.alpha.
Alpha Test	Compare pixel alpha with reference alpha and conditionally discard pixel.
Stencil Test	Compare pixel stencil value with reference and forward result to Buffer Update stage.
Depth Test	Compare pix.Z with corresponding Z value in the Depth Buffer and forward result to Buffer Update stage.
Color Blending	Combine pixel color with corresponding color in color buffer according to programmable function.
Gamma Correction	Adjust pixel's color according to gamma function for SRGB destination surfaces.
Color Quantization	Convert "full precision" pixel color values to fixed precision of the color buffer format.
Logic Ops	Combine pixel color logically with existing color buffer color (mutually exclusive with Color Blending).
Buffer Update	Write final pixel values to color and depth buffers or discard pixel without update.

The following logic describes the high-level operation of the Pixel Processing pipeline:

```
PixelProcessing() {
    AlphaCoverage()
    AlphaTest()
    DepthBufferCoordinateOffsetDisable
    StencilTest()
    DepthTest()
    ColorBufferBlending()
    GammaCorrection()
    ColorQuantization()
    LogicalOps()
    BufferUpdate()
}
```

Alpha Coverage

Alpha coverage logic is supported and can be controlled using three state variables:

- **AlphaToCoverage Enable**, when enabled Color Calculator modifies the sample mask. This function (along with AlphaToOne) come at the top of the pixel pipeline. The sample's Source0.Alpha value (possibly being replicated from the pixel's Source0.Alpha) is used to compute a (optionally dithered) 1/2/4-bit mask (depending on NumSamples).
- The **AlphaToCoverage Dither Enable** SV is used to control the dithering of the AlphaToCoverage mask. The bit corresponding to the sample# is then ANDed with the sample's incoming mask bits - allowing the sample to be masked off depending on alpha.
- **AlphaToOne Enable,** when enabled, Color Calculator must replace Source0.Alpha (if present) with 1.0f.

- If AlphaToCoverage is disabled, AlphaToCoverage Dither does not have any impact.

If Pixel Shader outputs oMask, AlphaToCoverage is disabled in hardware, regardless of the state setting for this feature.

Notes:

- Src0.alpha needs to be first multiplied with AA alpha before applying AlphaToCoverage and AlphaToOne functions.
- An alpha value of NaN results in a no coverage (zero) mask.
- Alpha values from the pixel shader are treated as FLOAT32 format for computing the AlphaToCoverage Mask.

Alpha Test

The Alpha Test function can be used to discard pixels based on a comparison between the incoming pixel's alpha value and the **Alpha Test Reference** state variable in COLOR_CALC_STATE. This operation can be used to remove transparent or nearly-transparent pixels, though other uses for the alpha channel and alpha test are certainly possible.

This function is enabled by the **Alpha Test Enable** state variable in COLOR_CALC_STATE. If ENABLED, this function compares the incoming pixel's alpha value (*pixColor.Alpha*) and the reference alpha value specified by via the **Alpha Test Reference** state variable in COLOR_CALC_STATE. The comparison performed is specified by the **Alpha Test Function** state variable in COLOR_CALC_STATE.

The **Alpha Test Format** state variable is used to specify whether Alpha Test is performed using fixed-point (UNORM8) or FLOAT32 values. Accordingly, it determines whether the **Alpha Reference Value** is passed in a UNORM8 or FLOAT32 format. If UNORM8 is selected, the pixel's alpha value will be converted from floating-point to UNORM8 before the comparison.

Pixels that pass the Alpha Test proceed for further processing. Those that fail are discarded at this point in the pipeline.

If **Alpha Test Enable** is DISABLED, this pipeline stage has no effect.

The Alpha Test function is supported in conjunction with Multiple Render Targets (MRTs). If delivered in the incoming render target write message, source 0 alpha is used to perform the alpha test. If source 0 alpha is not delivered, the normal alpha value is used to perform the alpha test.

Depth Coordinate Offset

The Depth Coordinate Offset function applies a programmable constant offset to the RenderTarget X,Y screen space coordinates in order to generate DepthBuffer coordinates.

The function has been specifically added to allow the OpenGL driver to deal with a RenderTarget and DepthBuffer of differing sizes.

This condition is not an issue for the D3D driver, as D3D defines an upper-left screen coordinate origin which matches the HW rasterizer; as long as the application limits rendering to the smaller of the RT/DepthBuffer extents, no special logic is required.

OpenGL defines a lower-left screen coordinate origin. This requires the driver to incorporate a "Y coordinate flipping" transformation into the viewport mapping function. The Y extent of the RT is used in this flipping transformation. If the DepthBuffer extent is different, the wrong pixel Y locations within the DepthBuffer will be accessed.

The least expensive solution is to provide a translation offset to be applied to the post-viewport-mapped DepthBuffer Y pixel coordinate, effectively allowing the alignment of the lower-left origins of the RT and DepthBuffer. [Note that the previous DBCOD feature performed an optional translation of post-viewport-mapping RT pixel (screen) coordinates to generate DepthBuffer pixel (window) coordinates. Specifically, the Draw Rect Origin X,Y state could be subtracted from the RT pixel coordinates.]

This function uses **Depth Coordinate Offset X,Y** state (signed 16-bit values in 3DSTATE_DEPTH_RECTANGLE) that is unconditionally added to the RT pixel coordinates to generate DepthBuffer pixel coordinates.

The previous DBCOB feature can be supported by having the driver program Depth Coordinate X,Y Offset to the two's complement of the Draw Rect Origin. By programming Depth Coordinate X,Y Offset to zeros, the current "normal" operation (DBCOD disabled) can be achieved.

Programming Note	
Context:	Depth Coordinate Offset
• Only simple 2D RTs are supported (no mipmaps). • Software must ensure that the resultant DepthBuffer Coordinate X,Y values are non-negative. • There are alignment restrictions - see 3DSTATE_DEPTH_BUFFER command.on SFID_DP_DC2) are IA coherent.	

Stencil Test

The Stencil Test function can be used to discard pixels based on a comparison between the [**Backface**] **Stencil Test Reference** state variable and the pixel's stencil value. This is a general-purpose function used for such effects as shadow volumes, per-pixel clipping, etc. The result of this comparison is used in the Stencil Buffer Update function later in the pipeline.

This function is enabled by the **Stencil Test Enable** state variable. If ENABLED, the current stencil buffer value for this pixel is read.

Programming Note	
Context:	Color Calculator - Stencil Test
If the Depth Buffer is either undefined or does not have a surface format of D32_FLOAT_S8X24_UINT or D24_UNORM_S8_UINT and separate stencil buffer is disabled, **Stencil Test Enable** must be DISABLED.	

A 2nd set of the stencil test state variables is provided so that pixels from back-facing objects, assuming they are not culled, can have a stencil test performed on them separate from the test for normal front-facing objects. The separate stencil test for back-facing objects can be enabled via the **Double Sided Stencil Enable** state variable. Otherwise, non-culled back-facing objects will use the same test function, mask and reference value as front-facing objects. The 2nd stencil state for back-facing objects is most

commonly used to improve the performance of rendering shadow volumes which require a different stencil buffer operation depending on whether pixels rendered are from a front-facing or back-facing object. The backface stencil state removes the requirement to render the shadow volumes in 2 passes or sort the objects into front-facing and back-facing lists.

The remainder of this subsection describes the function in term of [**Backface**] <**state variable name>**. The Backface set of state variables are only used if Double Sided Stencil Enable is ENABLED and the object is considered back-facing. Otherwise, the normal (front-facing) state variables are used.

This function then compares the [**Backface**] **Stencil Test Reference** value and the pixel's stencil value value after logically ANDing both values by [**Backface**] **Stencil Test Mask**. The comparison performed is specified by the [**Backface**] **Stencil Test Function** state variable. The result of the comparison is passed down the pipeline for use in the Stencil Buffer Update function. The Stencil Test function does not in itself discard pixels.

If **Stencil Test Enable** is DISABLED, a result of "stencil test passed" is propagated down the pipeline.

Depth Test

The Depth Test function can be used to discard pixels based on a comparison between the incoming pixel's depth value and the current depth buffer value associated with the pixel. This function is typically used to perform the "Z Buffer" hidden surface removal. The result of this pipeline function is used in the Stencil Buffer Update function later in the pipeline.

This function is enabled by the **Depth Test Enable** state variable. If enabled, the pixel's ("source") depth value is first computed. After computation the pixel's depth value is clamped to the range defined by **Minimum Depth** and **Maximum Depth** in the selected CC_VIEWPORT state. Then the current ("destination") depth buffer value for this pixel is read.

This function then compares the source and destination depth values. The comparison performed is specified by the **Depth Test Function** state variable.

The result of the comparison is propogated down the pipeline for use in the subsequent Depth Buffer Update function. The Depth Test function does not in itself discard pixels.

If **Depth Test Enable** is DISABLED, a result of "depth test passed" is propagated down the pipeline.

Programming Note:

- Enabling the Depth Test function without defining a Depth Buffer is UNDEFINED.

Pre-Blend Color Clamping

Pre-Blend Color Clamping, controlled via **Pre-Blend Color Clamp Enable** OR **Pre-Blend Source Only Clamp Enable** and **Color Clamp Range** states in COLOR_CALC_STATE, is affected by the enabling of Color Buffer Blend as described below.

The following table summarizes the requirements involved with Pre-/Post-Blend Color Clamping.

Programming Note	
Context:	Negative Values
Errata - Negative values on Unsigned Float channels are always clamped to 0 if blending is enabled, regardless of how pre-blend clamping is programmed.	

Programming Note	
Context:	errata
Errata : If the following conditions are true, blend is optimized to a fill operation in hardware which may cause corruption to occur if the pixel value is out of the RT format range. This is due to both pre-blend and post-blend clamping being skipped under these conditions which results in the unclamped value not getting down converted to RT format. RT format = r10g10b10_float_a2_unorm Blend State.Blend Enable = true Pre-Blend clamp enable = 0 Source blend factor is programmed such that it resolves to 1.0f and Destination blend factor is programmed such that it resolves to 0.0f (blend is converted to fill)	

Blending	Pre-Blend Color Clamp	Clamp Range
Off	Disabled: clamp to RT range (1)	Must set range = RT range (except if Pre Blend Source Only Clamp Enable is set)
	Enabled: clamp to RT range (1)	Must set range = RT range
On (if permitted)	Disabled: clamp to internal format (1) for float RTs and to RT range for UNORM/SNORM RTs	Must set range = RT range
	Enabled: clamp to RT range (1)	Must set range = RT range

(1) If Pre Blend Source Only Clamp Enable is set in BLEND STATE, SourceColor is clamped to COLORCLAMP_UNORM

Values written to a render target are always clamped to the RT range.

Pre-Blend Color Clamping When Blending is Disabled

The clamping of source color components is controlled by **Pre-Blend Color Clamp Enable**. If ENABLED, all source color components are clamped to the range specified by **Color Clamp Range**. If DISABLED, no clamping is performed.

Programming Note	
Context:	Pre-Blend Color Clamping When Blending is Disabled

- Given the possibility of writing UNPREDICTABLE values to the Color Buffer, it is expected and highly recommended that, when blending is disabled, software set **Pre-Blend Color Clamp Enable** to ENABLED and select an appropriate **Color Clamp Range**.
- When using SINT or UINT rendertarget surface formats, **Blending must** be DISABLED. The **Pre-Blend Color Clamp Enable** and **Color Clamp Range** fields are ignored, and an implied clamp to the rendertarget surface format is performed.

Pre-Blend Color Clamping When Blending is Enabled

The clamping of source, destination and constant color components is controlled by **Pre-Blend Color Clamp Enable**. If ENABLED, all these color components are clamped to the range specified by **Color Clamp Range**. If DISABLED, no clamping is performed on these color components prior to blending.

Color Buffer Blending

The Color Buffer Blending function is used to combine one or two incoming "source" pixel color+alpha values with the "destination" color+alpha read from the corresponding location in a RenderTarget.

Blending is enabled on a global basis by the **Color Buffer Blend Enable** state variable (in COLOR_CALC_STATE). If DISABLED, Blending and Post-Blend Clamp functions are disabled for all RenderTargets, and the pixel values (possibly subject to Pre-Blend Clamp) are passed through unchanged.

The Color Buffer Blend Enable is in the per-render-target BLEND_STATE, and the field in SURFACE_STATE is no longer supported.

Programming Note	
Context:	Color Buffer Blending, Logic Ops, DataPort, surface formats, render targets

- Color Buffer Blending and Logic Ops must not be enabled simultaneously, or behavior is UNDEFINED.
- Dual source blending: The DataPort only supports dual source blending with a SIMD8-style message.
- Only certain surface formats support Color Buffer Blending. Refer to the Surface Format tables in *Sampling Engine*. Blending must be disabled on a RenderTarget if blending is not supported.

The incoming "source" pixel values are modulated by a selected "source" blend factor, and the possibly gamma-decorrected "destination" values are modulated by a "destination" blend factor. These terms are then combined with a "blend function". In general:

src_term = src_blend_factor * src_color

dst_term = dst_blend_factor * dst_color

color output = blend_function(src_term, dst_term)

If there is no alpha value contained in the Color Buffer, a default value of 1.0 is used and, correspondingly, there is no alpha component computed by this function.

Dual Source Blending: When using "Dual Source" Render Target Write messages, the Source1 pixel color+alpha passed in the message can be selected as a src/dst blend factor (see "Color Buffer Blend Color Factors"). In single-source mode, those blend factor selections are invalid. If SRC1 is included in a src/dst blend factor and a DualSource RT Write message is not used, results are UNDEFINED. (This reflects the same restriction in DX APIs, where undefined results are produced if "o1" is not written by a PS - there are no default values defined). If SRC1 is not included in a src/dst blend factor, dual source blending must be disabled.

The blending of the color and alpha components is controlled with two separate (color and alpha) sets of state variables. However, if the **Independent Alpha Blend Enable** state variable in COLOR_CALC_STATE is DISABLED, then the "color" (rather than "alpha") set of state variables is used for both color and alpha. Note that this is the only use of the **Independent Alpha Blend Enable** state - it does not control whether Blending occurs, only how.

Per **Render Target Blend State**: Blend state is selected based on **Render Target Index** contained in the message header, and appropriate blend state is applied to Render Target Write messages.

The following table describes the color source and destination blend factors controlled by the **Source [Alpha] Blend Factor** and **Destination [Alpha] Blend Factor** state variables in COLOR_CALC_STATE. Note that the blend factors applied to the R,G,B channels are always controlled by the **Source/Destination Blend Factor**, while the blend factor applied to the alpha channel is controlled either by **Source/Destination Blend Factor** or **Source/Destination Alpha Blend Factor**.

Color Buffer Blend Color Factors

Blend Factor Selection	Blend Factor Applied for R,G,B,A channels (oN = output from PS to RT#N) (o1 = 2nd output from PS in Dual-Souce mode only) (rtN = destination color from RT#N) (CC = Constant Color)
BLENDFACTOR_ZERO	0.0, 0.0, 0.0, 0.0
BLENDFACTOR_ONE	1.0, 1.0, 1.0, 1.0
BLENDFACTOR_SRC_COLOR	oN.r, oN.g, oN.b, oN.a
BLENDFACTOR_INV_SRC_COLOR	1.0-oN.r, 1.0-oN.g, 1.0-oN.b, 1.0-oN.a
BLENDFACTOR_SRC_ALPHA	oN.a, oN.a, oN.a, oN.a
BLENDFACTOR_INV_SRC_ALPHA	1.0-oN.a, 1.0-oN.a, 1.0-oN.a, 1.0-oN.a
BLENDFACTOR_SRC1_COLOR	o1.r, o1.g, o1.b, o1.a
BLENDFACTOR_INV_SRC1_COLOR	1.0-o1.r, 1.0-o1.g, 1.0-o1.b, 1.0-o1.a
BLENDFACTOR_SRC1_ALPHA	o1.a, o1.a, o1.a, o1.a

Blend Factor Selection	Blend Factor Applied for R,G,B,A channels (oN = output from PS to RT#N) (o1 = 2nd output from PS in Dual-Souce mode only) (rtN = destination color from RT#N) (CC = Constant Color)
BLENDFACTOR_INV_SRC1_ALPHA	1.0-o1.a, 1.0-o1.a, 1.0-o1.a, 1.0-o1.a
BLENDFACTOR_DST_COLOR	rtN.r, rtN.g, rtN.b, rtN.a
BLENDFACTOR_INV_DST_COLOR	1.0-rtN.r, 1.0-rtN.g, 1.0-rtN.b, 1.0-rtN.a
BLENDFACTOR_DST_ALPHA	rtN.a, rtN.a, rtN.a, rtN.a
BLENDFACTOR_INV_DST_ALPHA	1.0-rtN.a, 1.0-rtN.a, 1.0-rtN.a, 1.0-rtN.a
BLENDFACTOR_CONST_COLOR	CC.r, CC.g, CC.b, CC.a
BLENDFACTOR_INV_CONST_COLOR	1.0-CC.r, 1.0-CC.g, 1.0-CC.b, 1.0-CC.a
BLENDFACTOR_CONST_ALPHA	CC.a, CC.a, CC.a, CC.a
BLENDFACTOR_INV_CONST_ALPHA	1.0-CC.a, 1.0-CC.a, 1.0-CC.a, 1.0-CC.a
BLENDFACTOR_SRC_ALPHA_SATURATE	f,f,f,1.0 where f = min(1.0 - rtN.a, oN.a)

The following table lists the supported blending operations defined by the **Color Blend Function** state variable and the **Alpha Blend Function** state variable (when in independent alpha blend mode).

Color Buffer Blend Functions

Blend Function	Operation (for each color component)
BLENDFUNCTION_ADD	SrcColor*SrcFactor + DstColor*DstFactor
BLENDFUNCTION_SUBTRACT	SrcColor*SrcFactor - DstColor*DstFactor
BLENDFUNCTION_REVERSE_SUBTRACT	DstColor*DstFactor - SrcColor*SrcFactor
BLENDFUNCTION_MIN	min (SrcColor*SrcFactor, DstColor*DstFactor) **Programming Note:** This is a superset of the OpenGL "min" function.
BLENDFUNCTION_MAX	max (SrcColor*SrcFactor, DstColor*DstFactor) **Programming Note:** This is a superset of the OpenGL "max" function.

Post-Blend Color Clamping

(See *Pre-Blend Color Clamping* above for a summary table regarding clamping)

Post-Blend Color clamping is available only if Blending is enabled.

If Blending is enabled, the clamping of blending output color components is controlled by **Post-Blend Color Clamp Enable**. If ENABLED, the color components output from blending are clamped to the range specified by **Color Clamp Range**. If DISABLED, no clamping is performed at this point.

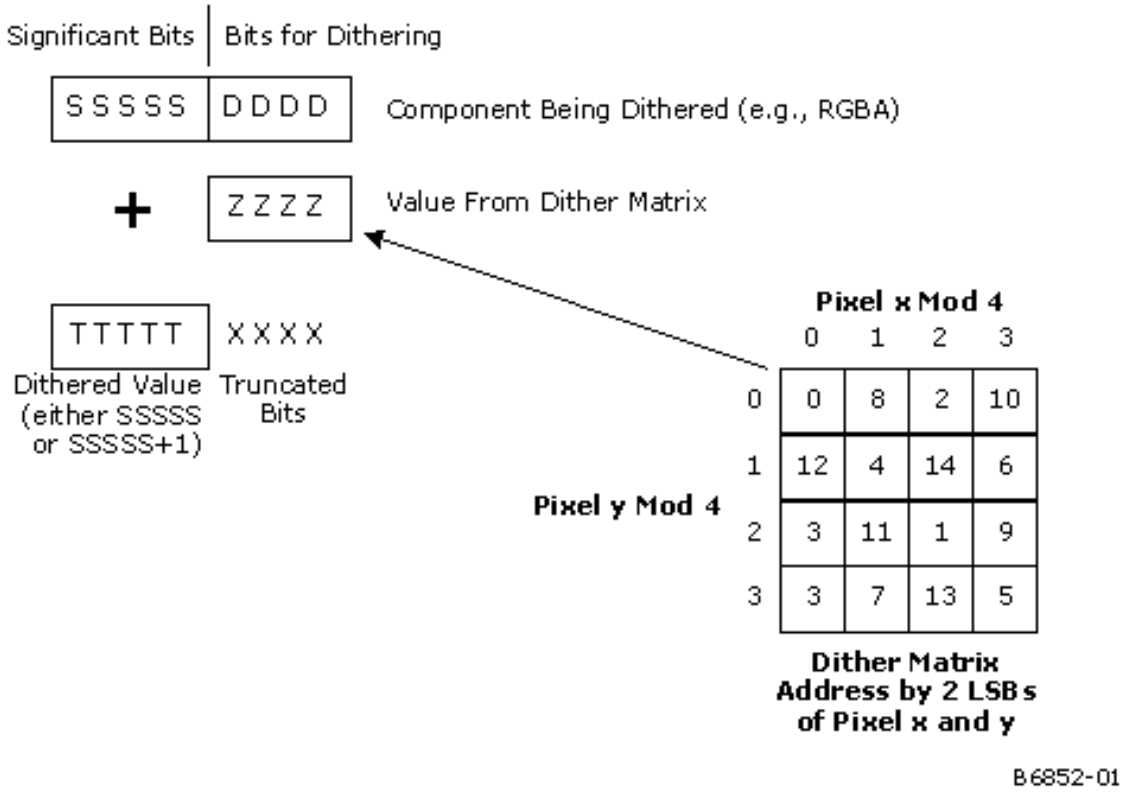
Regardless of the setting of **Post-Blend Color Clamp Enable**, when Blending is enabled color components will be automatically clamped to (at least) the rendertarget surface format range at this stage of the pipeline.

Dithering

Dithering is used to give the illusion of a higher resolution when using low-bpp channels in color buffers (e.g., with 16bpp color buffer). By carefully choosing an arrangement of lower resolution colors, colors otherwise not representable can be approximated, especially when seen at a distance where the viewer's eyes will average adjacent pixel colors. Color dithering tends to diffuse the sharp color bands seen on smooth-shaded objects.

A four-bit dither value is obtained from a 4x4 Dither Constant matrix depending on the pixel's X and Y screen coordinate. The pixel's X and Y screen coordinates are first offset by the **Dither Offset X** and **Dither Offset Y** state variables (these offsets are used to provide window-relative dithering). Then the two LSBs of the pixel's screen X coordinate are used to address a column in the dither matrix, and the two LSBs of the pixel's screen Y coordinate are used to address a row. This way, the matrix repeats every four pixels in both directions.

The value obtained is appropriately shifted to align with (what would be otherwise) truncated bits of the component being dithered. It is then added with the component and the result is truncated to the bit depth of the component given the color buffer format.



Dithering Process (5-Bit Example)

Logic Ops

The Logic Ops function is used to combine the incoming "source" pixel color/alpha values with the corresponding "destination" color/alpha contained in the ColorBuffer, using a logic function.

The Logic Op function is enabled by the **LogicOp Enable** state variable. If DISABLED, this function is ignored and the incoming pixel values are passed through unchanged.

Programming Notes

Programming Note
Color Buffer Blending and Logic Ops must not be enabled simultaneously, or behavior is UNDEFINED.
Logic Ops are supported on all blendable render targets and render targets with *INT formats.

The following table lists the supported logic ops. The logic op is selected using the **Logic Op Function** field in COLOR_CALC_STATE.

Logic Ops

LogicOp Function	Definition (S=Source, D=Destination)
LOGICOP_CLEAR	all 0's
LOGICOP_NOR	NOT (S OR D)
LOGICOP_AND_INVERTED	(NOT S) AND D
LOGICOP_COPY_INVERTED	NOT S
LOGICOP_AND_REVERSE	S AND NOT D
LOGICOP_INVERT	NOT D
LOGICOP_XOR	S XOR D
LOGICOP_NAND	NOT (S AND D)
LOGICOP_AND	S AND D
LOGICOP_EQUIV	NOT (S XOR D)
LOGICOP_NOOP	D
LOGICOP_OR_INVERTED	(NOT S) OR D
LOGICOP_COPY	S
LOGICOP_OR_REVERSE	S OR NOT D
LOGICOP_OR	S OR D
LOGICOP_SET	all 1's

Buffer Update

The Buffer Update function is responsible for updating the pixel's Stencil, Depth and Color Buffer contents based upon the results of the Stencil and Depth Test functions. Note that Kill Pixel and/or Alpha Test functions may have already discarded the pixel by this point.

Stencil Buffer Updates

If and only if stencil testing is enabled, the Stencil Buffer is updated according to the **Stencil Fail Op**, **Stencil Pass Depth Fail Op**, and **Stencil Pass Depth Pass Op** state (or their backface counterparts if **Double Sided Stencil Enable** is ENABLED and the pixel is from a back-facing object) and the results of the Stencil Test and Depth Test functions.

Stencil Fail Op and **Backface Stencil Fail Op** specify how/if the stencil buffer is modified if the stencil test fails. **Stencil Pass Depth Fail Op** and **Backface Stencil Pass Depth Fail Op** specify how/if the stencil buffer is modified if the stencil test passes but the depth test fails. **Stencil Pass Depth Pass Op** and **Backface Stencil Pass Depth Pass Op** specify how/if the stencil buffer is modified if both the stencil and depth tests pass. The operations (on the stencil buffer) that are to be performed under one of these (mutually exclusive) conditions is summarized in the following table.

Stencil Buffer Operations

Stencil Operation	Description
STENCILOP_KEEP	Do not modify the stencil buffer
STENCILOP_ZERO	Store a 0
STENCILOP_REPLACE	Store the *StencilTestReference* reference value
STENCILOP_INCRSAT	Saturating increment (clamp to max value)
STENCILOP_DECRSAT	Saturating decrement (clamp to 0)
STENCILOP_INCR	Increment (possible wrap around to 0)
STENCILOP_DECR	Decrement (possible wrap to max value)
STENCILOP_INVERT	Logically invert the stencil value

Any and all writes to the stencil portion of the depth buffer are enabled by the **Stencil Buffer Write Enable** state variable.

When writes are enabled, the **Stencil Buffer Write Mask** and **Backface Stencil Buffer Write Mask** state variables provide an 8-bit mask that selects which bits of the stencil write value are modified. Masked-off bits (i.e., mask bit == 0) are left unmodified in the Stencil Buffer.

Programming Note	
Context:	Stencil Buffer Updates
The Stencil Buffer can be written even if depth buffer writes are disabled via Depth Buffer Write Enable	

Depth Buffer Updates

Any and all writes to the Depth Buffer are enabled by the **Depth Buffer Write Enable** state variable. If there is no Depth Buffer, writes must be explicitly disabled with this state variable, or operation is UNDEFINED.

If depth testing is disabled or the depth test passed, the incoming pixel's depth value is written to the Depth Buffer. If depth testing is enabled and the depth test failed, the pixel is discarded - with no modification to the Depth or Color Buffers (though the Stencil Buffer may have been modified).

Color Gamma Correction

Computed RGB (not A) channels can be gamma-corrected prior to update of the Color Buffer.

This function is automatically invoked whenever the destination surface (render target) has an SRGB format (see surface formats in *Sampling Engine*). For these surfaces, the computed RGB values are converted from gamma=1.0 space to gamma=2.4 space by applying a ^(2.4) exponential function.

Color Buffer Updates

Finally, if the pixel has not been discarded by this point, the incoming pixel color is written into the Color Buffer. The **Surface Format** of the color buffer indicates which channel(s) are written (e.g., R8G8_UNORM are written with the Red and Green channels only). The **Color Buffer Component Write Disables** from the Color Buffer's SURFACE_STATE provide an independent write disable for each channel of the Color Buffer.

Pixel Pipeline State Summary

COLOR_CALC_STATE

3DSTATE_BLEND_STATE_POINTERS

3DSTATE_BLEND_STATE_POINTERS

3DSTATE_DEPTH_STENCIL_STATE_POINTERS

3DSTATE_DEPTH_STENCIL_STATE_POINTERS has been replaced by 3DSTATE_WM_DEPTH_STENCIL. (See Vol2a.11 3D Pipeline - Windower for details.)

COLOR_CALC_STATE

COLOR_CALC_STATE

DEPTH_STENCIL_STATE

DEPTH_STENCIL_STATE has been replaced by 3DSTATE_WM_DEPTH_STENCIL. (See *3D Pipeline - Windower* for details).

BLEND_STATE

BLEND_STATE

Signal	CC_INT::AlphaTestEnable

Description	AlphaTestEnable
Formula	= BLEND_STATE::AlphaTestEnable && !3DSTATE_WM_HZ_OP::DepthBufferResolveEnable && !3DSTATE_WM_HZ_OP::DepthBufferClear && !3DSTATE_WM_HZ_OP::StencilBufferClear

Signal	CC_INT::AlphaToCoverageEnable
Description	AlphaToCoverageEnable
Formula	= BLEND_STATE::AlphaToCoverageEnable && !3DSTATE_PS_EXTRA::PixelShaderDisableAlphaToCoverage

CC_VIEWPORT

CC_VIEWPORT

Other Pixel Pipeline Functions

Statistics Gathering

If **Statistics Enable** is set in 3DSTATE_WM, the PS_DEPTH_COUNT register (see Memory Interface Registers in Volume 1a, *GPU Overview*) is incremented once for each pixel (or sample) that passes the depth, stencil and alpha tests. Note that each of these tests is treated as passing if disabled. This count is accurate regardless of whether **Early Depth Test Enable** is set. To obtain the value from this register at a deterministic place in the primitive stream without flushing the pipeline, however, the PIPE_CONTROL command must be used. See Volume 2a, *3D Pipeline*, for details on PIPE_CONTROL.

GPGPU Compute Pipeline

This section discusses the programming the General Purpose GPU Pipeline, including:

- how GPGPU and Media threads are dispatched,
- thread resource management,
- how thread groups are tracked, and
- preemption and context switching.

General Purpose Compute Model

The General Purpose (GP) compute model consists of a host connected to one or more compute devices. Each compute device consists of several Processing Elements (PEs), also known as Execution Units (EUs). Applications are then architected as a combination of host software (per the host framework), and kernels that are submitted by the host to run on the PEs with a pre-defined decoupling point.

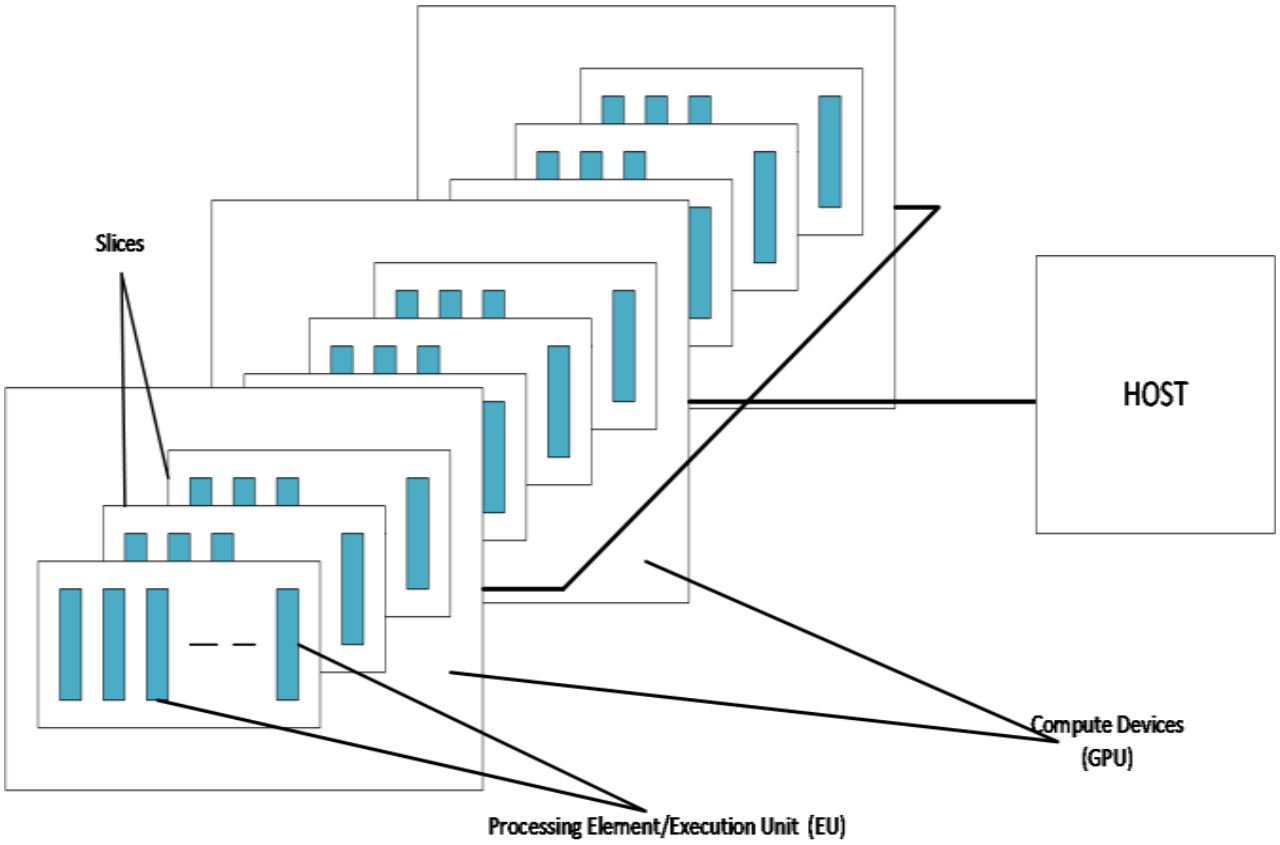


Figure: General Purpose Compute Model

The GP compute architecture contains two distinct units of execution: a host program and a set of kernels that execute within the context set by the host. The host interacts with these kernels through a command queue. Each device may have its own command queue. When a command is submitted into the command queue, the command is checked for dependencies and then executed on a PE inside the compute unit clusters. Once the command has finished executing, the kernel communicates an end of life cycle through "end of thread" message.

The GP execution model determines how to schedule and execute the kernels. When a kernel-enqueue command submits a kernel for execution, the command defines an index space or N-dimensional range. A kernel-instance consists of the kernel, the argument values associated with the kernel, and the parameters that define the index space. When a compute device executes a kernel-instance, the kernel function executes for each point in the defined index space or N-dimensional range.

An executing kernel function is called a work-item, and a collection of these work-items is called a work-group. A compute device manages work-items using work-groups. Individual work-items are identified by either a global ID, or a combination of the work-group ID and a local ID inside the work-group.

The work-group concept, which essentially runs the same kernel on several unit items in a group, captures the essence of data parallel computing. The PEs can organize work-items in vector format and run the same kernel on the vector, hence speeding up the compute for all such applications.

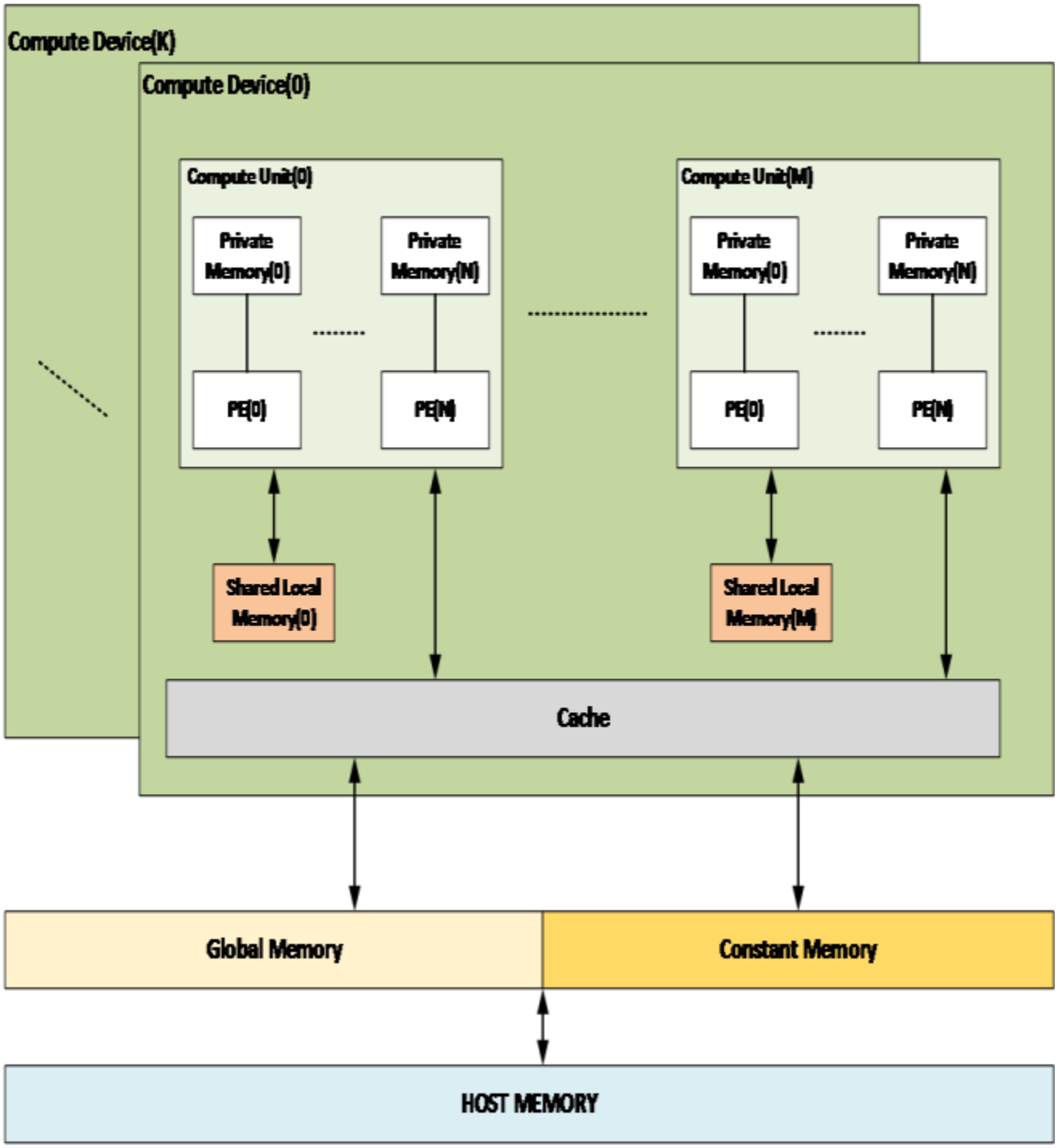
A device can compute each work-group in any arbitrary order. Also, the work-items within a single work-group execute concurrently, with no guarantee on the order of progress. A high level work-group function, like Barriers, applies to each work-item in a work-group, to facilitate the required synchronization points. Such a work-group function must be defined so that all work-items in the work-group encounter precisely the same work-group function.

Synchronization can also occur at the command level, where the synchronization can happen between commands in host command-queues. In this mode, one command can depend on execution points in another command or multiple commands.

Other types of synchronization based on memory-order constraints inside a program include Atomics and Fences. These synchronization types control how a memory operation of any particular work-item is made visible to another, which offers micro-level synchronization points in the data-parallel compute model.

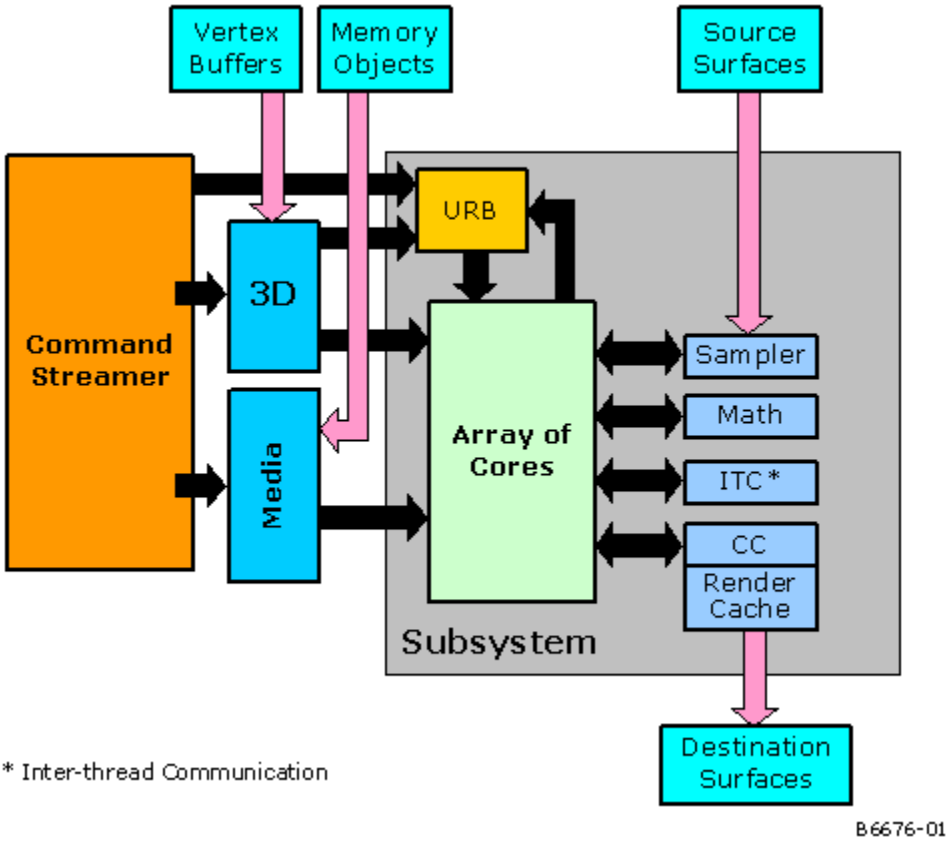
The memory model for a general-purpose engine is partitioned into host-side memory and device-side memory, using Shared Virtual Memory (SVM) to move objects between the two sides. The device-side memory is further divided into four distinct disjoint regions as defined below:

- Global Memory: Memory region accessible to all work-items within a context.
- Constant Memory: Memory region that remains constant during kernel execution.
- Local Memory: Memory region local and exclusive to a work-group (also called SLM).
- Private memory: Memory region private to a work-item.


Compute Device(K)
Compute Device(0)
Compute Unit(0)
Private Memory(0)
Private Memory(N)
PE(0)
PE(N)
Compute Unit(M)
Private Memory(0)
Private Memory(N)
PE(0)
PE(N)
Shared Local Memory(0)
Shared Local Memory(M)
Cache
Global Memory
Constant Memory
HOST MEMORY

GPGPU Context in GPU Hardware

The following block diagram provides an overview of the GPU Hardware. The media pipeline and the 3D fixed function pipeline are positioned in parallel. Concurrency of the media pipeline and the 3D pipeline is not supported. At any given time, only one of the two pipelines can be activated. A programmer can switch between the two pipelines within a single context by using the MI_PIPELINE_SELECT command.



The media pipeline provides media functions and has media-specific fixed function capability. These fixed functions are able to control shared functions and resources, feed generic threads to the Execution Units (EUs), and interact with generic threads during runtime. Since a programmer can use the media pipeline for non-media applications, the media pipeline can also be called the general purpose pipeline. Henceforth, we refer to this fixed function pipeline as either the media or GPGPU pipeline, keeping in mind its general-purpose capability.

The GPU subsystem consists of an array of EUs (also called an array of cores) and a set of shared functions outside the EUs. The EUs leverage these shared functions for I/O and dedicated computations. Programmers access the GPU subsystem via the 3D or media pipelines.

EUs are general-purpose programmable cores that support a rich instruction set. This instruction set has been optimized to support various 3D API shader languages, media functions processing, and compute kernels.

Shared functions are dedicated hardware accelerator units that provide specialized supplemental functionality for the EUs, to run filtered load, scatter, gather, and filter/blended store operations.

Programming the GPGPU Pipeline

The commands for either Programmable Media or GPGPU contexts uses mostly the same commands, with a few sequence differences as described in this section.

The Programmable Media Pipeline is programmed using command sequences. Media hardware threads are created through the parameterized media walker. Dispatch of a thread is controlled by a scoreboard mechanism.

The media pipeline can now use pipelined state changes, via the added MEDIA_STATE_FLUSH command. The MEDIA_STATE_FLUSH serves as a fence for state change by flushing the GPGPU Pipeline without waiting for threads to retire.

Note that media pipeline is the main user of the CURBE command. Media state commands deliver the interface descriptors directly, instead of loading them through indirect state.

Thread Lifecycle

Once a thread dispatcher (TS) selects a thread, the thread lifecycle is as follows:

- TS forwards the interface descriptor pointer to the L1 interface descriptor cache - a small fully associated cache containing up to four interface descriptors. If the interface descriptor is not found in the L1 interface descriptor cache, a corresponding request is forwarded to the L2 cache. Interface descriptors return back to TS in requested order.
- Once TS receives an interface descriptor, it checks whether the maximum concurrent thread number has been reached to determine whether to make a thread dispatch request or to stall the request until another thread retires. If the thread requests the use of scratch memory, TS also generates a pointer into the scratch space.
- TS then builds the transparent header and the R0 header, and TS makes a thread request to the thread dispatcher.
- TS keeps track of each dispatched thread, and monitors messages from each thread, including resource dereference and/or thread termination. When TS receives a thread termination message, it can recover the scratch space and thread slot allocated to the specified thread. The URB handle for a terminated thread may also be dereferenced for future reuse. Note that URB handle dereference may occur before a thread terminates. For more details, see the Media Message section.

The following flowchart outlines the thread lifecycle process.

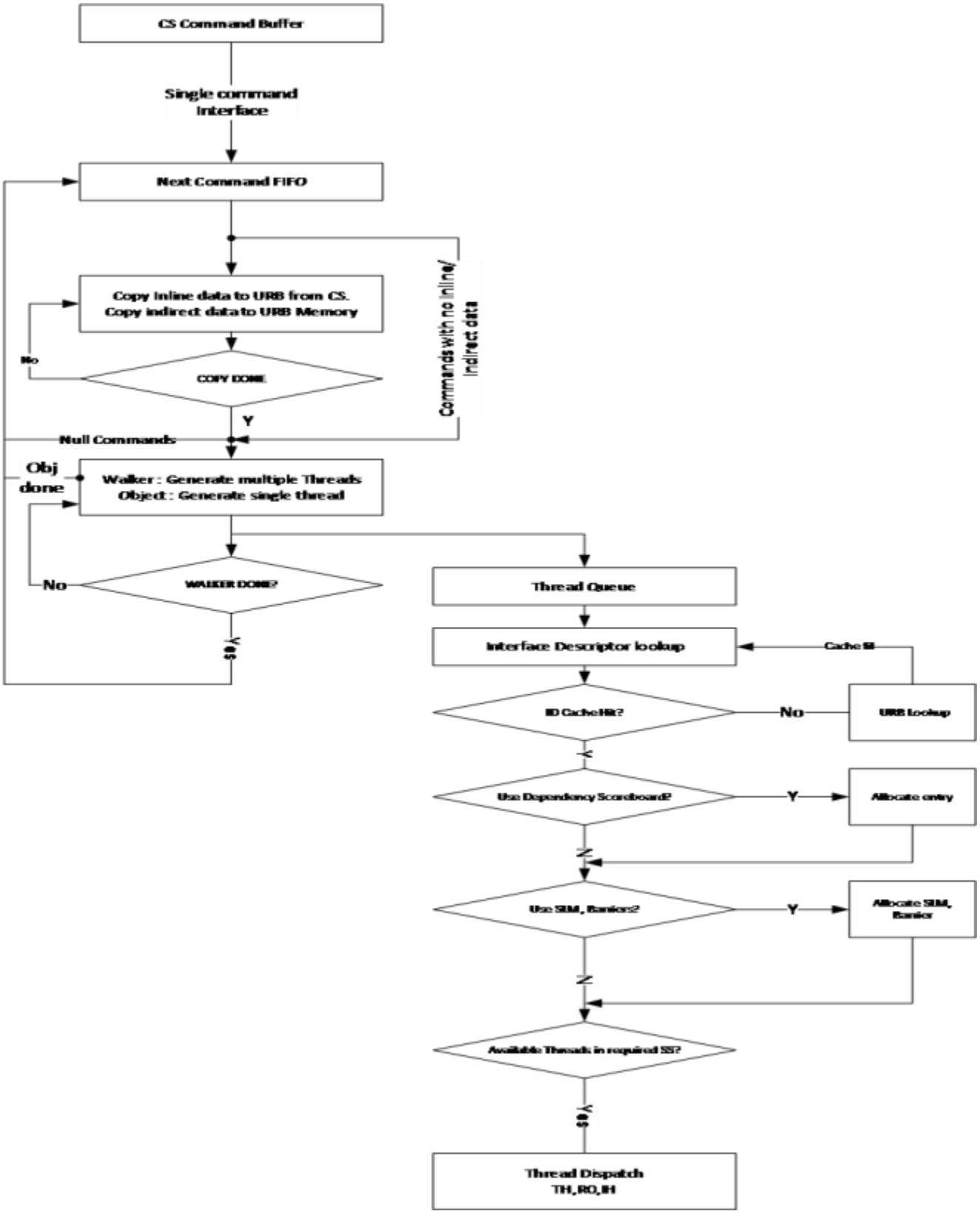


Figure: Thread Lifecycle Process

Command Sequence

The Command Sequence programming model for the GPGPU Pipeline (VFE/TSG) requires the following steps. Some of the steps listed are optional, but the order shown must be strictly followed. Some usage restrictions are emphasized for descriptive purposes. For more details, refer to the definitions of each command.

Step 1: MI_FLUSH/PIPE_CONTROL

- This step is mandatory.
- Each command in this step can be used multiple times, but that is not recommended for performance reasons.

Step 2: State command PIPELINE_SELECT

- This step is optional if it is known that, within the same context, the media pipeline was selected before Step 1.
- The command in this step can be used multiple time, but that is not recommended for performance reasons.

Step 3: State commands configuring pipeline states

STATE_BASE_ADDRESS:

- This command is mandatory for this step.
- Multiple instances of this command are allowed. The last instance overwrites previous ones.
- This command must precede any other state commands mentioned below.

The fields Indirect Object Base Address and Indirect Object Access Upper Bound are used to control indirect Media object load in VFE. The fields Dynamics Base Address and Dynamics Base Access Upper Bound are used to control indirect CURBE and interface descriptor object load in VFE.

On receiving the STATE_BASE_ADDRESS command, VFE stores the 48-bit Dynamics Base Address, Dynamics Base Access Upper Bound, Indirect Object Base Address, and Indirect Object Access Upper Bound in state registers. These values are used to calculate the fetch addresses for CURBE, Interface Descriptor, and Indirect Data respectively.

- Note that the STATE_BASE_ADDRESS command may be inserted before and after any commands listed in Steps 1 and 2 above. For example, STATE_BASE_ADDRESS may be placed in the ring buffer while other commands are put in a batch buffer.

STATE_SIP:

- This command is optional for this step, unless the kernels use SIP.

MEDIA_VFE_STATE:

- This command is mandatory for this step.

- This command destroys all outstanding URB handles in the system, then generates a new set of URB handles based on state parameters, number of URBs, and URB length from the VFE FF state.
- On receiving a MEDIA_VFE_STATE command, VFE allocates the number of handles specified by Number of URB Entries, with each handle size as specified in URB Entry Allocation Size, starting at CURBE Allocation Size + 64. These handles are stored in UEM.
- MEDIA_VFE_STATE also stores other state information, including Scratch Space, Gateway, Scoreboard, and Max Thread limit
- Multiple instances of the MEDIA_VFE_STATE command are allowed. The last instance overwrites previous ones.

MEDIA_CURBE_LOAD:

- This command is optional for this step.
- Multiple instances of this command are allowed. The last instance overwrites previous ones.
- On receiving a MEDIA_CURBE_LOAD command, VFE fetches the CURBE data from CURBE Data Start Address by reading CURBE Total Data Length bytes. The Address is defined relative to the Dynamics Base Address programmed through STATE_BASE_ADDRESS command.
- VFE breaks the CURBE Total Data Length into 64-byte chunks, and sends request through GAFS. The fetched data is written directly into URB at the address specified in the tag of the request. This address is set at 64. VFE tracks this fetch through its local up/down counter.

MEDIA_INTERFACE_DESCRIPTOR_LOAD:

- This command is mandatory for this step.
- Multiple instances of this command are allowed. The last instance overwrites previous ones.
- On receiving a MEDIA_INTERFACE_DESCRIPTOR_LOAD command, VFE fetches the Interface Descriptors from the Interface Descriptor Data Start Address by reading Interface Descriptor Total Length bytes. The Address is defined relative to the Dynamics Base Address, programmed through the STATE_BASE_ADDRESS command.
- VFE breaks the Interface Descriptor Total Length into 64 byte chunks and sends a request to GAFS. The fetched data is directly written into URB at the address specified in the tag of the request. That address is set as zero. VFE tracks this fetch through its local up/down counter.

Step 4: Primitive commands

This step must follow previous steps, which prepare the Media/GPU pipeline for the execution of primitive commands.

MEDIA_OBJECT/ MEDIA_OBJECT_GRPIP:

- You can issue this command multiple times, to continue processing media primitives.

On receiving a MEDIA_OBJECT/MEDIA_OBJECT_GRPID command, VFE fetches indirect data from Indirect Data Start Address by reading Indirect Data Length bytes. The Address is defined relative to the Indirect Object Base Address, programmed through the STATE_BASE_ADDRESS command.

VFE breaks the Indirect Data Length into 64-byte chunks and sends the request to GAFS. The fetched data is written directly into URB at the address specified in the request tag. This address is set as the handle address + inline data length (found from csdwlengthr [] - cmd_length), rounded up to the next 32-byte boundary. VFE tracks this fetch through its local up/down counter.

- If inline or indirect data is present in the command, the URB handle is allocated from UEM.
- Upon receiving this command, VFE stores the Media Object information to pass to the TSG unit, including the Interface Descriptor Offset, Child Present, Thread Synchronization, Scoreboard, and GRPID (for MEDIA_OBJECT_GRPID command).
- The command writes inline data at the starting URB handle address through the GAFS interface. If the inline data in not a multiple of 32-bytes, padding is added to the next 32-byte boundary.
- When the transfer of inline and indirect data is complete, VFE sends a thread dispatch signal to TSG.

GPGPU Pipeline Notes

- Set up the interface descriptor with the number of threads in barrier. The barrier ID is not specified here, because hardware can automatically assign barriers to available thread groups. The amount of CURBE data to deliver per thread dispatch is set in the interface descriptor.
- Set up CURBE with thread IDs and common data for all thread dispatches in the thread group.
- Set up a GPGPU_WALKER command with the thread group IDs to dispatch the threads. Each thread dispatch in the thread group sends CURBE data in sections; a new thread group starts sending the CURBE data from the beginning of the buffer.
- Place a MEDIA_STATE_FLUSH right before the MI_BATCH_BUFFER_END of any batch buffer that uses MEDIA_OBJECT.
- The kernel should handle barriers as follows:
    - The BarrierMsg contains the barrier ID and a way to reprogram the barrier count.
    - The barrier count reprogram is not normally used for GPGPU workloads.
    - When all threads in the group have reached the barrier, the gateway returns a notification bit 0.
    - The kernel must wait for the barrier to finish with a WAIT N0.

CURBE/Indirect Payload Dispatch

Example of a CURBE command sequence:

- MEDIA_STATE_FLUSH // Make sure dispatch is not accessing CURBE
- MEDIA_CURBE_LOAD
- GPGPU_WALKER

The GPGPU commands extend to allow indirect input as an alternative to CURBE using the exact same mechanism: the same offset specifies what data is delivered to all threads, and a count specifies how much data is delivered per thread. A single indirect pointer points to both the Cross-Thread and Per-Thread Constant Data, which are stored in the URB. The positions of the cross-thread and per-thread constants are swapped in the EU GRF.

To use indirect payload storage, the product (URB entry allocation size * number of URB entries) must be enough to cover the sum of the cross-thread and per-thread indirect data to be loaded. The MEDIA_VFE_STATE command sets these state variables. The URB entry allocation must be equal to or greater than the indirect data length in GPGPU_WALKER. Use the maximum number of URB entries that fills the URB space, to avoid a shortage of handles that could cause a performance problem.

Indirect input is defined as follows:

- Indirect uses a 32-bit memory pointer for the start address (compared to 64 bit for CURBE).
- Indirect does not use the constant URB entry read offset.
- Indirect multiplies the cross-thread constant data read length and constant entry read length by 32 to convert them into bytes.
- The X/Y/Z payload in the EU GRF comes before the cross-thread constant data.

Media GPGPU Payload Limitations

The Media/GPGPU instructions can have three types of payloads, but not all combinations are allowed. The following table lists the legal combinations.

Table: Media GPGPU Payload Limitations

Workload	Commands	Data Stored
GPGPU	GPGPU_WALKER	CURBE
	GPGPU_WALKER	INDIRECT
Media (Legacy)	Media_Object	CURBE
	Media_Object	INDIRECT
	Media_Object	INLINE
	Media_Object	CURBE+INLINE
	Media_Object	CURBE+INDIRECT
	Media_Object	INLINE+INDIRECT

	Media_Object	CURBE+ INLINE+INDIRECT
	Media_Object_Walker	CURBE
	Media_Object_Walker	INLINE
	Media_Object_Walker	CURBE+INLINE
	Media_Object_Walker	INDIRECT
Media using Barrier/SLM	Media_Object_GRPID	CURBE
	Media_Object_GRPID	INDIRECT
	Media_Object_GRPID	INLINE
	Media_Object_Walker (with group id)	CURBE
	Media_Object_Walker (with group id)	INLINE
	Media_Object_Walker (with group id)	INDIRECT

The Dynamic State MOCS, specified in **STATE_BASE_ADDRESS**, fetch indirect and CURBE payloads from memory during thread dispatch.

Media State Model

The media state model is based on an in-line state load mechanism. State commands load VFE state, URB configuration, and Interface Descriptors to VFE hardware.

All Interface Descriptors have the same size and are organized as a contiguous array in memory. The Interface Descriptor Index can select these descriptors for a given kernel, which allows different kinds of kernels to coexist in the system.

The state-command-related headers are as follows.

Pipeline (Media) Bits[28:27]	Opcode Bits[26:24]	Sub Opcode Bits[23:16]	Command
2h	0h	00h	MEDIA_VFE_STATE
2h	0h	01h	MEDIA_CURBE_LOAD
2h	0h	02h	MEDIA_INTERFACE_DESCRIPTOR_LOAD

Commands for GPGPU Pipe

This section lists and explains the various commands that VFE executes for media and GPGPU context on render pipe, including several commands that are shared with media mode. There are three types of VFE commands:

- Null command - may write URB, but does not place any request to the TSG unit.
- Object command - may write URB, and then place a single request to the TSG unit.
- Walker command - may write URB, and then trigger VFE_WALKER to generate multiple requests to the TSG unit.

Table: VFE Commands

CmdID	Command	Command type
1	MEDIA_CURBE_LOAD	null command
2	MEDIA_INTERFACE_DESCRIPTOR_LOAD	null command
3	MEDIA_VFE_STATE	null command
3	MEDIA_STATE_FLUSH[1]	object command
4	MEDIA_OBJECT	object command
5	MEDIA_OBJECT_GRPID	object command
7	MEDIA_WALKER	walker command
9	GPGPU_WALKER	walker command

[1] The MEDIA_STATE_FLUSH command is also an object command, to ensure that it reaches the top of the root FIFO in TSG unit only when all previous requests are dispatched.

MEDIA_VFE_STATE

MEDIA_VFE_STATE is the primary state command for GPGPU pipe, and sets up context for GPGPU pipe usage as follows:

VFE allocates URB handles (1-128) and specifies the size for each URB entry.

Defines Scratch Space allocation per thread and CURBE allocation size.

Sets up and controls the hardware scoreboard in the media pipeline. Sets up the type, mask and relative vertical and horizontal distance of the dependent instant for each of the eight possible scoreboards.

Sets up the gateway usage model, especially if the gateway protocol needs to be bypassed and timestamp reset.

Defines the maximum number of simultaneous active threads possible in the system and in the thread dispatch policy, chooses the best subslice for a thread, and selects a field to specify which slices or subslices the dispatch can disable.

Defines SLM bank selection policy, determined by a bit that allows selection between the least loaded SLM bank or a bank that is paired with the thread's subslice.

Note that a stalled PIPE_CONTROL (a top-end synchronization command) is required before the MEDIA_VFE_STATE command, unless the only bits changed are scoreboard-related. If bit changes are scoreboard-related, a MEDIA_STATE_FLUSH command is sufficient.

For more details, see the MEDIA_VFE_STATE definition.

MEDIA_STATE_FLUSH

The MEDIA_STATE_FLUSH command flushes the media/GPGPU parser. In general this command is the same as an MI_FLUSH, except MI_FLUSH clears the entire 3D/render pipeline.

A key function of the MEDIA_STATE_FLUSH command is to stall the Command Streamer (CS) in one of two ways based on a watermark:

(1) Stalls the CS until there is programmed amount of room in a subslice, or enough SLM, or a free barrier.

(2) Stalls the CS until the last-level thread dispatch unit is able to dispatch the last thread to the EU.

The MEDIA_STATE_FLUSH command also indexes the resource needed to support the above watermark. A bit in this command indicates whether the write data out of the current thread group needs to be flushed beyond the visibility of following commands.

Note: You should program the MEDIA_STATE_FLUSH command for any of the following:

(a) prior to new media state, CURBE, or interface descriptor commands;

(b) while programming for a new state in the same context; or

(c) while switching to a new context.

Also note that the interface descriptor offset field updates the MEDIA_STATE_FLUSH command to specify all resources required for the next thread group. If the resources are not available, the next thread group cannot start.

For more details, see the MEDIA_STATE_FLUSH definition.

MEDIA_VFE_STATE

MEDIA_VFE_STATE is the primary state command for GPGPU pipe, and sets up context for GPGPU pipe usage as follows:

- VFE allocates URB handles (1-128) and specifies the size for each URB entry.
- Defines Scratch Space allocation per thread and CURBE allocation size.
- Sets up and controls the hardware scoreboard in the media pipeline. Sets up the type, mask and relative vertical and horizontal distance of the dependent instant for each of the eight possible scoreboards.
- Sets up the gateway usage model, especially if the gateway protocol needs to be bypassed and timestamp reset.
- Defines the maximum number of simultaneous active threads possible in the system and in the thread dispatch policy, chooses the best subslice for a thread, and selects a field to specify which slices or subslices the dispatch can disable.
- Defines SLM bank selection policy, determined by a bit that allows selection between the least loaded SLM bank or a bank that is paired with the thread's subslice.

Note that a stalled PIPE_CONTROL (a top-end synchronization command) is required before the MEDIA_VFE_STATE command, unless the only bits changed are scoreboard-related. If bit changes are scoreboard-related, a MEDIA_VFE_STATE command is sufficient.

For more details, see the **MEDIA_VFE_STATE** definition.

MEDIA_STATE_FLUSH

The MEDIA_STATE_FLUSH command flushes the media/GPGPU parser. In general this command is the same as an MI_FLUSH, except MI_FLUSH clears the entire 3D/render pipeline.

A key function of the MEDIA_STATE_FLUSH command is to stall the Command Streamer (CS) in one of two ways based on a watermark:

(1) Stalls the CS until there is programmed amount of room in a subslice, or enough SLM, or a free barrier.

(2) Stalls the CS until the last-level thread dispatch unit is able to dispatch the last thread to the EU.

The MEDIA_STATE_FLUSH command also indexes the resource needed to support the above watermark. A bit in this command indicates whether the write data out of the current thread group needs to be flushed beyond the visibility of following commands.

Note: You should program the MEDIA_STATE_FLUSH command for any of the following:

(a) prior to new media state, CURBE, or interface descriptor commands;

(b) while programming for a new state in the same context; or

(c) while switching to a new context.

Also note that the interface descriptor offset field updates the MEDIA_STATE_FLUSH command to specify all resources required for the next thread group. If the resources are not available, the next thread group cannot start.

For more details, see the **MEDIA_STATE_FLUSH** definition.

MEDIA_CURBE_LOAD

The MEDIA_CURBE_LOAD command establishes the Constant URB Entry (CURBE) mode for Payloads by defining:

Total CURBE data length provides an address range up to 2^{17} bytes for the CURBE payload data. This field is 64-byte aligned.

CURBE start address specifies the 64-byte aligned address where CURBE data must start to reside. This is a relative address to the address established by the STATE_BASE_ADDRESS command.

For more details, see the **MEDIA_CURBE_LOAD** definition.

MEDIA_INTERFACE_DESCRIPTOR_LOAD

The MEDIA_INTERFACE_DESCRIPTOR_LOAD command sets up the VFE by defining where to fetch the Interface descriptor and how many Bytes to fetch:

Loads the descriptor total length in bytes and the 64-byte aligned pointer to the interface descriptor data. This interface descriptor pointer is relative to the Dynamic Base Address programmed through the STATE_BASE_ADDRESS command.

VFE fetches the data based on the pointer and loads the data into URB.

Note: Use a MEDIA_STATE_FLUSH (without watermark) before this command to ensure that the temporary interface descriptor storage is cleared.

For more details, see the **MEDIA_INTERFACE_DESCRIPTOR_LOAD** definition.

MEDIA_OBJECT

The MEDIA_OBJECT command is the basic media primitive command for the media pipeline, defined as follows:

- Supports loading of inline, CURBE, and indirect data.
- Can be used in the following three VFE modes: Generic mode, IS mode, and VLD mode.
    - Specifies the length of inline data in all of the above three modes and all the associated rules.
    - For indirect data, defines the rules for determining the length and start address for the same three modes.
- Cannot be used in AVC-IT, AVC-MC and VC1-IT modes.

Note: If neither SLM nor a barrier is used, MEDIA_OBJECT must be used instead of the MEDIA_OBJECT_GRPID command.

For more details, see the **MEDIA_OBJECT** definition.

MEDIA_OBJECT_GRPID

The MEDIA_OBJECT_GRPID command is a variation of the MEDIA_OBJECT command and includes a 32-bit group ID, which is a unique identifying number describing threads that share the same barriers and/or SLMs.

- The interface descriptor command specifies how much SLM is needed and how many threads report to a barrier. This command tracks and executes such groups.
- All MEDIA_OBJECT_GRPIDs with the same group ID need to have the same interface descriptor. They must also be dispatched to the same Subslice, which is ensured by the hardware dispatcher if Force Destination = 0, or by software if Force Destination = 1.
- Slice/SS destination select parameters can ensure where the dispatch happens for the group. Software can also ensure that all threads needed for the Barrier fit into a subslice, or the Barrier would never be satisfied due to insufficient condition.
- Hardware only dispatches a single group at a time, and that must be enforced by the programmer for MEDIA_OBJECT_GRPID commands - unlike the MEDIA_OBJECT_WALKER command.
- A bitfield defines whether threads associated with MEDIA_OBJECT_GRPID need to use the scoreboard, and whether there are supporting fields to provide the X, Y, and color terms for the scoreboard.

Note: MEDIA_OBJECT_GRPID supports the GPGPU version of payload delivery using the following:

- Per-thread payload containing a split of either indirect or CURBE between the threads, and
- Cross-thread payload sent to all threads.

For indirect payload splitting to work properly, you must send the same pointer with all the commands associated with the thread group. A kernel does not split inline data, but sends the payload attached to each command with the corresponding thread. Only one of inline, indirect, or CURBE is allowed, and you must send at least one form of payload.

Also note that you must place MEDIA_STATE_FLUSH, with the watermark bit, between groups created by MEDIA_OBJECT_GRPID. The Interface Descriptor associated with the watermark must match the Interface Descriptor used for the subsequent group.

There are two walker commands: MEDIA_OBJECT_WALKER and GPGPU_WALKER.

For more details, see the **MEDIA_OBJECT_GRPID** definition.

MEDIA_OBJECT_WALKER

The MEDIA_OBJECT_WALKER command can create multiple threads with various patterns, per the walker parameters provided in the command. Thread generation for "parameterized media walker" works as follows:

The MEDIA_OBJECT_WALKER command uses the VFE hardware walker to generate threads associated with a rectangular-shaped object. You can use this command only in generic mode. This command can generate one or more threads.

MEDIA_OBJECT_WALKER supports loading only of inline data or CURBE, but not indirect data. You must send at least one form of payload.

When inline data is present, MEDIA_OBJECT_WALKER repeats that data for all dispatched threads, so pipeline flushes are not required. When commands are issued without a pipeline flush in between, changes to the 'global' (shared) data are supported. (Unlike CURBE, which requires pipeline flushes for such changes.)

Note: You must place a MEDIA_STATE_FLUSH command after each MEDIA_OBJECT_WALKER with inline data.

For more details, see the **MEDIA_OBJECT_WALKER** definition.

MEDIA_WALKER

Each MEDIA_WALKER command dispatches multiple threads covering the (X,Y) dimensions of an image frame.

MEDIA_WALKER is similar to MEDIA_OBJECT_WALKER command but is enhanced so that multiple MEDIA_WALKER commands can be executing at the same time from the same command stream. The number of simultaneously active MEDIA_WALKER commands is specified in MEDIA_VFE_STATE command. Simultaneously active MEDIA_WALKER commands dispatch their threads to non-overlapping DSS.

Like MEDIA_OBJECT_WALKER, the dispatch order of the MEDIA_WALKER threads is programmable for various patterns using inner, outer, local, and global loop parameters. Furthermore, a color parameter can be specified to support the simultaneous processing of 2 or more frames using the same parameters. This is useful because the inter-thread dependencies can significantly reduce the number of threads actively running in the walker command.

MEDIA_WALKER specifies the Interface Descriptor as the first part of its Indirect Data. Each dispatched thread receives a payload that is specified in the remainder of the Indirect Data.

For more details, see the definition.

GPGPU Walker

A single GPGPU job may require thousands or even millions of threads. Creating these threads individually would be inefficient, so the GPGPU_WALKER command generates the threads algorithmically (rather than modifying the Media Walker).

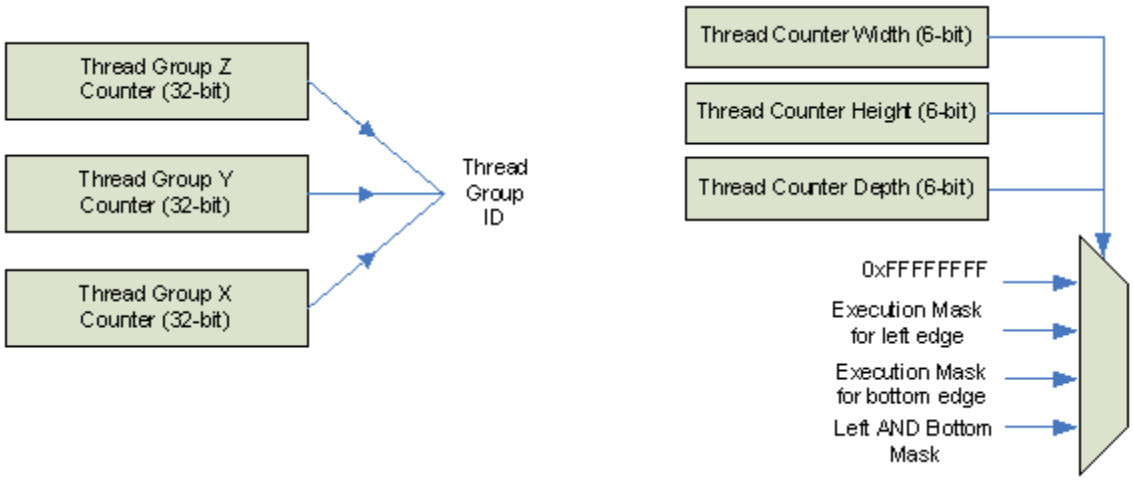


Figure: GPGPU Thread Group Walker

The X, Y, and Z counters for a thread group each have an initial and maximum value. Combining the three counter values forms the thread group ID sent with each dispatch. Each counter contains 32 bits; there is no limit on the size of the thread ID.

The GPGPU requires that 1024 SIMD channels be available for a maximum-size thread group. Thread groups that do not need access to a barrier or SLM are not limited to 1024 SIMD channels.

In general, the formula for "number of Threads" = "number of Channels" / SIMD size.

For example, in a system with ten EUs per subslice, each with seven hardware threads, a maximum-size thread group fits in a subslice if SIMD16 instructions are used, but not if SIMD8 instructions are used.

See the Configurations section for an exact determination of what SIMD option is required to fit in the subslice of the targeted configuration.

The three thread counters count the number of dispatches in a single thread group: up to 32 dispatches for SIMD32 or 64 dispatches for SIMD16/8. There are three thread counters to select the execution masks correctly - see the Execution Masks section.

Each thread counter is six bits to allow flexibility for any dimension going to 64, while the rest do not increment. A thread is generated each time one of the thread counters increment. When the thread group X counter reaches the maximum it is reset to 0, and the Y counter is incremented. When the thread group Y counter reaches the maximum it is reset to 0, and the Z counter is incremented. When all the counters reach their maximum values, the thread group is done, and the thread group counter can increment to start a new thread group.

The compiler determines how many SIMD channels are needed per thread group, and then decides how these are split among EU threads. The number of threads is programmed in the thread counter, and the SIMD mode (SIMD8/SIMD16/SIMD32) is specified in the GPGPU_WALKER command.

The maximum thread group size must fit into a single subslice (or DSS) and run in parallel, so the number of EU threads must be less than the number specified in Configurations for threads per subslice (DSS).

Thread Payload

The payload for each thread dispatched is as follows:

1. A thread group ID that identifies the group to which the threads belong. This consists of three 32-bit X/Y/Z values.
2. The set of X/Y/Z counters that form the thread ID for each channel. If the Z counter is not used, then only X and Y are needed.
3. The execution mask that indicates what channels are active.

Thread IDs form a 2D or 3D surface that is mapped into SIMD32, SIMD16, or SIMD8 dispatches. The compiler supplies the mapping, rather than having the hardware force a particular mapping of thread IDs to channels.

The VFE receives a simple count of the number of threads per thread group, which counts the number of dispatches. A constant buffer in the MEDIA_CURBE_LOAD command contains the thread IDs for all threads in a thread group. You can use a single set of thread IDs repeatedly for all thread groups, since the thread IDs are the same for each thread group ID output by the GPGPU_WALKER.

The following figure is an example set of payloads for a 2 Z x 2 Y x 12 X and a SIMD16 dispatch. This thread group requires three dispatches:

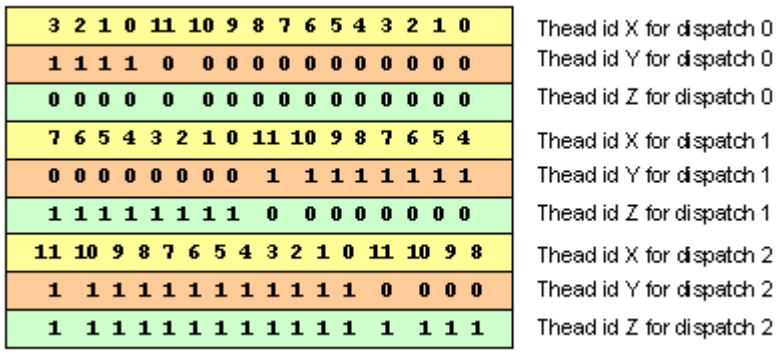


Figure: Dispatch Arrangement

In this case the thread counter width is programmed with a maximum value of three. Since the execution masks are all F, it doesn't matter how the thread counters are programmed, as long as they count to three before finishing the thread group.

The first dispatch would tell the TS that the payload starts at the beginning of the constant buffer and has a length of three. The next dispatch would have a payload starting at Constant_Buffer_Start + 3. The final dispatch payload starts at Constant_Buffer_Start + 6. If there are more thread groups in the command, they would each get exactly the same payload, except for a different thread group ID, barrier, and shared local memory space.

Execution Masks

The number of channels required by a GPGPU job may not fit evenly into the number of SIMD channels, which can leave some channels idle. The execution mask tells the hardware what channels are actively used.

The payload for a general-purpose thread must include the execution mask. The execution mask has one bit set per channel for up to 32 channels. SIMD16 and SIMD8 use the LSB bits of the execution mask. The 5-bit number transferred from VFE expands to produce the 32-bit mask. This mask uses the Dmask from the pixel shader dispatch in the transparent header.

A thread group is modeled as a 3D solid, with each channel acting as one X/Y/Z point in the solid. This group can have the following forms:

A line with 1024 X channels from 0 to 1023, and constant Y/Z;

A square with X ranging from 0 to 32 and Y ranging from 0 to32;

A cube with X ranging from 0 to 9, Y ranging from 0 to 9, and Z ranging from 0 to 9.

Software needs to determine how to map these shapes onto the 32 SIMD32 channels per dispatch, or 16 SIMD16, or 8 SIMD8. The mapping per thread is a 2D square of channels such as 8x4, 16x2, or 32x1. Below is a diagram of a 22x6 thread group that is mapped onto a set of 8x4 SIMD32 channels:

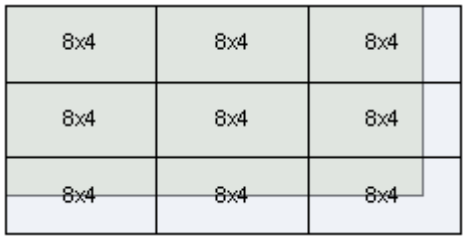


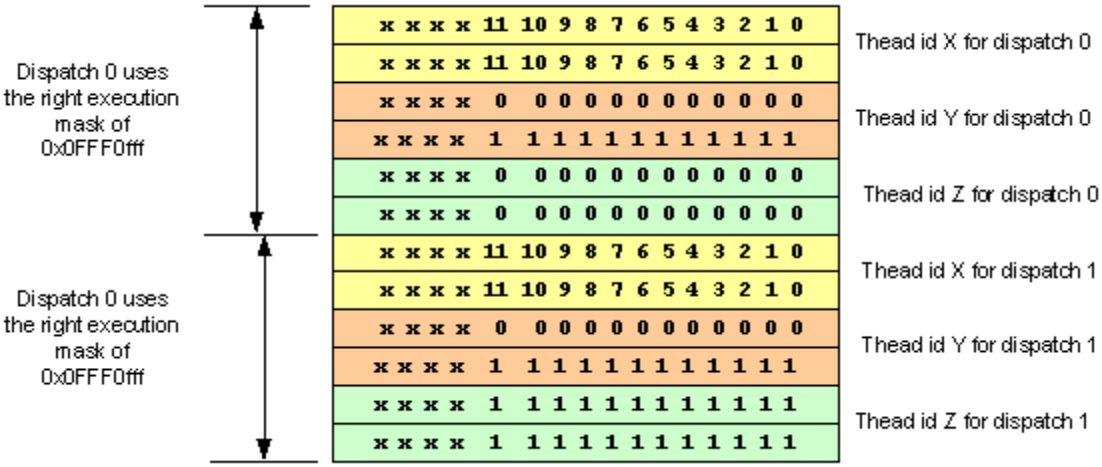
Figure: Mapping onto a Set of SIMD32 Channels

Note that the dispatches in the upper left have execution masks of all-F, while all the right edge dispatches have the same execution mask; likewise, all the bottom edge dispatches have the same execution mask. The bottom right is the logical-AND of the right and bottom edge dispatches.

A 32-bit right and bottom mask is sent with the GPGPU_WALKER command, and the thread width, height and depth counters determine when the masks are used. Width, height and depth are used instead of X/Y/Z, since it is not required that width = X; width and height are the two variables that are changing in a single SIMD dispatch even if they are Y and Z.

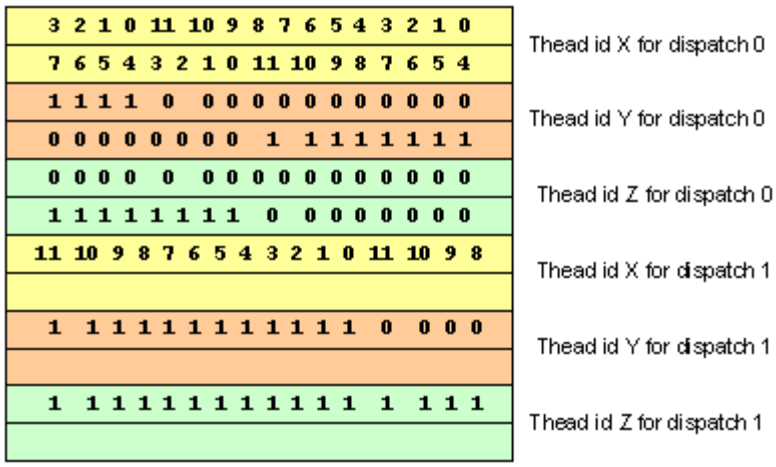
For each dispatch, the width counter increments until it reaches the maximum. The dispatch with width=max uses the right execution mask. The height counter is then incremented, and the process repeated. If at any time the height counter = max then the execution mask is the bottom execution mask. When the height and width counters are both max then the dispatch is the AND of the right and bottom masks, and the depth counter increments.

The same 2Z x 2Y x 12X thread group described above dispatched as SIMD32 with each dispatch delivering a 16X x 2Y shape requires two dispatches with empty bits in the right execution mask and all F in the bottom.



The width and height counter would have a maximum of one, and the depth counter would have a maximum of two. The two dispatches use the AND of the two masks, but since the bottom mask is F it would be the same as just the right mask.

The execution masks can also be used when the software wants to pack the channels rather than lay them out in a regular pattern:



In this case the width counter can have a maximum of two, and the height and depth counters have a maximum of one. The first dispatch uses the bottom mask only (all F), and the second uses the right AND bottom mask to remove unused channels.

Synchronization of the Media-GPGPU Pipeline

The PIPE_CONTROL command synchronizes the Media/GPGPU Pipeline in the same way as for the 3D pipeline. PIPE_CONTROL for the Media pipe case stalls at the top of the pipe until the Media FFs finish processing commands parsed before PIPE_CONTROL. Post-synchronization operations, flushing of caches, and interrupts then occur per the enabled PIPE_CONTROL parameter fields. Due to the stalling behavior at the top, only one PIPE_CONTROL command can be outstanding at a time on the Media Pipe.

The top of the pipe synchronization enforces the read-only cache invalidation. This synchronization guarantees that primitives rendered after a synchronization event fetch the latest read-only data from memory.

For more details, see the **PIPE_CONTROL** command definition.

Supporting Commands for MEDIA-GPGPU Pipe

PIPE_CONTROL

Use the **PIPE_CONTROL** command to implement synchronization in the render pipe. For Media and GPPIPE, this causes a synchronization that stalls at the top of the pipe until the Media FFs finish processing commands parsed before PIPE_CONTROL.

PIPE_CONTROL with flush replaces MI_FLUSH.

PIPELINE_SELECT

Use the **PIPELINE_SELECT** command to select the currently active pipe in the render engine. Only one of either the 3D pipeline or the GPGPU pipeline can be selected at a time; selection of these pipelines is mutually exclusive.

STATE_BASE_ADDRESS

The **STATE_BASE_ADDRESS** command sets the base pointers for subsequent state, instruction, and media indirect object accesses by the GPPIPE.

3DSTATE_BINDING_TABLE_POOL_ALLOC

The **3DSTATE_BINDING_TABLE_POOL_ALLOC** command when enables Binding Table Pool sets the base pointer for subsequent Binding Table Pointer access by the GPPIPE.

Thread Spawner (TS)

Thread spawning is fundamental to the functionality of GPGPU and programmable media. The VFE can create sets of threads using the GPGPU_WALKER, or the VFE can assemble general-purpose threads using MEDIA_OBJECT_WALKER in generic media mode.

A command streamer interprets and forwards VFE-decodable commands to the VFE unit. The VFE first runs one set of commands to set up the GP pipe, then runs WALKER commands to do automatic thread generation. A single GPGPU WALKER command can create a whole list of threads by walking either a one-dimensional range (X-pattern), a two-dimensional range (X-Y pattern), or a three-dimensional range (X-Y-Z pattern). Similarly, a media WALKER command can create a list of threads through a walking pattern of left, right, diagonal, forward, or backward.

The thread request queue stores thread requests and is in arbitration only if the number of outstanding threads does not exceed the maximum thread state variable. Otherwise, the thread request queue stalls until some other threads retire or terminate. TS keeps everything needed to get the threads ready for dispatch, and then tracks dispatched threads until their retirement.

Memory Resources for Threads

The GPGPU memory model uses URB, SLM, scratch space, and GRF. Hardware, URB, SLM, and scratch space are physically in the L3 Cache and logically segregated; GRF is exclusively inside the EUs. The following section covers these items and their usage model in more detail.

URB

The Unified Return Buffer (URB) is as a temporary holding area that handles the process of thread creation and dispatch. VFE manages the URB in both GPGPU and generic media modes and allocates one URB handle for each thread. The URB handle delivers the payload into the thread. The TS signals VFE to de-reference the URB handle immediately after receiving acknowledgement from TD that the thread has been dispatched.

The first 64 URB entries are reserved for the interface description, and CURBE data is placed after the IDs. URB handles are required for indirect data. When the VFE starts, it creates up to 128 handles by partitioning the remaining URB space into evenly spaced addresses and saving the resulting handles to a buffer. The resulting handles are removed from the buffer on use and returned to the buffer when they retire.

MEDIA_VFE_STATE specifies the amount of CURBE space, the URB handle size, and the number of URB handles.

The driver must ensure that:

((URB_handle_size * URB_num_handle) - CURBE - 64) <= URB_allocation_in_L3.

Shared Local Memory (SLM)

The Shared Local Memory is a 64KB block per subslice that must be shared between all thread groups on that half-slice. An SLM memory manager, similar to the scratch space memory manager, allocates the SLM space.

At a given time, a single interface descriptor dispatches most threads. For a new interface descriptor request, the GP pipe is drained, and all shared memory recovered before allocating new shared memory. Hence only a single size of shared memory needs to be supported at one time.

For simplicity, only power-of-2 shared memory sizes from 4KB to 64KB are allowed. A thread request specifies what size is needed. The first thread of a Thread Group is marked whether it requires new shared local memory. If not, the old SLM offset is sent with the dispatch.

A simple set of 16 bits allocates 4KB shared memory, with more bits used for larger sizes. A priority encoder finds the first unused bit and the offset associated with a particular barrier ID. The barrier ID then tracks the thread group. When thread group tracking indicates that a thread group is completely retired, that section of SLM can be reclaimed.

Local Memory/Scratch Space

Local memory is allocated per thread dispatched to the EU. The scratch space manager provides between 1KB and 2MB memory per thread.

GRF

A General Register File (GRF) inside the EU represents the per-thread private memory of the general-purpose compute framework memory model. The GRFs are register files dedicated to read/write operations of the threads. There are 128 GRFs of 32 bytes each per thread.

Media and GPGPU Indirect Thread Dispatch

Indirect thread dispatch allows one thread group to control the group size of a following thread group. To enable GPGPU indirect thread dispatch, use the following sequence of commands in the ring buffer:

GPGPU_OBJECT/WALKER	// Either a set of objects or a walker to dispatch a thread group to write the next group's properties to memory.
MI_FLUSH	// Make sure the thread group has finished executing.
MEDIA_CURBE_LOAD	// Load the thread ids for new group.
MI_LOAD_REGISTER_MEMORY	// Load the indirect MMIO GPGPU registers from the mem written by the previous group.
GPGPU_WALKER (indirect)	// A walker with the indirect bit set.

The first thread group writes the following data to memory:

1. The thread IDs delivered in the CURBE, written where the following MEDIA_CURBE_LOAD reads them.
2. The GPGPU_WALKER parameters, written where the MI_LOAD_REGISTER_MEMORY reads them:
    a. GPGPU_DISPATCHDIMX - the X dimension of the number of thread groups in which to dispatch; DW offset 7
    b. GPGPU_DISPATCHDIMY - the Y dimension of the number of thread groups in which to dispatch; DW offset 10
    c. GPGPU_DISPATCHDIMZ - the Z dimension of the number of thread groups in which to dispatch; DW offset 12

Use CURBE (rather than indirect) for the payload when using indirect dispatch.

Parameterized Media Walker

The Parameterized Media Walker is a hardware thread-generation mechanism to create threads associated with units in a generalized two-dimensional space (e.g. blocks in a 2D image). With a small number of unit step vectors, the walker can implement a large number of walking patterns. This walker may provide functions that are normally handled by the host software, thus simplifying the host software to GPU interface.

The walker described here is doubly nested, meaning a "local" walker can perform a variety of two-dimensional walking patterns while a "global" walker performs similar two-dimensional patterns upon many local walkers. The local walker has three levels (outer, middle, and inner) while the global walker has two levels (outer and inner). Hence the walker algorithm has five-nested loops that modify local state based on user-defined unit step vectors.

The parameterized media walker's programmability is derived from the following.

- The walker traverses a unit-normalized surface. Some example unit sizes:
  - 1x1: Walking pixels,
  - 4x4: Walking sub-blocks,
  - 16x16: Walking macro-blocks, and
  - 32x16: Walking macro-block-pairs.
- Unit step vectors to describe the motion at each of level of nesting.
- Starting locations for the local and global walkers.
- Block sizes of the local and global walker.
- A small number of special mode controls for the inner-most loop, designed to efficiently divide an image into two balanced workloads for dual-slice designs.

Walker Parameter Description

The global and local loops are both described by the same four parameters:

- Resolution,
- Starting location,
- Outer unit vector, and
- Inner unit vector.

The local inner loop has some special modes that are described later. A table of the user inputs and some example values are shown below:

GLOBAL LOOP PARAMETERS							
Global Resolution		Global Start		Outer Loop Unit Vector		Inner Loop Unit Vector	
X	Y	X	Y	X	Y	X	Y
120	68	0	0	32	0	0	32

LOCAL LOOP PARAMETERS							
Block Resolution		Local Start		Outer Loop Unit Vector		Inner Loop Unit Vector	
X	Y	X	Y	X	Y	X	Y
32	32	0	0	1	0	-2	2

LOCAL INNER LOOP SPECIAL MODE SELECTS							
Dual Mode	Repel	Attract			ExtraSteps	X	Y
TRUE	FALSE	FALSE			1	0	1

A programmer implicitly defines what a "unit" represents, which means the walker traverses a "unit normalized space" that is not inherently bound to pixel walking.

If the smallest unit of work to be walked is a 4x3 block of pixels, you can program the inner loop to step (4,3) or (1,1) as follows:

- In the (4,3) case, the user is walking units of pixels.
- In the (1,1) case, the user is walking units of 4x3 blocks of pixels.

Note that hardware doesn't contain enough bits for pixel walking for resolutions like 1920x1088 pixels. The intended usage of the walker is for block walking where the block size is not relevant to the walker parameters.

Basic Parameters for the Local Loop

Basic Parameters for the Local Loop The local inner and outer loop xy-pair parameters alone can describe a large variety of primitive walking patterns. Below are nine primitive walking patterns generated by varying only the inner and outer unit step vectors of the local loop:

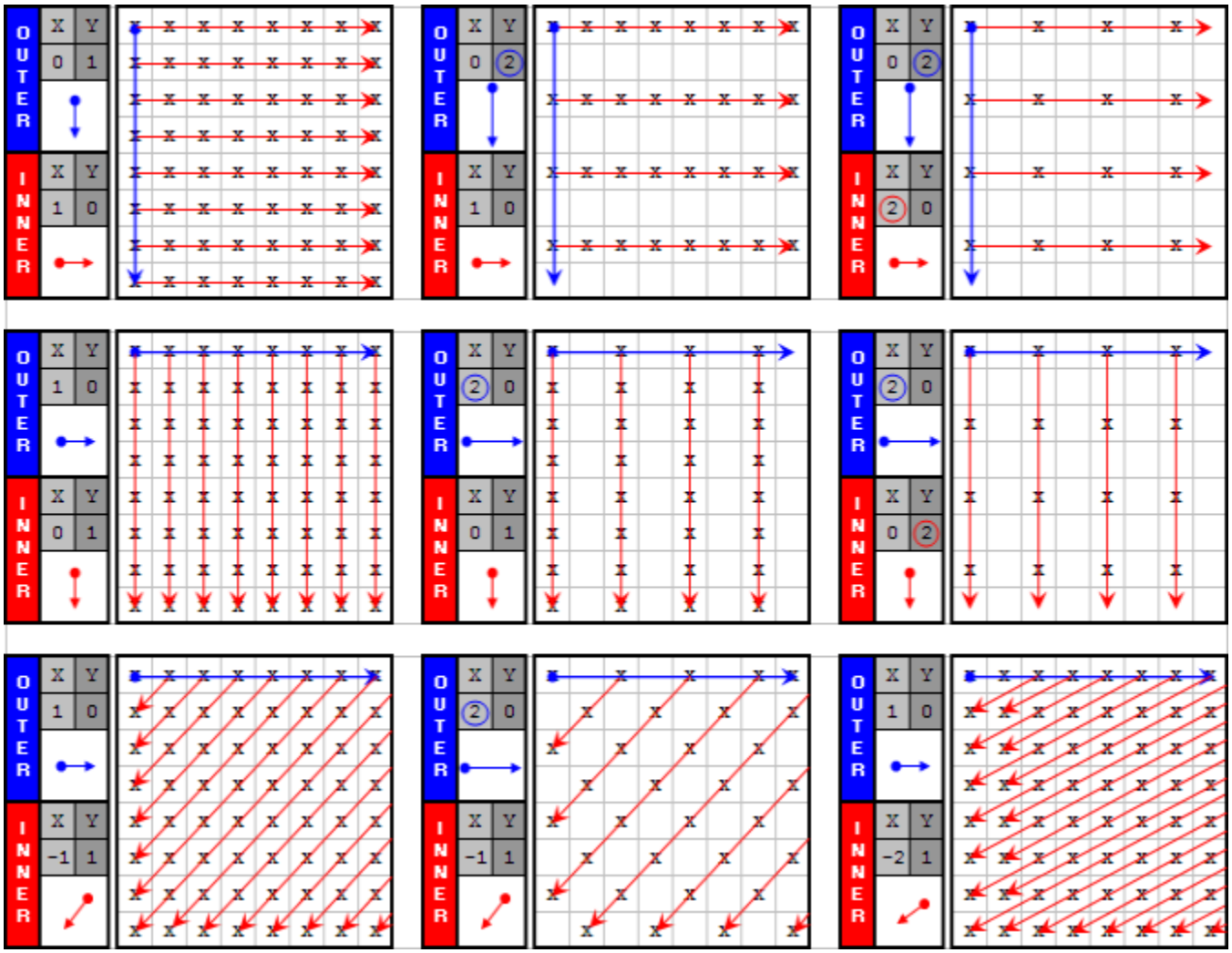


Figure: Local Loop Walking Patterns

- The top row of the figure above shows the outer unit vector pointing down (+Y) and the inner unit vector pointing right (+X). Skip rows and columns by increasing the unit step vectors beyond one.
- The middle row of the figure shows the outer unit vector pointing right (+X) and the inner unit vector pointing down (+Y). Skip rows and columns by increasing the unit step vectors beyond one.
- The last row of the figure shows the option to walk patterns not aligned to the edges. The left option yields a 45^o walking pattern by setting the inner unit vector to (-1,1). The middle option yields a checkerboard pattern by skipping every other outer loop and retaining the inner unit vector of (-1,1). The right option yields a 26.5^o walking pattern by setting the inner unit vector to (-2,1).

The block resolution, shown as [8,8], and the starting location, shown as [0,0], can be varied and the walking patterns can be stretched and rotated in many ways. The diagram below shows an example of where the start position and unit step vectors can be set to achieve a full rotation of the same pattern:

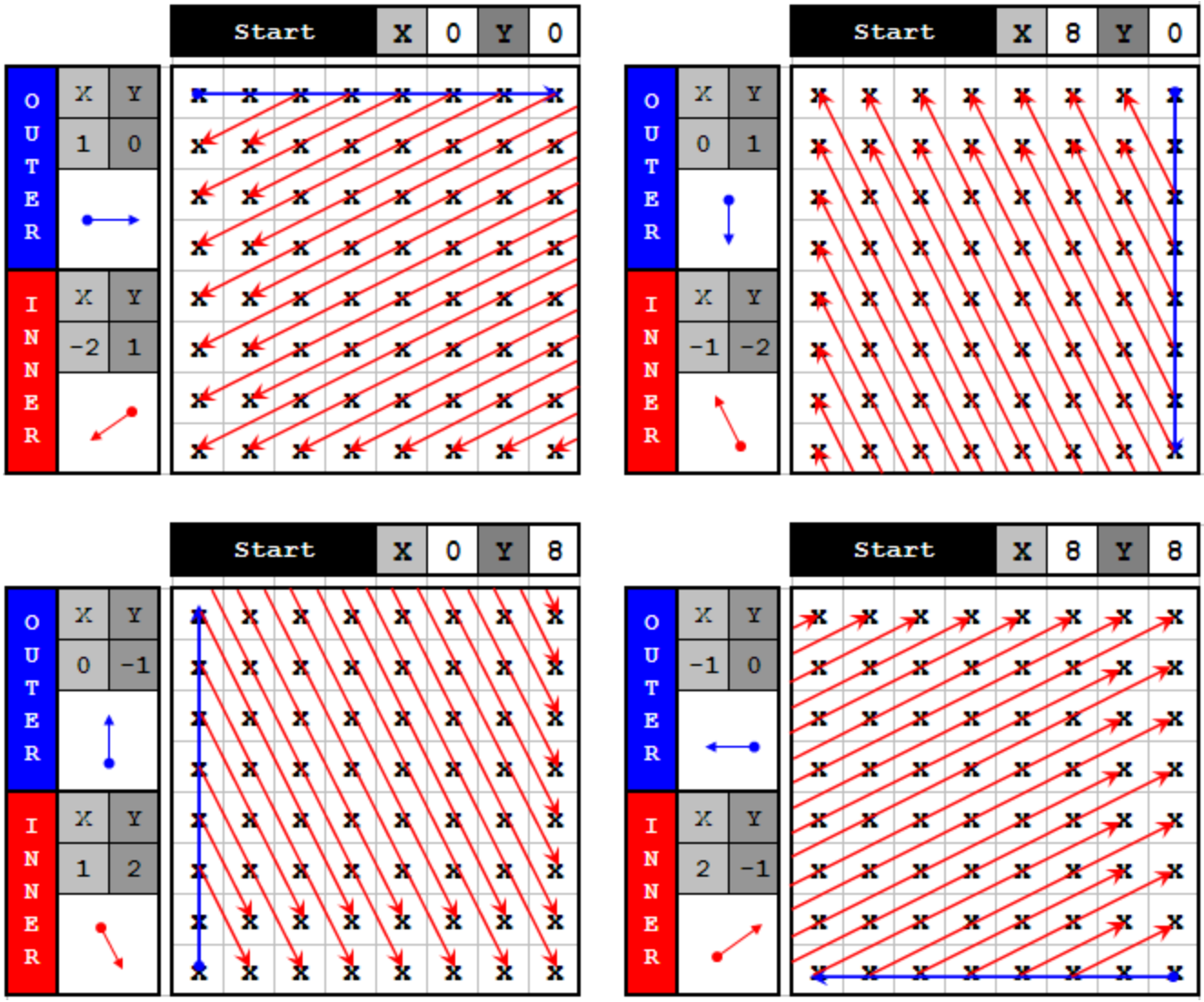


Figure: More Local Loop Walking Patterns

Walker Algorithm Description

The walker algorithm has been tested and optimized in software. A high-level pseudo-code description is given below:

```
Walker(){ //C-Style Pseudo-Code of Walker Algorithm
    Load_Inputs_And_Initialize();
    While (Global_Outer_Loop_In_Bounds()){
        Global_Inner_Loop_Intialization();
        While (Global_Inner_Loop_In_Bounds()){
            Local_Block_Boundary_Adjustment();
            Local_Outer_Loop_Initialization();
            While (Local_Outer_Loop_In_Bounds()){
                Local_Middle_Loop_Initialization();
                While (Local_Middle_Steps_Remaining()){
                    Local_Inner_Loop_Initialization();
                    While (Local_Inner_Loop_Is_Shrinking()){
                        Execute();
                        Calculate_Next_Local_Inner_X_Y();
                    } //End Local Inner Loop
                    Calculate_Next_Local_Middle_X_Y();
                } //End Local Middle Loop
                Calculate_Next_Local_Outer_X_Y();
                Calculate_Next_Local_Inverse_Outer_X_Y();
            } //End Local Outer Loop
            Calculate_Next_Global_Inner_X_Y();
        } //End Global Inner Loop
        Calculate_Next_Global_Outer_X_Y();
    } //End Global Outer Loop
} //End Walker
```

The above pseudo-code has the following characteristics:

- There are five levels of iteration.
- The highest two levels are called "global" and the lowest three levels are called "local":
    - The global loop contains an outer and an inner loop;
    - The local loop contains an outer, middle, and inner loop;
    - The user defines a bounding box for the global and local resolution;
    - The user also specifies the starting location within each bounding box.
- Each of the five loops has its own persistence:
    - Current position (x,y),
    - Unit step vector (x,y).
- The final output (x,y) is a summation of the global x,y and the local x,y.
- The next (x,y) for a given level can be calculated while the next lower level is still executing. The (x,y) calculation result tests whether the current level needs to execute again once control is returned.

The flow of the global outer and inner loops is as follows:

1. Check a bound condition;
2. Initialize the next level loop;

3. Execute the next level loop;
4. When the next level loop fails its bound condition, calculate the next position for the current loop level and repeat.

Walker Algorithm Flowchart for the Global Loop

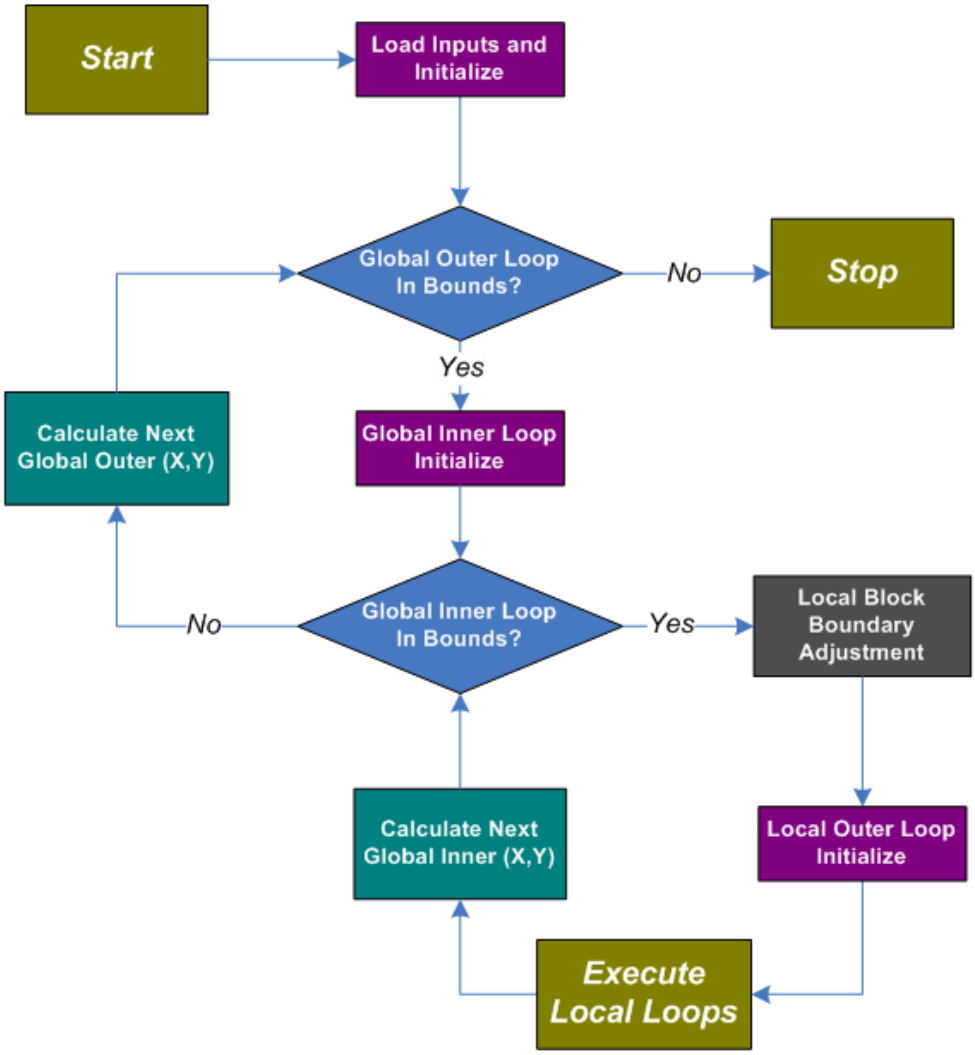


Figure: Walker Algorithm Flowchart for the Global Loop

Note the grey box above labeled, "Local Block Boundary Adjustment." Use this logic to adjust the local block size, when the distance between the current global position and the edge of the image is less than the local resolution. Local starting positions may also be modified by the same logic, if the defined starting position is larger than the new local block size.

The flow of the three local loops does not vary much from the two global loops. The differences are as follows:

- The local middle loop, in addition to a boundary check, also ensures the number of middle steps is less than or equal to the user-defined "number of extra steps."

- The local inner loop only checks to see whether the prior distance between the x,y starting and ending points is greater than their current distance. If this comparison is true, that implies the two inner loops are converging towards each other.
- When the middle loop check fails, both the starting points (local outer) and ending points (local inner) are updated.

Walker Algorithm Flowchart for the Local Loop

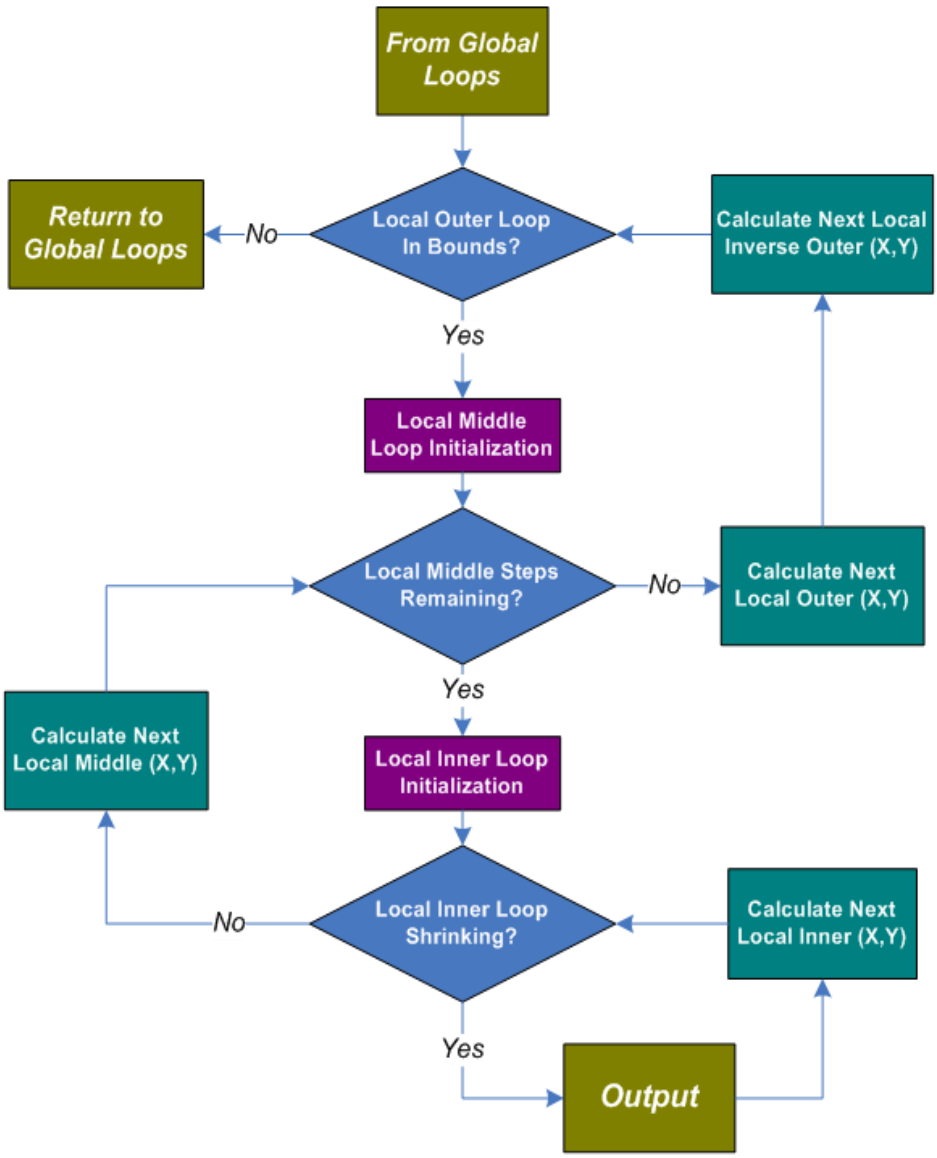


Figure: Walker Algorithm Flowchart for the Local Loop

Flexible Dispatch of Local Loop

A local loop automatically splits between available subslices with adjacent blocks kept next to each other, to improve cache and execution efficiency. A single iteration of the local loop splits into equal segments, one for each currently powered-up subslice. The dispatches for the local loop are done so that one

thread from each segment is dispatched before repeating the process. For example: segment 0, thread 0 is followed by segment 1, thread 0 rather than segment 0, thread 1.

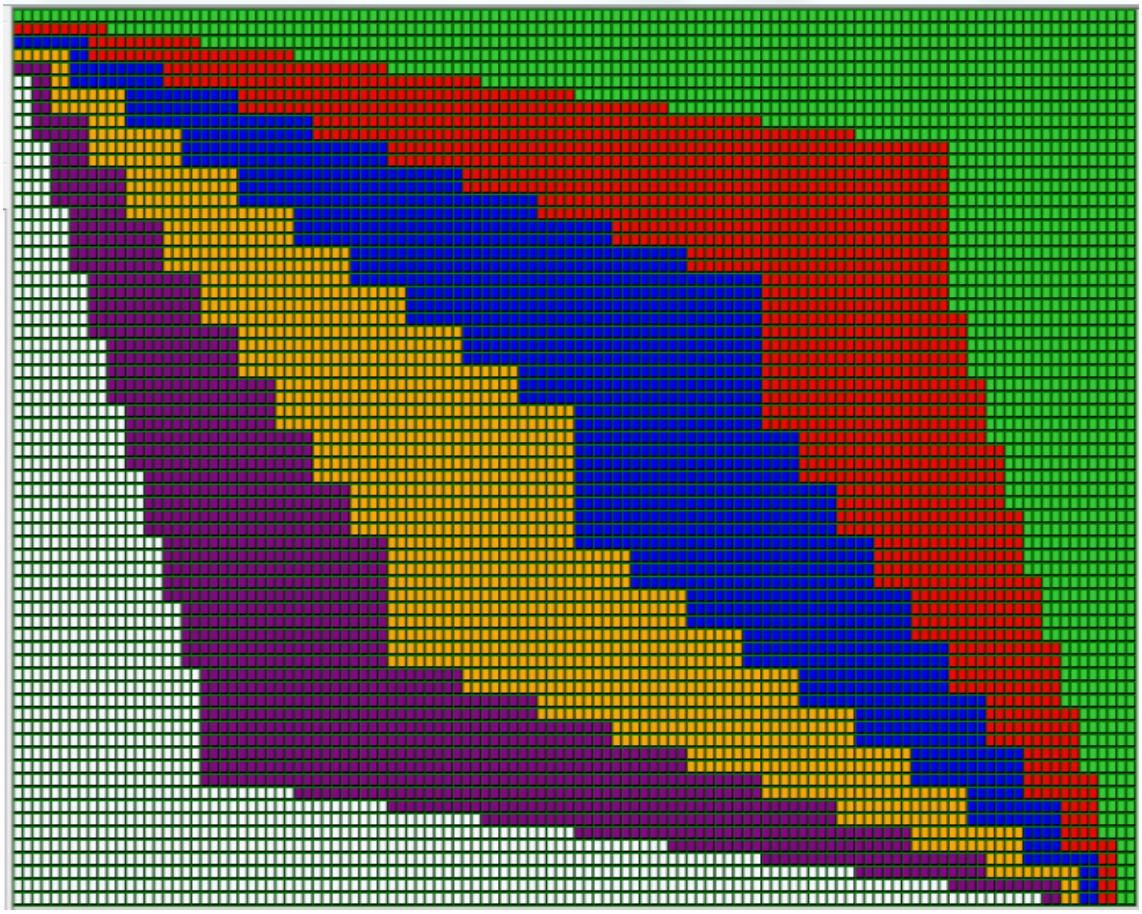


Figure: Splitting a Wavefront Between Six Segments

Each color in the figure above indicates a separate segment dispatched to a specific subslice. In this example, each local loop walks a diagonal line from the lower left to the upper right, while the global loop steps between the lines. Dependencies are to the left and above, which is a typical usage model.

The first few iterations are in the upper right corner and have few blocks dispatched per local loop. Farther down, the length of the local loop gets large enough that the six available subslices are full of threads from each segment running in parallel, and being dispatched in an even manner.

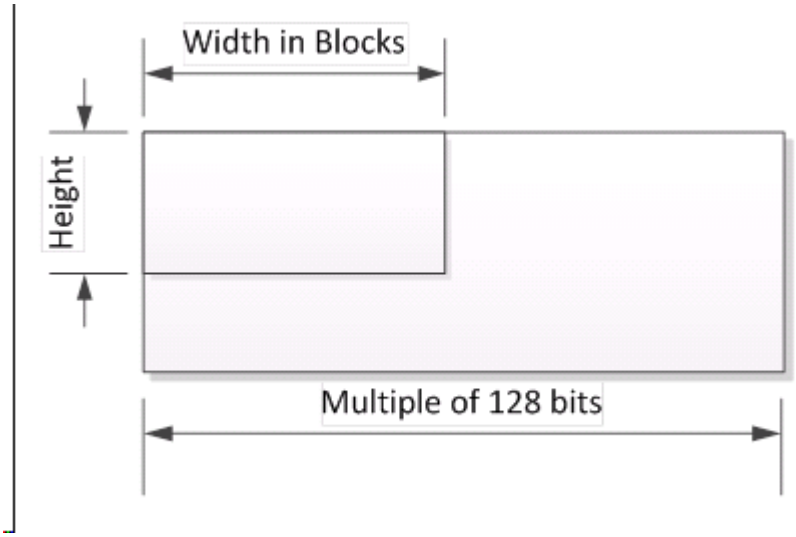
Masked Dispatch

Masked dispatch adds a dispatch mask to the MEDIA_OBJECT_WALKER command, which provides a bit per dispatch to determine whether to drop the dispatch. This allows use of MEDIA_OBJECT_WALKER with irregularly shaped areas. When using the masked dispatch mode, program the walker to walk the surface as a simple rectangle, with X changing in the innermost loop.

CURBE contains the dispatch mask, so a MEDIA_OBJECT_WALKER command that uses the dispatch mask cannot use CURBE in the dispatch payload. CURBE contains a bitmask for which each bit corresponds to the X/Y blockID (not the pixel X/Y). The byte address of each bit is defined as follows:

Byte_address = (Y * pitch + X) » 8

Pitch is the number of horizontal blocks in the surface, rounded up to the next power of two bits (e.g. 128, 256, or 512).



The MEDIA_OBJECT_WALKER command allows X and Y to have a maximum of nine bits, which corresponds to a 512 pixels x 512 pixels surface in CURBE. For 16 pixel by 16 pixel blocks, this would represent a total of 8KB x 8KB pixels. Note that in this mode, the walker dispatches in sequence, rather than splitting the local loop into segments that dispatch to individual subslices (as described in "Flexible Dispatch of Local Loop" above).

Thread Dispatch

The following sections cover the thread dispatch process in the GPGPU pipe.

Thread Dispatch Format

The thread dispatch for GPPIPE consists of a series of headers and payloads, routed through the dispatch pipeline from the thread spawner unit up until the EU. The thread dispatch consists of transparent and R0 headers and an associated payload that may consist of per-thread and cross-thread data. The transparent header part gets translated as part of the EU state surrounding the threads, becomes the context for a thread while the rest of the payload (including R0 and any optional payload) gets copied to the GRF, and acts as parameters for the thread. Both the transparent and R0 headers are crafted dynamically by hardware based on the interface descriptor.

All threads begin with a transparent header, followed by an R0 header. The format of R0 is different for GPGPU and media threads, as well as for other 3D fixed functions. While the thread spawner generates the headers, the payload is fetched from URB and attached by the dispatch pipeline before dispatch to the EU.

Table: Dispatch Combinations and Associated TSG Phases

Dispatch Type	Dispatch Parameters				TSG Phases (2 or 3 clocks / thread)		
	GPGPU Mode	CURBE Length	Cross Length	Payload Length	Ph-1	Ph-2	Ph-3
Media Thread no CURBE no Payload	0	0	0	0	TH	R0	
Media Thread no CURBE with Payload	0	0	0	>0	TH	R0	Root Handle (DW0)
Media Thread with CURBE no Payload	0	>0	0	0	TH	R0	
Media Thread with CURBE with Payload	0	>0	0	>0	TH	R0	Root Handle (DW0)
GPGPU Thread no Thread ID no CURBE	1	0	0	0	TH	R0	
GPGPU Thread with Thread ID no CURBE	1	>0	0	0	TH	R0	
GPGPU Thread with Thread ID with CURBE	1	>0	>0	0	TH	R0	Cross Thread CURBE Handle (DW0)

The interface descriptor of a given thread determines whether the CURBE data field is present.

INTERFACE_DESCRIPTOR_DATA

Interface descriptor data is the data structure fetched from the URB, using the interface descriptor data start address. The interface descriptor data serves as the key parameter file to set up the modes of execution, formation, and usage model of threads and thread groups.

Some of the key parameters controlled by the interface descriptor are as follows:

- Basic thread execution attributes such as the kernel start pointer, floating point mode selection, thread priority definition, handling of de-normalized numbers, whether the kernel uses >1 SIMDness, and whether the thread can be pre-empted if a mid-thread pre-emption request comes through.
- Number of threads in a GPGPU thread group. This parameter needs to be accurate because it is used in the pre-emption sequence to ensure proper pre-emption. The minimum value is one and

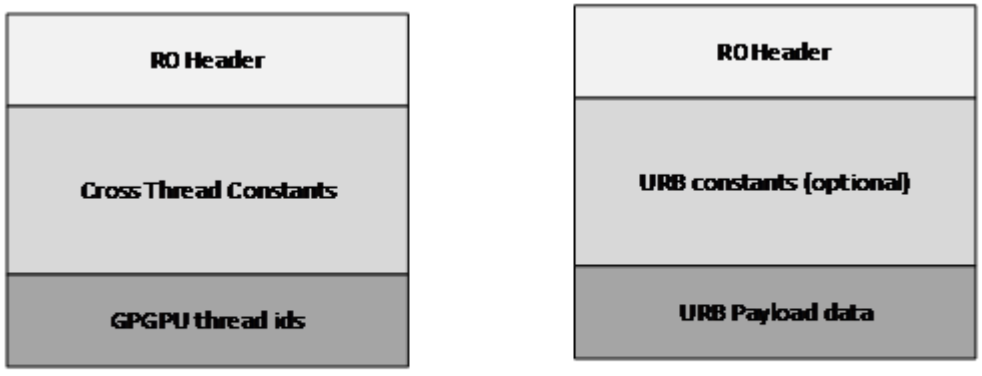
the maximum is the number of threads in two subslices for local barrier. For global barrier it may be higher.

- SLM size for a thread group, which is specified in powers of two and minimum increments of 4K blocks up to 64K per subslice.
- Barrier enable and global barrier enable/disable. These parameters define whether there is a barrier associated with a thread group and, if so, can it be local within a subslice or cross-subslice as a global barrier (or both). Setting a global barrier affects performance.
- Constant URB Entry (CURBE) data is a combination of fields. First is the address offset where the constant URB data can be found, followed by a combination of "constant/indirect URB entry read length" and "cross-thread constant data read length." The latter two items define whether there is any payload - and, if so, whether that is per-thread only payload, CURBE only, or both.
- The binding table pointer, and the number of binding table entries that the kernel uses. Also, the sampler state table pointer and sampler count for kernels in use.
- Some exception enables that get loaded directly into the EU CR0 registers.

For more details, see the **INTERFACE_DESCRIPTOR_DATA** definition.

R0 Headers

GPGPU and Media R0 headers differ in their organization. Even for Media threads, those with groups attached have a different R0 header than the generic mode R0 header.



GPGPU Payload format Media Payload format

GPGPU R0 Header

The thread dispatch payload for a GPGPU thread consists of:

An R0 header;

If present, cross-thread CURBE data in R1 and subsequent registers; and

Per-thread payload (e.g. X/Y/Z thread IDs) in the next available registers of the thread - starting in R1 if there is no CURBE data.

DWord	Bits	Description
R0.7	31:0	**Thread Group ID Z:** This field identifies the Z component of the thread group that this thread belongs to.
R0.6	31:0	**Thread Group ID Y:** This field identifies the Y component of the thread group that this thread belongs to.
R0.5	31:10	**Scratch Space Pointer.** Specifies the 1K-byte aligned pointer to the scratch space (used for the GPGPU local memory space). Format = GeneralStateOffset[31:10]
	9:0	**FFTID.** This ID is assigned by TS and is a unique identifier for the thread in comparison to other concurrent threads (of any thread group). It is used to free up resources used by the thread upon thread completion. Format = U9
R0.4	31:5	**Binding Table Pointer.** Specifies the 32-byte aligned pointer to the Binding Table. It is specified as an offset from the **Surface State Base Address**. Format = SurfaceStateOffset[31:5]
	4	Reserved: MBZ
	3:0	Indicates the stack memory size. Range = [0,11] indicating [1K bytes, 2M bytes] **Programming Note**: Exception handling on stack overflow is not supported when GPGPU mid-thread pre-emption is desired.
R0.3	31:5	**Sampler State Pointer.** Specifies the 32-byte aligned pointer to the sampler state table. Format = GeneralStateOffset[31:5]
	4	Reserved: MBZ
	3:0	**Per Thread Scratch Space.** Specifies the amount of scratch space, in 16-byte quantities, allowed to be used by this thread. The value specifies the power that two is raised to, to determine the amount of scratch space. Format = U4

DWord	Bits	Description
		Range = [0,11] indicating [1K bytes, 2M bytes] in powers of two.
R0.2	31	**Barrier Enable.** This field indicates that a barrier has been allocated for this kernel.
	30:24	**BarrierID.** This field indicates the barrier that this kernel is associated with. Format: U7 Range = [0,63]
	14:11	Reserved: MBZ
	10	**GPGPU Dispatch** - This field indicates that the dispatch is from GPGPU_WALKER rather than the various media dispatch instructions.
	3:1	Reserved: MBZ
	0	**FreeThread** - This bit is set in the first thread of a set of fused threads to help properly track the available threads. In non-fused mode it is set on every thread.
R0.1	31:0	**Thread Group ID X:** This field identifies the X component of the thread group that this thread belongs to.
R0.0	31:24	**SLM ID.** This field encodes whether SLM was allocated for this thread, and if so, the SLM base offset used with SLM operations. Format: U8
	23:17	Reserved: MBZ
	16	**FFSID:** Fixed Function Stack ID. This field indicates which stack the thread belongs to.
	15:0	**URB Handle:** This is the URB handle indicating the URB space for use by the thread.

Cross-thread CURBE if present is in R1 and above, followed by the X/Y/Z thread id values for each channel in the thread.

MEDIA R0 Header

The thread payload messages for generic mode and for MEDIA_OBJECT_WALKER are in the same format. The R0 header is the same in both cases, then there are constants from CURBE when CURBE is enabled, followed by inline data when inline is enabled. The inline data block field is the same as in the MEDIA_OBJECT command, with zero-filled partial GRF, and there is no indirect data block field.

The R0 header of the MEDIA_OBJECT_GRPID and MEDIA_OBJECT_WALKER payload with groups enabled is different from the generic mode and MEDIA_OBJECT* payload without groups enabled. This header payload difference is shown in the table below.

Table: Media R0 Header Payload

DWord	Bits	Description
R0.7	31:0	**Group ID LSB.** This is the LSBs of the Group ID. For MEDIA_OBJECT_GRPID threads this is the entire group id. For MEDIA_OBJECT_WALKER threads the interpretation depends on the Group ID Loop Select: 0: No groups, field is 0. 1: cat(InnerGlobalCnt[6:0], OuterLocalCnt[9:0], MidLocalCnt[4:0], InnerLocalCnt[9:0]); rest of group id is in R0.2. 2: cat(OuterGlobalCnt[6:0], InnerGlobalCnt[9:0], OuterLocalCnt[9:0], MidLocalCnt[4:0]); rest of group id is in R0.2. 3: cat(2'b0, OuterGlobalCnt[9:0], InnerGlobalCnt[9:0], OuterLocalCnt[9:0]). 4: cat(12'b0, OuterGlobalCnt[9:0], InnerGlobalCnt[9:0]). 5: cat(22'b0, OuterGlobalCnt[9:0]).
R0.5	31:10	**Scratch Space Pointer.** Specifies the 1k-byte aligned pointer to the scratch space. This field is only valid when Scratch Space is enabled. Format = GeneralStateOffset[31:10]
	9:0	**FFTID**. This ID is assigned by TS and is a unique identifier for the thread in comparison to other concurrent root threads. It is used to free up resources used by the thread upon thread completion.
R0.4	31:5	**Binding Table Pointer.** Specifies the 32-byte aligned pointer to the Binding Table. It is specified as an offset from the **Surface State Base Address**. Format = SurfaceStateOffset[31:5]
	4:0	Reserved: MBZ
R0.3	31:5	**Sampler State Pointer.** Specifies the 32-byte aligned pointer to the sampler state table. Format = GeneralStateOffset[31:5]
	4	Reserved: MBZ
	3:0	**Per Thread Scratch Space.** Specifies the amount of scratch space, in 1K-byte quantities, allowed to be used by this thread. The value specifies the power to which two is raised to determine the amount of scratch space. Format = U4 Range = [0,11] indicating [1K bytes, 2M bytes] in powers of two
R0.2	31	**Barrier Enable.** This field indicates that a barrier has been allocated for this kernel.
	30:24	**BarrierID.** This field indicates the barrier that this kernel is associated with. Format: U7 Range = [0,63]
	23:11	**Group ID MSB.** This field is the MSBs of the Group ID. These bits are 0 for MEDIA_OBJECT_GRPID. For MEDIA_OBJECT_WALKER threads the interpretation depends on the Group ID Loop Select: 0: No groups, field is 0. 1: cat(OuterGlobalCnt[9:0], InnerGlobalCnt[9:7]). 2: cat(10'b0, OuterGlobalCnt[9:7]). 3-5: All zero.
	10	Reserved: MBZ
	9:4	**Interface Descriptor Offset.** This field specifies the offset from the interface descriptor base pointer to the interface descriptor that is applied to this object. It is specified in units of interface descriptors.

DWord	Bits	Description
		Format = U6
	3:0	Reserved: MBZ
R0.1	31:24	**Block Color.** This field is the 8-bit Media block Color value. It specifies which dependency color the current thread belongs to Format = U8
	23	Reserved: MBZ
	22:12	**Block Y.** This field provides the Y media block coordinate of current thread. Format = U11
	11	Reserved: MBZ
	10:0	**Block X.** This field provides the X media block coordinate of current thread. Format = U11
R0.0	31:24	**SLM ID.** This field encodes whether SLM was allocated for this thread, and if so, the SLM base offset used with SLM operations.
	23:17	Reserved: MBZ
	16	**FFSID:** Fixed Function Stack ID. This field indicates which stack the thread belongs to.
	15:0	**URB Handle.** This is the URB handle where indicating the URB space for use by the root thread and its children.

Thread Tracking and Synchronization

You must track a dispatched thread for the End Of Thread (EOT) lifecycle. The tracking process in TSG creates renewable capacity in the EU farms and renews thread resources by releasing them once the thread lifecycle finishes. For GPGPU, the synchronization between threads is done through barrier; in Media, the synchronization is done through the scoreboard.

Thread Synchronization Monitors

Thread Synchronization Monitors (TSM) are a hardware feature used for inter-thread communication. It can be used to implement a software scoreboard or other synchronization method with a higher performance than using polling. Unlike barriers, TSM allows communication across all thread in the entire machine, rather than just the threads in a thread group.

TSM uses messages to the Message Gateway to monitor and signal events. The messages used are:

1) MonitorEvent - this message indicates to the Gateway that a particular Event ID is of interest to the sending thread and should be monitored. When the corresponding event occurs, it is recorded in the Gateway but no message is sent until a WaitForEvent message is sent.

2) WaitForEvent - this message tells the Gateway that the thread is ready to go to sleep until the event or a timeout occurs. After sending this message the thread should read the writeback register to allow the EU to switch to other threads until the event occurs and the dependency is cleared. The message payload specifies how long the Gateway should wait before signaling a timeout, and the writeback register specified gets a value of 0 if a timeout occurs or 1 if the event occurs. Note that a pre-emption will always cause a timeout.

3) SignalEvent - this message is broadcast to every Gateway (including the source Gateway). The event is recorded for each thread that has a MonitorEvent with a matching event ID, while a message is sent to each thread which has a WaitForEvent outstanding. If the WaitForEvent occurs after the SignalEvent then the message will go out immediately upon receiving the WaitForEvent. After the message goes out thread will still be monitoring that event, so it can be reused as long as care is taken that every thread gets the same number of events - if an event happens twice while a thread is merely monitoring before WaitForEvent it will only see the event once.

4) MonitorNoEvent - this message disables the monitoring of events for this thread. This is also done when the thread exits with the EOT (End of Thread) message.

See the Message Gateway section of the Bspec for more details of these messages.

Here is an example of using TSM for a software scoreboard which is used to determine if macroblocks that the current thread is dependent on have completed:

```
For (all dependencies for this macroblock) {
 MonitorEvent(scoreboard[dx][dy]); // Start monitor before checking dependency bit
 While (!atomic_read(scoreboard[dx][dy])) {
 WaitForEvent(EventID(dx,dy), timeout); // EventID determined from dx & dy
 }
 }
// MonitorNoEvent is not needed here
 DoBlock(); // All normal processing of block
 Atomic_write(scoreboard[x][y], 1); // Set the dependency bit for following blocks
 SignalEvent(EventID(x,y));
```

Barriers and Shared Local Memory

Barriers and Shared Local Memory (SLM) offer advantages for both general purpose and media applications. Barriers can synchronize between GPGPU and media threads more efficiently than atomics can do, while SLM can be used to share data between tightly associated threads.

Barriers and SLM can also be used with a more generalized walker, to add the ability to use the scoreboard at the same time. Implementation requires an identifying number similar to the thread group ID that can be used to track and free resources. The MEDIA_OBJECT_WALKER can use various loop counts to define the group ID.

For flexibility, you can specify which loop counts form the group ID, and which ones count the threads inside each group. If cross-slice barriers are disabled, the walker must ensure that all threads in a group are dispatched to a single subslice so that the barrier and SLM are available to all group members. Make sure that the number of threads generated per group is no larger than the number of threads available in a subslice.

MEDIA_OBJECT_WALKER adds the X/Y values for the various loops together to produce a single X/Y point. Since some walking patterns have overlapping X/Y values, the combined X/Y point cannot be used for the global ID; instead, execution counts are concatenated together to produce an ID number. (An

execution count is a tally of how many times each loop is executed.) The GPGPU thread group ID is a 96-bit number, so it easily contains the 49 bits created by all the execution counts.

The media walker has five nested loops for producing the X/Y point. In addition, there is an innermost color loop defined as follows:

1. Color loop - 4 bits
2. Inner local loop execution count - 10 bits
3. Mid local loop execution count - 5 bits
4. Outer local loop execution count - 10 bits
5. Inner global loop execution count - 10 bits
6. Outer global loop execution count - 10 bits

A media walker parameter specifies the global ID bits by identifying at what point to switch between the group ID and per-thread ID. Unused loops always have execution counts of zero. The media walker forms group IDs by concatenating execution counts that are enabled in the Group ID Loop Select to create the LSBs of the IDs and sets any unused MSBs to zero.

Barriers and SLM Example:

If a MEDIA_OBJECT_WALKER specifies that the outer local loop count and above form the group ID, then every iteration of the color, inner local, and mid local dispatches a thread with the same group ID. TSG allocates a shared barrier and SLM (as specified in the interface descriptor), and ensures that all threads go to the same subslice. If no global barriers are used, the number of threads must fit on a single subslice; if global barriers are used, the number of threads must fit on the system.

The group ID for this example is formed by concatenating the outer global exec count, inner global exec count, and outer local exec count. When the group ID increments, the TSG allocates a new barrier and SLM and pick an available subslice.

Media-GPGPU Thread EOT Message

The thread EOT message is issued to the Gateway unit by a GPGPU or Media thread running on an EU. This message contains only one 8-DWord register which is intended to be a direct copy of R0 in the thread payload. It indicates that the thread is terminating and the thread spawner should dereference the barrier and SLM resources that were optionally allocated with the thread when it was dispatched.

Message Descriptor

Thread EOT Message Descriptor

Message Payload

DWord	Bits	Description
M0.5	31:10	Ignored.
	9:0	Ignored.
M0.4	31:0	Ignored.
M0.3	31:0	Ignored.
M0.2	31	**Barrier Enable.** This field indicates whether a barrier was allocated for this thread. Format: Enable
	30:24	**BarrierID.** When Barrier Enable is set, this field identifies which barrier was allocated for this thread. Format: U7
	23:0	Ignored.
M0.1	31:0	Ignored.
M0.0	31:24	**SLM ID.** This field encodes whether SLM was allocated for this thread, and if so, the SLM base offset used with SLM operations. Format: U8
	23:0	Ignored

Context Switch for GPGPU and Media

In the ExecList scheduling mode, software triggers preemption by submitting a new pending execlist to the ExecList Submit Port (ELSP). On detecting the availability of the new pending execlist, hardware triggers preemption on a preemptable command. Following successful preemption, the newly submitted execlist undergoes a context switch. As part of the context switch, the Logical Ring Context Address (LRCA) saves the preempted context state. The context state contains details for hardware to resume execution of a resubmitted context from the point where the context was preempted.

The GPGPU Pipe supports several types of preemption boundaries:

GPGPU Pipe Preemption Boundaries

Preemption Boundary	Description
Command (coarsest granularity)	The context is preempted after the current command is completed. The GPGPU Pipe is flushed, and the current context state is saved so that the next command in the command buffer will be executed when the context is resumed.
Threadgroup	The context is preempted in the middle of a GPGPU Walker or Media Walker command. After all previously dispatched threadgroups have completed, the current walker position is saved along with the current context state. The current walker command will be executed when the context is resumed and then re-start at the next theadgroup dispatch in the walker.

Preemption Boundary	Description
Mid-Thread (finest granularity)	The context is preempted in the middle of a GPGPU Walker command. After all previously dispatched threadgroups have started running in an EU, each running thread is preempted at the next interruptible EU instruction, all per-thread data is saved, the current walker position is saved along with the current context state. When the context is resumed, all the per-thread data is restored, the threads are re-started, and the current walker command is executed to re-start at the next threadgroup dispatch in the walker.

Programming Note	
Context:	Mid-Thread Preemption
Usually software selects either command or threadgroup preemption because they save less data as part of the context image and are responsive enough for the applications. Mid-thread preemption has a lot more overhead but sometimes guarantees a shorter preemption time when threadgroups run a long time.	

Programming Note	
Context:	Per-Context vs Global Selection of Preemption Boundary
The GPGPU Pipe preemption boundary is set by software by the **Media GPGPU Preemption Control**. There are two registers used set this mode: the Global power context preemption boundary in register FF_SLICE_CS_CHICKEN2 (0x20e4[2:1]), and the per-context preemption boundary in register CS_CHICKEN1 (0x2580[2:1]). When **Per Context Preemption Granularity Control** is clear in register FF_SLICE_CS_CHICKEN1 (0x20e0[14]), only the Global power context register is used to determine the preemption boundary. When **Per Context Preemption Granularity Control** is set in register FF_SLICE_CS_CHICKEN1 (0x20e0[14]), then the preemption boundary is set to the coarsest level specified by the Global and Per-Context preemption boundaries. This allows kernel mode drivers to override user mode settings of the preemption granularity.	

If preemption is signaled and the Preemption Control specifies a boundary not supported the currently running command or command streamer, then the preemption will occur on the boundary supported by that command and command streamer.

If a particular sequence of commands within a context needs to be protected from preemption, then it must be explicitly put between the MI_ARB_OFF/MI_ARB_ON command pair.

Preemption Boundaries supported by GPGPU Pipe Commands and Command Streamers

Recognized Preemption Boundaries	Current GPGPU Pipe Command	Which Render Engine Command Streamer is running the Command
• Mid-thread • Threadgroup • Command	GPGPU_WALKER	RCS

Recognized Preemption Boundaries	Current GPGPU Pipe Command	Which Render Engine Command Streamer is running the Command
• Threadgroup • Command	GPGPU_WALKER	CCS
• Threadgroup • Command	MEDIA_OBJECT_WALKER	RCS
• Command	All other GPGPU Pipe Commands, including: • MEDIA_VFE_STATE, • MEDIA_STATE_FLUSH, • MEDIA_LOAD_INTERFACE_DESCRIPTOR • MEDIA_CURBE_LOAD Any MI_ Command when MI_ARB_ON, including: • PIPE_CONTROL • PIPELINE_SELECT • MI_ARB_CHECK • MI_WAIT_FOR_EVENT • MI_SEMAPHORE_WAIT	RCS

3D and GPGPU Programs

Overview of kernel execution with EU instructions and Shared Function messages.

EU Overview

The instruction set is a general-purpose data-parallel instruction set optimized for graphics and media computations. Support for 3D graphics API (Application Programming Interface) Shader instructions is mostly native, meaning that GFX efficiently executes Shader programs. Depending on Shader program operation modes (for example, a Vertex Shader may be executed on a base of a vertex pair, while a Pixel Shader may be executed on a base of a 16-pixel group), translation from 3D graphics API Shader instruction streams into native instructions may be required. In addition, there are many specific capabilities that accelerate media applications. The following feature list summarizes the instruction set architecture:

- SIMD (single instruction multiple data) instructions. The maximum number of data elements per instruction depends on the data type.
- SIMD parallel arithmetic, vector arithmetic, logical, and SIMD control/branch instructions.
- Instruction level variable-width SIMD execution.
- Conditional SIMD execution via destination mask, predication, and execution mask.
- Instruction compaction.

- An instruction may executed in multiple cycles over a SIMD execution pipeline.
- Most instructions have three operands. Some instructions have additional implied source or destination operands. Some instructions have explicit dual destinations.
- Region-based register addressing.
- Direct or indirect (indexed) register addressing.
- Scalar or vector immediate source operand.
- Higher precision accumulator registers are architecturally visible.
- Self-modifying code is not allowed (instruction streams, including instruction caches, are read-only).

CoIssue/Multi Issue:

Multiple execution pipes operate simultaneously, scheduler selects an execution pipe based on their availability and capability. Each instruction will occupy multiple **execution slots** of the execution pipe based on their SIMD width,. e.g. a SIMD16 instruction executed on a SIMD4 wide execution pipe will occupy 4 slots.

Thread scheduling:

Threads are scheduled with the "oldest first" policy: a thread runs as long as no dependency is encountered. When a switch is required, the oldest thread i.e., the thread which has been spawned the first is the next to execute. After scheduling the next instruction from the currently executing thread, if any of the four units are free, the EU tries to fill them from instructions from other threads (processed in oldest to newest order).

A new "round-robin" scheduling policy is also introduced. This new policy issues the thread in a *round robin* fashion as opposed to *oldest first*. It is a global selection policy meaning that all the EUs are selected to run one policy or the other in an any given time.

Primary Usage Models

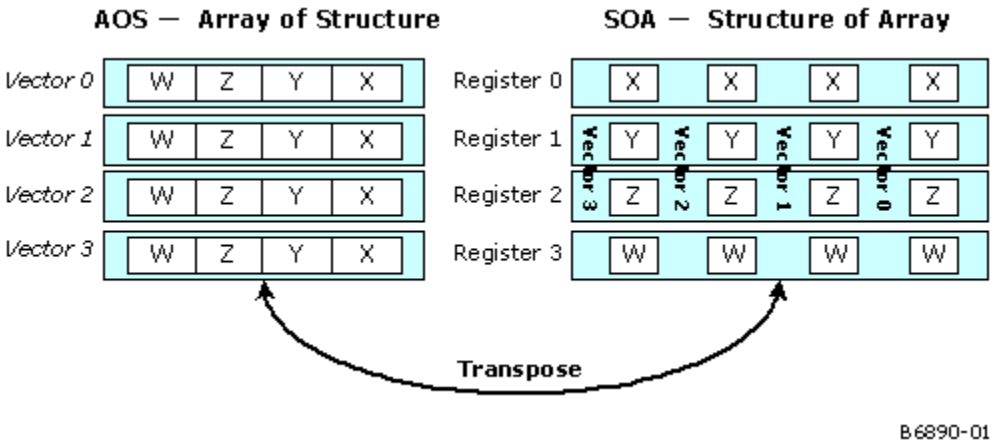
In describing the usage models of the instruction set, the following sections forward reference terminology, syntax, and instructions described later in this specification. For clarity reasons, not all forward references are explained at the point of reference. See the Instruction Set Summary chapter for information about instruction fields and syntax.

AOS and SOA Data Structures

With the Align1 and Align16 access modes, the instruction set provides effective SIMD computation whether data is arranged in array of structures (AOS) form or in structure of arrays (SOA) form. The AOS and SOA data structures are illustrated by the examples in *AOS and SOA Data Structures*. The example shows two different ways of storing four vectors in four SIMD registers. For simplicity, the data vector and the SIMD register both have four data elements. The four data elements in a vector are denoted by X, Y, Z, and W just as for a vertex in 3D geometry. The AOS structure stores one vector in a register and

the next vector in another register. The SOA structure stores one data element of each vector in a register and the next element of each vector in the next register and so on. The two structures can be related by a matrix transpose operation.

AOS and SOA Data Structures



3D and media applications take advantage of such broad architecture support and use both AOS and SOA data arrangements.

- Vertices in 3D Geometry (Vertex Shader and Geometry Shader) are arranged in AOS form and use SIMD4x2 and SIMD4 modes, respectively, as detailed below.
- Pixels in 3D Rasterization (Pixel Shader) are arranged in SOA form and use SIMD8 and SIMD16 modes as detailed below.
- Pixels in media are primarily arranged in SOA form, and occasionally in AOS form with possibly mixed modes of operation that uses region-based addressing extensively.

These are preferred methods; alternative arrangements may also be possible. Shared function resources provide data transpose capability to support both modes of operations:

The sampler has a transpose for sample reads, the data port has a transpose for render cache writes, and the URB unit has a transpose for URB writes.

The following 3D graphics API Shader instruction is used in the following sections to illustrate various operation modes:

```
add dst.xyz src0.yxzw src1.zwxy
```

This example is a SIMD instruction that takes two source operands src0 and src1, adds them, and stores the result to the destination operand dst. Each operand contains four floating-point data elements. The data type is determined by the instruction opcode. This instruction also uses source swizzles (.yxzw for src0 and .zwxy for src1) and a destination mask (.xyz). Please refer to the programming specifications of 3D graphics API Shader instructions for more details.

A general register has 256 bits, which can store 8 floating point data elements. For 3D graphics, the mode of operation is (loosely) termed after the data structure as SIMD*mxn*, where *m* is the size of the vector and *n* is the number of concurrent program flows executed in SIMD.

Execution with AOS data structures:

- **SIMD4** (short for SIMD4x1) indicates that a SIMD instruction operates on 4-element vectors storedin registers. There is one program flow.
- **SIMD4x2** indicates that a SIMD instruction operates on a pair of 4-element vectors in registers. There are effectively two programs running side by side with one vector per program.

Execution with SOA data structures, also referred to as "channel serial" execution, mostly uses:

- **SIMD8** (short for SIMD1x8) indicates a SIMD instruction based on the SOA data structure where one register contains one data element (the same one) for each of 8 vectors. Effectively, there are 8 concurrent program flows.
- **SIMD16** (short for SIMD1x16) indicates that a SIMD instruction operates on a pair of registers that contain one data element (the same one) for each of 16 vectors. SIMD16 has 16 concurrent program flows.

SIMD16 Mode of Operation

With 16 concurrent program flows, one element of a vector would take two GRF registers. In this mode, two corresponding vectors from the two program flows fill a register.

With the following register mappings,

src0:r2-r9 (with 16 X data elements in r2-r3, Y in r4-5, Z in r6-7 and W in r8-9),

src1:r10-r17,

dst:r18-r25,

the example 3D graphics API Shader instruction can be translated into the following three instructions:

add (16) r18<1>:f r4<8;8,1>:f r14<8;8,1>:f// dst.x = src0.y + src1.z

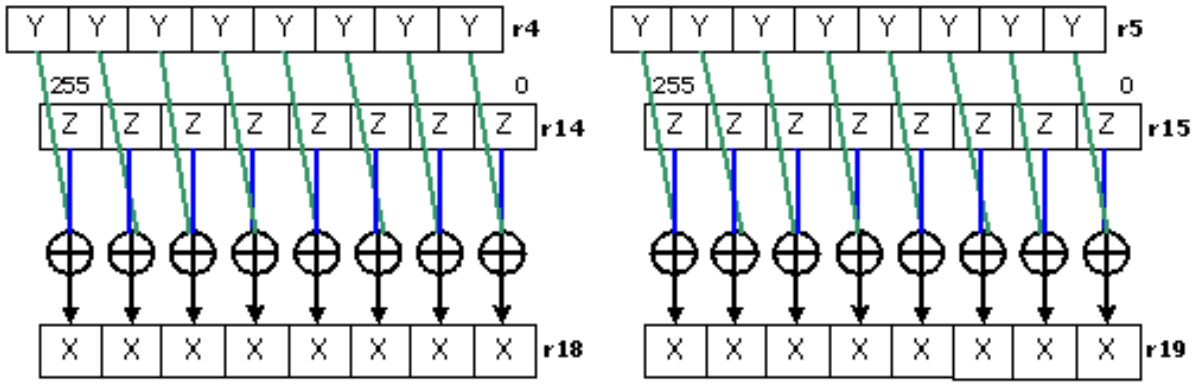
add (16) r20<1>:f r6<8;8,1>:f r16<8;8,1>:f// dst.y = src0.z + src1.w

add (16) r22<1>:f r8<8;8,1>:f r10<8;8,1>:f // dst.z = src0.w + src1.x

The three instructions correspond to the three enabled destination masks As there is no output for the W elements of dst, no instruction is needed for that element. The first instruction inputs the Y elements of src0 and the Z elements of src1 and outputs the X elements of dst. The operation of this instruction is shown in *SIMD16 Mode of Operation*.

With more than one program flow, the above instructions are also subject to the execution mask. The 16-bit dispatch mask is partitioned into four groups with four bits each. For Pixel Shader generated by the Windower, each 4-bit group corresponds to a 2x2 pixel subspan. If a subspan is not valid for a Pixel Shader instance, the corresponding 4-bit group in the dispatch mask is not set. Therefore, the same instructions can be used independent of the number of available subspans without creating bogus data in the subspans that are not valid.

A SIMD16 Example




```
Add (16) r18<1>:f r4<8;8,1>:f r14<8;8,1>:f {Compr}  // dst.x=src0.y+src1.z
```

B6894-01

Similar to SIMD4x2 mode, a constant may also be shared for the 16 program flows. For example, the first source operand could be a 4-element vector (e.g. a constant) stored in doublewords 0-3 in r2 (AOS format). The example 3D graphics API Shader instruction can then be translated into the following instruction:

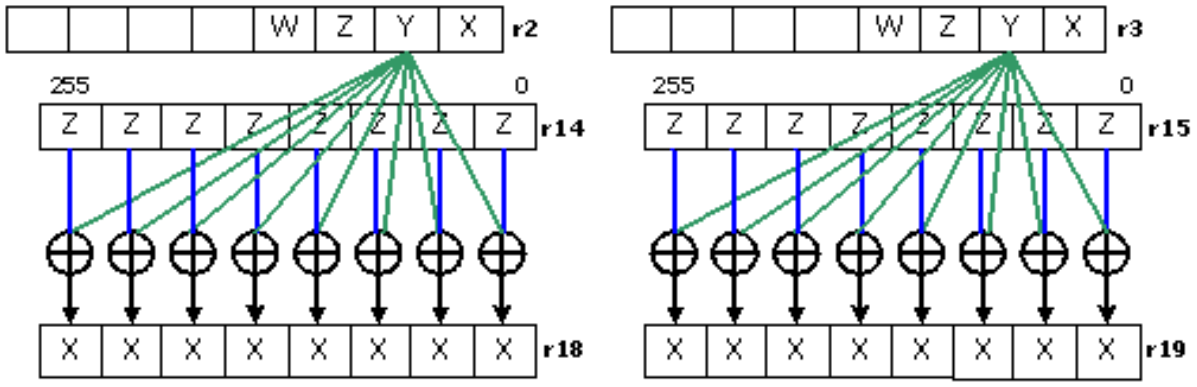
add (16) r18<1>:f r2.1<0;1,0>:f r14<8;8,1>:f {Compr}// dst.x = src0.y + src1.z

add (16) r20<1>:f r2.2<0;1,0>:f r16<8;8,1>:f {Compr}// dst.y = src0.z + src1.w

add (16) r22<1>:f r2.3<0;1,0>:f r10<8;8,1>:f {Compr}// dst.z = src0.w + src1.x

The register region of the first source operand represents a replicated scalar. The operation of the first instruction is illustrated in *SIMD16 Mode of Operation*.

Another SIMD16 Example with an AOS Shared Constant




```
Add (16) r18<1>:f r2.1<0;1,0>:f r14<8;8,1>:f {Compr}  // dst.x=src0.y+src1.z
```

B6895-01

SIMD8 Mode of Operation

SIMD8 Mode of Operation

Each compressed instruction has two corresponding native instructions. Taking the example instruction shown in Another SIMD16 Example with an AOS Shared Constant, it is equivalent to the following two instructions.

```
add (8) r18<1>:f r4<8;8,1>:f r14<8;8,1>:f // dst.x[7:0] = src0.y + src1.z
add (8) r19<1>:f r5<8;8,1>:f r15<8;8,1>:f {SecHalf}
// dst.x[15:8] = src0.y + src1.z
```

Therefore, SIMD8 can be viewed as a special case for SIMD16.

There are other reasons that SIMD8 instructions may be used. Within a program with 16 concurrent program flows, sometime SIMD8 instruction must be used due to architecture restrictions. For example, the address register a0 only have 8 elements, if an indirect GRF addressing is used, SIMD16 instructions are not allowed.

Messages

Communication between the EUs and the shared functions and between the fixed function pipelines (which are not considered part of the "Subsystem") and the EUs is accomplished via packets of information called *messages*. Message transmission is requested via the send instruction. Refer to the send instruction definition in the *ISA Reference* chapter for details.

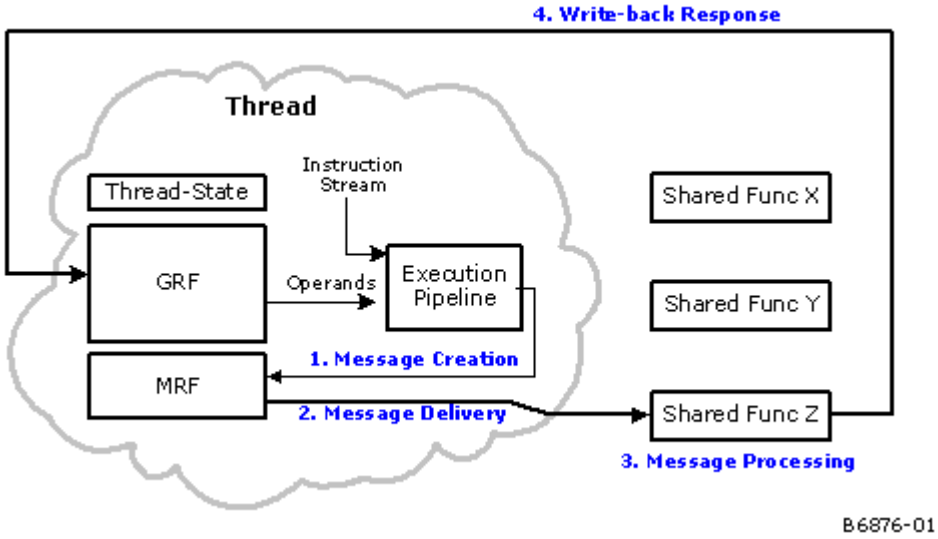
The information transmitted in a message falls into two categories:

- **Message Payload.**
- Associated ("sideband") information provided by:
  - **Message Descriptor.** Specified with the send instruction. Included in the message descriptor is control and routing information such as the target function ID, message payload length, response length, etc.
  - Additional information provided by the send instruction, e.g., the starting destination register number, the execution mask (EMASK), etc.
  - A small subset of Thread State, such as the Thread ID, EUID, etc.

The software view of messages is shown in Data Flow Associated With Messages. There are four basic phases to a message's lifetime as illustrated below:

1. **Creation.**
2. **Delivery.** The thread issues the message for delivery via the send instruction. The send instruction also specifies the destination shared function ID (SFID), and where in the GRF any response is to be directed. The messaging subsystem enqueues the message for delivery and eventually routes the message to the specified shared function.
3. **Processing.** The shared function receives the message and services it accordingly, as defined by the shared function definition.
4. **Writeback.** If called for, the shared function delivers an integral number of registers of data to the thread's GRF in response to the message.

Data Flow Associated With Messages



Message Payload Containing a Header

For most shared functions, the first register of the message payload contains the *header payload* of the message (or simply the *message header*).

Consequently, the rest of the message payload is referred to as the *data payload*.

Messages to Extended Math do not have a header and only contain data payload. Those messages may be referred to as header-less messages. Messages to Gateway combine the header and data payloads in a single message register.

Writebacks

Some messages generate return data as dictated by the 'function-control' (opcode) field of the 'send' instruction (part of the <desc> field). The execution unit and message passing infrastructure do not interpret this field in any way to determine if writeback data is to be expected. Instead, explicit fields in the 'send' instruction to the execution unit the starting GRF register and count of returning data. The execution unit uses this information to set in-flight bits on those registers to prevent execution of any instruction which uses them as an operand until the register(s) is(are) eventually written in response to the message. If a message is not expected to return data, the 'send' instruction's writeback destination specifier (<post_dest>) must be set to 'null' and the response length field of <desc> must be 0 (see 'send' instruction for more details).

The writeback data, if called for, arrives as a series of register writes to the GRF at the location specified by the starting GRF register and length as specified in the 'send' instruction. As each register is written back to the GRF, its in-flight flag is cleared, and it becomes available for use as an instruction operand. If a thread was suspended pending return of that register, the dependency is lifted, and the thread is allowed to continue execution (assuming no other dependency for that thread remains outstanding).

Message Delivery Ordering Rules

All messages between a thread and an individual shared function are delivered in the ordered they were sent. Messages to different shared functions originating from a single thread may arrive at their respective shared functions out of order.

The writebacks of various messages from the shared functions may return in any order. Further individual destination registers resulting from a single message may return out of order, potentially allowing execution to continue before the entire response has returned (depending on the dependency chain inherent in the thread).

Execution Mask and Messages

The Architecture defines an Execution Mask (EMask) for each instruction issued. This 32b bit-field identifies which SIMD computation channels are enabled for that instruction. Since the 'send' instruction is inherently scalar, the EMask is ignored as far as instruction dispatch is concerned. Further the execution size has no impact on the size of the 'send' instruction's implicit move (it is always 1 register regardless of specified execution size).

The 32b EMask is forwarded with the message to the destination shared function to indicate which SIMD channels were enabled at the time of the 'send'. A shared function may interpret or ignore this field as dictated by the functionality it exposes. For instance, the DataPort writes to the render cache ignoring this field completely, instead it uses the pixel mask included in-band in the message payload to indicate which channels carry valid data.

End-Of-Thread (EOT) Message

The final instruction of all threads must be a *send* instruction that signals 'End-Of-Thread' (EOT). An EOT message is one in which the EOT bit is set in the *send* instruction's 32b <desc> field. When issuing instructions, the EU looks for an EOT message, and when issued, shuts down the thread from further execution and considers the thread completed.

Only a subset of the shared functions can be specified as the target function of an EOT message, as shown in the table below.

Target Shared Functions supporting EOT messages	Target Shared Functions not supporting EOT messages
PixelPort, URB, ThreadSpawner	DataPort

Target Shared Functions supporting EOT messages	Target Shared Functions not supporting EOT messages
PixelPort, URB, ThreadSpawner	DataPort, Sampler

Both the fixed-functions and the thread dispatcher require EOT notification at the completion of each thread. The thread dispatcher and fixed functions in the 3D pipeline obtain EOT notification via the target shared functions.

The thread dispatcher, upon detecting an end-of-thread message, updates its accounting of resource usage by that thread, and is free to issue a new thread to take the place of the ended thread. Fixed functions require end-of-thread notification to maintain accounting as to which threads it issued have completed and which remain outstanding, and their associated resources such as URB handles.

Unlike the thread dispatcher, fixed-functions discriminate end-of-thread messages, only acting upon those from threads which they originated, as indicated by the 4b fixed-function ID present in R0 of end-of-thread message payload. This 4b field is attached to the thread at new-thread dispatch time and is placed in its designated field in the R0 contents delivered to the GRF. Thus, to satisfy the inclusion of the fixed-function ID, the typical end-of-thread message generally supplies R0 from the GRF as the first register of an end-of-thread message.

As an optimization, an end-of-thread message may be overload upon another "productive" message, saving the cost in execution and bandwidth of a dedicated end-of-thread message. Outside of the end-of-thread message, most threads issue a message just prior to their termination (for instance, a Pixel Port write to the framebuffer) so the overloaded end-of-thread is the common case. The requirement is that the message contains R0 from the GRF (to supply the fixed-function ID), and that destination shared function be either (a) the URB; (b) the Pixel Port; or, (c) the Thread Spawner, as these functions reside on the O-Bus. In the case where the last real message of a thread is to some other shared function, the thread must issue a separate message for the purposes of signaling end-of-thread.

Performance

The Architecture imposes no requirement as to a shared function's latency or throughput. Due to this as well as factors such as message queuing, shared bus arbitration, implementation choices in bus bandwidth, and instantaneous demand for that function, the latency in delivering and obtaining a response to a message is non-deterministic. It is expected that an implementation has some notion of fairness in transmission and servicing of messages so as to keep latency outliers to a minimum.

Other factors to consider with regard to performance:

Description
Software prefetching techniques may be beneficial for long latency data fetches (i.e. issue a load early in the thread for data that is required late in the thread).

Message Description Syntax

All message formats are defined in terms of DWords (32 bits). The message in all cases has the same width as general purpose register. The registers and DWords within the message are named as follows, where n is the register number, and d is the DWord number from 0. For writeback messages, the register number indicates the offset from the specified starting destination register.

Dispatch Messages: **R**n.d

Dispatch messages are sent by the fixed functions to dispatch threads. See the fixed function chapters in the *3D and Media* volume.

SEND Instruction Messages: **M**n.d

These are the messages initiated by the thread via the SEND instruction to access shared functions. See the chapters on the shared functions later in this volume.

Writeback Messages: **W**n.d

These messages return data from the shared function to the GRF where it can be accessed by thread that initiated the message.

The bits within each DWord are given in the second column in each table.

Message Errors

Messages are constructed via software, and not all possible bit encodings are legal, thus there is the possibility that a message may be sent containing one or more errors in its descriptor or payload contents. There are two points of error detection in the message passing system: (a) the message delivery subsystem is capable of detecting bad FunctionIDs and some cases of bad message lengths; (b) the shared functions contain various error detection mechanisms which identify bad sub-function codes, bad message lengths, and other misc errors. The error detection capabilities are specific to each shared function. The execution unit hardware itself does not perform message validation prior to transmission.

In both cases, information regarding the erroneous message is captured and made visible through MMIO registers, and the driver notified via an interrupt mechanism.

The set of possible errors is listed in Error Cases with the associated outcome.

Error Cases

Error	Outcome
Bad Shared Function ID	The message is discarded before reaching any shared function. If the message specified a destination, those registers will be marked as in-flight, and any future usage by the thread of those registers will cause a dependency which will never clear, resulting in a hung thread and eventual time-out.
Unknown opcode Incorrect message length	The destination shared function detects unknown opcodes (as specified in the 'send' instructions <desc> field), and known opcodes where the message payload is either too long or too short, and threats these cases as errors. When detected, the shared function latches and makes available via MMIO registers the following information: the EU and thread ID which sent the message, the length of the message and expected response, and any relevant portions of the first register (R0) of the message payload. The shared function alerts the driver of an erroneous message through and interrupt mechanism then continues normal operation with the subsequent message.
Bad message contents in payload	Detection of bad data is an implementation decision of the shared function. Not all fields may be checked by the shared function, so an erroneous payload may return bogus data or no data at all. If an erroneous value is detected by the shared function, it is free to discard the message and continue with the subsequent message. If the thread was expecting a response, the destination registers specified in the associated 'send'

Error	Outcome
	instruction are never cleared potentially resulting in a hung thread and time-out.
Incorrect response length	Case: too few registers specified - the thread may proceed with execution prior to all the data returning from the shared function, resulting in the thread operating on bad data in the GRF. Case: too many registers specified - the message response does not clear all the registers of the destination. In this case, if the thread references any of the residual registers, it may hand and result in an eventual time-out.
Improper use of End-Of-Thread (EOT)	Any 'send' instruction which specifies EOT must have a 'null' destination register. The EU enforces this and, if detected, will not issue the 'send' instruction, resulting in a hung thread and an eventual time-out. The 'send' instruction specifies that EOT is only recognized if the <desc> field of the instruction is immediate. Should a thread attempt to end a thread using a <desc> sourced from a register, the EOT bit will not be recognized. In this case, the thread will continue to execute beyond the intended end of thread, resulting in a wide range of error conditions.
Two outstanding messages using overlapping GRF destinations ranges	This is not checked by HW. Due to varying latencies between two messages, and out-of-order, non-contiguous writeback cycles, the outcome in the GRF is indeterminate; may be the result from the first message, or the result from the second message, or a combination of both.

Registers and Register Regions

Register Files

Registers are grouped into different name spaces called register files. There are two register files, the General Register File and the Architecture Register File. A third encoding of some register file instruction fields indicates immediate operands within instructions rather than a register file.

- General Register File (GRF): The GRF contains general-purpose read-write registers.
- Architecture Register File (ARF): The ARF contains all architectural registers defined for specific purposes, including address registers (*a#*), accumulators (*acc#*), flags (*f#*), notification count (*n#*), instruction pointer (*ip*), null register (*null*), etc.
- Immediate: Certain instructions can take immediate source operands. A distinct register file field encoding indicates an immediate operand.

Each thread executed in an EU has its own thread context, a dedicated register space that is not shared between threads, whether executing on a common EU or on a different EU. In the rest of the chapters in this volume, register access is relative to a given thread.

GRF Registers

Number of Registers:	Various
Default Value:	None
Normal Access:	RW
Elements:	Various
Element Size:	Various
Element Type:	Various
Access Granularity:	Byte
Write Mask Granularity:	Byte
Indexable?	Yes

Registers in the General Register File are the most commonly used read-write registers. During the execution of a thread, GRF registers are used to store the temporary data, and serve as the destination to receive data from shared function units (and sometimes from a fixed function unit). They are also used to store the input (initialization) data when a thread is created. By allowing fixed function hardware to initialize some portion of GRF registers during thread dispatch time, architecture can achieve better parallelism. A thread's execution efficiency can also be improved as some data are already in the register to be executed upon. Besides these registers containing thread's payload, the rest of GRF registers of a thread are not initialized.

Summary of GRF Registers

Register File	Register Name	Description
General Register File (GRF)	r#	General purpose read write registers

Each execution unit has a fixed size physical GRF register RAM. The GRF register RAM is shared by all threads on the EU. Each thread has a dedicated space of 128 or 256 register, r0 through r127 or r255.

GRF registers can be accessed using region-based addressing at byte granularity (both read and write). A source operand must be contained within two adjacent registers. A destination operand must be contained within one register. GRF registers support direct addressing and register-indirect addressing. Register-indirect addressing uses the address registers (ARF registers a#) and an immediate address offset value.

When accessing (read and/or write) outside the GRF register range allocated for a given thread either through direct or indirect addressing, the result is unpredictable.

Register Size	Allocation Granularity	Number per Thread
256 bits	Fixed allocation of 128 registers	128 registers

ARF Registers

ARF Registers Overview

GRF registers are directly indicated by a unique register file encoding. Besides registers that are directly indicated by a unique register file coding, all other registers belong to the Architecture Register File (ARF). Encodings of architecture register types are based on the MSBs of the register number field, RegNum, in the instruction word. The RegNum field has 8 bits.

The 4 MSBs, RegNum[7:4], represent the Architecture **Register Type**. This is summarized in the *Summary of Architecture Registers* table below.

RegNum[3:0], represent the Architecture **Register Number**.

Summary of Architecture Registers

Register Type	Register Name	Description
0000b	*null*	Null register
0001b	*a0.#*	Address register
0010b	*acc#*	Accumulator register
0011b	*f#.#*	Flag register
0100b	*ce#*	Channel Enable register
0101b	*msg#*	Message Control Register
0110b	*sp*	Stack Pointer Register
0111b	*sr0.#*	State register
1000b	*cr0.#*	Control register
1001b	*n#*	Notification Count register
1010b	*ip*	Instruction Pointer register
1011b	*tdr*	Thread Dependency register
1100b	*tm0*	Pause register
1101b	*fc#.#*	Flow Control register
1110b	*Reserved*	Reserved

Description
The remaining **Register Number** field RegNum[3:0] is used to identify the register number of a given architecture register type. Therefore, the maximum number of registers for a given architecture register type is limited to 16. The subregister number field, SubRegNum, in the instruction word has 5 bits. It is used to address subregister regions for an architecture register supporting register subdivision. The SubRegNum field is in units of bytes. Therefore, the maximum number of bytes of an architecture register is limited to 32. Depending on the alignment restriction of a register type, only certain encodings of SubRegNum field apply for an architecture register. The detailed definitions are provided in subsequent sections.

In general, an ARF register can be dst (destination) or src0 (source 0, first source operand) for an instruction. Depending on the register and the instruction, other restrictions may apply.

Access Granularity

ARF registers may be accessed with subregister granularity according to the descriptions below and following the same rule of region-based addressing for GRF.

The machine code for register number and subregister number of ARF follows the same rule as for other register files with byte granularity. For an ARF as a source operand, the region-based address controls the source swizzle mux. The destination subregister number and destination horizontal stride can be used to generate the destination write mask at byte level.

Subregister fields of an ARF register may not all be populated (indicated by the subregister being indicated as reserved). Writes to unpopulated subregisters are dropped; there are no side effect. Reads from unpopulated subregisters, if not specified, return unpredictable data.

Some ARF registers are read-only. Writes to read-only ARF registers are dropped and there are no side effects.

Null Register

Null Register Summary

Attribute	Value
ARF Register Type Encoding (RegNum[7:4]):	0000b
Number of Registers:	1
Default Value:	N/A
Normal Access:	N/A
Elements:	N/A
Element Size:	N/A
Element Type:	N/A
Access Granularity:	N/A
Write Mask Granularity:	N/A
SecHalf Control?	N/A
Indexable?	No

The null register is a special encoding for an operand that does not have a physical mapping. It is primarily used in instructions to indicate non-existent operands. Writing to the null register has no side effect. Reading from the null register returns an undefined result.

The null register can be used where a source operand is absent. For example, for a single source operand instruction such as MOV or NOT, the second source operand src1 must be a null register.

When the null register is used as the destination operand of an instruction, it indicates the computed results are not stored in any registers. However, implied writes to the accumulator register, if applicable,

may still occur for the instruction. When the conditional modifier is present, updates to the selected flag register also occur. In this case, the register region fields of the 'null' operand are valid.

Another example use is to use the null register as the posted destination of a *send* instruction for data write to indicate that no write completion acknowledgement is required. In this case, however, the register region fields are still valid. The null register can also be the first source operand for a send instruction indicating the absent of the implied move. See the *send* instruction for details.

Address Register

Address Register Summary

Attribute	Value
ARF Register Type Encoding (RegNum[7:4]):	0001b
Number of Registers:	1
Default Value:	None
Normal Access:	RW
Elements:	16
Element Size:	16 bits
Element Type:	UW or UD
Access Granularity:	Word
Write Mask Granularity:	Word
SecHalf Control?	N/A
Indexable?	No

There are sixteen address subregisters forming a 16-element vector. Each address subregister contains 16 bits. Address subregisters can be used as regular source and destination operands, as the indexing addresses for register-indirect-addressed access of GRF registers, and also as the source of the message descriptor for the *send* instruction.

Register and Subregister Numbers for Address Register

RegNum[3:0]	SubRegNum[4:0]
0000b = a0 All other encodings are reserved.	When register a0 or subregisters in a0 are used as the address register for register-indirect addressing, the address subregisters must be accessed as unsigned word integers. Therefore, the subregister number field must also be word-aligned. 00000b = **a0.0:uw** 00010b = **a0.1:uw** 00100b = **a0.2:uw** 00110b = **a0.3:uw** 01000b = **a0.4:uw** 01010b = **a0.5:uw**

RegNum[3:0]	SubRegNum[4:0]
	01100b = **a0.6:uw** 01110b = **a0.7:uw** 10000b = **a0.8:uw** 10010b = **a0.9:uw** 10100b = **a0.10:uw** 10110b = **a0.11:uw** 11000b = **a0.12:uw** 11010b = **a0.13:uw** 11100b = **a0.14:uw** 11110b = **a0.15:uw** All other encodings are reserved. However, when register a0 or subregisters in a0 are explicit source and/or destination registers, other data types are allowed as long as the register region falls in the 128-bit range.

Address Register Fields

DWord	Bits	Description
7	31:16	**Address subregister a0.15:uw**. Follows the same format as **a0.3**.
	15:0	**Address subregister a0.14:uw**. Follows the same format as **a0.2**.
6	31:16	**Address subregister a0.13:uw**. Follows the same format as **a0.3**.
	15:0	**Address subregister a0.12:uw**. Follows the same format as **a0.2**.
5	31:16	**Address subregister a0.11:uw**. Follows the same format as **a0.3**.
	15:0	**Address subregister a0.10:uw**. Follows the same format as **a0.2**.
4	31:16	**Address subregister a0.9:uw**. Follows the same format as **a0.3**.
	15:0	**Address subregister a0.8:uw**. Follows the same format as **a0.2**.
3	31:16	**Address subregister a0.7:uw**. Follows the same format as **a0.3**.
	15:0	**Address subregister a0.6:uw**. Follows the same format as **a0.2**.
2	31:16	**Address subregister a0.5:uw**. Follows the same format as **a0.3**.
	15:0	**Address subregister a0.4:uw**. Follows the same format as **a0.2**.
1	31:16	**Address subregister a0.3:uw**. This field can be used for register-indirect register addressing or serve as extended descriptor for a *send* instruction. When used for register-indirect register addressing, it is a 12-bit unsigned integer. For a *send* instruction, it provides the higher 16 bits of <exdesc>. Format: U12 or U16

DWord	Bits	Description
	15:0	**Address subregister a0.2:uw**. This field can be used for register-indirect register addressing or serve as extended descriptor for a *send* instruction. When used for register-indirect register addressing, it is a 12-bit unsigned integer. For a *send* instruction, it provides the lower 16 bits of <exdesc>. Format: U12 or U16
0	31:16	**Address subregister a0.1:uw**. This field can be used for register-indirect register addressing or serve as message descriptor or extended descriptor for a *send* instruction. When used for register-indirect register addressing, it is a 12-bit unsigned integer. For a *send* instruction, it provides the higher 16 bits of <desc> or <exdesc>. Format: U12 or U16.
	15:0	**Address subregister a0.0:uw**. This field can be used for register-indirect register addressing or serve as message descriptor or extended descriptor for a *send* instruction. When used for register-indirect register addressing, it is a 12-bit unsigned integer. For a *send* instruction, it provides the lower 16 bits of <desc> or <exdesc>. Format: U12 or U16.

When used as a source or destination operand, the address subregisters can be accessed individually or as a group. In the following example, the first instruction moves 8 address subregisters to the first half of GRF register r1, the second instruction replicates a0.4:uw as an unsigned word to the second half of r1, the third instruction moves the first 4 words in r1 into the first 4 address subregisters, and the fourth instruction replicates r1.4 as a unsigned word to the next 4 address subregisters.

```
mov (8) r1.0<1>:uw a0.0<8;8,1>:uw // r1.n = a0.n for n = 0 to 7 in words
mov (8) r1.8<1>:uw a0.4<0;1,0>:uw // r1.m = a0.4 for m = 8 to 15 in words
mov (4) a0.0<1>:uw r1.0<4;4,1>:uw // a0.n = r1.n for n = 0 to 3 in words
mov (4) a0.4<1>:uw r1.4<0;1,0>:uw // a0.m = r1.4 for m = 4 to 7 in words
```

When used as the register-indirect addressing for GRF registers, the address subregisters can be accessed individually or as a group. When accessed as a group, the address subregisters must be group-aligned. For example, when two address subregisters are used for register indirect addressing, they must be aligned to even address subregisters. In the following example, the first instruction is legal. However, the second one is not. As ExecSize = 8 and the width of src0 is 4, two address subregisters are used as row indices, each pointing to 4 data elements spaced by HorzStride = 1 dword. For the first instruction, the two address subregisters are a0.2:uw and a0.3:uw. The two align to a DWord group in the address register. However, the two address subregisters for the second instruction are a0.3:uw and a0.4:uw. They are not DWord-aligned in the address register and therefore violate the above mentioned alignment rule.

```
mov (8) r1.0<1>:d r[a0.2]<4,1>:d // a0.2 and a0.3 are used for src1
mov (8) r1.0<1>:d r[a0.3]<4,1>:d // ILLEGAL use of register indirect
```

Programming Note	
Context:	ARF Registers
Implementation restriction: When used as the source operand <desc> for the send instruction, only the first dword subregister of a0 register is allowed (i.e. a0.0:ud, which can be viewed as the combination of a0.0:uw and a0.1:uw). In addition, it must be of UD type and in the following form <desc> = a0.0<0;1,0>:ud.	

Programming Note	
Context:	ARF Registers
Performance Note: There is only one scoreboard for the whole address register. When a write to some subregisters is in flight, hardware stalls any instruction writing to other subregisters. Software may use the destination dependency control {NoDDChk, NoDDClr} to improve performance in this case. Similarly, when a write to some subregisters is in flight, hardware stalls any instruction sourcing other subregisters until the write retires.	

Accumulator Registers

Accumulator Registers Summary

Attribute	Value
ARF Register Type Encoding (RegNum[7:4]):	0010b
Number of Registers:	2+8
Default Value:	None
Normal Access:	RW
Elements:	8
Element Size:	33 bits
Element Type:	UW & UD
Access Granularity:	Word
Write Mask Granularity:	Word
SecHalf Control?	No
Indexable?	No

Accumulator registers can be accessed either as explicit or implied source and/or destination registers. To a programmer, each accumulator register may contain either 8 DWords or 16 Words of data elements. However, as described in the Implementation Precision Restriction notes below, each data element may have higher precision with added guard bits not indicated by the numeric data type.

Accumulator capabilities vary by data type, not just data size, as described in the Accumulator Channel Precision table below. For example, D and F are both 32-bit data types, but differ in accumulator support.

See the Accumulator Restrictions section for information about additional general accumulator restrictions and also accumulator restrictions for specific instructions.

Register and Subregister Numbers for Accumulator Registers

RegNum[3:0]	SubRegNum[4]	SubRegNum[3:0]
0000b-1001b = acc0-acc9 All other encodings are reserved.	0 : Lower half 1 : Upper half	Reserved: MBZ

- Accumulators are updated implicitly only if the AccWrCtrl bit is set in the instruction. The Accumulator Disable bit in control register cr0.0 allows software to disable the use of AccWrCtrl for implicit accumulator updates. The write enable in word granularity for the instruction is used to update the accumulator. Data in disabled channels is not updated.
- When an accumulator register is an implicit source or destination operand, hardware always uses acc0 by default and also uses acc1 if the execution size exceeds the number of elements in acc0. When implicit access to acc1 is required, QtrCtrl is used. Note that QtrCtrl can be used only if acc1 is accessible for a given data type. If acc1 is not accessible for a given data type, QtrCtrl defaults to acc0.

Description
acc0 and acc1 are supported for half-precision (HF, Half Float) and single-precision (F, Float). Use QtrCtrl of Q2 or Q4 to access acc1 for Float. use QtrCtrl of H2 to access acc1 for Half Float.

Examples:

```
// Updates acc0 and acc1 because it is SIMD16:
add (16) r10:f r11:f r12:f {AccWrEn}
// Updates acc0 because it is SIMD8:
add (8) r10:f r11:f r12:f {AccWrEn}
// Updates acc1. Implicit access to acc1 using QtrCtrl:
add (8) r10:f r11:f r12:f {AccWrEn, Q2}
// Updates acc1 for Half Floats using QtrCtrl:
add (16) r10:hf r11:hf r12:hf {AccWrEn, H2}
```

- It is illegal to specify different accumulator registers for source and destination operands in an instruction (e.g. "*add (8) acc1:f acc0:f*"). The result of such an instruction is unpredictable.
- Accumulator registers may be accessed explicitly as src0 operands only.
- Register Regioning patterns where register data bit locations are changed between source and destination are not supported when an accumulator is used as an implicit source or an explicit source in an instruction.
- Reading accumulator content with a datatype different from the previous write will result in undeterministic values.
- Word datatype write to accumulator is not allowed when destination is odd offset strided by 2.
- For any DWord operation, including DWord multiply, accumulator can store up to 8 channels of data, with only acc0 supported.
- When an accumulator register is an explicit destination, it follows the rules of a destination register. If an accumulator is an explicit source operand, its register region must match that of the destination register with the exception(s) described below.

Implementation Precision Restriction: As there are only 64 bits per channel in DWord mode (D and UD), it is sufficient to store the multiplication result of two DWord operands as long as the post source modified sources are still within 32 bits. If any one source is type UD and is negated, the negated result becomes 33 bits. The DWord multiplication result is then 65 bits, bigger than the storage capacity of accumulators. Consequently, the results are unpredictable.

Implementation Precision Restriction: As there are only 33 bits per channel in Word mode (W and UW), it is sufficient to store the multiplication result of two Word operands with and without source modifier as the result is up to 33 bits. Integers are stored in accumulator in 2's complement form with bit 32 as the sign bit. As there is no guard bit left, the accumulator can only be sourced once before running into a risk of overflowing. When overflow occurs, only modular addition can generate a correct result. But in this case, conditional flags may be incorrect. When saturation is used, the output is unpredictable. This is also true for other operations that may result in more than 33 bits of data. For example, adding UD (FFFFFFFFh) with D (FFFFFFFFh) results in 1FFFFFFFEh. The sign bit is now at bit 34 and is lost when stored in the accumulator. When it is read out later from the accumulator, it becomes a negative number as bit 32 now becomes the sign bit.

Accumulator Channel Precision

Data Type	Accumulator Number	Number of Channels	Bits Per Channel	Description
DF	acc0	4	64	When accumulator is used for Double Float, it has the exact same precision as any GRF register.
F	acc0/acc1	8	32	When accumulator is used for Float, it has the exact same precision as any GRF register.
HF	acc0/acc1	16	16	When accumulator is used for Half Float, it has the exact same precision as any GRF register.
Q	N/A	N/A	N/A	Not supported data type.
D (UD)	acc0	8	33/64	When the internal execution data type is doubleword integer, each accumulator register contains 8 channels of (extended) doubleword integer values. The data are always stored in accumulator in 2's complement form with 64 bits total regardless of the source data type. This is sufficient to construct the 64-bit D or UD multiplication results using an instruction macro sequence consisting of *mul*, *mach*, and *shr* (or *mov*). Writing to acc1 using any datatype may corrupt this result.
W (UW)	acc0	16	33	When the internal execution data type is word integer, each accumulator register contains 16 channels of (extended) word integer values. The data are always stored in accumulator in 2's complement form with 33 bits total. This supports single instruction multiplication of two-word sources in W and/or UW format. Writing to acc1 using any datatype may corrupt this result.
B (UB)	N/A	N/A	N/A	Not supported data type.

Math Macro Extended Accumulators
Special Accumulators acc2-acc9

These are accumulator registers defined for a special purpose. They are used to emulate IEEE-compliant fdiv and sqrt macro-operations. The access is different from *acc0* and *acc1*, which are defined as full 256-bit registers having 8 DWords and may be accessed explicitly or implicitly. Conversely, these math macro extended accumulators consist of just a few bits and have very restricted access.

Example:

These registers may be accessed explicitly by a *mov* operation, with no source modifiers, condition modifiers, or saturation. When accessed explicitly, the datatype must be D. On reads, the low 2 bits of each DWord are valid data. The other bits are undefined. On writes, the low two bits are updated, and other bits are dropped.
// Move 256 bits from acc8 to r10. Just low two bits of each DWord are valid:
mov (8) r10:ud acc8:ud
// Move 256 bits from r10 to acc8. Just low two bits of each DWord are updated:
mov (8) acc8:ud r10:ud

These registers are accessed directly by math macro opcodes only. **Note:** These macro operations are *madm* and some others defined under the *math* opcode section. The macro descriptions also define the restrictive implicit uses of these registers.

Implicit access across accumulator registers is required for each source operand for these macro instructions. The SubRegNum bits in the instruction are used to implicitly address the different accumulators. The noacc value is specified when no write to accumulator is required; think of it as a null.

SpecialAcc[3:0] Encoding	Accumulator Register
0000b	acc2
0001b	acc3
0010b	acc4
0011b	acc5
0100b	acc6
0101b	acc7
0110b	acc8
0111b	acc9
1000b	noacc

The MSB bits [4:1] of the source SubRegNum are used to specify the SpecialAcc[3:0], similarly the destination SubRegNum bits are used to specify SpecialAcc[3:2] with the lower bits SpecialAcc[1:0] directly specified in the instruction.

Flag Register

Flag Register Summary

Attribute	Value
ARF Register Type Encoding (RegNum[7:4]):	0011b
Number of Registers:	2
Default Value:	None
Normal Access:	RW
Elements:	2
Element Size:	16 bits
Element Type:	UD & UW
Access Granularity:	Word
Write Mask Granularity:	Word
SecHalf Control?	Yes
Indexable?	No

Each flag register contains two 16-bit subregisters. Each flag bit corresponds to a data channel. Predication uses flag values to enable or disable channels. Conditional modifiers assign flag values. If an instruction uses both predication and conditional modifiers, both features use the same flag register or subregisters.

Flags can be split to halfs, quarters, or eighths using the QtrCtrl and NibCtrl instruction fields. Those fields affect the selection of flags for predication and conditional modifiers, but do not affect reading or writing flags as explicit instruction operands.

The values held in the individual bits of a flag register are the result of the most recent instruction with a conditional modifier and specifying that flag register. For example:

```
add.nz.f0.0 ...
```

Updates flag subregister f0.0 with the per-channel results of the not zero condition.

The flag register has per-bit write enables. When being updated as the secondary destination associated with a conditional modifier, only the bits corresponding to the enabled channels in *EMask* are updated. Other bits in the flag subregister are unchanged.

Flag registers and subregisters can also be explicit source or destination operands.

The *sel* instruction does not update flags.

Register and Subregister Numbers for Flag Register

RegNum[3:0]	SubRegNum[4:0]
0000b = **f0:ud** 0001b = **f1:ud** Other encodings are reserved.	00000b = **f*n*.0:uw** 00010b = **f*n*.1:uw** Other encodings are reserved.

f0:ud or f1:ud etc reference an entire flag register. f0.0:uw, f0.1:uw, fn.0:uw, and fn.1:uw reference the flag subregisters.

Channel Enable Register

Channel Enable Register Summary

Attribute	Value
ARF Register Type Encoding (RegNum[7:4]):	0100b
Number of Registers:	1
Default Value:	N/A
Normal Access:	RO
Elements:	1
Element Size:	32 bits
Element Type:	UD
Access Granularity:	DWord
Write Mask Granularity:	N/A
SecHalf Control?	No
Indexable?	No

Register and Subregister Numbers for Channel Enable Register

RegNum[3:0]	SubRegNum[4:0]
0000b = ce All other encodings are reserved.	00000b = **ce:ud**. All other encodings are reserved.

Channel Enable Register Fields

DWord	Bits	Description
0 (ce:ud)	31:0	**Channel Enable Register** Format: U32 This field contains 32 bits of Channel Enables for the current instruction.

Message Control Registers

Message Control Register Summary

Attribute	Value
ARF Register Type Encoding (RegNum[7:4]):	0101b
Number of Registers:	1
Default Value:	None
Normal Access:	RO/PW
Elements:	1
Element Size:	32 bits
Element Type:	UD
Access Granularity:	Word
Write Mask Granularity:	Word
SecHalf Control?	No
Indexable?	No

Register and SubRegister Numbers for Message Control Registers

RegNum[3:0]	SubRegNum[4:0]
0000b = **msg0**	00000b = **msg0.0:ud**. All other encodings are reserved.

These are specific control registers used to track messaging. These may be saved and restored by the kernel only when a thread is in the context save/restore mode. Access of these registers otherwise, will result in undeterministic behaviour.

Each thread has 1 registers. The granularity of access is always one full register. These registers must be accessed with a MOV with no predication, src modifiers or conditional modifiers. They MUST be accessed in direct addressing more only.

DWord	Bits	Description
0 (msg0:ud)	31:22	**Reserved**. MBZ.

Stack Pointer Register

Stack Pointer Register Summary

Attribute	Value
ARF Register Type Encoding (RegNum[7:4]):	0110b
Number of Registers:	1
Default Value:	Provided by the Dispatcher
Normal Access:	RW
Elements:	2
Element Size:	64 bits
Element Type:	UD
Access Granularity:	DWord
Write Mask Granularity:	DWord
SecHalf Control	No
Indexable	No

The stack pointer register can be accessed as a unsigned DWord or QWord integer. It consists of a read-write register, containing the current stack pointer (sp.0), which is relative to the Generate State Base Address, and a read-only stack limit register (sp.1). The stack pointer is inserted into the message header when data is stored into scratch space as a stack. The stack pointer is managed by software. If the stack pointer exceeds the limit or the space allocated, an exception is triggered. See the Stack Pointer Exception in the Exceptions Section. Writes to the stack pointer must use the {Switch} ThreadCtrl option.

Register and Subregister Numbers for SP Register

RegNum[3:0]	SubRegNum[4:0]
0000b = sp All other encodings are reserved.	00000b = **sp.0:uq**. 01000b = **sp.1:uq** All other encodings are reserved.

SP Register Fields

DWord	Bits	Description
0..1 (sp.0:ud)	63:48	**Reserved.** MBZ.
	47:0	**sp.0**. Specifies the current stack pointer. This pointer is relative to the General State Base Address. This register is initialized at thread load to the top of the per thread Scratch Space. The register is R/W. *sp = [scratch space pointer] + [scratch space] - 1* Alternatively, this register may be updated by Software to any flat address space. In such cases, the stack is NOT relative to the General State Base Address. Software must ensure that the address is

DWord	Bits	Description
		exclusive for the thread.
2..3 (sp.1:ud)	63:48	**Reserved.** MBZ.
	47:0	**sp.1**. Specifies the upper limit for the stack pointer. This pointer is relative to the General State Base Address. This register is initialized at thread load to the limit allocated for stack in the state. See the GPGPU Thread Payload description for details. The register is R/W. *sp_limit = [scratch space pointer] + [stack space limit]* Alternatively, this register may be updated by software, similar to the sp register. In such cases, software is responsible for allocating the right thread stack pointer limit.

State Register

State Register Summary

Attribute	Value
ARF Register Type Encoding (RegNum[7:4]):	0111b
Number of Registers:	1
Default Value:	Provided by the Dispatcher
Normal Access:	RW
Elements:	4
Element Size:	32 bits
Element Type:	UD
Access Granularity:	Byte
Write Mask Granularity:	N/A
SecHalf Control?	No
Indexable?	No

Register and Subregister Numbers for State Register

RegNum[3:0]	SubRegNum[4:0]
0000b = sr0 All other encodings are reserved.	Valid encoding range: 00000b = **sr0.0:ud** 00100b = **sr0.1:ud** 01000b = **sr0.2:ud** 01100b = **sr0.3:ud** All other encodings are reserved.

State Register Fields

DWord	Bits	Description
0 (sr0.0:ud)	31:28	**Reserved**. MBZ.
	27:24	**FFID** (Fixed Function Identifier). Specifies which fixed function unit generates the current thread. This field is set at thread dispatch and is forwarded on the message bus for all out-going messages from this thread. Initialized at dispatch. Read-only.
	23	**Priority Class**. This field, when set, indicates the thread belongs to the high priority class, which has higher scheduling priority over any thread with this field cleared. The priority field below determines the relative priority within the same priority class. This field is initialized by the thread dispatcher at thread dispatch time and stays unchanged throughout the life span of the thread. This field is forwarded on the message bus to the message bus arbiter for all out-going messages. It serves as a priority hint for the target shared function. See the Shared Function chapters for whether and how a shared function uses this priority hint. This bit is honored only when Threads Execution Arbitration Mode is set to Age Based. 0 = Low priority class. 1 = High priority class. Initialized at dispatch. Read-only.
	22:20	**Priority**. This field is the relative aging priority of the thread. This field indicates the 'age' of this thread relative to other threads within the EU. No two threads in the same EU can have the same priority number (independent of the priority class value). Within the same priority class, an older thread (with a larger priority number) has higher schedule priority over a younger thread. This field is set to zero at a thread's dispatch. During a thread's run time, this field may or may not be incremented when a new thread is dispatched to the same EU. It is only incremented when another thread's priority number is incremented and reaches the same value. For example, if currently there is a thread with priority 0 on an EU, then dispatching a new thread to that EU causes the old thread's priority number to increment to 1. However, if the active thread (assuming for simplicity that there is only one) on an EU has a priority number 1 (or 2 or 3), then dispatching a new thread to this EU does not change the old thread's priority number. As threads on an EU may terminate in arbitrary order, the exact number for a thread depends on the dynamic execution of threads. When thread context is saved and restored after pre-emption, the Priority is not restored to the original state. Instead the priority is initiated as if new threads were loaded. Initialized to 0. Read-only.
	19:14	**Reserved**. MBZ.
	13:4	[13:11] Slice ID. [10:9] Dual-SubSlice ID

<table>
<tr><th>DWord</th><th>Bits</th><th>Description</th></tr>
<tr><td></td><td></td><td>[8] SubSlice ID.
[7] : EUID[2]
[6] : Reserved
[5:4] EUID[1:0]
Initialized by hardware. Read-only.</td></tr>
<tr><td></td><td>3</td><td>Reserved. MBZ.</td></tr>
<tr><td></td><td>2:0</td><td>TID (The thread identifier). Specifies the thread slot that the current thread is assigned to. This field is set at thread dispatch.
Initialized by hardware. Read-only.</td></tr>
<tr><td>1
(sr0.1:ud)</td><td>31:21</td><td>FFTID (Fixed Function Thread ID). There is no connection between this thread ID, assigned in fixed functions, and the TID assigned in the EUs.</td></tr>
<tr><td></td><td>15:7</td><td>Reserved. MBZ</td></tr>
<tr><td></td><td>5:0</td><td>IEEE Exception. The exception bits are sticky bits set by the opcode when one of the exception is triggered. These bits are defined per thread and all channels update one sticky bit. These bits may be cleared by software or on a thread load. Updates to these bits may be turned OFF by the IEEE Exception trap enable in the CR register. When these bits are required as source of an operation, the previous instruction must use a {Switch}. This ensures all asynchronous flag updates are complete before using as source operand. The following table describes these bits:
Workaround: These bits will have undefined value if a previously saved GPGPU context is restored for execution. All new contexts will have these bits initialized to zero.
<table>
<tr><th>Bits</th><th>Definition</th></tr>
<tr><td>5</td><td>Reserved</td></tr>
<tr><td>4</td><td>Inexact Exception</td></tr>
<tr><td>3</td><td>Overflow</td></tr>
<tr><td>2</td><td>Underflow</td></tr>
<tr><td>1</td><td>Divide by Zero</td></tr>
<tr><td>0</td><td>Invalid Operation</td></tr>
</table>
Initialized to 0. Read/Write</td></tr>
<tr><td>2
(sr0.2:ud)</td><td>31:0</td><td>Dispatch Mask (DMask). This 32-bit field specifies which channels are active at Dispatch time. This field is used by hardware to initialize the mask register.
Format: U32
Initialized at dispatch. Read/Write.</td></tr>
<tr><td>3</td><td>31:0</td><td>Vector Mask (VMask). This 32-bit field contains, for each 4-bit group, the OR of the corresponding 4-bit group in the dispatch mask. This field is used by hardware to initialize the</td></tr>
</table>

DWord	Bits	Description
(sr0.3:ud)		mask register. Format: U32 Initialized at dispatch. Read-only.

Implementation Restriction on Register Access: When the state register is used as a source and/or destination, hardware does not ensure execution pipeline coherency. Software must make the immediate next instruction dependent on an instruction that uses state register as an explicit operand. This is important as the state register is an implicit source or destination for many instructions. For example, fields like IEEE Exception may be an implicit destination updated by multiple back-to-back instructions. Therefore, if the instructions updating the state register doesn't make the next instruction stall, subsequent instructions may have undefined results.

Control Register

Control Register Summary

Attribute	Value
ARF Register Type Encoding (RegNum[7:4]):	1000b
Number of Registers:	1
Default Value:	Provided by the Dispatcher
Normal Access:	RW/PW
Elements:	4
Element Size:	32 bits
Element Type:	UD
Access Granularity:	DWord
Write Mask Granularity:	DWord
SecHalf Control?	No
Indexable?	No

The Control register is a read-write register. It contains four 32-bit subregisters that can be accessed individually.

Subregister *cr0.0:ud* contains normal operation control fields such as the floating-point mode and the accumulator disable. It also contains the master exception status/control field that allows software to switch back to the application thread from the System Routine.

Subregister *cr0.1:ud* contains the mask and status/control fields for all exceptions. The exception fields are arranged in significance-decreasing order from MSB to LSB. This arrangement allows the System Routine to use the *lzd* instruction to find the high priority exceptions and handle them first. As each exception is mapped to a single bit, another exception priority order may be implemented by software. The System Routine may choose to handle one exception at a time, by handling the exception detected by an *lzd*instruction and returning to the application thread. Or it may choose to handle all the concurrent exceptions, by looping through the exception fields until all outstanding exceptions are

handled before returning back to the application thread.Exception enable bits (bits 15:0 in *cr0.1:ud*) control whether an exception causes hardware to jump to the System Routine or not. Exception status and control bits (bits 31:16 in *cr0.1:ud*) indicate which exceptions have occurred, and are used by the system routine to clear the exception. Even if a given exception is disabled, the corresponding exception status and control bit still reflects its status, whether an exception event has occurred or not.

cr0.2:ud contains the **Application IP (AIP)** indicating the current thread IP when an exception occurs.

cr0.3:ud is reserved. Values written to this subregister are dropped; the result of reading from this subregister is unpredictable.

Fields in Control registers also reference a virtual register called **System IP (SIP)**. SIP is the virtual register holding the global System IP, which is the initial instruction pointer for the System Routine. The SIP is a GraphicsAddress. There is only one SIP for the whole system. It is virtual only from a thread's point of view, as it is not visible (i.e. not readable and not writeable) to the thread software executed on an EU. It can only be accessed indirectly by the hardware to respond to exception events. Upon an exception, hardware performs some bookkeeping (e.g. saving the current IP into AIP) and then jumps to SIP. Upon finishing exception handling, the System Routine may return back to the application by clearing the Master Exception Status and Control field in *cr0*, which causes the hardware to load AIP to IP register. See the STATE_SIP command for how to set SIP.

Although the SIP may be more than 32 bits wide, the EU still only uses the low 32 bits.

Register and Subregister Numbers for Control Register

RegNum[3:0]	SubRegNum[4:0]
0000b = cr0 All other encodings are reserved.	00000b = **cr0.0:ud**. It contains general thread control fields. 00100b = **cr0.1:ud**. It contains exception status and control. 01000b = **cr0.2:ud**. It contains AIP. All other encodings are reserved.

Control Register Fields

DWord	Bits	Description
0	31	**Master Exception State and Control.** This bit is the master state and control for all exceptions. Reading a 0 indicates that the thread is in normal operation state and a 1 means the thread is in exception handle state. Upon an exception event, hardware sets this bit to 1 and switches to SIP. Writing 1 to this bit has no effect. Writing 0 to this bit also has no effect if the previous value is 0. In both cases, the bit keeps the previous value. If the previous value of this bit is 1, software writing a 0 causes the thread to return to AIP. This transition is automatic - software does not have to move AIP to IP. The value of this bit then stays as 0. 0 = The thread is in normal state. 1 = The thread is in exception state. Initialized by hardware. Read/Clear.

DWord	Bits	Description
	30:16	**Reserved.** MBZ.
	14	**Thread preemption disable:** This field specifies whether the thread is allowed to stop in middle on receiving mid-thread preemption request. 0 = Thread is preempted on receiving preemption indication. 1 = Thread is preempted only in case of page-fault.
	13	**Reserved.** MBZ.
	12	**IEEE Float to Integer Rounding Mode.** This bit determines how rounding modes are handled in float to integer conversion operation. This bit is initialized to 0 during Thread Dispatch. This bit must be set for IEEE compliant float to integer conversion operation. 0= The result of float to integer conversion operation is with RTZ rounding mode. 1= The result of float to integer conversion operation is with rounding mode programmed in cr0.0[5:4]. Initialized to 0. Read/Write.
	11	**IEEE MinMax.** This bit determines how SNAN is handled in min/max operations. Refer to IEEE Floating Point Mode for details. This bit is initialized to 0 during Thread Dispatch. This bit must be set for IEEE compliant min/max operation. 0 = The result of min/max is a non-SNAN source. 1 = The result of min/max is the SNAN source. Initialized to 0. Read/Write.
	10	**Half Precision Denorm Mode.** This bit determines how denormal numbers are handled for the HF (Half Float) type. This bit is initialized to 0 during Thread Dispatch. 0 = Flush denorms to zero when reading source operands and flush denorm calculation results to zero. Denorm flushing preserves sign. 1 = Allow denorm source values and denorm results. Initialized to 0. Read/Write.
	9	**IEEE Exception Trap Enable.** This bit enables trapping IEEE exception flags. This control bit may be updated by software. It is initially zero on thread load. If enabled, IEEE floating-point exceptions set sticky bits in the IEEE Exceptions field of sr0.1. Note that IEEE floating-point exceptions do *not* transfer control to any handler. 0 = IEEE Exception flags are NOT trapped. 1 = IEEE Exception flags are trapped. Initialized to 0. Read/Write.
	7	**Single Precision Denorm Mode.** This bit determines how denormal numbers are handled for the

DWord	Bits	Description
		F (Float) type when using the IEEE floating-point mode. It is ignored in the ALT floating-point mode, which always flushs denorms. This bit is initialized by Thread Dispatch. 0 = Flush denorms to zero when reading source operands and flush denorm calculation results to zero. Denorm flushing preserves sign. 1 = Allow denorm source values and denorm results. Initialized at dispatch. Read/Write.
	6	**Double Precision Denorm Mode.** This bit determines how denormal numbers are handled for the DF (Double Float) type. It is initialized by Thread Dispatch. 0 = Flush denorms to zero when reading source operands and flush denorm calculation results to zero. Denorm flushing preserves sign. 1 = Allow denorm source values and denorm results. Initialized to 0. Read/Write.
	5:4	**Rounding Mode.** This field specifies the FPU rounding mode. It is initialized by Thread Dispatch. 00b = Round to Nearest or Even (RTNE) 01b = Round Up, toward +inf (RU) 10b = Round Down, toward -inf (RD) 11b = Round Toward Zero (RTZ) Initialized at dispatch. Read/Write.
	3	**Vector Mask Enable (VME).** This bit indicates DMask or Vmask should be used by EU for execution. This bit is set by the Thread Dispatch. 0: Use Dispatch Mask (DMASK) 1: Use Vector Mask (VMASK) Initialized at dispatch. Read/Write.
	2	**Single Program Flow (SPF).** Specifies whether the thread has a single program flow (SIMDnxm with m = 1) or multiple program flows (SIMDnxm with m > 1). This bit affects the operation of all branch instructions. In Single Program Flow mode, all execution channels branch and/or loop identically. This bit is initialized by the Thread Dispatch. 0: Multiple Program Flows 1: Single Program Flow **Programming Restrictions:** Only H1/Q1/N1 are allowed in SPF mode. Power Optimization: If an entire shader does not do SIMD branching, the driver can set the SPF bit to 1 to save power in HW.

DWord	Bits	Description
		SPF mode must be set to 0 in Fused threads. Initialized at dispatch. Read/Write.
	1	**Accumulator Disable.** This bit controls the update of the accumulator by the instruction field AccWrCtrl. If this bit is cleared, the accumulator is updated for all instructions with AccWrCtrl enabled. If set, the accumulator is disabled for all update operations, maintaining its value prior to being disabled. Setting this bit has no effect if the accumulator is the explicit destination operand for an instruction. This bit is initialized to 0. 0: Enable accumulator update. 1: Disable accumulator update. **Usage Notes:** This control bit is primarily designed for the System Routine. That routine is not expected to use the accumulator, though it may need to use instructions that implicitly update the accumulator. To use such instructions in the System Routine, but still preserve the accumulator contents on returning to the application kernel, the System Routine would either (a) save and restore the accumulator, or (b) prevent the accumulator from being unintentionally modified. This control bit has been added for the latter method. Software has the option to limit the setting of this control bit to strictly within the System Routine. If, by convention, this bit is clear within application kernels, the System Routine can simply set the bit upon entry and clear it before returning control to the application kernel. This usage model would not necessarily require cr0.0 to be saved/restored in the System Routine. However, if by convention application kernels are permitted to set this bit, then the System Routine is required to preserve the content of this bit. Initialized to 0. Read/Write.
	0	**Single Precision Floating Point Mode (FP Mode).** This bit specifies whether the current single-precision floating-point operation mode is IEEE mode (IEEE Standard 754) or the ALT (alternative mode). This bit does not affect the floating-point mode used for other floating-point data types. Refer to Alternative Floating Point Mode for details. This bit is also forwarded on the message sideband for all out-going messages, for example, to control the floating-point mode of the Sampler. Software may modify this bit to dynamically switch between the two floating-point modes. This bit is initialized by Thread Dispatch. 0 = IEEE floating-point mode for the F (Float) type. 1 = ALT (alternative) floating-point mode for the F (Float) type. Initialized at dispatch. Read/Write.
1	30	**External Halt Exception Status and Control**. This bit indicates the External Halt exception. It is set by EU hardware on receiving the broadcast External Halt signal. The System Routine should reset this bit before returning to an application routine to avoid infinite loops. This bit may be set or cleared by software. This bit is initialized to 0. Initialized by hardware. Read/Clear.

DWord	Bits	Description
	29	**Software Exception Control**. This bit is the control bit for software exceptions. Setting this bit to 1 in an application routine causes an exception. Clearing this bit in an application routine has no effect. Upon entering the system routine, the hardware maintains this bit as 1 to signify a software exception. The System Routine should reset this bit before returning to an application routine. This bit may be set or cleared by software. This bit is initialized to 0. Initialized to 0. Read/Write.
	28	**Illegal Opcode Exception Status**. This bit, when set, indicates an illegal opcode exception. The exception handler routine normally does not return back to the application thread upon an illegal opcode exception. Leaving this bit set has no effect on hardware; if system software adversely returns to an application routine leaving this bit set, it doesn't cause any exception. This bit should not be set by software or left set by the system routine to avoid confusion. Initialized to 0. Read/Clear.
	27	**Stack Overflow Exception Status**. This bit when set, indicates a stack overflow exception. The exception handler routine normally does not return back to the application thread upon a stack overflow exception. Leaving this bit set has no effect on hardware; if system software adversely returns to an application routine leaving this bit set, it doesn't cause any exception. This bit should not be set by software or left set by the system routine to avoid confusion. This bit is initialized to 0.
	25	**Context Save Status.** This bit when set, indicates a Context Save process has been initiated. The system routine must reset this bit after saving the context to terminate the thread.
	24	**Context Restore Status.** This bit when set, indicates a Context Restore process has been initiated. The system routine must reset this bit after restoring the context. The reset of this bit is required before invoking application routine.
	23:16	**Reserved**. MBZ.
	13	**Software Exception Enable**. This bit enables or disables the software exception. Enabling or disabling this bit may allow host software to turn on/off certain features (such as profiling) without changing the kernel program. This bit is initialized by the Thread Dispatcher. Format = ENABLED: 0: Disabled 1: Enabled Initialized at dispatch. Read/Write.
	12	**Illegal Opcode Exception Enable**. This bit specifies whether the illegal opcode exception is enabled or not. The Illegal opcode exception includes illegal opcodes and undefined opcodes, caused by bad programs or run-time data corruption.

<table>
<tr><th>DWord</th><th>Bits</th><th>Description</th></tr>
<tr><td></td><td></td><td>This bit is initialized by the Thread Dispatcher.

Software should normally assign this bit in the interface descriptor. Even though this mechanism is provided to disable the illegal opcode exception, it should be used with extreme caution.

Format = ENABLED:

0: Disabled

1: Enabled

Initialized at dispatch. Read/Write.</td></tr>
<tr><td></td><td>11</td><td>Stack Overflow Exception Enable. This bit specifies whether the stack overflow exception is enabled or not. The stack overflow exception includes an overflow or an underflow in the stack space allocated for the thread.

This bit is initialized by the Thread Dispatcher.

Software should normally assign this bit in the interface descriptor.

Format = ENABLED:

0: Disabled

1: Enabled

Initialized at dispatch. Read/Write.</td></tr>
<tr><td></td><td>10:0</td><td>Reserved. MBZ.</td></tr>
<tr><td>2

(cr0.2:ud)</td><td>31:3</td><td>Application IP (AIP). This is the register storing the instruction pointer before an exception is handled. Upon an exception, hardware automatically saves the current IP into the AIP register, and then sets the Master Exception State and Control field to 1, which forces a switch to the System IP (SIP). The AIP register may contain either the pointer to the instruction that causes the exception (such as breakpoint) or the one after (such as masked stack overflow/underflow exceptions). This is shown in the following table, where IP is the instruction that generated the exception.

<table><tr><th>Exception Type</th><th>AIP Value</th></tr><tr><td>External Halt</td><td>N/A (1)</td></tr><tr><td>Software Exception</td><td>IP + 1</td></tr><tr><td>Illegal Opcode</td><td>IP</td></tr></table>
(1) External Halt exception is asynchronous and not associated with an instruction.

When the System Routine changes the Master Exception State and Control field from 1 to 0, hardware restores IP from this register. This field is writable allowing the returning IP to be altered after an exception is handled.

Format = U32[31:3]

Not Initialized. Read/Write.</td></tr>
<tr><td></td><td>2:0</td><td>Reserved. MBZ.</td></tr>
</table>

Implementation Restriction on Register Access: When the control register is used as an explicit source and/or destination, hardware does not ensure execution pipeline coherency. Software must make the immediate next instruction dependent on an instruction that uses control register as an explicit operand. This is important as the control register is an implicit source for most instructions. For example, fields like FPMode and Accumulator Disable control the arithmetic and/or logic instructions. Therefore, if the instruction updating the control register doesn't make the next instruction stall, subsequent instructions may have undefined results.

Notification Registers

Notification Registers Summary

Attribute	Value
ARF Register Type Encoding (RegNum[7:4]):	1001b
Number of Registers:	1
Default Value:	No
Normal Access:	RO/PW
Elements:	2
Element Size:	32 bits
Element Type:	UD
Access Granularity:	DWord
Write Mask Granularity:	DWord
SecHalf Control?	No
Indexable?	No

Register and Subregister Numbers for Notification Registers

RegNum[3:0]	SubRegNum[4:0]
0000b = n0 All other encodings are reserved.	00000b = **n0.0:ud** 00100b = **n0.1:ud** 01000b = **n0.2:ud** All other encodings are reserved.

There are three notification registers (*n0.0:ud*, *n0.1:ud, and n0.2:ud*) used by the *wait* instruction. These registers are read-only, except under context restore, and can be accessed in 32-bit granularity. Write access to this register is allowed only when context is restored.

Programming Note	
Context:	Notification Registers - Workaround
The sub-register numbers for n0.0 and n0.2 are swapped on a write, i.e., a destination of n0.0 is required to update n0.2 and n0.2 is required to update n0.0.	

It should be noted that in the extreme case, it is possible to have more notifications to a thread than the maximum allowed number of concurrent threads in the system. Therefore, the range of the thread-to-thread notification count in n0, is larger than the maximum number of threads computed by EUID * TID.

There is only one bit for the host-to-thread notification count in n1.

When directly accessed, this register is read-only. If the value is non zero, the only way to alter the value is to use the wait instruction to decrement the value until zero is reached. A wait instruction on a zero notification subregister causes the thread to stall, waiting for a notification signal from outside targeting the same subregister. See the wait instruction for details.

Implementation Restriction: The notification registers are initialized to 0 after hardware/software reset. However, these registers are not reset at thread dispatch time.

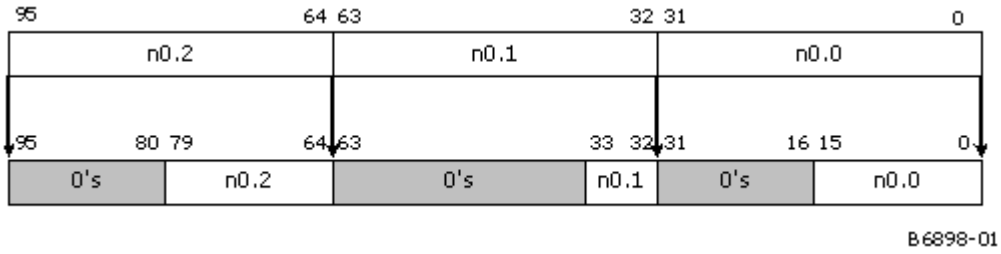
Notification Register 0 Fields

Dword	Bits	Description
0	31:1	**Reserved**. MBZ.
	0	**Thread to Thread Notification Count**. This register is used by the sync.bar instruction for thread-to-thread synchronization. The value read from this register specifies the outstanding notifications received from other threads. It can be changed indirectly by using the sync.bar instruction. See the sync.bar instruction for details. Format: U1 Initiallized to 0. Read/PWrite.

Notification Register 1 Fields

DWord	Bits	Description
0	31:1	**Reserved**. MBZ.
	0	**Host to Thread Notification**. This register is used by the sync.host instruction for host-to-thread synchronization via MMIO registers. Format: U1 Initiallized to 0. Read/PWrite.

Format of the Notification Register



IP Register

IP Register Summary

Attribute	Value
ARF Register Type Encoding (RegNum[7:4]):	1010b
Number of Registers:	1
Default Value:	Provided by the Dispatcher
Normal Access:	RW
Elements:	1
Element Size:	32 bits
Element Type:	UD
Access Granularity:	DWord
Write Mask Granularity:	DWord
SecHalf Control?	No
Indexable?	No

The ip register can be accessed as a 32-bit quantity. It is a read-write register, containing the current instruction pointer, which is relative to the **Instruction Base Address**. Reading this register returns the instruction pointer of the current instruction. The 3 LSBs are always read as zero. Writing this register causes program flow to jump to the new address. Writes to this register should use the Switch ThreadCtrl option. When it is written, the 3 LSBs are dropped by hardware.

Register and Subregister Numbers for IP Register

RegNum[3:0]	SubRegNum[4:0]
0000b = ip All other encodings are reserved.	00000b = **ip:ud** All other encodings are reserved.

IP Register Fields

DWord	Bits	Subfield Description
0	31:3	**Ip**. Specifies the current instruction pointer. This pointer is relative to the **Instruction Base Address**.
	2:0	**Reserved**. MBZ.

Performance Registers

Performance Registers Summary

Attribute	Value
ARF Register Type Encoding (RegNum[7:4]):	1100b
Number of Registers:	1
Default Value:	0h
Normal Access:	RO/RW
Elements:	5
Element Size:	32 bits
Element Type:	UD
Access Granularity:	DWord
Write Mask Granularity:	DWord
SecHalf Control?	No
Indexable?	No

Timestamp Register

This register is a low latency timestamp source, "TM", available as part of a thread's Architectural Register File (ARF). This is a is free running counter, 64b in size, and exposed to the ISA as individual 32b high 'TmHigh' and low 'TmLow' unsigned integer source operands. As part of the EU's register space, access to the timestamp has a low and deterministic latency and therefore can be used for intra-kernel high resolution performance profiling.

The TM features provides a 1-bit indicator 'TmEvent' which identifies the occurrence of a time-impacting event such as context switch or frequency change since the last time any portion of the Timestamp register value was read by that thread. Software that uses the Timestamp capability should check this bit to identify when a relative time calculation may be suspect. To properly use this additional information, the instrumentation code should operate on the Timestamp register value as a whole (i.e. as an 8 dword register) so that the 64b time and this 1b value are captured simultaneously, as opposed to 32b portions, to eliminate the chance of missing a TmEvent that might occur between accesses to 32b portions of this register.

Programming Note	
Context:	Performance Registers
The Timestamp register need not be saved as part of thread state on context-save, but only 'TmEvent' must be set to 1 as part of context restored to indicate that context switch had occurred).	

Performance Counter Register

This is a counter intended to provide finer grained visibility into the EUs performance inside kernels. This counter is a 32-bit free-running counter that increments if the EU flexible performance event selected for OA counter A7 is true (please refer to OA documentation for details on how to count various EU flexible

events on OA counter A7). The pm0 count continues to increment during a thread's active/standby state transitions as well as context switches. It is read-only and not pre- or resettable under any software control, either kernel or driver, other than a full gfx reset.

Pause Register

This register provides the mechanism for a thread to pause itself from further execution for a short amount of time. This may be useful in situations where a periodic polling operation on an external resource is required, but polling loop time needs to be controlled to prevent excessive consumption of execution slots and resource bandwidth. To mitigate excessive polling rates, this 'pause' operation can be placed in the polling loop to cap the periodic polling at some maximum rate.

The 'pause' operation is invoked by the kernel itself through the writing of the Pause Register with an unsigned-word value. The register over the course of many clocks will count down from the written value to zero, in steps of 32 decrements every 32 EU clocks (first decrement event has an uncertainty between 1 to 32 clocks). Upon reaching 0x0, the decrement stops. During the time the Pause Register is non-zero, no new instruction issue occurs; upon reaching a 0-count, the thread once again becomes a candidate for execution. Note that any instruction or message issued prior to the invocation of the 'pause' continues to execute and retire. Actual resumption of instruction issues after the pause duration may further be delayed through normal operational policies such as thread priority and/or outstanding register dependencies.

Therefore, the value written for 'pause' should be considered a number of instruction issue slots (divide by 2 for SIMD-8 instruction slots, or 4 for SIMD-16 instruction slots), as opposed to some fixed time duration.

The Pause Register is reset to 0x0 when a new thread is loaded. It is also reset to 0x0 upon invocation of the System Thread IP (SIP) and at the commencement of a context save or context restore event. The value of the register is not saved/restored as part of context save/restore.

The actual duration of the pause is considered approximate; generally the duration will be somewhat longer than the value written, as the counting does don't commence until the write to the 'Pause' register actually retires.

Register and Subregister Numbers for Performance Register

RegNum[3:0]	SubRegNum[4:0]
0000b = *tm0* All other encodings are reserved.	00000b = **tm0.0:ud.** 00100b = **tm0.1:ud.** 01000b = **tm0.2:ud** 01100b = **tm0.3:ud** 10000b = **tm0.4:ud** **All other encodings are reserved.**

Performance Register Fields

DWord	Bits	Description
0 (tm0.0:ud)	31:0	**TmLow.** The lower 32b of the 64b timestamp value sourced from Cr clock. Read-only. Format: U32
1 tm0.1:ud	31:0	**TmHigh.** The upper 32b of the 64b timestamp value sourced from Cr clock. Read-only. Format: U32
2 tm0.2:ud	31:1	**Reserved**
	0	**TmEvent.** Indicates a discontinuous time-impacting event (e.g. context switch, frequency change) occurred since any portion of the Timestamp register was last read, thus making any relative duration calculation based on this counter suspect. This bit is reset at the time a new thread is loaded, and on each read of any portion of the 'Timestamp' register.
3 tm0.3 (pm0)	31:0	pm0. Increments based on the EU flexible performance event currently selected being true. Format: U32
4 tm0.4:ud (tp0)	31:16	**Reserved**
	15:0	**Pause Counter.** The pause duration. A non-zero value written to this register causes execution of the thread to halt for the corresponding number of clocks. Lower 5 bits are always zero and therefore, writing value less than 64 may not result in a pause. [15:10] - Reserved, must be written as zero; when read, returns zero. [9:5] - Count value. [4:0] - Reserved, must be zero. Format: U16

Flow Control Registers

Summary

Attribute	Value
ARF Register Type Encoding (RegNum[7:4]):	1101b
Number of Registers:	4
Default Value:	None
Normal Access:	R/PW
Elements:	32
Element Size:	64 bits
Element Type:	UD
Access Granularity:	Dword

Attribute	Value
Write Mask Granularity:	Dword
SecHalf Control?	No
Indexable?	No

Register and Subregister Numbers for Flow Control Registers

RegNum[3:0]	SubRegNum[4:0]
0000b = fc0	00000b-11111b = **fc0.0-fc0.31**.
0001b = fc1	00000b = **fc1.0**. All other encodings are reserved.
0010b = fc2	00000b = **fc2.0**. All other encodings are reserved.
0011b = fc3	00000b = **fc3.0**. All other encodings are reserved.

These are special hardware registers used in handling flow control operations. These registers may be accessed ONLY in context save/restore operation using the SIP. These registers are accessed with the 'MOV' opcode. Use of any other opcode or access of these registers in non-context save/restore modes may result in undeterministic behaviour of hardware.

These registers are accessed as 256b registers. Parts of the 256b register may be redundant, depending on the hardware implementation of each register. The fields "RegNum" and "SubRegNum" are used together to address these registers.

Immediate

Two forms of immediate are provided as a source operand: scalar and vector.

The immediate field may be 64 bits or 32 bits. For a word, unsigned word, or half-float immediate data, software must replicate the same 16-bit immediate value to both the lower word and the high word of the 32-bit immediate field in an instruction. The 64-bit immediate takes up two DWords of the instruction bit field. Hence a 64-bit immediate is supported ONLY for a MOV operation. The field is denoted by ***imm32:type*** for 32-bit immediates and ***imm64:type*** for 64-bit immediates.

The immediate form of vector allows a constant vector to be in-lined in the instruction stream. Both integer and float immediate vectors are supported.

An immediate integer vector is denoted by type **v** or **uv** as *imm32:v* or *imm32:uv*, where the 32-bit immediate field is partitioned into 8 4-bit subfields. Refer to the Numeric DataType topic for description of the packing of vector integers to a DWord.

For a scalar immediate, the numeric data types supported are :uw, :w, :ud, :d, :uq, :q for integers AND :hf, :f, :df for floats. Refer to the Instruction Machine format topics for the encoding of these immediates.

An immediate float vector is denoted by type **vf** as *imm32:vf*, where the 32-bit immediate field is partitioned into 4 8-bit subfields. Refer to the Numeric DataType topic for the description of the packing of vector floats to a DWord.

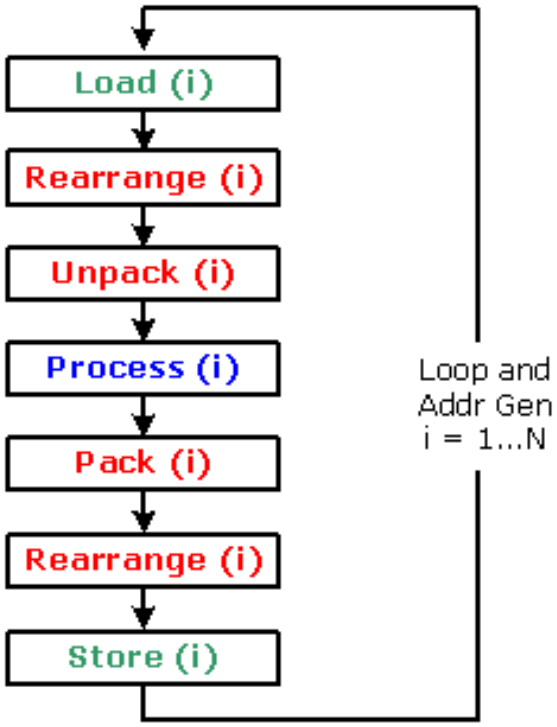
When an immediate vector is used in an instruction, the destination must be 128-bit aligned with destination horizontal stride equivalent to a word for an immediate integer vector (***v***) and equivalent to a DWord for an immediate float vector (***vf***).

Region Parameters

Unlike conventional SIMD architectures where an N-bit wide SIMD instruction can only operate on N-bit aligned SIMD data registers, a region-based register addressing scheme is employed in architecture. The region-based register addressing capability significantly improves the SIMD computation efficiency by providing per-instruction-based multiple data gathering from register file. This avoids instruction overhead to perform data pack, unpack, and shuffling, which has been observed on other SIMD architectures. One benefit of such capability is allowing various kinds of 3D Graphics API Shader compute models to run efficiently on GFX. Another benefit is allowing high throughput of media applications, which tend to operate on byte or word data elements.

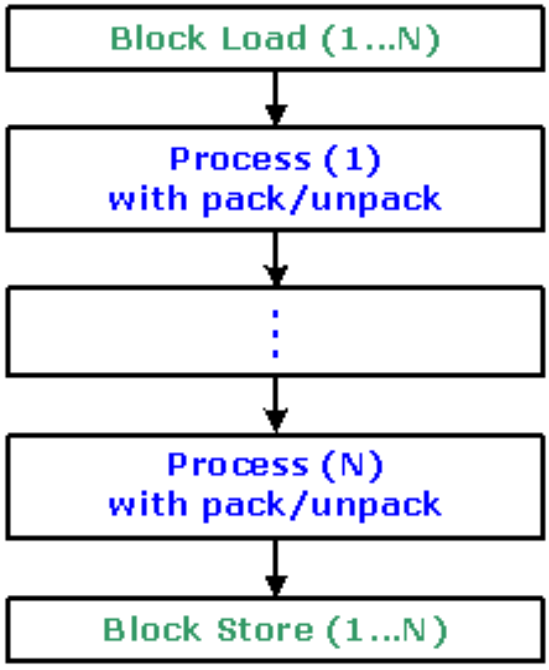
This can be illustrated by the example shown in Conventional SIMD Instruction Sequence and SIMD Instruction Sequence for the Same Program. As shown in Conventional SIMD Instruction Sequence, a sequence of SIMD instruction is executed on a conventional load/store based superscalar machine with SIMD instruction extension. The data parallelism can be achieved by first level of loop unrolling. As shown, there is a second level of loop for the task. Before a given SIMD compute instruction, *Process (i)*, can proceed, there might be a load, a data rearrange, and a data unpack (and conversion) instruction to load and prepare the input data. After the compute instruction is complete, it might also require pack, re-arrange and store instructions, to format and save the same to memory. At the loop, other scalar computations such as loop count and address generation may be needed. For the same program, when the data can fit in the large GRF register file, the outer loop may be unrolled for GFX. Here one or a few block loads (using *send* instruction) may be sufficient to move the working set into GRF. Then the data shuffle can be combined with the processing operation with region-based addressing capability. Per operand float type and mixed data type operation may also allow GFX to combine data conditioning operations with computing operations. These techniques in architecture help to achieve high compute efficiency and throughput for graphics and media applications.

Conventional SIMD Instruction Sequence



B6899-01

SIMD Instruction Sequence for the Same Program



B6900-01

In an instruction, each operand defines a region in the register file. A region may contain multiple data elements. Each data element is assigned to an execution channel in the EU. The total number of data elements of a region is called the **size** of the region, or the size of the operand. The number of execution channels is called the **execution size** (*ExecSize*), which is specified in the instruction word. ExecSize determines the size of region for source and destination operands in an instruction.

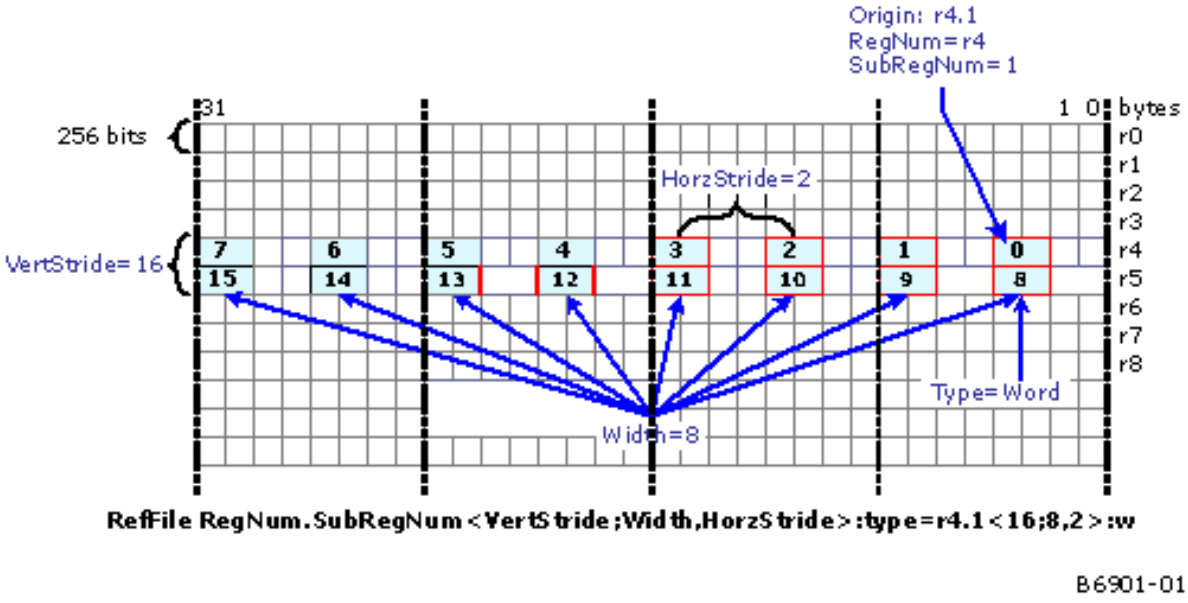
- For an instruction with two source operands, the sizes of the two source operands must be the same.
- The size of a destination operand generally matches the execution size, therefore equals to the number of source operand(s) in the same instruction.
    - Exception of this rule is present for the integer reduction instructions (such as sad2 and sada2) where the destination area is smaller than the source area.

Regions are **generalized 2-dimensional** (2D) arrays in row-major order. The first dimension is named the **horizontal** dimension (data elements within a row) and the second dimension is termed the **vertical** dimension (data elements in a column). Here, horizontal/vertical and row/column are just symbolic notations.

When the GRF registers are viewed as a row-major 2D array of memory, such a notation normally matches well with the geometric locations of the data elements of an operand.

However, as the register region is fully described by the parameters discussed below, the data elements of a register region may not form a regular rectangular shape. For example, Vertical Stride parameter is allowed to be smaller than Horizontal Stride, making the rows of a register region interleave with each other. It should also note that the meanings of horizontal/vertical here is different than that used for the flag control in Section Flag Register.

An example of a register region (*r4.1<16;8,2>:w*) with 16 elements

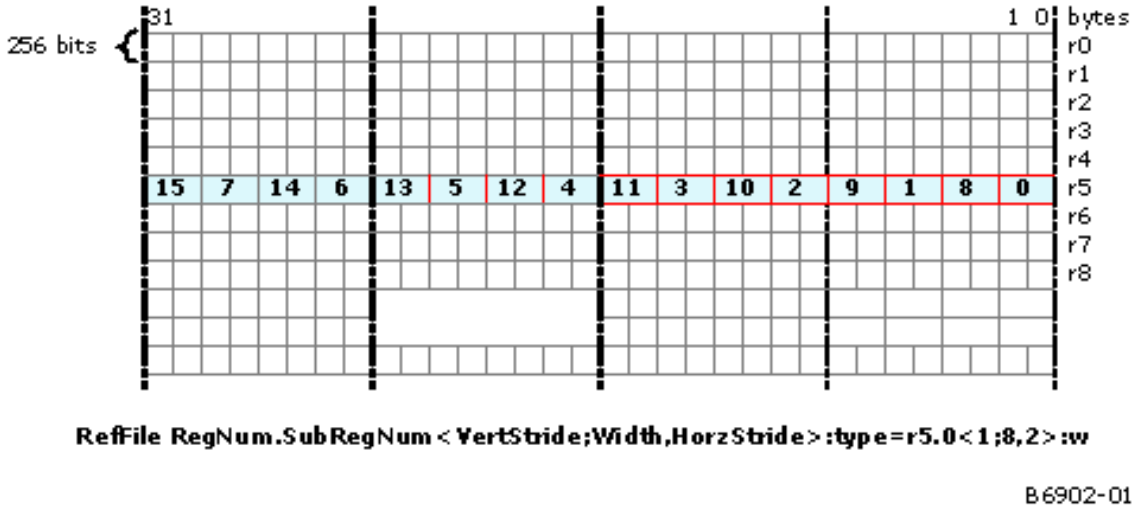


A 16-element register region with interleaved rows (r5.0<1;8,2>:w) shows another example where the rows are interleaved. The region, having word data elements, starts at location r5.0:w. HorzStride, the

distance within a row, is 2 words. So the second element (channel number 1) is at location 5.2:w. And there are 8 elements per row. VertStride, the distance between two rows, is only 1 word, which is less than HorzStride. Therefore, the first element of the second row (channel number 8) is at r5.1:w, just next to channel number 0. It is clear from the picture that the two rows are interleaved.

By varying the region parameters, reader may construct other configurations. The next section provides more details on the region-based register addressing. However, there are restrictions imposed by hardware implementation, which can be found in the later sections of this chapter.

A 16-element register region with interleaved rows (*r5.0<1;8,2>:w*)



Without considering the source channel swizzle and destination register region description, the above row-major-order region description provides the data assignment to each execution channel. The following pseudo code computes the addresses of data elements assigned to execution channels for a special case when the destination register is aligned to 256-bit register boundary.

```
// Input: Type: ub | b | uw | w | ud | d | f | v
//RegNum: In unit of 256-bit register
//SubRegNum: In unit of data element size
//ExecSize, Width, VertStride, HorzStride: In unit of data elements
// Output: Address[0:ExecSize-1] for execution channels
int ElementSize = (Type=="b"||Type=="ub") ? 1 : (Type=="w"|Type=="uw") ? 2 : 4;
int Height = ExecSize / Width;
int Channel = 0;
int RowBase = RegNum«5 + SubRegNum * ElementSize;
for (int y=0; y<Height; y++) {
        int Offset = RowBase;
        for (int x=0; x<Width; x++) {
```

Address [Channel++] = Offset;

*Offset += HorzStride*ElementSize;*

}

*RowBase += VertStride * ElementSize;*

}

As *HorzStride* and *VertStride* are specified independently (note that *VertStride* might be smaller than or equal to *HorzStride*), the region may take various shapes from a replicated scalar, a replicated vector, a vector of replicated scalars, a sliding window, to a non-overlapped 2D array.

A region-based description of a destination operand can take the following simplified format

RegFile RegNum.SubRegNum<HorzStride>:type

The destination operand is only allowed to have a 1-dimensional region. The Register Region Origin and Type are the same as for a source operand. The total number of elements is given by *ExecSize*. However, only *HorzStride* is required to describe the 1D region, not *VertStride* and *Width*.

As a source register region may cross multiple physical GRF registers, an instruction with such source operands may take more than two execution cycles to gather source data elements for execution. The destination register region of a non-compressed instruction is restricted to be within a physical GRF register. In other words, destination scatter writes over multiple registers are not supported.

Region Addressing Modes

There are two different register addressing modes: Direct register addressing and register-indirect register addressing. Depending on the register region description, the register-indirect register addressing mode can be further divided into three usages: 1x1 index region where only the origin of register region is provided by the address register, Vx1 index region where the offset of each row of the register region is provided by an address register, VxH index region where the offset of each data element is provided by an address register.

The 16-bit address register has byte address of the GRF register, additionally an immediate byte address offset can also be specified in the instruction. The sum of the address register and immediate offset can be divided by GRF register size to determine the RegNum and SubRegNum.

The examples in this section use 256-bit physical GRF registers to demonstrate the concept.

Direct Register Addressing

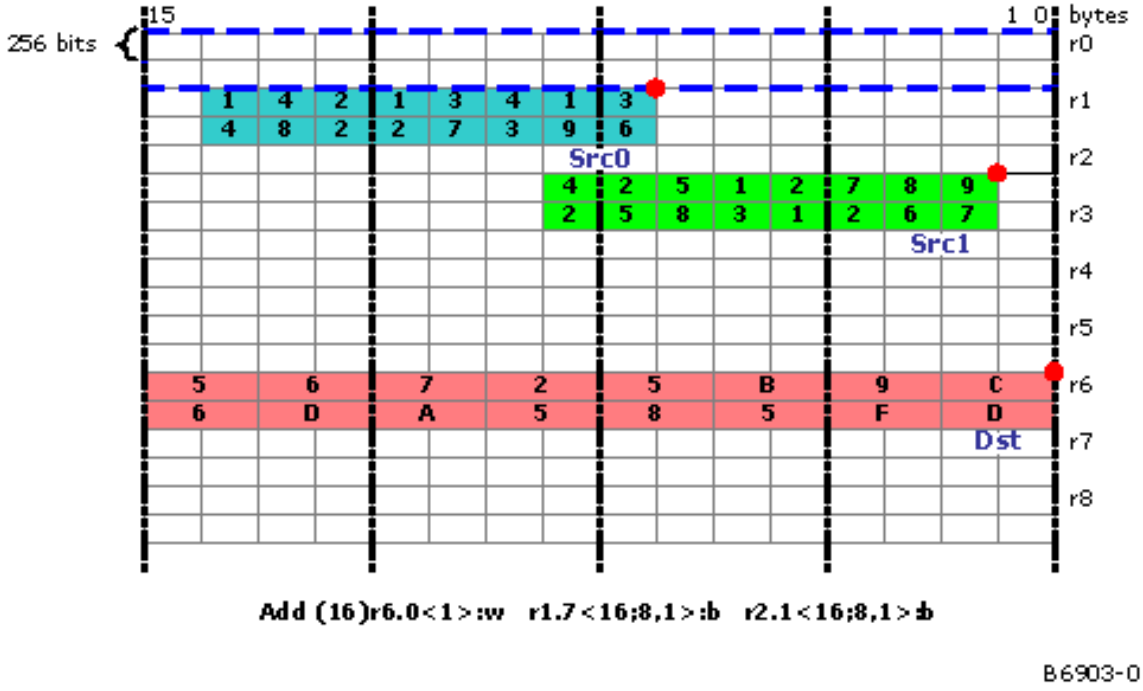
In this mode, all register region parameters are specified for an operand using fields in the instruction word.

Direct Register Addressing and *Direct Register Addressing* are two examples of direct register addressing.

For the example in *Direct Register Addressing*, all operands are 2D rectangular regions having the same size of 16 data elements. The two source operands, *Src0* and *Src1*, have 16 bytes. The destination operand, *Dst*, has 16 words. There are 8 elements in a row for *Src0* and *Src1*. The vertical stride of 16 bytes for *Src0* and *Src1* indicates that the first element and the 9'th element are 16 bytes apart in the

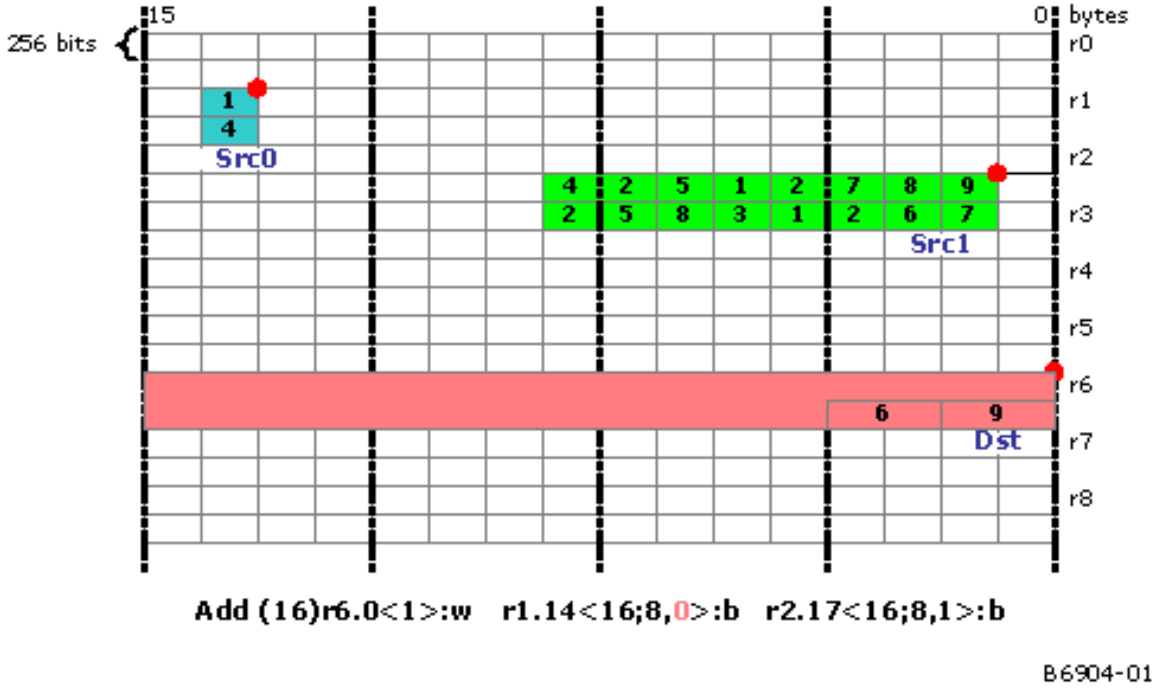
register file. Note that *Src0* falls into the 256-bit physical GRF register starting at r1.0, but Src1 crosses the 256-bit physical GRF register boundary between r2 and r3. The numbers in the shaded regions are the values of the data elements. Observing the upper right corners of the source/destination regions (first data element), we have C = 3+9.

A region description example in direct register addressing



For the example in *Direct Register Addressing*, the sizes of areas of *Src0* and *Src1* are the same, but *Src0* contains a vector of replicated scalars. With HorzStride = 0 and Width = 8, the first row of 8 elements in Src0 is a replication of the byte at r1.14. Comparing *ExecSize* of 16 to Width of 8 indicates that there is a second row of 8 elements in *Src0*. With VertStride = 16, the second row in *Src0* is a replication of the byte at r1.30 (30 = 14+16). Effectively, the 16 data elements of *Src0* are {1,1,1,1,1,1,1,1, 4,4,4,4,4,4,4,4}.

A region description example in direct register addressing with src0 as a vector of replicated scalars

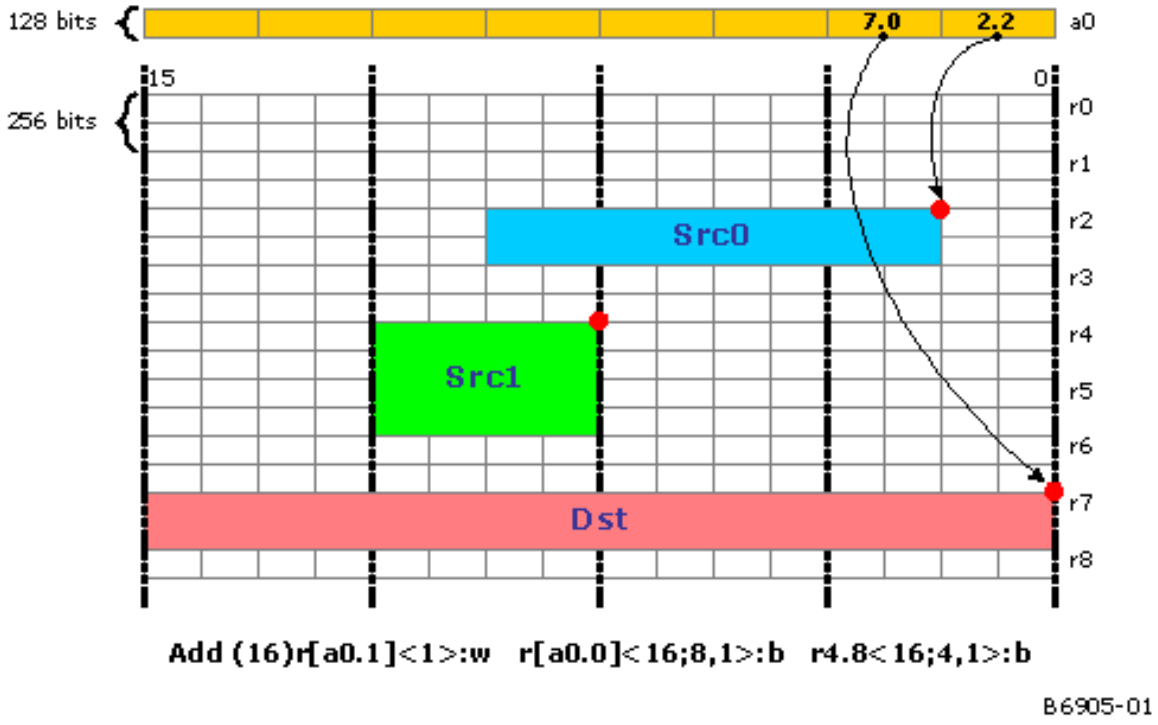


Register-Indirect Register Addressing with a 1x1 Index Region

In the register-indirect register addressing mode with 1x1 index region, the region origin is provided by the content of the address register, the rest of region parameters are provided by the fields in the instruction word.

Register-Indirect Register Addressing with a 1x1 Index Region depicts an example for this addressing mode. For example, the presence of a full region description <16;8,1> for Src0 indicates that only the origin of the region is provided by the address register a0.0.

An example illustrating register-indirect register addressing mode with a 1x1 index region

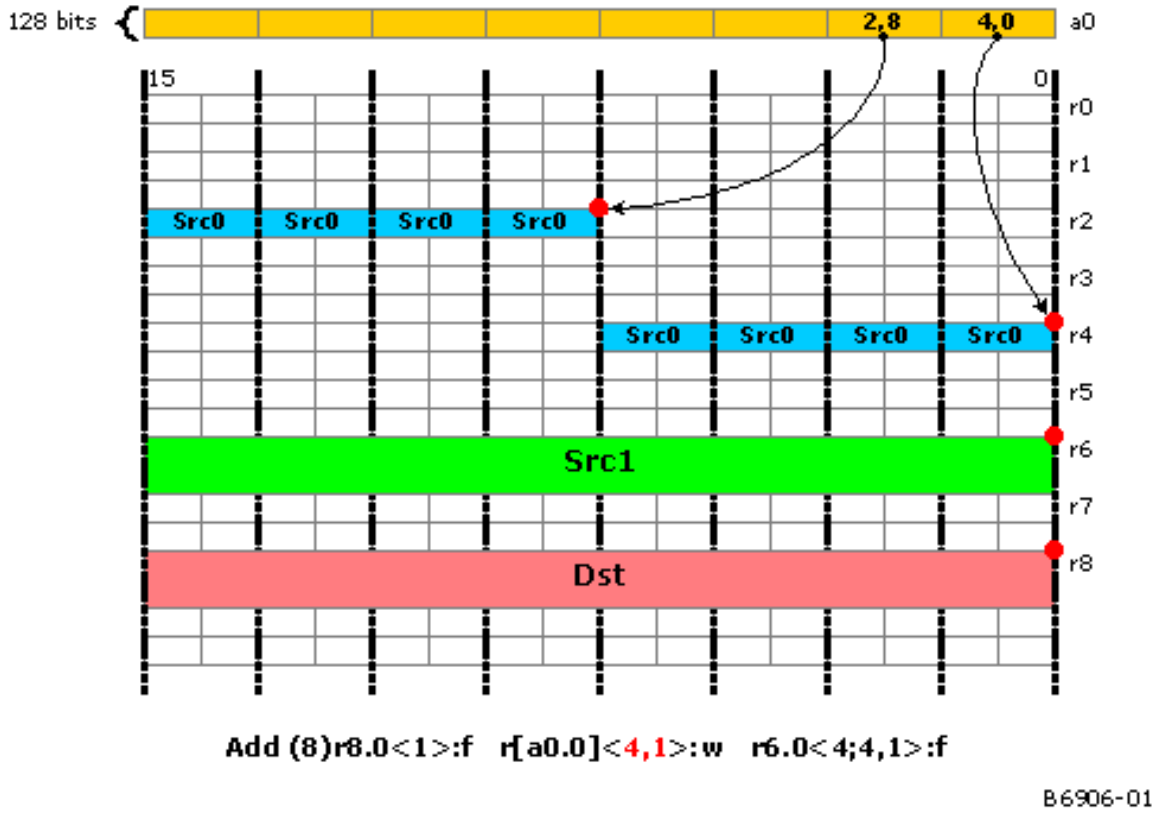


Register-Indirect Register Addressing with a Vx1 Index Region

In the register-indirect register addressing mode with Vx1 index region, the horizontal dimension is described by the fields in the instruction word and the vertical dimension is described by an address register region. Specifically, the origin of each row of the data region is provided by the contents of an address register region. The rows are described by the width and the horizontal stride. The first address register is provided and the following contiguous address registers are for the following rows. The total number of address registers used is inferred from the parameters *ExecSize* and *Width*.

An example is provided in *Register-Indirect Register Addressing with a Vx1 Index Region*. The assembly syntax notion of a register region without vertical stride, <4,1>, corresponding to the special encoding of vertical stride of 0xF in the instruction word, indicates the VxH or Vx1 mode of indirect register addressing. In this case, the origin for each row of src0 is provided by the address register. As ExecSize/Width = 2, there are two address registers a0.0 and a0.1, each pointing to a row of 4 data elements.

An example illustrating register-indirect-register addressing mode with a Vx1 index region (src0)

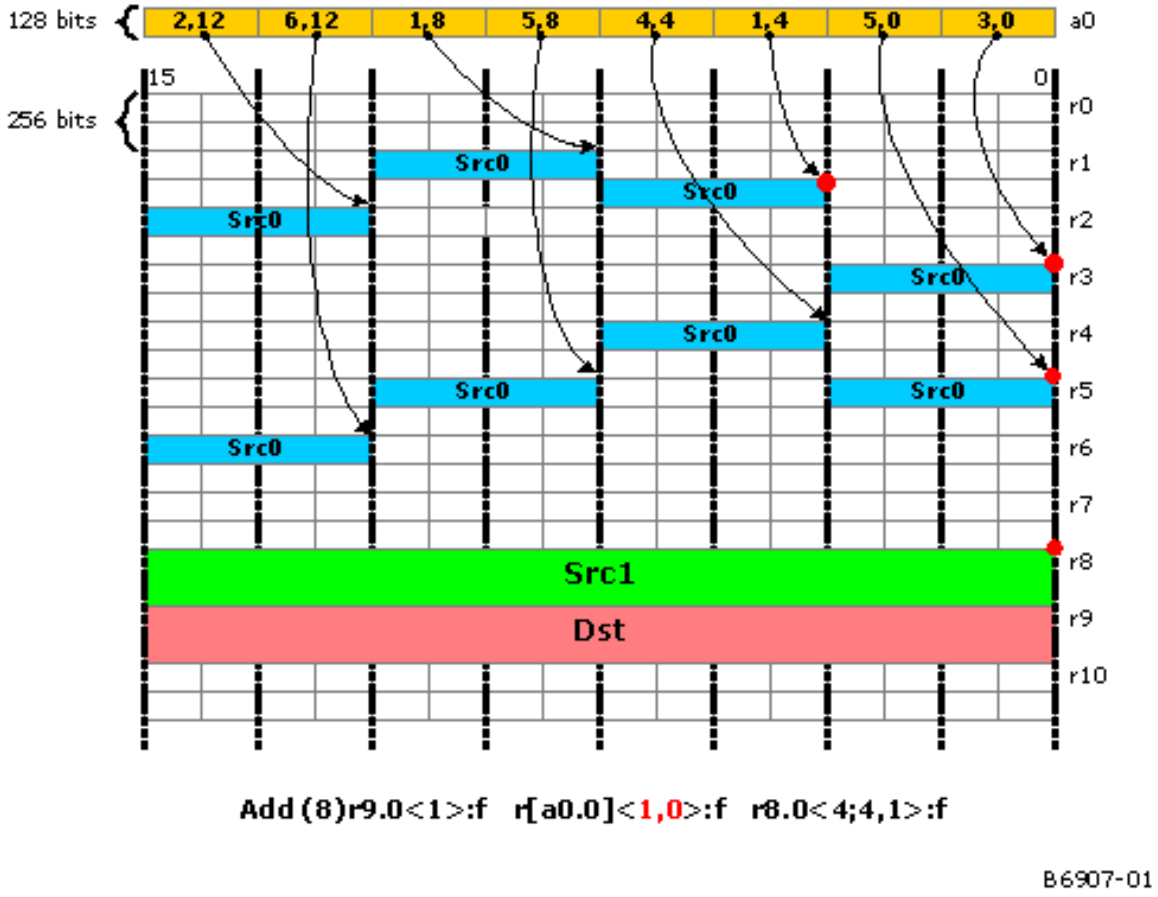


Register-Indirect Register Addressing with a VxH Index Region

In the register-indirect register addressing mode with VxH index region, the position of each data element is provided by the contexts in an address register region. This mode has the identical syntax as the Vx1 index region mode, and in fact, can be viewed as a special case of the Vx1 mode. When *Width* of the region is 1, the number of address registers used equals *ExecSize*.

An example is provided in *Register-Indirect Register Addressing with a VxH Index Region*. The absent of vertical stride in the region description <1,0> with width = 1 indicates that the origin for each row of 1 data element of Src0 is provided by the address register. As ExecSize/Width = 8, there are 8 address registers from a0.0 to a0.7, each pointing to a single data elements.

An example illustrating register-indirect register addressing mode with a VxH index region (Src0).



Access Modes

There are two basic register access modes controlled by a single bit instruction subfield called Access Mode.

- 16-byte Aligned Access Mode (**align16**): In this mode, the origins of all operands (sources and destination), whether it is by direct addressing or register-indirect addressing, are 16-byte aligned. For example, a row in the region description starts at 16-bype aligned and the width the row must be 4 and the 4 data elements within a row must span 16-bytes. In this access mode (and with other restrictions put forward later), full-channel swizzle for both source operands and per-channel mask for destination operand are supported on a 4-component basis. In other words, the control and setting of full source swizzle and destination mask are repeated for every 4 components up to total of *ExecSize* channels.
    - The **align16** access mode can be used for AOS operations. See examples provided in the Primary Usage Model section for SIMD4x2 and SIMD4x1 modes of operation to support 3D API Vertex Shader and Geometric Shader execution.
- 1-byte Aligned Access Mode (**align1**): In this mode, the origins of all operands may be aligned to their data type and could be 1-byte if the operand is of byte type. In this access mode, full region

register descriptions are supported, however, source swizzle or destination mask are not supported.

    - The **align1** access mode can be used for SOA operations. See examples provided in the Primary Usage Model section for SIMD8 and SIMD16 modes of operation to support 3D API Pixel Shader. Many media applications also operate well in **align1** access mode.

- Align16 accessmode is restricted to IEEE macro instructions only. All other non-IEEE macro instructions must use Align1 accessmode.

Execution Data Type

The architecture carries out arithmetic and logical operations using a smaller set of data types than the variety supported as source or destination operands. These are the *execution data types*. A particular arithmetic or logical instruction has one execution data type, from those listed in the table.

Execution Data Types

Type	Description
W	Word. 16-bit signed integer.
D	Doubleword. 32-bit signed integer.
Q	Quadword. 64-bit signed integer.
F	Float. 32-bit single precision floating-point number.
DF	Double Float. 64-bit double precision floating-point number.
HF	Half Float. 16-bit half precision floating-point number.

The following rules explain the conversion of multiple source operand types, possibly a mix of different types, to one common execution type:

- For floating-point sources, all source operands must have the same floating-point type, with the exceptions below:
    - A two-source floating-point instruction can have Float as the src0 type and VF (Packed Restricted Float Vector) as the immediate src1 type.
- Mixing floating-point and integer source types is not allowed. Either all source types must be one floating-point type or all source types must be integer types.
- Unsigned integers are converted to signed integers.
- Byte (B) or Unsigned Byte (UB) values are converted to a Word or wider integer execution type.
- If source operands have different integer widths, use the widest width specified to choose the signed integer execution type.

Note that when the execution data type is an integer type, it is always a signed integer type. For integer execution types, extra precision is provided within the hardware, including the accumulators, so that conversions from unsigned to signed do not affect instruction correctness.

Register Region Restrictions

A register region is described as *packed* if its elements are adjacent in memory, with no intervening space, no overlap, and no replicated values. If there is more than one element in a row, elements must be adjacent. If there is more than one row, rows must be adjacent. When two registers are used, the registers must be adjacent and both must exist.

The following register region rules apply to ALU instructions:

1. **General Restrictions Based on Operand Types**

   There are these general restrictions based on operand types:

   1. Where *n* is the largest element size in bytes for any source or destination operand type, *ExecSize* * *n* must be <= 64.
   2. When the Execution Data Type is wider than the destination data type, the destination must be aligned as required by the wider execution data type and specify a *HorzStride* equal to the ratio in sizes of the two data types. For example, a *mov* with a D source and B destination must use a 4-byte aligned destination and a *Dst.HorzStride* of 4.

2. **General Restrictions on Regioning Parameters**

   The mapping of data elements within the region of a source operand is in row-major order and is determined by the region description of the source operand, the destination operand, and the *ExecSize*, with these restrictions:

   1. *ExecSize* must be greater than or equal to *Width*.
   2. If *ExecSize* = *Width* and *HorzStride* != 0, *VertStride* must be set to *Width* * *HorzStride*.
   3. If *ExecSize* = *Width* and *HorzStride* = 0, there is no restriction on *VertStride*.
   4. If *Width* = 1, *HorzStride* must be 0 regardless of the values of *ExecSize* and *VertStride*.
   5. If *ExecSize* = *Width* = 1, both *VertStride* and *HorzStride* must be 0.
   6. If *VertStride* = *HorzStride* = 0, *Width* must be 1 regardless of the value of *ExecSize*.
   7. *Dst.HorzStride* must not be 0.
   8. *VertStride* must be used to cross GRF register boundaries. This rule implies that elements within a '*Width*' cannot cross GRF boundaries.

3. **Region Alignment Rules for Direct Register Addressing**
   1. In Direct Addressing mode, a source cannot span more than 2 adjacent GRF registers.
   2. A destination cannot span more than 2 adjacent GRF registers.

4. **Special Cases for Byte Operations**
   1. When the destination type is byte (UB or B) only a 'raw move' using the *mov* instruction supports a packed byte destination register region: *Dst.HorzStride* = 1 and *Dst.DstType* = (UB or B). This packed byte destination register region is not allowed for any other instructions, including a 'raw move' using the *sel*instruction, because the *sel* instruction is based on Word or DWord wide execution channels.
   2. There is a relaxed alignment rule for byte destinations. When the destination type is byte (UB or B), destination data types can be aligned to either the lowest byte or the second lowest

byte of the execution channel. For example, if one of the source operands is in word mode (a signed or unsigned word integer), the execution data type will be signed word integer. In this case the destination data bytes can be either all in the even byte locations or all in the odd byte locations.

This rule has two implications illustrated by this example:

```
// Example:
mov (8) r10.0<2>:b r11.0<8;8,1>:w
mov (8) r10.1<2>:b r11.0<8;8,1>:w

// Dst.HorzStride must be 2 in the above example so that the destination
// subregisters are aligned to the execution data type, which is :w.
// However, the offset may be .0 or .1.
// This special handling applies to byte destinations ONLY.
```

5. **Special Cases for Word Operations**

   There are some special cases for word operations for specific projects, described in the following table. If you are viewing a version of the BSpec limited to other particular projects, the table may not show and there are no special cases in this category.

   There is a relaxed alignment rule for word destinations. When the destination type is word (UW, W, HF), destination data types can be aligned to either the lowest word or the second lowest word of the execution channel. This means the destination data words can be either all in the even word locations or all in the odd word locations.

   ```
   // Example:
   add (8)  r10.0<2>:hf r11.0<8;8,1>:f r12.0<8;8,1>:hf
   add (8)  r10.1<2>:hf r11.0<8;8,1>:f r12.0<8;8,1>:hf

   // Note: The destination offset may be .0 or .1 although the destination subregister
   // is required to be aligned to execution datatype.
   ```

6. **Special Requirements for Handling Double and Qword Precision Data Types**

   There are special requirements for handling double precision data types that vary by project, described in the following table. If you are viewing a version of the BSpec limited to other particular projects, the table may appear with no data rows.

   All regioning parameters like stride, execution size, and width are in units of element size.

   ```
   // Example:
   mov (4) r10.0<1>:df r11.0<4;4,1>:df
   // The above instruction moves four double floats.
   // Note the difference between [DevIVB] and [DevHSW+].
   ```

7. **Special Requirements for Handling Mixed Mode Float Operations**

There are some special requirements for handling mixed mode float operations for specific projects, described in the following table. If you are viewing a version of the BSpec limited to other particular projects, the table may appear with no data rows.

Requirement

No SIMD16 in mixed mode when destination is packed f16.
`mad(8) r3.xyzw:hf r4.xyzw:f r6.xyzw:hf r7.xyzw:hf`
`add(8) r20.0<1>:hf r3<8;8,1>:f r6.0<8;8,1>:hf {Q1}`

When source is float or half float from accumulator register and destination is half float with a stride of 1, the source must register aligned. i.e., source must have offset zero.

No swizzle is allowed when an accumulator is used as an implicit source or an explicit source in an instruction. i.e. when destination is half float with an implicit accumulator source, destination stride needs to be 2.
`mac(8) r3<2>:hf r4.0<8;8,1>:f r6.0<8;4,2>:hf`
`mov(8) r3<1>:f acc0.0<8;4,2>:hf`

Math operations for mixed mode: - In Align1, f16 inputs need to be strided `math(8) r3<1>:hf r4.0<8;8,1>:f r6.0<8;4,2>:hf`
For mixed float operations, f16 datatype write to accumulator cannot be packed destination.
Mixed mode instruction allows fp16 operands in the following cases: • dst, src0 and src1 for 2-source instructions not involving multiplier (add, cmp etc). • dst and src0 for 2-source instructions involving multiplier (mul, mac etc). • dst, src0 and src1 for 3-source instructions (mad). Mixed mode instruction allows packed fp16 destination.

8. **Regioning Rules for Register Indirect Addressing**

   Regioning rules for register indirect addressing vary for specific projects, described in the following table.

Rules

1. When the execution size and destination regioning parameters require two adjacent registers, these registers are accessed using one index register ONLY.
`// Example:`
`mov (16) r[a0.0]:f r10:f`
`// The above instruction behaves the same as the following two instructions:`
`mov (8) r[a0.0]:f r10:f`
`mov (8) r[a0.0, 8*4]:f r11:f`
2. When the destination requires two registers and the sources are 1x1 indirect mode, the sources must be assembled from two GRF registers accessed by a single index register. The data for each destination GRF register is entirely derived from one source register. This is ensured by appropriate use of regioning parameters. The exception to this is the use of indirect scalar sources, where the same element is used across the execution size.

Rules

``
// Example:
// Case (a)
add (16) r[a0.0]:f r[a0.2]:f r[a0.4]:f
// The above instruction behaves the same as the following two instructions:
add (8) r[a0.0]:f r[a0.2]:f r[a0.4]:f
add (8) r[a0.0, 8*4]:f r[a0.2, 8*4]:f r[a0.4, 8*4]:f
// Note that the immediate for the second instruction is based on regioning.
// In this case, it is 8 DWs.
 	
// Case (b)
add (16) r[a0.0]:ud r[a0.2]<4;8,1>:w r10<8;8,1>:ud
// The above instruction behaves the same as the following two instructions:
add (8) r[a0.0]:f r[a0.2]<4;8,1>:w r10<8;8,1>:ud
add (8) r[a0.0, 8*4]:f r[a0.2, 4*2]<4;8,1>:w r11<8;8,1>:ud
// Note that the immediate for the second instruction is based on regioning.
// VertStride of 4 with data type of word.
 	
// Case (c):
add (16) r[a0.0]:f r[a0.2]:f r[a0.4]<0;1,0>:f
// The above instruction behaves the same as the following two instructions:
add (8) r[a0.0]:f r[a0.2]:f r[a0.4]<0;1,0>:f
add (8) r[a0.0, 8*4]:f r[a0.2, 8*4]:f r[a0.4]<0;1,0>:f
// Note that the src1 indirect address does not change.
``
3. Indirect addressing on src1 must be a 1x1 indexed region mode.
4. When a Vx1 or a VxH addressing mode is used on src0, the destination may use one or two registers.
``
// Example:
// Case (a)
add (16) r[a0.0]<1>:d r[a0.0]<4,1>:ud r16.0<8;8,1>:ud
// The above instruction behaves the same as the following two instructions:
add (8) r[a0.0]<1>:d r[a0.0]<4,1>:ud r16.0<8;8,1>:ud
add (8) r[a0.0, 8*4]<1>:d r[a0.2]<4,1>:ud r17.0<8;8,1>:ud
// Since the pointer (index register) is incremented every 4 elements
// (width), the second instruction moves from a0.0 to a0.2.
 	
// Case (b)
add (16) r10.0<2>:uw r[a0.0, 0]<1,0>:uw r16.0<8;8,1>:uw
// The above instruction behaves the same as the following two instructions:
add (8) r10.0<2>:uw r[a0.0, 0]<1,0>:uw r16.0<8;8,1>:uw
add (8) r11.0<2>:uw r[a0.8, 0]<1,0>:uw r17.0<8;8,1>:uw
// Since the pointer (index register) is incremented every 1 element
// (width), the second instruction moves from a0.0 to a0.8.
``
5. Indirect addressing on the destination must be a 1x1 indexed region mode. The destination cannot span more than 2 adjacent GRF registers irrespective of predicates.

Execution size of 32 is NOT supported in Vx1 or VxH modes.
Accumulator as destination is NOT supported in Vx1 or VxH modes.

9. **Special Restrictions**

   There are some special restrictions on register region access for specific projects, described in the following table. If you are viewing a version of the BSpec limited to other particular projects, the table may appear with no data rows.

Restriction

Conversion between Integer and HF (Half Float) must be DWord-aligned and strided by a DWord on the destination.
`// Example:`
`add (8) r10.0<2>:hf r11.0<8;8,1>:w r12.0<8;8,1>:w`
`// Destination stride must be 2.`
`mov (8) r10.0<2>:w r11.0<8;8,1>:hf`
`// Destination stride must be 2.`

The src, dst overlapping behavior with the second half src and the first half destination to the same register must not be used with any compressed instruction.
Regioning Rules for Align1 Ternary Operations Width is an implied regioning parameter. 1. Width is 1 when Vertical and Horizontal Strides are both zero (broadcast access). 2. Width is equal to Vertical Stride when Horizontal Stride is zero. 3. Width is equal to Vertical Stride/Horizontal Stride when both Strides are non-zero. 4. Vertical Stride must not be zero if Horizontal Stride is non-zero. This implies Vertical Stride is always greater than Horizontal Stride. 5. For Source 2, if Horizontal Stride is non-zero, then Width is the a register's width of elements (e.g. 8 for a 32-bit data type). Otherwise, if Horizontal Stride is 0, then so is the Vertical (and rule 1 applies). This means Vertical Stride is always 'Width' * 'Horizontal Stride'.
Byte data type is not supported for src1 register regioning. This includes byte broadcast as well.
Byte data type is not supported for src1 register regioning.

In case of conversion/mixed-mode instruction with fp32 execution datatype and fp16 datatype destination with stride of 1 (packed) the following apply
• Oword boundary must not be crossed by packed fp16 destination.
• Subregister byte offset within the oword must be half of the execution subregister byte offset
`add (8) r3.0<1>:hf r4.0<1;1,0>:f r6.0<2;1,0>:hf // mixed mode`
`mov (8) r3.0<1>:hf r4.0<1;1,0>:f // conversion`
`mov (4) r3.2<1>:hf r4.2<1;1,0>:f // conversion (2)*sizeof(hf) mod oword = half of 2*sizeof(f)`
`mov (4) r3.11<1>:hf r4.3<1;1,0>:f // conversion (11)*sizeof(hf) mod oword = half of 3*sizeof(f)`

Src2 Restrictions 1. Byte data type is not supported. 2. Register Regioning patterns where register data bit locations are changed between source and destination are not supported except for broadcast of a scalar. 3. The Source 2 operand must be 64-bit aligned except for broadcast of a scalar.
In an atomic instruction sequence forming a macro the following must apply:

Restriction
• Only first instruction can have SWSB check dependency. • Internal producer-consumer relationship must not be present.

Destination Operand Description

Destination Region Parameters

Based on the above restrictions, a subset of register region parameters are sufficient to describe the destination operand:

- Destination Register Origin
  - Destination Register Number and Destination Subregister Number for direct register addressing mode
  - A Scalar Destination Register Index for register-indirect-register addressing mode
- Destination Register 'Region' - Note that destination register region does not have full region description parameters
  - Destination Horizontal Stride

SIMD Execution Control

This section discusses SIMD execution, both with and without predication.

Predication

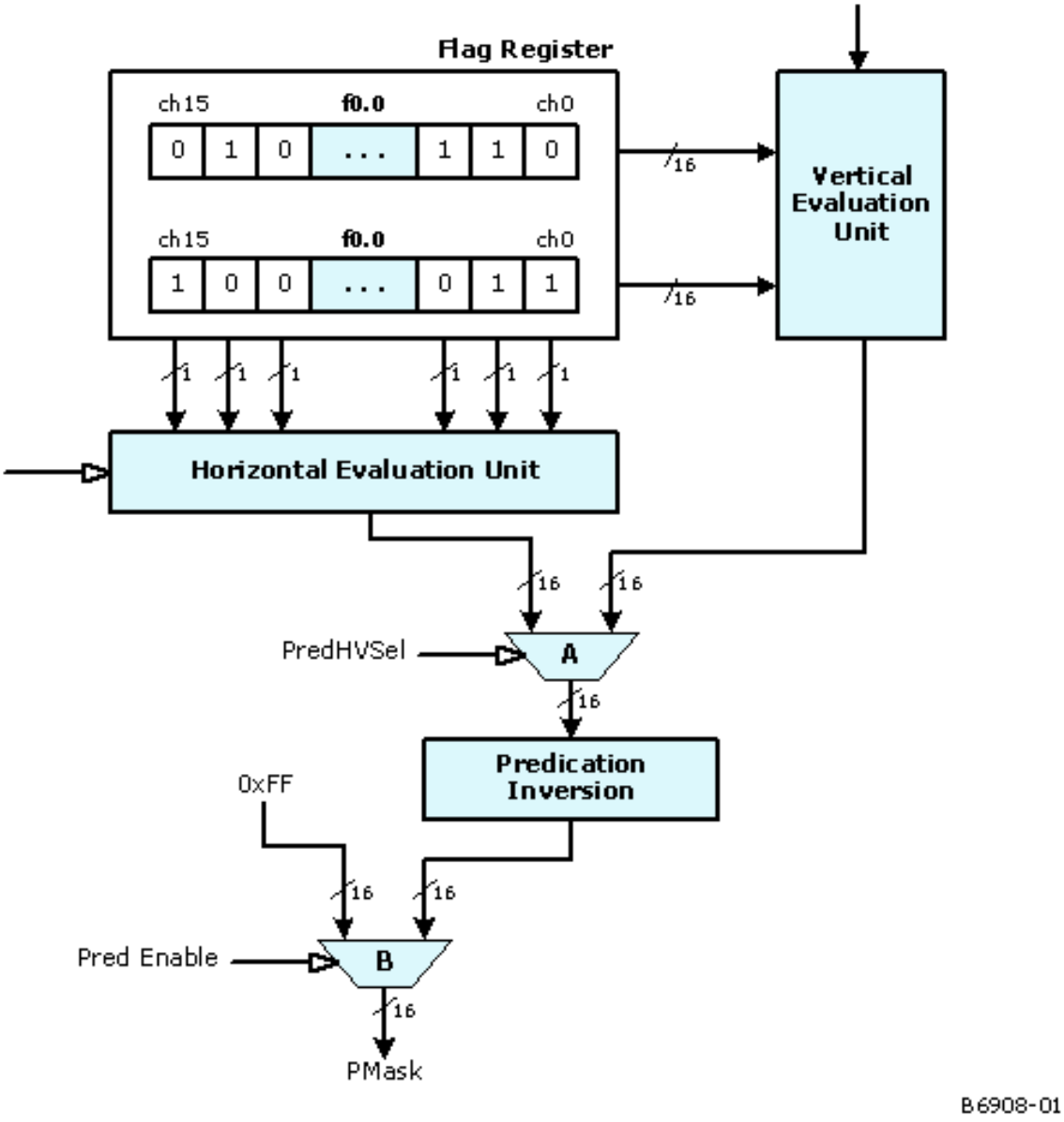
Predication is the conditional SIMD channel selection for execution on a per instruction basis. It is an efficient way of dynamic SIMD channel enabling without paying branch instruction overhead. When predication is enabled for an instruction, a Predicate Mask (PMask), which contains 16-bit channel enables, is generated internally in EU. Note that PMask is not a software visible register. It is provided here to explain how SIMD execution control works. PMask generation is based on the Predication Control (*PredCtrl*) field, Predication Inversion (*PredInv*) field and the flag source register in the instruction word. See Instruction Summary chapter for definition of these fields.

*Predication*shows the block diagram of the hardware logic to generate PMask. PMask is generated based on combinatory logic operation of the bits in the flag register. Instruction field *PredCtrl* controls the horizontal evaluation unit and vertical evaluation unit. MUX A in the figure selects whether horizontally evaluated results or vertically-evaluated results are sent to the Predication Invertion unit. The *PredInv* field controls the Prediction Inversion unit. Either one 16-bit flag subregister or the whole flag register may be selected to generate the PMask depending on the predication control modes. MUX B indicates that predication can be enabled and disabled. Predication can be grouped into the following three categories. Predication functionality also depends on the Access Mode of the instruction.

- No predication: Of course, predication can be disabled. This is the most commonly used case.

- Predication with horizontal combination: the predicate mask is generated based on combinatory logic operation of bits within a selected flag subregister.
- Predication with vertical combination: the predicate mask is generated based on combinatory logic operation of bits across flag multiple subregisters.

Generation of predication mask



No Predication

When PredCtrl field of a given instruction is set to 0 ("no predication"), it indicates that no predication is applied to this instruction. Effectively, the resulting PMask is all 1's. This is shown by the 2:1 multiplexer B controlled by the Pred Enable signal in *Predication*. Where predication is not enabled for an instruction, multiplex B is selected to output 0xFF to PMask.

Predication with Horizontal Combination

Predication with horizontal combination inputs the 16 bits of a single flag subregister (f0.0:uw or f0.1:uw) and passes them through combinatory logic of the Horizontal Evaluation unit to create PMask.

The simplest combination is 'no combination' - the same 16 bits from selected flag subregister are output to MUX A. In this case, a bit in the selected flag subregister controls the conditional execution of the corresponding execution channel. Let the selected flag subregister be denoted as f0.#, the following pseudo code describes the predicate mask generation for predication with sequential flag channel mapping.

```
If (PredCtrl == "Sequential flag channel mapping") {
    For (ch=0; ch<16; ch++)
        PMask[ch] = (PredInv == TRUE) ? ~f0.#[ch] : f0.#[ch];
}
```

More complex horizontal evaluation is based on channel grouping. A group of adjacent channels (bits from flag subregister) are evaluated together, and a single bit is replicated to the group. The size of groups is in power of 2. The supported combination depends on the Access Mode of an instruction.

In **Align16** access mode, horizontal combination is based on 4-channel groups.

- Channel replication: PredCtrl of '.x', '.y', '.z' and '.w' select a single channel from each 4-channel group and replicate it as the output for the group. For example, PredCtrl = '.x' means that channel 0 in each group is replicated.
- OR combination: PredCtrl of '.any4h' means that if **any** of the channel in a group is enabled, outputs for the 4 channels in the group are all enabled.
- AND combination: PredCtrl of '.all4h' means that only when **all** of the channels in a group are enabled, the output for the group is enabled.

These combinations in **Align16** mode can be described by the following pseudo-code.

```
If (Access Mode == Align16) {
    For (ch = 0; ch < 16; ch += 4)
        Switch (PredCtrl) {
            Case     '.x':   bTmp = f0.#[ch]; break;
            Case     '.y':   bTmp = f0.#[ch+1]; break;
            Case     '.z':   bTmp = f0.#[ch+2]; break;
            Case     '.w':   bTmp = f0.#[ch+3]; break;
            Case '.any4h':   bTmp = f0.#[ch] | f0.#[ch+1] | f0.#[ch+2] | f0.#[ch+3]; break;
Case '.all4h':   bTmp = f0.#[ch] & f0.#[ch+1] & f0.#[ch+2] & f0.#[ch+3]; break;
        }
        bTmp = (PredInv == TRUE) ? ~bTmp : bTmp;
        PMask[ch] = PMask[ch+1] = PMask[ch+2] = PMask[ch+3] = bTmp;
    }
}
```

In **Align1** access mode, horizontal combination is based on AND combination '.any#h' and OR combination '.all#h' on channel groups with various sizes, where # is the number of channels in a group ranging from 2 to 16. This is described by the following pseudo-code.

```
If (Access Mode == Align1) {
    Switch (PredCtrl) {
        Case  '.any2h':   groupSize = 2; <op> = '|'; break;         Case  '.all2h':
groupSize = 2; <op> = '&'; break;
        Case  '.any4h':   groupSize = 4; <op> = '|'; break;         Case  '.all4h':
```

```
groupSize = 4; <op> = '&'; break;
         Case  '.any8h':   groupSize = 8; <op> = '|'; break;         Case  '.all8h':
groupSize = 8; <op> = '&'; break;
         Case '.any16h':   groupSize = 16; <op> = '|'; break;         Case '.all16h':
groupSize = 16; <op> = ' break;     }
     For (ch = 0; ch < 16; ch += groupSize) {
         For (inc = 0, bTmp = FALSE; inc < groupSize; inc ++)
             bTmp = bTmp <op> f0.#[ch+inc];
         For (inc = 0; inc < groupSize; inc ++)
             PMask[ch+inc] = bTmp;
     }
 }
```

Predication with Vertical Combination

Predication with vertical combination uses both flag subregister as inputs. The AND or OR combination is across the subregisters on a channel-by-channel basis. This is shown by the following pseudo-code.

```
If (Access Mode == Align1) {
    For (ch = 0; ch < 16; ch ++) {
        If (PredCtrl == 'any2v')
            PMask[ch] = f0.0[ch] | f0.1[ch]
        Else If (PredCtrl == 'all2v')
            PMask[ch] = f0.0[ch] & f0.1[ch]
    }
}
```

Predication with Vertical Combination

Predication with vertical combination uses both flag register as inputs. The AND or OR combination is across the registers on a channel-by-channel basis. This is shown by the following pseudo-code.

```
If (Access Mode == Align1) {
    For (ch = 0; ch < 32; ch ++) {
        If (PredCtrl == 'anyv')
            PMask[ch] = f0.0[ch] | f1.0[ch]
        Else If (PredCtrl == 'allv')
            PMask[ch] = f0.0[ch] & f1.0[ch]
    }
}
```

End of Thread

There is no special instruction opcode (such as an END instruction) to cause the thread to terminate execution. Instead, the end of thread is signified by a *send* instruction with the end-of-thread (EOT) sideband bit set. Upon executing a *send* instruction with EOT set, the EU stops on the thread. Upon observing an EOT signal on the output message bus, the Thread Dispatcher makes the thread's resource available. If a thread uses pre-allocated resource managed by a fixed function, such as URB handles and scratch memory, some fixed function protocol also requires the thread to terminate with the message header phase to carry the information in order for the fixed function to release the pre-allocated resource.

EU hardware guarantees that if a terminated thread has in-flight read messages or loads at the time of 'end' that their writebacks will not interfere with either other threads in the system or new threads loaded in the system in the future.

More details can be found in the *send* instruction description in Instruction Reference chapter.

Assigning Conditional Flags

Instructions can output two sets of conditional signals, one set from before the outputs clamping/re-normalizing/format conversion logic, we call this the pre conditional signals. The second set is generated from the final results after clamping and re-normalizing/format conversion logic, and we call this the post conditional signals. The post conditional signals are used for fusing the DirectX compare instruction. **Note:** The flags generated from the post conditional signals should be equivalent to the flags generated by a separate *cmp* instruction after the current arithmetic instruction.

The pre conditional signals are used to generated flags for *cmp*/*cmpn* instructions only, this logically does the compare of the two input sources. The post conditional signals are used to generated flags for all the other arithmetic instructions, this logically does the compare of the result with zero.

cmpn with both sources as NaNs is a don't care case as this doesn't impact the MIN/MAX operations.

The pre conditional signals include the following:

- **pre_sign** bit: This bit reflects the sign of the computed result before going through any kind of clamping, normalizing, or format conversion logic.
- **pre_zero** bit: This bit reflects whether the computed result is zero before any kind of clamping, normalizing, or format conversion logic.

The post conditional signals include the following:

- **post_sign** bit: This bit reflects the sign of the final result after all the clamping, normalizing, or format conversion logic.
- **post_zero** bit: This bit reflects whether the final result is zero after all the clamping, normalizing, or format conversion logic.
- **OF** bit: This bit reflects whether an overflow occured in any of the computation of the current instruction, including clamping, re-normalizing, and format conversion.
- **NC** bit: The NaN computed bit indicates whether the computed result is not a number. It carries valid information for instructions operating on floating point values. For an operation on integer operands, this bit is always 0.
- **NS0** bit: The NaN Source 0 bit indicates whether src0 of an execution channel is not a number. It carries valid information for instructions operating on floating point values. For an operation on integer operands, this bit is always 0.
- **NS1** bit: The NaN Source 1 bit indicates whether src1 of an execution channel is not a number. It carries valid information for instructions operating on floating point values. For an operation on integer operands, this bit is always 0. For an operation with one source operand, this bit is also set to 0. This bit is only used for the comparison instruction *cmpn*, which is specifically provided to emulate MIN/MAX operations. For any other instructions, this bit is undefined.

Flag Generation for *cmp* Instructions (The Supported Conditional Modifiers are .e, .ne, .g, .ge, .l, and .le.)

Conditional Modifier	Meaning	Resulting Flag Value (for an execution channel)
.e	Equal-to	**(pre_zero & ! (NS0 | NS1)).** This conditional modifier tests whether the two sources are equal. If either source is NaN (i.e. NC is true), the flag is forced to false.
.ne	Not-Equal-to	**! (pre_zero & ! (NS0 | NS1)).** This conditional modifier test whether the two sources are equal. It takes exactly the reverse polarity as the modifier **.e**.
.g	Greater-than	**(! pre_sign & ! pre_zero & ! (NS0 | NS1)).** This conditional modifier tests whether src0 is greater than src1. If either source is a NaN (i.e. NC is true), the flag is forced to false.
.ge	Greater-than-or-equal-to	**((! pre_sign | pre_zero) & ! (NS0 | NS1)).** This conditional modifier tests whether src0 is greater than or equal to src1. If either source is a NaN (i.e. NC is true), the flag is forced to false.
.l	Less-than	**(pre_sign & !pre_zero & ! (NS0 | NS1)).** This conditional modifier tests whether src0 is less than src1. If either source is a NaN (i.e. NC is true), the flag is forced to false.
.le	Less-than-or-equal-to	**((pre_sign | pre_zero) & ! (NS0 | NS1)).** This conditional modifier tests whether src0 is less than or equal to src1. If either source is a NaN (i.e. NC is true), the flag is forced to false.

Flag Generation for All Instructions Other than *cmp/cmpn* Instructions (The Supported Conditional Modifiers are .e, .ne, .g, .ge, .l, .le, .o, and .u.)

Conditional Modifier	Meaning	Resulting Flag Value (for an execution channel)
.e	Equal-to	**(post_zero & ! NC).** This conditional modifier tests whether the result is equal to zero. If either source is NaN (i.e. NC is true), the flag is forced to false.
.ne	Not-Equal-to	**! (post_zero & ! NC).** This conditional modifier test whether the result is not equal to zero. It takes exactly the reverse polarity as modifier **.e**.
.g	Greater-than	**(! post_sign & ! post_zero & ! NC).** This conditional modifier tests whether

Conditional Modifier	Meaning	Resulting Flag Value (for an execution channel)
		result is greater than zero. If either source is a NaN (i.e. NC is true), the flag is forced to false.
.ge	Greater-than-or-equal-to	**((! post_sign | post_zero) & ! NC).** This conditional modifier tests whether result is greater than or equal to zero. If either source is a NaN (i.e. NC is true), the flag is forced to false.
.l	Less-than	**(post_sign & ! post_zero & ! NC).** This conditional modifier tests whether result is equal to zero. If either source is a NaN (i.e. NC is true), the flag is forced to false.
.le	Less-than-or-equal-to	**((post_sign | post_zero) & ! NC).** This conditional modifier tests whether result is equal to or less than zero. If either source is a NaN (i.e. NC is true), the flag is forced to false.
.o	Overflow	**OF.** This conditional modifier tests whether the computed result causes overflow - the computed result is outside the range of the destination data type. Note: the computed result here should represent the infinite precision for intermediate result before converted to destination result for both float and integer operation. Note: The legacy condition modifier behavior is different from IEEE exception Overflow flag. For inf float to int conversion, .o will set the legacy Overflow flag, but IEEE exception Overflow flag won't be set. All other internal conditional signals are ignored.
.u	Unordered	**NC.** This conditional modifier tests whether the computed result is a NaN (unordered). All other internal conditional signals are ignored.

Destination Hazard

Architecture has built-in hardware to avoid destination hazard.

Destination Hazard stands for the risk condition when multiple operations are trying to write to the same destination and the result of the destination may be ambiguous. This may or may not happen on GFX for two instructions with the same destination, or with destinations that have overlapped register region, depending on the ordering of the arrival of destination results. Let's consider two instructions in a thread with potential destination hazard. There may be other instruction between them as long as there is no instruction sourcing the same destination. Using register scoreboards, hardware automatically takes care of the destination hazard by not issuing the second instruction until the destination scoreboard is cleared. However, for certain cases, in fact for most cases, such destination hazard indicated by the

register scoreboard is false, causing unnecessary delay of instruction issuing. This may result in lower performance. The destination dependency control field in the instruction word {*NoDDClr, NoDDChk*} allows software to selectively override such hardware destination dependency mechanism. Such performance optimization hooks must be used with extreme caution. When it is not certain that it is a false destination hazard, the programmer should rely on hardware to resolve the dependency.

As the destination dependency control field does not apply to *send* instruction, there is only one condition that a programmer may use the {*NoDDClr, NoDDChk*} capability.

Instructions other than send, may use this control as long as operations that have different pipeline latencies are not mixed. The operations that have longer latencies are:

- Opcodes *pln*, *lrp*, *dp**.
- Operations involving double precision computation.

Integer DW multiplication where both source operands are DWs.

Non-present Operands

Some instructions do not have two source operands and one destination operand. If an operand is not present for an instruction the operand field in the binary instruction must be filed with null. Otherwise, results are unpredictable.

Specifically, for instructions with a single source, it only uses the first source operand src0. In this case, the second source operand src1 must be set to null and also with the same type as the first source operand src0. It is a special case when src0 is an immediate, as an immediate src0 uses DW3 of the instruction word, which is normally used by src1. In this case, src1 must be programmed with register file ARF and the same data type as src0.

Instruction Prefetch

Due to prefetch of the instruction stream, the EUs may attempt to access up to 8 cachelines (512b) beyond the end of the kernel program - possibly into the next memory page. Although these instructions will not be executed, they must be accounted for the prefetch in order to avoid invalid page access faults. GFX software is required to pad the end of all kernel programs with 512b data. A more efficient approach would be to ensure that the page after all kernel programs is at least valid (even if mapped to a dummy page). Note that the **General State Access Upper Bound** field of the STATE_BASE_ADDRESS command can be used to prevent memory accesses past the end of the General State heap (where kernel programs must reside).

ISA Introduction

ISA Introduction

This chapter contains these sections that introduce this volume.

- Introducing the Execution Unit
- EU Terms and Acronyms
- EU Changes by Processor Generation
- EU Notation

Subsequent chapters cover:

- EU Data Types
- Execution Environment
- Exceptions
- Instruction Set Summary
- Instruction Set Reference
- EU Programming Guide

The EU Programming Guide provides some useful examples and information but is not a complete or comprehensive programming guide.

Introducing the Execution Unit

This section introduces the Execution Unit (EU), a simple and capable processor within the GPU that supports graphics processing within the graphics pipelines, can do general purpose computing (GPGPU), and responds to exceptional conditions via the System Routine.

The EU provides parallelism at two levels: thread and data element. Multiple threads can execute on the EU; the number executing concurrently depends on the processor and is transparent to EU code. Each thread has its own registers (GRF and ARF, described below). Most EU instructions operate on arrays of data elements; the number of data elements is normally the *ExecSize (execution size)* or number of channels for the instruction. A *channel* is a logical unit of execution for data element access, masking, and flow control within instructions. The number of channels is independent of the number of physical ALUs or FPUs for a particular graphics processor.

EU native instructions are 128 bits (16 bytes) wide. Some combinations of instruction options can use compact instruction formats that are 64 bits (8 bytes) wide. Identifying instructions that can be compacted and creating the compact representations is done by software tools, including compilers and assemblers.

Data manipulation instructions have a destination operand (dst) and one, two, or three source operands (src0, src1, or src2). The instruction opcode determines the number of source operands. An instruction's last source operand can be an immediate value rather than a register.

Data read or written by a thread is generally in the thread's *GRF (General Register File)*, 128 general registers, each 32 bytes. A data element address within the GRF is denoted by a register number (r0 to

r127) and a subregister number. In the instruction syntax, subregister numbers are in units of data element size. For example, a :d (Signed Doubleword Integer) element can be in subregister 0 to 7, corresponding to byte numbers in the instruction encoding of 0, 4, ... 28.

The EU cannot directly read or write data in system memory.

Specialized registers used to implement the ISA are in a distinct per thread *Architecture Register File (ARF)*. Each such register or group of related registers has its own distinct name. For example, ip is the instruction pointer and f0 is a flags register. An ARF register can be a src0 or dst operand but not a src1 or src2 operand. There are restrictions on how particular ARF registers are accessed that should be understood before directly reading or writing those registers. See the ARF Registers section for more information.

The EU supports both integer and floating-point data types, as described in the Numeric Data Types section.

For EU flow control, each channel has its own per-channel instruction pointer (PcIP[n]) and only executes an instruction when IP == PcIP[n] and any other masks enable the channel. Most flow control instructions use signed offsets from the current instruction address to reference their targets. Unconditional branches are done using *mov* with IP as the destination. Flow control can also use SPF (Single Program Flow) mode to execute with a single instruction pointer (IP).

The EU ISA supports predication, masking, regioning, swizzling, some type conversions, source modification, saturation, accumulator updates, and flag updates as part of instruction execution:

- *Predication* creates a bit mask (*PMask*) to enable or disable channels for a particular instruction execution. Pmask is derived from flag register and subregister values using boolean formulas determined by the PredCtrl (Predicate Control) and PredInv (Predicate Inversion) instruction fields. See the Predication section.
- *Masking* is the overall process of determining which channels execute for a given instruction based on five factors:
  - Number of channels (only channels in [0, *ExecSize* - 1] can execute)
  - Execution mask (*EMask*)
  - Whether the channel is on the instruction (if not in Single Program Flow mode and MaskCtrl is not NoMask)
  - Predicate mask (*PMask*)
  - In Align16 mode, any enabling of channels using the Dst.ChanEn instruction field (if MaskCtrl is not NoMask).
- *Regioning* specifies an array of data elements contained in one or two registers, with options for scattering, interleaving, or repeating data elements in registers using width and stride values, subject to significant constraints. Regioning also includes access mode (Align1 or Align16) and addressing mode (Direct or Indirect). See the Registers and Register Regions section.
- *Swizzling* allows small scale reordering of data elements within groups of four at the input using the modulo 4 channel names x, y, z, and w. For example, a swizzle of .wzyx with an *ExecSize* of 8 reads execution channels 0 to 7 from these input channels: 3, 2, 1, 0, 7, 6, 5, and 4. Swizzling is only available in the Align16 access mode, described in the Execution Environment chapter.

- *Type Conversions* do any needed conversion from source data type to execution data type and from execution data type to destination data type. See Execution Data Type for more information. Each instruction description indicates what combinations of data types are supported.
- *Source Modification* modifies a source operand just before doing the requested operation. For a numeric operation, the choices are:
  - No modification (normal).
  - - indicating negation.
  - (abs) indicating absolute value.
  - -(abs) indicating a forced negative value.

  Source modification logically occurs after any conversion from source data type to execution data type. Each instruction description indicates whether it supports source modification.
- *Saturation* clamps result values to the nearest value within a saturation range determined by the destination type. For a floating-point type, the saturation range is [0.0, 1.0]. For an integer type, the saturation range is the entire range for that type, for example [0, 65535] for the UW (Unsigned Word) type. Each instruction description indicates whether it supports saturation.
- *Accumulator Updates* optionally update the accumulator register or registers in the ARF with destination values as a side effect of instruction execution. The AccWrCtrl instruction field enables accumulator updates. The Accumulator Disable flag in control register 0 (cr0) can be used to disable accumulator updates, regardless of AccWrCtrl values; for example, this flag may be used in the System Routine.
- *Flag Updates* optionally update a flags register and subregister (f0.0, f0.1, f1.0, or f1.1) with conditional flags based on the CondModifier (Condition Modifier) instruction field. For example, a CondModifier of **.nz** (not zero) assigns flag bits based on whether result elements are not zero (1) or zero (0). Each instruction description indicates whether it supports the Condition Modifier and any restrictions on the values supported.

The EU is not required to execute steps in its internal pipeline sequentially or in order, so long as it produces correct results.

The assembler syntax uses spaces between operands and encloses ExecSize and any predicate in parentheses. Instruction mnemonics, register names, conditional modifiers, predicate controls, and type designators use lowercase. Function names used with the math instruction are UPPERCASE.

```
( pred ) inst cmod sat ( exec_size ) dst src0 src1 { inst_opt, ... }
```

General register destination regions use the syntax r*m.n*<*HorzStride*>:type. General register directly addressed source regions use the syntax r*m.n*<*VertStride*;*Width*,*HorzStride*>:type. You need to understand more about register regioning to understand all of these terms.

The following example assembly language instruction adds two packed 16-element single-precision Float arrays in r4/r5 and r2/r3 writing results to r0/r1, only on those channels enabled by the predicate in f0.0 along with any other applicable masks.

```
(f0.0) add (16) r0.0<1>:f r2.0<8;8,1>:f r4.0<8;8,1>:f
```

EU Terms and Acronyms

This section provides three tables describing EU general terms and acronyms, EU data types, and EU selected ARF registers.

EU General Terms and Acronyms

Term	Description
ALT mode	A floating-point execution mode that maps +/- inf to +/- fmax, +/- denorm to +/-0, and NaN to +0 at the FPU inputs and never produces infinities, denormals, or NaN values as outputs. See IEEE mode.
ALU	Arithmetic Logic Unit. A functional block that performs integer arithmetic and logic operations, as distinct from instruction fetch and decode, floating-point operations (see FPU), or messaging.
AOS	Array Of Structures. Also see SOA.
ARF	Architecture Register File, a distinct register file containing registers used to implement specific ISA features. For example the Instruction Pointer and condition flags are in ARF registers. See GRF.
byte	An 8-bit value aligned on an 8-bit boundary and the basic unit of addressing. Bits within a byte are denoted 0 to 7 from LSB to MSB.
channel	A logical unit of SIMD data parallel execution within a thread and within the EU. The number of physical ALUs or FPUs is not directly related to the number of channels. Up to 32 channels are supported.
compact instruction	A 64-bit instruction encoded as described in the EU Compact Instructions section. Only some combinations of instruction parameters can be encoded as compact instructions. See native instruction.
compressed instruction	An instruction that writes to two destination registers. For example, a SIMD16 instruction with Float operands can write channels 0 to 7 to one 32-byte general register and channels 8 to 15 to a second, consecutive 32-byte general register.
denorm	A very small but nonzero number in IEEE mode, with a magnitude less than the smallest normalized floating-point number representable in a particular floating-point format. Denormals lose precision as their values approach zero, called *gradual underflow*.
DWord	Doubleword. A 32-bit (4-byte) value aligned on a 32-bit (4-byte) boundary. Bits within a DWord are denoted 0 to 31 from LSB to MSB.
EOT	End of Thread. A flag set on a *send* or *sendc* instruction to terminate a thread's execution on the EU.
EU	Execution Unit. The single GPU unit described in this volume. This volume describes individual data parallel execution paths within a thread in the EU as *channels*. A few fields, like EUID, use EU to refer to a particular hardware resource used to implement the overall EU.
exception	An error or interrupt condition that arises during execution that may transfer control to the System Routine. Some exceptions can be disabled, preventing such transfers. As defined in this volume, some errors do not produce exceptions.
ExecSize	The number of execution channels for a particular instruction. Channels within that number are enabled or disabled by various masks.
floating-point	Numeric types that allow fractional values and often a wider range than integer types. The EU

Term	Description
	supports binary floating-point types including the single precision type and the double precision type defined by the IEEE 754 standard.
GRF	General Register File, a distinct register file containing 128 general registers, r0 to r127. Each general register is 256 bits (32 bytes), can contain any type of data, and can be accessed with any valid combination of addressing mode, access mode, and region parameters. A general register is directly addressed using a register number and subregister number, or indirectly addressed using an address subregister (index register) and an address immediate offset.
IEEE mode	A floating-point execution mode that supports all the kinds of floating-point values described by the IEEE 754 standard: normalized finite nonzero binary floating-point numbers, signed zeros, signed infinities, signed denormals that are closer to zero than any normalized value but still nonzero, and NaN (not a number) values. See ALT mode.
index register	An address subregister when used for indirect addressing.
inf	Infinity, +inf or -inf, as a floating-point value in IEEE mode.
instruction	In this volume, *instruction* always refers to an EU instruction.
ISA	Instruction Set Architecture, processor aspects visible to programs and programmers and independent of a particular implementation, including data types, registers, memory access, addressing modes, exceptions, instruction encodings, and the instruction set itself. An ISA does not include instruction timing, hardware pipeline details, or the number of physical resources (ALUs, FPUs, instruction decoders) mapped to logical constructs (threads, channels). This volume also includes a recommended assembly language syntax, closely related to the ISA but logically distinct from it.
LSB	Least significant bit.
message	A data structure transmitted from a thread to another thread, to a shared function, or to a fixed function. Message passing is the primary communication mechanism of the GFX architecture.
MSB	Most significant bit.
NaN	Not a Number. A non-numeric value allowed in the standard single precision and double precision floating-point number formats. Quiet NaNs propagate through calculations and signaling NaNs cause exceptions. NaNs are not used in the ALT floating-point mode.
native instruction	A 128-bit instruction, the regular instruction format that allows all defined instruction parameters and options. Some instructions can also be encoded using a 64-bit compact instruction format.
OWord	Octword. A 128-bit (16-byte) value aligned on a 128-bit (16-byte) boundary. Bits within an OWord are denoted 0 to 127 from LSB to MSB. This term is used rarely and may be dropped from future versions of this volume.
packed	A register region is described as *packed* if its elements are adjacent in memory, with no intervening space, no overlap, and no replicated values. If there is more than one element in a row, elements must be adjacent. If there is more than one row, rows must be adjacent. When two registers are used, the registers must be adjacent, and both must exist. The immediate vector data types are all described as *Packed* because each such type packs several small data elements into a 32-bit immediate value.
QWord	Quadword. A 64-bit (8-byte) value aligned on a 64-bit (8-byte) boundary. Bits within a QWord are denoted 0 to 63 from LSB to MSB.

Term	Description
region	A collection of data locations in registers and subregisters for a source or destination operand. The associated regioning parameters allow regions to be arrays with various layouts.
register	Part of the directly accessible state of an EU program, such as a general register in the GRF or an architecture register in the ARF. Note that system memory is not directly accessible.
SIMD	Single Instruction Multiple Data. Each EU instruction can operate on multiple data elements in parallel, as specified by the instruction's ExecSize.
SIP	System Instruction Pointer, the starting IP value for the System Routine.
SOA	Structure of Arrays. Also see AOS.
SPF	Single Program Flow. A mode in which every execution channel uses the common instruction pointer, IP in the ip register. The SPF bit in the control register is 1 to enable SPF and 0 to disable it. If SPF is disabled, then each execution channel n has its own instruction pointer, PcIP[n] and each channel n is only eligible to execute, subject to other masking, when PcIP[n] == IP.
swizzle	Rearrange data elements within a vector. The EU supports modulo four swizzling of register source operands at the input in the Align16 access mode.
System Routine	A global EU exception handling routine. Any enabled exception from any EU thread transfers control to this routine.
thread	An instance of a program executing on the EU. The life cycle for a thread on the EU starts with the first instruction after being dispatched to the EU by the Thread Dispatcher and ends after executing a *send* or *sendc* instruction with EOT set, signaling thread termination. Threads can be independent or can communicate with each other via the Message Gateway shared function.
word	A 16-bit (2-byte) value aligned on a 16-bit (2-byte) boundary. Bits within a word are denoted 0 to 15 from LSB to MSB. *Word* has denoted a 16-bit unit for Intel processors since the 8086 and 8088 processors were introduced in 1978.

The next table lists all EU numeric data types. See the Numeric Data Types section for more information about each data type.

EU Numeric Data Types (Listed Alphabetically by Short Name)

Short Name	Assembler Syntax	Long Name	Size in Bytes	Size in Bits	Integral or Float	Description
B	:b	Signed Byte Integer	1	8	I	Signed integer in the range -128 to 127.
D	:d	Signed Doubleword Integer	4	32	I	Signed integer in the range -2^{31} to 2^{31} - 1.
DF	:df	Double Float	8	64	F	Double precision floating-point number.
F	:f	Float	4	32	F	Single precision floating-point number.
HF	:hf	Half Float	2	16	F	Half precision floating-point number.
Q	:q	Signed Quadword Integer	8	64	I	Signed integer in the range -2^{63} to 2^{63} - 1.
UB	:ub	Unsigned Byte Integer	1	8	I	Unsigned integer in the range 0 to 255.
UD	:ud	Unsigned Doubleword Integer	4	32	I	Unsigned integer in the range 0 to 2^{32} - 1.
UQ	:uq	Unsigned Quadword Integer	8	64	I	Unsigned integer in the range 0 to 2^{64} - 1.

Short Name	Assembler Syntax	Long Name	Size in Bytes	Size in Bits	Integral or Float	Description
UV	:uv	Packed Unsigned Half Byte Integer Vector	4	32	I	Eight 4-bit unsigned integer values each in the range 0 to 15. Only used as an immediate value.
UW	:uw	Unsigned Word Integer	2	16	I	Unsigned integer in the range 0 to 65,535.
V	:v	Packed Signed Half Byte Integer Vector	4	32	I	Eight 4-bit signed integer values each in the range -8 to 7. Only used as an immediate value.
VF	:vf	Packed Restricted Float Vector	4	32	F	Four 8-bit restricted float values. Only used as an immediate value.
W	:w	Signed Word Integer	2	16	I	Signed integer in the range -32,768 to 32,767.

The next table lists the seven ARF registers that you should understand first, omitting several others. See the ARF Registers section for more information, including descriptions of additional registers not listed below.

EU Selected ARF Registers (Listed Alphabetically by Name)

Name	Assembler Syntax	Description
Accumulators	acc0, acc1	Data registers that can hold integer or floating-point values of various sizes. Many instructions can implicitly update accumulators with a copy of destination values, done by setting the AccWrCtrl instruction option. A few instructions, like *mac* (Multiply Accumulate), use the accumulators as an implicit source operand, useful for some iterative calculations. There are added accumulator registers acc2 to acc9 for special purposes; these added accumulators are not generally used.
Address Register	a0.*s*	Holds subregisters primarily used for indirect addressing. Each subregister is a 16-bit UW (Unsigned Word) value. For an indirectly addressed operand or element, the subregister value plus an AddrImm signed offset field determines the byte address (RegNum and SubRegNum) within the register file (GRF). There are 16 address subregisters.
Control Register	cr0.*s*	Contains bit fields for floating-point modes, flow control modes, and exception enable/disable. Also contains exception indicator flags and saves the AIP (Application Instruction Pointer) on transferring control to the System Routine to handle an exception.
Flags	f*r.s*	Used as the outputs for various channel conditional signals, such as equality/zero or overflow. Used as the inputs for predication. There are two 32-bit flags registers each containing two 16-bit subregisters.
Instruction Pointer (IP)	ip	References the current instruction in memory, as an unsigned offset from the General State Base Address. IP is the thread's overall instruction pointer. Each channel n can have its own instruction pointer (PcIP[n]). If not in Single Program Flow mode (SPF is 0) then only those channels where PcIP[n] == IP are eligible to execute the

Name	Assembler Syntax	Description
		instruction, if enabled by all other applicable masks.
Null Register	null	Indicates a non-existent operand. Unused operands in the instruction format, like the unused second source operand field in a *mov* instruction, are encoded as null. For present source operands, reading a null source operand returns undefined values. For null destination operands, results are discarded but any implicit updates to accumulators or flags still occur.
State Register	sr0.*s*	Contains thread identification and scheduling fields, and mask fields for enabling or disabling channels.

Execution Units (EUs)

Each EU is a vector machine capable of performing a given operation on as many as 16 pieces of data of the same type in parallel (though not necessarily on the same instant in time). In addition, each EU can support a number of execution contexts called *threads* that are used to avoid stalling the EU during a high-latency operation (external to the EU) by providing an opportunity for the EU to switch to a completely different workload with minimal latency while waiting for the high-latency operation to complete.

For example, if a program executing on an EU requires a texture read by the sampling engine, the EU may not necessarily idle while the data is fetched from memory, arranged, filtered and returned to the EU. Instead, the EU will likely switch execution to another (unrelated) thread associated with that EU. If that thread encounters a stall, the EU may switch to yet another thread and so on.

Once the Sampler result arrives back at the EU, the EU can switch back to the original thread and use the returned data as it continues execution of that thread.

The fact that there are multiple EU cores each with multiple threads can generally be ignored by software. There are some exceptions to this rule: e.g., for:

Description
thread-to-thread communication (see *Message Gateway*, *Media*)
synchronization of thread output to memory buffers (see *Geometry Shader*)

In contrast, the internal SIMD aspects of the EU are very much exposed to software.

This volume will not deal with the details of the EUs.

EU Changes by Processor Generation

This section describes how the EU changes for particular processor generations. Instruction compaction tables can differ for each generation, so that is not mentioned in these lists. Particular readers and audiences can see only certain content in this section. Workarounds for particular generations, SKUs, or steppings are not included in these lists. Some small changes in instruction layouts are not included in these lists.

These are the features or behaviors:

- The maximum *ExecSize* increases to 32, for byte or word operands.
- Increase the number of flag registers from one to two.
- Add the NibCtrl field, used with QtrCtrl to select groups of channels or flags.
- Add the DF (Double Float) data type, the first time an 8-byte data type is supported. DF only supports the IEEE floating-point mode and not the ALT floating-point mode.
- Add a shared source data type field and a destination data type field for instructions with three source operands, allowing F (Float), DF (Double Float), D (Signed Doubleword Integer), or UD (Unsigned Doubleword Integer) types to be specified.
- Add bit manipulation instructions: *bfi1*, *bfi2*, *bfrev*, *cbit*, *fbh*, and *fbl*.
- Add the integer *addc* (Add with Carry) and *subb* (Subtract with Borrow) instructions.
- Add the *brc* (Branch Converging) and *brd* (Branch Diverging) instructions.
- For the *cmp* and *cmpn* instructions, relax the accumulator restrictions.
- For the *sel* instruction, remove the accumulator restriction.
- Add the Rounding Mode and Double Precision Denorm Mode fields in Control Register 0.
- DF (Double Float) operands use an element size of 8. Regioning and channel parameters for the DF type are determined normally, in the same way as for other types.
- Add the channel enable register, flow control registers, and stack pointer register in the ARF.
- In the Control Register, add the Force Exception Status and Control, Context Save Status, and Context Restore Status bits.
- Relative instruction offsets (JIP, UIP) are now 32-bit values in units of bytes (rather than 16-bit values using 8-byte units) for some instructions: *brc*, *brd*, *call*, and *jmpi*.
- A *call* instruction can get the relative instruction offset (JIP) from a register.
- Add the *calla* (Call Absolute) instruction.
- A *mov* instruction with different source and destination types can now use conditional modifiers.
- Add the HF (Half Float) type and a corresponding HF execution data type and execution path.
- Add flags to indicate IEEE floating-point exceptions and to enable or disable exception reporting to those flags.
- Add the Single Precision Denorm Mode bit in Control Register 0. It can be enabled to allow calculations using the F (Float) type in IEEE floating-point mode to support denormals and gradual underflow.

- Add the Q (Signed Quadword Integer) and UQ (Unsigned Quadword Integer) types. Integer source types cannot mix 64-bit and 8-bit operands. Some integer instructions (e.g., *avg*) do not support Q or UQ source types.
- Instructions with one source operand and a 64-bit source type can have immediate 64-bit source operands.
- The JIP and UIP relative instruction offset fields in all remaining flow control instructions are 32-bit values in units of bytes (rather than 16-bit values using 8-byte units).
- The instruction layout is noticeably different. The SrcType and DstType instruction fields are widened to allow for more type encodings as three types are added. The AddrSubRegNum instruction field is widened to allow for 16 address subregisters rather than 8. The layout now supports 64-bit immediate source operands for one-source instructions and 32-bit relative instruction offset fields for flow control instructions.
- In the 3-source instruction format, widen the SrcType and DstType fields and add an encoding for the HF (Half Float) type.
- Add a compact instruction format for 3-source instructions.
- Use a different source modifier interpretation for logical (*and*, *not*, *or*, *xor*) instructions.
- An accumulator source operand for a logical instruction can now have a source modifier.
- Add eight address subregisters, increasing the number of address subregisters from 8 to 16.
- For the *brc* and *brd* instructions do not allow the **Switch** instruction option.
- For the *cmp* and *cmpn* instructions, remove the accumulator restrictions.
- Add the *goto* instruction, reusing the opcode for the discontinued *fork* instruction.
- Add the *join* instruction.
- For the *lzd* instruction, remove the accumulator restriction.
- The *mach* instruction reverses the roles of the two source operands compared to previous generations.
- Add the *madm* instruction.
- Enhance the *math* instruction to allow some immediate source values and support the INVM and RSQRTM functions.
- For the *mul* instruction, relax the accumulator restriction on source operands so it applies for only integer source operands.
- For the rounding instructions (*rndd*, *rnde*, *rndu*, and *rndz*), remove the accumulator restrictions.
- Revise the *shl* and *shr* instructions to use the low 6 bits of the shift count in QWord mode, versus the low 5 bits otherwise.
- Add the *smov* instruction.
- Add eight accumulator registers, acc2 to acc9, used only for the special purpose of emulating IEEE-compliant fdiv and sqrt operations.
- Add message control registers.
- Widen the sp (Stack Pointer) register to 64 bits.
- Add the IEEE Exception, Page Fault Status, and Page Fault Code bit fields in the State Register.

- Remove some regioning restrictions when operands span two registers.
- Add the Half Precision Denorm Mode bit in the Control Register. It can be enabled to allow calculations using the HF (Half Float) type to support denormals and gradual underflow.
- Restrict Align 16 access mode only to IEEE macros
- Remove the Align 16 access mode
- Add *dp4a* instruction.
- Add *rol* and *ror* instruction for dword datatype.
- Remove *pln, line* and *dp* product instruction.
- Restriction on source2 regioning - byte datatype support removed, swizzling of data also removed except for broadcast.
- Add 8bits of software hints in instruction to facilitate scheduling.

Byte datatype support removed from source1.

EU Notation

The Courier New font is used for code examples and for the Syntax, Format, and Pseudocode sections in the instruction reference.

The *italic* font style is used for instruction mnemonics outside of code (e.g., the *send* instruction), for syntactic production names, for key values in algorithms (*ExecSize*), and to emphasize a word or phrase. For example: When bit 10 is set, the destination register scoreboard is *not* cleared.

The **bold** font weight is used for the short name and long name of a bit field being described, for value names being defined, for syntactic terminals, for unnumbered subheadings, and for the terms Note or Workaround used to introduce a paragraph.

Bit field names and value names used where not being defined and not as syntactic terminals are in plain text.

Bit field values in hex use the 0x prefix. The PRM currently uses the 0x prefix for hex in some parts and the h suffix for hex in other parts. For single bits, values appear as simply 0 or 1. For multi-bit binary values, the appropriate number of binary digits appears with a b suffix.

Instruction mnemonics are lowercase. Function names invoked using the *math* instruction are UPPERCASE. For example, SQRT.

Device names in the new syntax do not use the Dev prefix or square brackets and often appear in tagged tables with a blue background. For example:

Tables describing bit field layouts or registers proceed from most significant to least significant bits. Figures showing bit fields or registers show most significant bits on the left and least significant bits on the right.

Any bit, field, or register described as Reserved should be regarded as undefined and unpredictable. Such bits should be treated as follows:

- When testing values, do not depend on the state of reserved bits. Mask out or otherwise ignore such bits.

- Sometimes software must initialize reserved bits. For example, a compiler must write complete instruction values when creating an instruction stream, including reserved bits. In such cases, write reserved bits as zeros unless otherwise indicated.
- Do not use reserved bits as extra storage for software-defined values; put nothing in such bits.
- When saving state and restoring state, save and restore any reserved bits as well.
- Do not assume that reserved bits are invariant between explicit writes. Software should function even if reserved bits change in undefined and unpredictable ways.

Any value, encoding, or combination of values or encodings described as Reserved must not be used. The EU's behavior is undefined in this case.

When a combination of instruction parameters or an EU state is described as producing undefined results or behavior, do not assume that undefined results or behavior are confined to specific instructions, operands, registers, or channels.

Execution Environment

This topic is currently under development.

EU Data Types

Fundamental Data Types

Numeric Data Types

Floating Point Modes

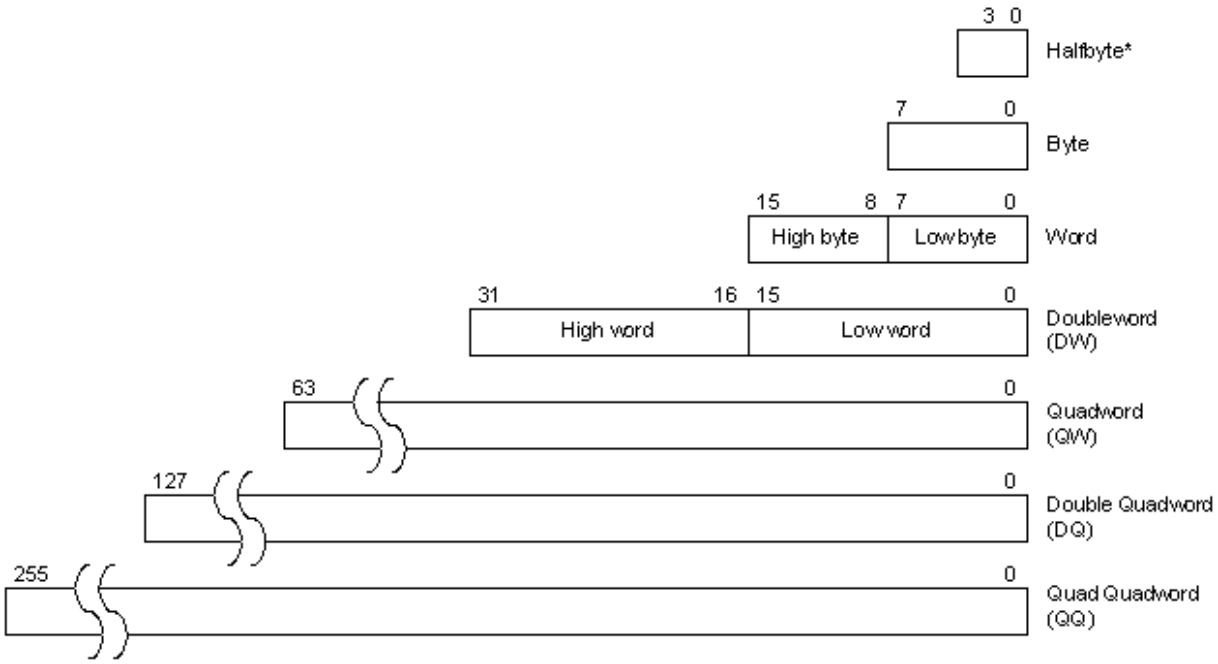
- IEEE Floating Point Mode
  - Partial Listing of Honored IEEE 754 Rules
  - Complete Listing of Deviations or Additional Requirements vs IEEE 754
  - Min/Max of Floating Point Numbers
- Alternative Floating Point Mode
- Floating-Point Support
  - IEEE Floating-Point Exceptions
  - Floating-Point Compare Operations
  - Type Conversion

Fundamental Data Types

The fundamental data types in the architecture are halfbyte, byte, word, doubleword (DW), quadword (QW), double quadword (DQ) and quad quadword (QQ). They are defined based on the number of bits of the data type, ranging from 4 bits to 256 bits. As shown in the figure below, a halfbyte contains 4 bits, a byte contains 8 bits, a word contains two bytes, a doubleword (DWord) contains two words, and so on. Halfbyte is a special data type that is not accessed directly as a standalone data element; it is only

allowed as a subfield of the numeric data type of "packed signed halfbyte integer vector" described in the next section.

Fundamental Data Types



With the exception of halfbyte, the access of a data element to/from a register or to/from memory must be aligned on the natural boundaries of the data type. The natural boundary for a word has an even-numbered address in units of bytes. The natural boundary for a doubleword has an address divisible by 4 bytes. Similarly, the natural boundary for a quadword, double quadword, and quad quadword has an address divisible by 8, 16, and 32 bytes, respectively. Double quadword, and quad quadword do not have corresponding numeric data types. Instead, they are used to describe a group (a vector) of numeric data elements of smaller size aligned to larger natural boundaries.

Numeric Data Types

The numeric data types defined in the architecture include signed and unsigned integers and floating-point numbers (floats) of various sizes. These numeric data types are described below.

Integer Numeric Data Types

The Execution Unit supports the following integer data types. Signed integer types use two's complement representation for negative numbers.

UB: Unsigned Byte, 8-bit Unsigned Integer

7		0

B: Byte, 8-bit Signed Integer

7	6		0
S			

UW: Unsigned Word, 16-bit Unsigned Integer

15		0

W: Word, 16-bit Signed Integer

15	14		0
S			

UD: Unsigned Doubleword, 32-bit Unsigned Integer

31		0

D: Doubleword, 32-bit Signed Integer

31	30		0
S			

UQ: Unsigned Quadword, 64-bit Unsigned Integer

63		0

Q: Quadword, 64-bit Signed Integer

63	62		0
S			

UV: Packed Unsigned Half-Byte Integer Vector, 8 x 4-Bit Unsigned Integer

31		28	27		24	23		20	19		16	15		12	11		8	7		4	3		0

V: Packed Signed Half-Byte Integer Vector, 8 x 4-Bit Signed Integer

31		28	27		24	23		20	19		16	15		12	11		8	7		4	3		0
S			S			S			S			S			S			S			S		

The following table summarizes the EU integer data types.

Execution Unit Integer Data Types

Notation	Size in Bits	Name	Range
UB	8	Unsigned Byte Integer	[0, 255]
B	8	Signed Byte Integer	[-128, 127]
UW	16	Unsigned Word Integer	[0, 65535]
W	16	Signed Word Integer	[-32768, 32767]
UD	32	Unsigned Doubleword Integer	[0, 2^{32} - 1]
D	32	Signed Doubleword Integer	[-2^{31}, 2^{31} - 1]
UQ	64	Unsigned Quadword Integer	[0, 2^{64} - 1]
Q	64	Signed Quadword Integer	[-2^{63}, 2^{63} - 1]
UV	32	Packed Unsigned Half-Byte Integer Vector	[0, 15] in each of eight 4-bit immediate vector elements.
V	32	Packed Signed Half-Byte Integer Vector	[-8, 7] in each of eight 4-bit immediate vector elements.

Restriction: Only a raw move using the *mov* instruction supports a packed byte destination register region. For information about raw moves, refer to the **Description** in *mov - Move*.

Floating-Point Numeric Data Types

The Execution Unit supports the following floating-point data types.

Type or Description
The Float type uses the single precision format specified in IEEE Standard 754-1985 for Binary Floating-Point Arithmetic.
The Double Float type uses the double precision format specified in IEEE Standard 754-1985 for Binary Floating-Point Arithmetic.
In the ALT floating-point mode, representations for infinities, denorms, and NaNs within those formats are not used. The EU does not support the double extended precision (80-bit) floating-point format found in the x86/x87/Intel 64 floating-point registers. All floating-point formats are signed using signed magnitude representation (a distinct sign bit, separate from the magnitude information).
The Half Float type uses the binary16 format specified in IEEE Standard 754-2008.
The F (Float) type supports both the ALT and IEEE floating-point modes, controlled by the Single Precision Floating-

Type or Description
Point Mode bit in the Control Register.
Whether IEEE mode F calculations support denorms or flush denormalized values to zero is controlled by the Single Precision Denorm Mode bit in the Control Register.
The DF (Double Float) type only supports the IEEE floating-point mode. Whether DF calculations support denorms or flush denormalized values to zero is controlled by the Double Precision Denorm Mode bit in the Control Register.
The HF (Half Float) type only supports the IEEE floating-point mode.
Whether HF calculations support denorms or flush denormalized values to zero is controlled by the Half Precision Denorm Mode bit.

HF: Half Float, 16-bit Half-Precision Floating-Point Number

15	14		10	9		0
S	biased exp.			fraction		

F: Float, 32-bit Single-Precision Floating-Point Number

31	30		23	22		0
S	biased exponent			fraction		

DF: Double Float, 64-bit Double-Precision Floating-Point Number

63	62		52	51		0
S	biased exponent			fraction		

VF: Packed Restricted Float Vector, 4 x 8-Bit Restricted Precision Floating-Point Number

31	30		28	27		24	23	22		20	19		16	15	14		12	11		8	7	6		4	3		0
S	b. exp.			frac.			S	b. exp.			frac.			S	b. exp.			frac.		S	b. exp.			frac.			

The following table summarizes the EU floating-point data types.

Execution Unit Floating-Point Data Types

Notation	Size in Bits	Name	Range
HF	16	Half Float	Half precision, 1 sign bit, 5 bits for the biased exponent, and 10 bits for the significand: $[-(2-2^{-10})^{15}...-2^{-24}, 0.0, 2^{-24}... (2-2^{-10})^{15}]$
F	32	Float	Single precision, 1 sign bit, 8 bits for the biased exponent, and 23 bits for

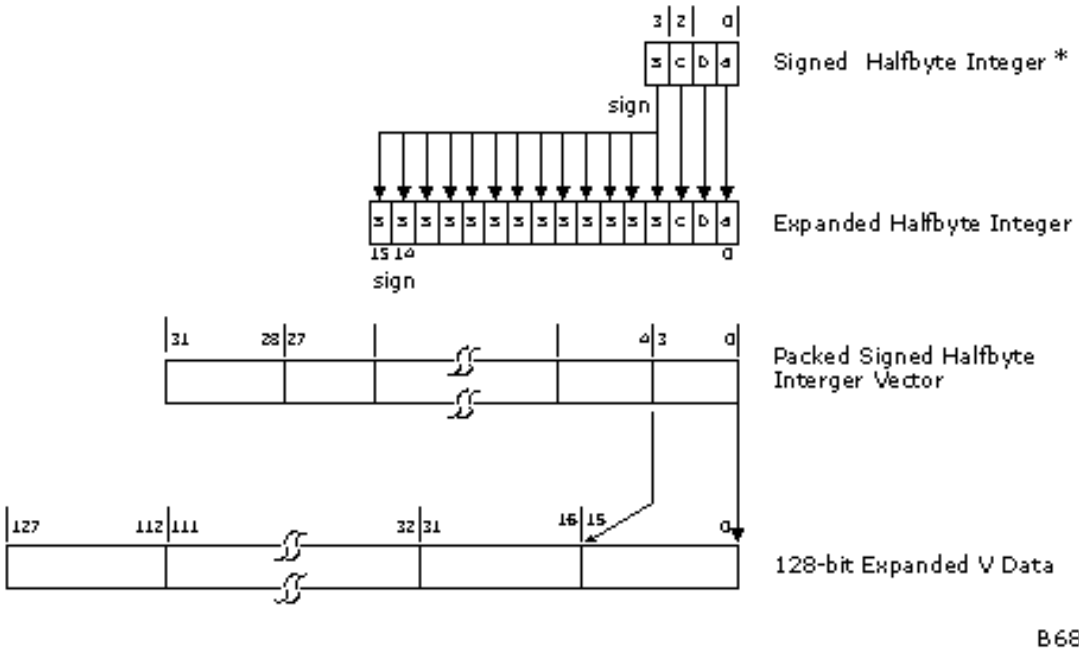
Notation	Size in Bits	Name	Range
			the significand: $[-(2-2^{-23})^{127}...-2^{-149}, 0.0, 2^{-149}... (2-2^{-23})^{127}]$
DF	64	Double Float	Double precision, 1 sign bit, 11 bits for the biased exponent, and 52 bits for the significand: $[-(2-2^{-52})^{1023}...-2^{-1074}, 0.0, 2^{-1074}... (2-2^{-52})^{1023}]$
VF	32	Packed Restricted Float Vector	Restricted precision. Each of four 8-bit immediate vector elements has 1 sign bit, 3 bits for the biased exponent (bias of 3), and 4 bits for the significand: [-31... -0.1328125, -0, 0, 0.1328125 ... 31]

Packed Signed Half-Byte Integer Vector

A packed signed halfbyte integer vector consists of 8 signed halfbyte integers contained in a doubleword. Each signed halfbyte integer element has a range from -8 to 7 with the sign on bit 3. This numeric data type is only used by an immediate source operand of doubleword in an instruction. It cannot be used for the destination operand or a non-immediate source operand. Hardware converts the vector into an 8-element signed word vector by sign extension. This is illustrated in Converting a packed half-byte vector to a 128-bit signed integer vector.

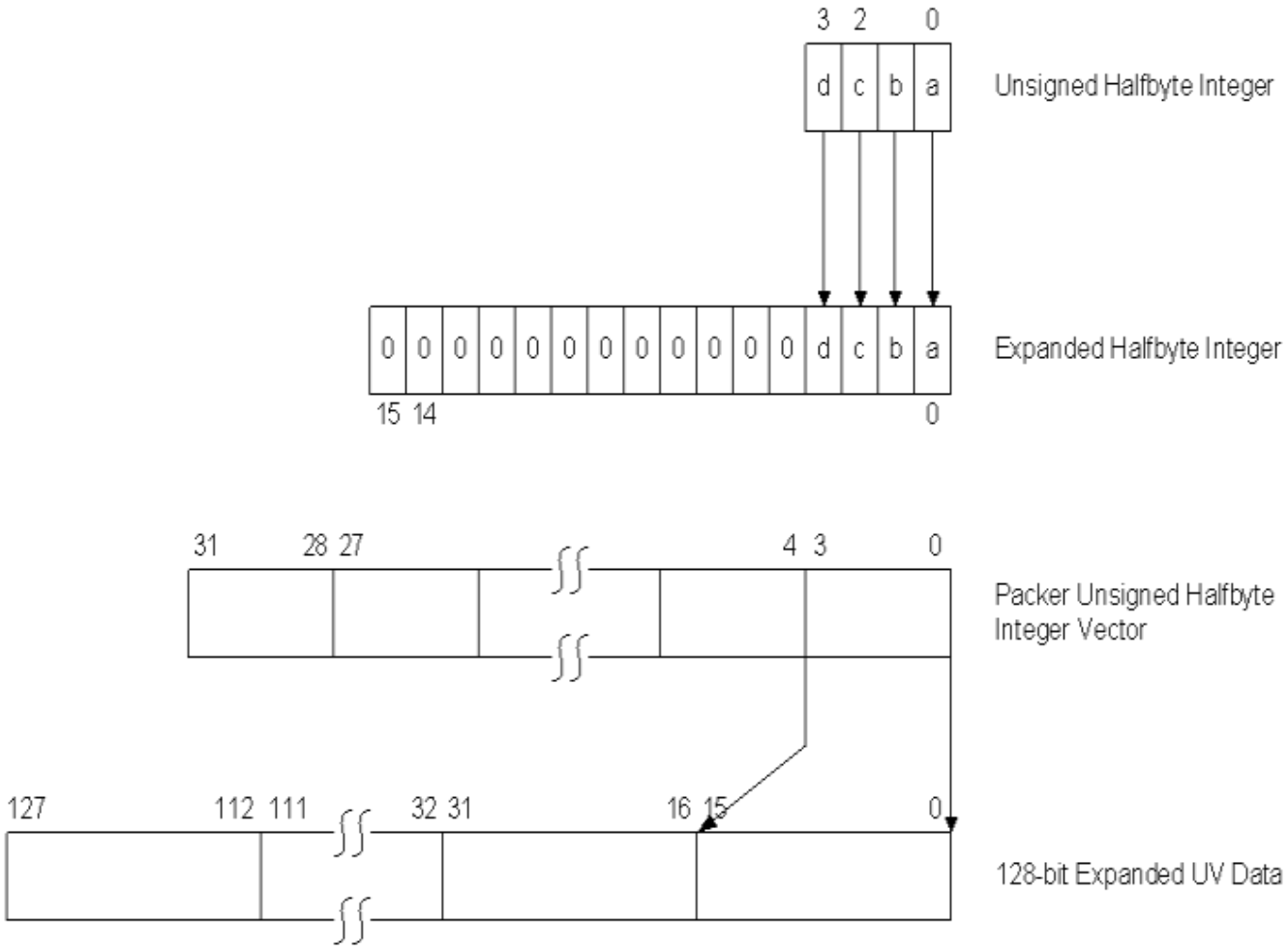
The short hand format notation for a packed signed half-byte vector is **V**.

Converting a Packed Half-Byte Vector to a 128-bit Signed Integer Vector



Packed UnSigned Half-Byte Integer Vector

A packed unsigned halfbyte integer vector consists of 8 unsigned halfbyte integers contained in a doubleword. Each unsigned halfbyte integer element has a range from 0 to 15. This numeric data type is only used by an immediate source operand of doubleword in an instruction. It cannot be used for the destination operand or a non-immediate source operand. Hardware converts the vector into an 8-element signed word vector.

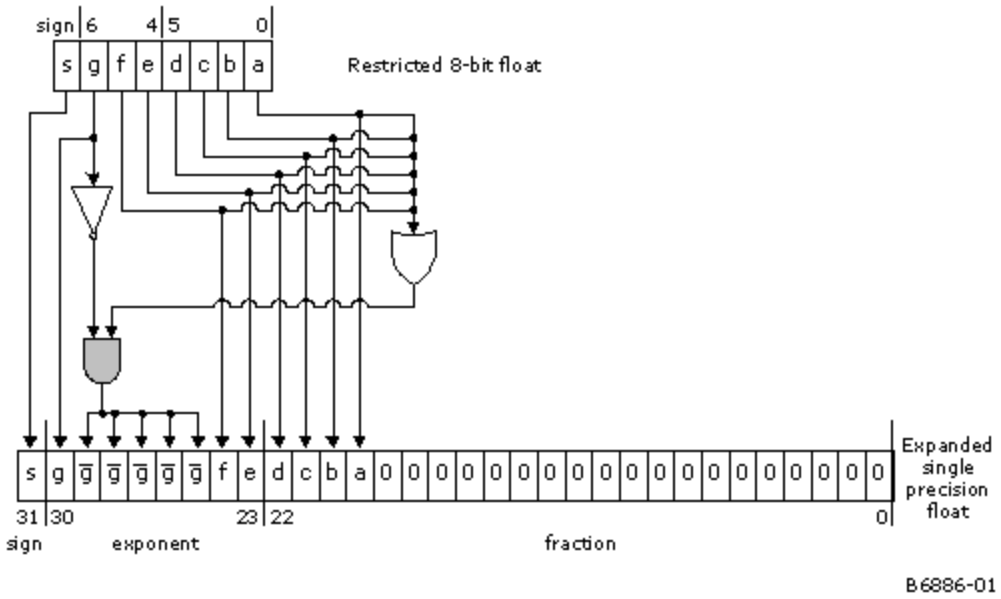


Packed Restricted Float Vector

A packed restricted float vector consists of 4 8-bit restricted floats contained in a doubleword. Each restricted float has the sign at bit 7, a 3-bit coded exponent in bits 4 to 6, a 4-bit fraction in bits 0 to 3, and an implied integer 1. The exponent is in excess-3 format - having a bias of 3. Restricted float provides zero, positive/negative normalized numbers with a small range (3-bit exponent) and small precision (4-bit fraction). This numeric data type is only used by an immediate source operand of doubleword in an instruction. It cannot be used for the destination operand, or a non-immediate source operand.

The following figure shows how to convert an 8-bit restricted float into a single precision float. Converting a 3-bit exponent with a bias of 3 to an 8-bit exponent with a bias of 127 is by adding 4, or equivalently copying bit 2 to bit 7 and putting the inverted bit 2 to bits 6:2. A special logic is also needed to take care of positive/negative zeros.

Conversion from a Restricted 8-bit Float to a Single-Precision Float



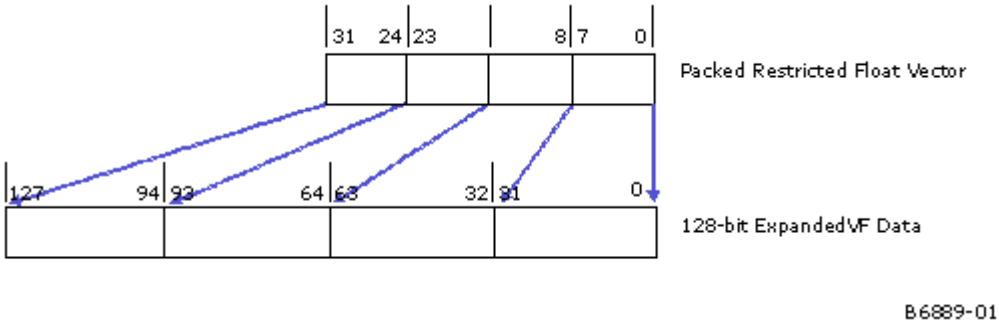
The following table shows all possible numbers of the restricted 8-bit float. Only normalized float numbers can be represented, including positive and negative zero, and positive and negative finite numbers. Normalized infinites, NaN, and denormalized float numbers cannot be represented by this type. It should be noted that this 8-bit floating point format does not follow IEEE-754 convention in describing numbers with small magnitudes. Specifically, when the exponent field is zero and the fraction field is not zero, an implied one is still present instead of taking a denormalized form (without an implied one). This results in a simple implementation but with a smaller dynamic range - the magnitude of the smallest non-zero number is 0.1328125.

Examples of Restricted 8-bit Float Numbers

Class	Hex #	Sign [7]	Exponent [6:4]	Fraction [3:0]	Extended 8-bit Exponent	Floating Number in Decimal
Positive Normalized Float	0x70-0x7F	0	111	0000 ... 1111	1000 0011	16 ... 31
	0x60-0x6F	0	110	0000 ... 1111	1000 0010	8 ... 15.5
	0x50-0x5F	0	101	0000 ... 1111	1000 0001	4 ... 7.75
	0x40-0x4F	0	100	0000 ... 1111	1000 0000	2 ... 3.875
	0x30-0x3F	0	011	0000 ... 1111	0111 1111	1 ... 1.9375
	0x20-0x2F	0	010	0000 ... 1111	0111 1110	0.5 ... 0.96875
	0x10-0x1F	0	001	0000 ... 1111	0111 1101	0.25 ... 0.484375
	0x01-0x0F	0	000	0001 ... 1111	0111 1100	0.1328125 ... 0.2421875
	0x00	0	000	0000	0000 0000	0 (+zero)
Negative Normalized Float	0xF0-0xFF	1	111	0000 ... 1111	1000 0011	-16 ... -31
	0xE0-0xEF	1	110	0000 ... 1111	1000 0010	-8 ... -15.5
	0xD0-0xDF	1	101	0000 ... 1111	1000 0001	-4 ... -7.75
	0xC0-0xCF	1	100	0000 ... 1111	1000 0000	-2 ... -3.875
	0xB0-0xBF	1	011	0000 ... 1111	0111 1111	-1 ... -1.9375
	0xA0-0xAF	1	010	0000 ... 1111	0111 1110	-0.5 ... -0.96875
	0x90-0x9F	1	001	0000 ... 1111	0111 1101	-0.25 ... -0.484375
	0x81-0x8F	1	000	0001 ... 1111	0111 1100	-0.1328125 ... -0.2421875
	0x80	1	000	0000	0000 0000	-0 (-zero)

The following figure shows the conversion of a packed exponent-only float to a 4-element vector of single precision floats.

The shorthand format notation for a packed signed half-byte vector is VF.



Floating Point Modes

Architecture supports two floating point operation modes, namely IEEE floating point mode (IEEE mode) and alternative floating-point mode (ALT mode). Both modes follow mostly the requirements in IEEE-754 but with different deviations. The deviations will be described in detail in later sections. The primary difference between these modes is on the handling of Infs, NaNs and denorms. The IEEE floating point mode may be used to support newer versions of 3D graphics API Shaders and the alternative floating-point mode may be used to support early Shader versions. Taking DirectX 3D graphics API Shaders for example, shader models before version 3.0 may use the alternative floating-point mode, while version 3.0 and following shader models may use the IEEE floating point mode.

These two modes are supported by all units that perform floating point computations, including execution units, shared functions like Extended Math and the Render Cache color calculator, and fixed functions like VF, Clipper, SF and WIZ. Host software sets floating point mode through the fixed function state descriptors for 3D pipeline and the interface descriptor for media pipeline. Therefore, different modes may be associated with different threads running concurrently. Floating point mode control for EU and shared functions are based on the floating-point mode field (bit 0) of *cr0* register.

These two modes are supported by all units that perform floating point computations, including execution units, shared functions like Extended Math, the Sampler and the Render Cache color calculator, and fixed functions like VF, Clipper, SF and WIZ. Host software sets floating point mode through the fixed function state descriptors for 3D pipeline and the interface descriptor for media pipeline. Therefore, different modes may be associated with different threads running concurrently. Floating point mode control for EU and shared functions are based on the floating-point mode field (bit 0) of *cr0* register.

IEEE Floating Point Mode

Partial Listing of Honored IEEE-754 Rules

Here is a summary of expected 32-bit floating point behaviors in architecture. Refer to IEEE-754 for topics not mentioned.

- INF - INF = NaN
- 0 * (+/-)INF = NaN
- 1 / (+INF) = +0 and 1 / (-INF) = -0
    - (+/-)INF / (+/-)INF = NaN as A/B = A * (1/B)

- INV (+0) = RSQ (+0) = +INF, INV (-0) = RSQ (-0) = -INF, and SQRT (-0) = -0
- RSQ (-finite) = SQRT (-finite) = NaN
- LOG (+0) = LOG (-0) = -INF, LOG (-finite) = LOG (-INF) = NaN
- NaN (any OP) any-value = NaN with one exception for min/max mentioned below. Resulting NaN may have different bit pattern than the source NaN.
- Normal comparison with conditional modifier of EQ, GT, GE, LT, LE, when either or both operands is NaN, returns FALSE. Normal comparison of NE, when either or both operands is NaN, returns TRUE.
    - **Note:** Normal comparison is either a **cmp** instruction or an instruction with conditional modifier
- Special comparison **cmpn** with conditional modifier of EQ, GT, GE, LT, LE, when the second source operand is NaN, returns TRUE, regardless of the first source operand, and when the second source operand is not NaN, but first one is, returns FALSE. **Cmpn** of NE, when the second source operand is NaN, returns FALSE, regardless of the first source operand, and when the second source operand is not NaN, but first one is, returns TRUE.
    - This is used to support the proposed IEEE-754R rule on **min** or **max** operations. For which, if only one operand is NaN, min and max operations return the other operand as the result.
- Both normal and special comparisons of any non-NaN value against +/- INF return exact result according to the conditional modifier. This is because that infinities are exact representation in the sense that +INF = +INF and -INF = -INF.
    - NaN is unordered in the sense that NaN != NaN.
- IEEE-754 requires floating point operations to produce a result that is the nearest representable value to an infinitely precise result, known as "round to nearest even" (RTNE). 32-bit floating point operations must produce a result that is within 0.5 Unit-Last-Place (0.5 ULP) of the infinitely precise result. This applies to addition, subtraction, and multiplication.
- All arithmetic floating-point instructions does Round To Nearest Even at the end of the computation, except the round instructions.

Complete Listing of Deviations or Additional Requirements vs IEEE-754

For a result that cannot be represented precisely by the floating point format, the EU uses rounding to nearest or even to produce a result that is within 0.5 Unit-Last-Place(0.5 ULP) of the infinitely precise result.

Description
The rounding mode is specified by the Rounding Mode field in the Control Register.
Handle denorms as follows: • When inputs are denorms in mixed mode (one of the source operand is half float and other is single precision float OR source is half float and destination is single precision float), on upconversion, they are non-denorms. So computes in 32b can handle this. • When outputs are denorms, the denorms are retained if there is a format conversion.

Description
• Mixed mode operations should always behave like sum of individual operations.

Other information regarding floating-point behaviors:

- NaN input to an operation always produces NaN on output, however the exact bit pattern of the NaN is not required to stay the same (unless the operation is a raw "mov" instruction which does not alter data at all.)
- x*1.0f must always result in x (except denorm flushed and possible bit pattern change for NaN).
- x +/- 0.0f must always result in x (except denorm flushed and possible bit pattern change for NaN). But -0 + 0 = +0.
- Fused operations (such as *mac*, *dp4*, *dp3*, etc.) may produce intermediate results out of 32-bit float range, but whose final results would be within 32-bit float range if intermediate results were kept at greater precision. In this case, implementations are permitted to produce either the correct result, or else +/-inf. Thus, compatibility between a fused operation, such as *mac*, with the unfused equivalent, *mul* followed by *add* in this case, is not guaranteed.
- As the accumulator registers have more precision than 32-bit float, any instruction with accumulator as a source/destination operand may produce a different result than that using more general registers, as indicated in this table:
- Such an instruction may produce a different result than that using GRF registers.
- API Shader divide operations are implemented as x*(1.0f/y). With the two-step method, x*(1.0f/y), the multiply and the divide each independently operate at the 32-bit floating point precision level (accuracy to 1 ULP).
- See the Type Conversion section for rules on converting to and from float representations.

Min Max of Floating Point Numbers

A special comparison called Compare-NaN is introduced in the architecture to handle the difference of above-mentioned floating-point comparison and the rules on supporting MIN/MAX. To compute the MIN or MAX of two floating-point numbers, if one of the numbers is NaN and the other is not, MIN or MAX of the two numbers returns the one that is not NaN. When two numbers are NaN, MIN or MAX of the two numbers returns source1.

When both the sources are NaN inputs, the special case in the section 2.4.8 Floating point Min/Max Operations describe the results.

When one source is SNAN, DX,OCL and IEEE treat the outputs differently. The special case section 2.4.8 Floating point Min/Max operations described the results.

Min and Max is supported by conditional select.

Note even though f0.0 is specified in the instruction, the flag register is not touched by this instruction.

The following tables detail the rules for this special compare-NaN operation for floating-point numbers. Notice that excepting "Not-Equal-To" comparison-NaN, last columns in all other tables have 'T'.

Results of "Less-Than" Comparison-NaN - CMPN.L

src0 src1	-inf	-Fin	-denorm	-0	+0	+denorm	+Fin	+inf	NaN
-inf	F	T	T	T	T	T	T	T	T
-Fin	F	T/F	T	T	T	T	T	T	T
-denorm	F	F	F	F	F	F	T	T	T
-0	F	F	F	F	T	F	T	T	T
+0	F	F	F	F	F	F	T	T	T
+denorm	F	F	F	F	F	F	T	T	T
+Fin	F	F	F	F	F	F	T/F	T	T
+inf	F	F	F	F	F	F	F	F	T
NaN	F	F	F	F	F	F	F	F	F

Results of "Greater-Than or Equal-To" Comparison-NaN - CMPN.GE

src0 src1	-inf	-Fin	-denorm	-0	+0	+denorm	+Fin	+inf	NaN
-inf	T	F	F	F	F	F	F	F	T
-Fin	T	T/F	F	F	F	F	F	F	T
-denorm	T	T	T	T	T	T	F	F	T
-0	T	T	T	T	F	T	F	F	T
+0	T	T	T	T	T	T	F	F	T
+denorm	T	T	T	T	T	T	F	F	T
+Fin	T	T	T	T	T	T	T/F	F	T
+inf	T	T	T	T	T	T	T	T	T
NaN	F	F	F	F	F	F	F	F	F

Alternative Floating Point Mode

The key characteristics of the alternative floating point mode is that NaN, Inf, and denorm are not expected for an application to pass into the graphics pipeline, and the graphics hardware must not generate NaN, Inf, or denorm as a single precision result of a computation/conversion operation. For example, a result that is larger than the maximum representable floating point number is expected to be flushed to the largest representable floating point number, i.e., +fmax. The fmax has an exponent of 0xFE and a mantissa of all one's, which is the same for IEEE floating point mode.

Here is the complete list of the differences of legacy graphics mode from the relaxed IEEE-754 floating point mode.

- Any +/- INF result must be flushed to +/- fmax, instead of being output as +/- INF.
- Extended mathematics functions of log(), rsq(), and sqrt() take the absolute value of the sources before computation to avoid generating INF and NaN results.

Supported Legacy Float Mode and Impacted Units shows the support of these differences in various hardware units.

Supported Legacy Float Mode and Impacted Units

IEEE-754 Deviations	VF	Clipper	SF	WIZ	EU	EM	Sampler	RC
Any +/- INF result flushed to +/- fmax	Y	Y	Y	Y	Y	Y	Y	Y
Log, rsq, sqrt take abs() of sources	N/A	N/A	N/A	N/A	N/A	Y	N/A	N/A

Supported Legacy Float Mode and Impacted Units

IEEE-754 Deviations	VF	Clipper	SF	WIZ	EU	EM	RC
Any +/- INF result flushed to +/- fmax	Y	Y	Y	Y	Y	Y	Y
Log, rsq, sqrt take abs() of sources	N/A	N/A	N/A	N/A	N/A	Y	N/A

Dismissed legacy behaviors shows some of the desired or recommended alternative floating point mode behaviors that do not have hardware design impact. The reasons of not needing special hardware support for these items are also provided. This is based on the compliance requirement that can be found in the DirectX 9 specification: "**Handling of NaNs, Infs, and denorms is undefined. Applications should not pass in such values into the graphics pipeline.**"

Dismissed Legacy Behaviors

Suggested IEEE-754 Deviations	Reason for Dismiss
Mov forces (+/-)INF to (+/-)fmax	(+/-)INF is never present as input
(+/-)INF - (+/-)INF = +/- fmax instead of NaN	(+/-)INF is never present as input
Denorm must be flushed to zero in all cases (including trivial mov and point sampling)	Denorm is never present as input
Anything*0=0 (including NaN*0=0 and INF*0=0)	NaN and INF are never present as input
Except propagated NaN, NaN is never generated	NaN is never present as input and GFX never generates NaN based on rules in the previous table
An input NaN gets propagated excepting (a)-(d)	NaN is never present as input
(a) Rcp (and rsq) of 0 yields fmax	N/A, as it is already covered by the general rule "Any +/- INF result flushed to +/- fmax"
(b) Sampler honors 0/0 = 0 as if (1/0)*0	There is no divide in Sampler
I Rcp (and rsq) of INF yields +/- 0	(+/-)INF is never present as input
(d) Sampler honors INF/INF = 0 as if (1/INF)=0 followed by Anything*0 = 0	There is no divide in Sampler

Floating-Point Support

The following sections describe the floating-point support relative to the IEEE Standard for Floating-Point Arithmetic, currently IEEE Standard 754-2008. These sections cover binary floating-point arithmetic, as the EU provides no support for decimal floating-point arithmetic.

Note: Hardware alone is usually not fully conformant to the IEEE standard. It requires software functions to supplement the hardware.

Compared to previous generations, the following has been added:

- Adds the HF (Half Float) type and a corresponding HF execution data type and execution path.
- Adds flags to indicate IEEE floating-point exceptions and to enable or disable exception reporting to those flags.
- Adds the Single Precision Denorm Mode bit in Control Register 0. It can be enabled to allow calculations using the F (Float) type in IEEE floating-point mode to support denormals and gradual underflow.

Floating-Point Types and Values

The EU supports 16-bit (HF, Half Float), 32-bit (F, Float), and 64-bit (DF, Double Float) types in the IEEE Standard formats (respectively *binary16*, *binary32*, and *binary64* in IEEE 754-2008). See Floating-Point Numeric Data Types for the layout of the supported floating-point types.

Any bit pattern for a floating-point value corresponds to a value defined by the standard: +/- finite (normalized nonzero finite number), +/- 0 (signed zero), +/- inf (signed infinity), +/- denorm (denormalized, very small but nonzero number), or NaN (Not a Number). A NaN can be a signaling NaN (sNaN) or quiet NaN (qNaN).

These operating modes are available for the different floating-point types:

- Half Float uses the IEEE floating-point mode.
- Half Float support for denormals and gradual underflow is controlled by the Half Precision Denorm Mode bit.
- Float can use the ALT (Alternative Floating-Point Mode) or the IEEE floating-point mode. In IEEE mode, support for denormals and gradual underflow is controlled by the Single Precision Denorm Mode bit in the Control Register.
- Double Float uses the IEEE floating-point mode. Support for denormals and gradual underflow is controlled by the Double Precision Denorm Mode bit in the Control Register.

Flush to zero is not defined by IEEE Standard 754, but is implementation-specific and required by some APIs (including DirectX), thus the EU ISA supports either using or flushing Float or Double Float denorms based on the respective Denorm Mode bits.

Specifications in this volume sometimes reference +/-fmax, the largest finite magnitude representable in a format, and +/-fmin, the smallest normalized nonzero magnitude representable in a format. Calculating those values uses the extreme exponent values for finite nonzero floating-point values, Emax and Emin below, along with the number of explicit fraction bits (not counting the implicit bit in the significand). The following table provides these values, with the fmax and fmin values generally approximate.

Floating-Point Type Parameters

Type	Exp. Bits	Exp. Bias	Emax	Emin	Explicit Fraction Bits	fmax	fmin
HF	5	15	15	-14	10	65504.0	about 6.1E-5
F	8	127	127	-126	23	about 3.4E38	about 1.18E-38
DF	11	1023	1023	-1022	52	about 1.79E308	about 2.23E-308

Where f is the number of explicit fraction bits, the general formula for fmax is $(2.0 - 2^{-f}) * 2^{Emax}$.

The general formula for fmin is 2^{Emin}.

Not a Number (NaN) Formats

A NaN has a biased exponent field with all bits set (as if encoding an exponent of Emax + 1), a nonzero fraction field (as a zero fraction field with that exponent indicates infinity), and either sign bit.

As specified in IEEE Standard 754-2008, the MSB of the fraction field, what would be the first bit following the binary point in a numeric value, determines a NaN's type:

0 - Signaling NaN (sNaN). The remaining fraction bits cannot all be zero.

1 - Quiet NaN (qNaN). The remaining fraction bits can have any value.

When an sNaN is an input, an operation normally signals the Invalid Operation exception and *quietizes* the NaN, producing the equivalent qNaN value, with MSB set to 1, at the output. Raw moves do not check for NaNs and do not signal exceptions or quietize NaN values.

When QNAN as one of the input to an operation this results in QNAN without raising the exception flag. This silently propagates and the output is the same QNAN as in the input.

Intel specifies the value *qNaN Indefinite* as a quiet NaN with all zeros in the remaining fraction bits, those other than the MSB. This value is useful because it is never produced by quietizing an sNaN, thus qNaN Indefinite may be used to initialize floating-point values that are not otherwise initialized by software, allowing the uninitialized case to be distinguished.

The EU applies numeric source modifiers (**-**, **(abs)**, or **-(abs)**) to NaN source values as well as to other values, possibly changing the sign bit of a NaN value when it is propagated. NaN sign bits are normally don't care values.

Per IEEE Standard 754-2008, a NaN's *payload* is contained in all fraction field bits other than the fraction MSB. Thus in the overall floating-point format, the sign bit, biased exponent, and fraction MSB are not part of the payload. NaN payload values are not affected by quietizing or by source modifiers. As noted above, an sNaN must have a nonzero payload and a qNaN can have any payload.

Floating-Point Rounding Modes

The EU supports the four rounding modes required in IEEE Standard 754 for binary floating-point arithmetic. If the unrounded result of infinite precision and range is exactly representable in the destination format, then that exact result is produced and no exception is signaled. For an unrounded

result of infinite precision and range that is not exactly representable in the destination format (an *inexact* result), rounding chooses a numerically adjacent value in the destination format, while signaling the Inexact, Overflow, or Underflow exceptions when appropriate. The four rounding modes are:

RNE - Round to nearest or even. Choose the value in the destination precision nearest to the unrounded result. If the unrounded result is midway between two such values, choose the value with its least significant fraction bit as 0 (*even*).

RD - Round down, toward minus infinity.

RU - Round up, toward plus infinity.

RZ - Round toward zero.

The rounding mode is specified by the Rounding Mode field in the Control Register. It is initialized by Thread Dispatch. The normal default value is round to nearest or even. The Rounding Mode can be read to check the mode and written to change it. The Control Register and the Rounding Mode value are thread-specific; the Rounding Mode applies to all floating-point types, all execution channels, and all floating-point instructions executed by the thread after it is assigned.

Rounding an inexact result signals the Inexact Exception.

Rounding an inexact result preserves the sign of the result.

Infinities and NaNs are exact results and are not affected by the rounding mode.

Zeros are exact results, but the signs of zero results are affected by the rounding mode in certain cases:

X - X = +0 for RNE, RU, RZ

X - X = -0 for RD

(+0) + (-0) = (-0) + (+0) = +0 for RNE, RU, RZ

(+0) + (-0) = (-0) + (+0) = -0 for RD

Regardless of the rounding mode, (+0) + (+0) = +0 and (-0) + (-0) = -0.

The *directed rounding modes* are round down, round up, and round toward zero.

In IEEE mode, when a floating-point overflow occurs the result is determined by the sign of the result and the rounding mode:

+ and (RNE or RU): + inf

+ and (RD or RZ): + fmax

- and (RU or RZ): - fmax

- and (RNE or RD): - inf

Note that for floating-point overflow in IEEE mode, RNE always produces +/- inf and RZ always produces +/- fmax.

Floating-Point Operations and Precision

IEEE Standard 754-2008 requires the following floating-point operations to be precise within <= 0.5 *ulp* (unit in the last place) when using the round to nearest or even rounding mode:

ADD (*add* instruction)

DIV (*math* instruction with FDIV function code)

FMA (fused multiply add, *mad* instruction)

MUL (*mul* instruction)

SQRT (*math* instruction with SQRT function code)

SUB (*add* instruction using one **-** source modifier)

Conversions (float to float, float to int, and int to float)

Min/Max

Compare

Single Precision Floating-Point Rounding to Integral Values

The **rndd** (Round Down), **rnde** (Round to Nearest or Even), **rndu** (Round Up), and the **rndz** (Round to Zero) instructions round arbitary Float values to integral Float values. Each instruction specifies its rounding mode so these instructions are not affected by the Rounding Mode in the Control Register.

An integral source value produces the same value for the destination (ignoring any saturation). For magnitudes >= 8,388,608 (2^{23}) all Float values are integral.

The rounding instructions are sign preserving.

Signed zeros are propagated. In IEEE mode, signed infinities are propagated. In IEEE mode, sNaN inputs are quietized, the equivalent qNaN is produced, and the Invalid Operation exception is indicated. In IEEE mode, qNaN inputs are propagated.

No other exceptions are signaled for these instructions. For example if the source and result values differ, the Inexact exception is not signaled.

The Single Precision Denorm Mode in the Control Register affects the results of the **rndd** and **rndu** instructions for denorm source values.

Floating-Point to Integer Conversion

The *mov* and *sel* instructions can be used to convert floating-point values to integers. In the tables below, Imin is the smallest representable value in a signed integer type, Imax is the largest representable value in an integer type, and f is a finite floating-point value after rounding to an integral value using the current rounding mode.

Converting unrepresentable floating-point values, including infinities, NaNs, and values that convert to integers outside of the destination type's range, signal Invalid Operation exceptions. When the

destination integer type is unsigned, normalized nonzero negative inputs signal Invalid Operation exceptions and negative denorm inputs signal Inexact exceptions.

Data written in accumulator using implicit or explicit destination will not be IEEE compliant.

Floating-Point to Integer Conversion Results and Exceptions for Signed Integer Types

FP Value	Integer Result	FP Exception
qNaN	0	Invalid Operation
sNaN	0	Invalid Operation
+inf	Imax	Invalid Operation
f > Imax	Imax	Invalid Operation
Imin <= f <= Imax	f	Inexact if rounding changed f
f < Imin	Imin	Invalid Operation
-inf	Imin	Invalid Operation

Floating-Point to Integer Conversion Results and Exceptions for Unsigned Integer Types

FP Value	Integer Result	FP Exception
qNaN	0	Invalid Operation
sNaN	0	Invalid Operation
+inf	Imax	Invalid Operation
f > Imax	Imax	Invalid Operation
0 <= f <= Imax	f	Inexact if rounding changed f
f = -0	0	Inexact
-1 < f < -0	0	Inexact
-1 < f < -0	Real Indefinite	Invalid Operation
-fmax <= f <= -1	Real Indefinite	Invalid Operation
-inf	Real Indefinite	Invalid Operation

Note: Real Indefinite is encoded as Integer value 0.

Integer to Floating-Point Conversion

Integer to floating-point conversion follows these rules in IEEE mode:

- The result is never +/- inf, never NaN, and never -0.
- If the integer source value is not exactly representable in the destination floating-point format, use the current rounding mode to choose an adjacent floating-point value and signal the Inexact Exception.
- If the integer source value is too large to represent in the destination floating-point format (only possible when converting to Half Float from D, UD, or UW) then signal the Overflow Exception. Based on the sign and the current rounding mode, the result is +/- fmax or +/- inf, as described in the Overflow Exception section.

Floating-Point Min/Max Operations

In the following Min/Max operations, sNaN inputs are preferred to non-NaN inputs and non-NaN inputs are preferred to qNaN inputs.

Min(x, qNaN) = Min(qNaN, x) = x with no exceptions signaled.

Min(x, sNaN) = Min(sNaN, x) = qNaN (quietized value corresponding to the input sNaN) and signal the Invalid Operation exception.

Note: DX deviates from this rule:

The DX behavior is inferred from the denorm bit.

Min(x, sNaN) = Min(sNaN, x) = x

Max(x, qNaN) = Max(qNaN, x) = x with no exceptions signaled.

Max(x, sNaN) = Max(sNaN, x) = qNaN (quietized value corresponding to the input sNaN) and signal the Invalid Operation exception.

Note: DX deviates from this rule:

The DX behavior is inferred from the denorm bit.

Max(x, sNaN) = Max(sNaN, x) = x

Special cases(when both sources are NaN inputs)

Min(qNaN, qNan) = qNaN (of the first source) and no exception raised

Min(qNaN, sNaN) = Min(sNaN, qNaN) = qNaN (quiet-ized sNaN and signal the Invalid Operation exception)

Min(sNaN, sNaN) = qNaN(of the first source and signal Invalid Operation Exception)

Max(qNaN, qNan) = qNaN (of the first source) and no exception raised

Max(qNaN, sNaN) = Max(sNaN, qNaN) = qNaN (quiet-ized sNaN and signal the Invalid Operation exception)

Max(sNaN, sNaN) = qNaN(of the first source and signal Invalid Operation Exception)

IEEE Floating-Point Exceptions

The EU detects the five floating-point exceptions defined by IEEE Standard 754:

- Invalid Operation
- Division by Zero
- Overflow
- Underflow
- Inexact

Signaling Floating-Point Exceptions

When enabled in the Control Register, floating-point exceptions are detected and set *sticky* flag bits in the State Register. There is no mechanism for automatic transfer to a handler, so floating-point exceptions are not handled like other exceptions described in the Exceptions chapter.

Setting flags to indicating Floating-Point exceptions is the default exception handling approach specified by IEEE Standard 754. The flag bits are sticky because in normal operation the EU only sets these bits as exceptions occur, and does not clear these bits, so a set value sticks until cleared by software. The Control Register and State Register are cleared at reset and initialized at thread load. Both are read/write registers. These fields are used:

- **IEEE Exception Trap Enable** (Control Register cr0.0:ud bit 9). This bit enables trapping IEEE exception flags. This control bit may be updated by software. It is initially zero on thread load. If enabled, IEEE floating-point exceptions set sticky bits in the IEEE Exception field of sr0.1, in the State Register. **Note:** Software must set this flag at thread start to use the IEEE Exception flags.
- **IEEE Exception**. (State Register sr0.1:ud bits 7:0). The IEEE exception bits are sticky bits set by the opcodes when floating-point exceptions are triggered. These bits are defined per thread and all channels update one set of sticky bits. These bits are cleared on thread load and may be cleared by software. Exception updates to these bits may be disabled by clearing the IEEE Exception Trap Enable bit in the Control Register. The following table specifies the IEEE exception bits:

Bits	Definition
7:5	Reserved
4	Inexact Exception
3	Overflow Exception
2	Underflow Exception
1	Division by Zero Exception
0	Invalid Operation Exception

The IEEE exception flags are per thread, shared by all channels. (Maintaining separate per channel exception flags for 32 channels would require 160 bits per thread.). Operations involving non IEEE float formats (e.g. bfloat16, bfloat8, tfloat32 etc) may not trigger these exceptions.

Invalid Operation Exception

An Invalid Operation exception is signaled by any operation on a signaling NaN (sNaN) or by certain combinations of operations and operands with undefined results, always producing a quiet NaN (qNaN) result.

The following specific operations signal Invalid Operation, where *x* is a positive, finite, nonzero, and normalized number:

+inf - (+inf) or (-inf) - (-inf)

+/- 0 / +/- 0

+/- inf / +/- inf

+/- 0 x +/- inf or +/- inf x +/- 0

Remainder(+/- *x*, +/- 0)

Remainder(+/- inf, +/- *x*)

Sqrt(-*x*)

Note that Sqrt(-0) is -0 per IEEE Standard 754 and does not cause any exception.

These instructions can signal specific Invalid Operation exceptions (and also on sNaN inputs except for Float inputs in ALT mode), producing a qNaN result:

add

dp2, *dp3*, *dp4*, and *dph* (the Dot Product instructions)

line

lrp

mac

mad

madm

math with the FDIV, SQRT, or RSQRTM function codes

mul

pln

These other instructions signal Invalid Operation exceptions on sNaN inputs (except for Float inputs in ALT mode), producing a qNaN result:

cmp, *cmpn*

frc

math with all other function codes for floating-point operations

mov except for raw moves

rndd, *rnde*, *rndu*, and *rndz* (the Round instructions)

sel

smov except for raw moves

Division by Zero Exception

The operation +/- *x* / +/- 0, where *x* is a positive, finite, nonzero, and normalized number, signals the Division by Zero Exception and produces a correctly signed +/- inf result. Note that in accordance with the standard, +/- 0 / +/- 0 signals Invalid Operation, does not signal Division by Zero, and produces a qNaN result. This behavior can occur for the *math* instruction with the FDIV function code.

The operation $LOG_2(+/- 0)$ signals the Division by Zero Exception and produces -inf as the result. This behavior can occur for the *math* instruction with the LOG function code.

Overflow Exception

A floating-point *overflow* occurs when an operation with a finite result produces an internal result with magnitude > fmax, the maximum representable finite value in the destination format. The internal result is rounded to the destination precision with the current rounding mode but has an unbounded exponent. An overflow produces +/- inf or +/- fmax as the result, depending on the sign and the rounding mode.

The following algorithm describes floating-point overflow processing:

1. Compute the result with infinite precision and unbounded range.
2. Normalize the result using an unbounded exponent.
3. Round the result to the destination precision using an unbounded exponent and the current rounding mode.
4. If (abs(rounded unbounded result) > fmax(destination format)) {

    Set the Overflow Exception flag to 1.

    Output = +/- inf or +/- fmax depending on the sign and the rounding mode:

        + and (RNE or RU): + inf

        + and (RD or RZ): + fmax

        - and (RU or RZ): - fmax

        - and (RNE or RD): - inf

    }

}

Note: The first three steps of the overflow and underflow algorithms are identical.

Underflow Exception

An *underflow* occurs when an operation produces a tiny but nonzero inexact result *x*, with abs(*x*) < fmin, where fmin = 2^{Emin}. See the Floating-Point Type Parameters table for the fmin and Emin values for different floating-point types.

IEEE Standard 754 allows underflow to be determined before or after rounding. The Execution Unit determines underflow after rounding, which is consistent with the behavior of the x87 and SSE instructions in the CPU.

When denorms are flushed to zero, no underflow exceptions are signaled. Flush to zero is not defined by IEEE Standard 754, but is implementation specific and required by some APIs (including DirectX), thus the EU ISA supports either using or flushing Float or Double Float denorms based on the respective Denorm Mode bits.

When denorms are enabled, if an operation's internal result rounded to the destination precision, but using an unbounded exponent range, has a magnitude that is less than fmin but nonzero AND the denorm result in the destination format is inexact, then signal the Underflow Exception.

The rounded result can be +/- 0, +/- fmin, or (when denorms are enabled) +/- denorm.

IEEE Standard 754 requires the non-intuitive behavior that an exact denorm result does not set the Underflow Exception flag.

The following algorithm describes floating-point underflow processing:

1. Compute the result with infinite precision and unbounded range.
2. Normalize the result using an unbounded exponent.
3. Round the result to the destination precision using an unbounded exponent and the current rounding mode.
4. If (0 < abs(rounded unbounded result) < fmin(destination format) {

    if (flush denorms to zero) {

        Output = 0; // No underflow exception.

    }

    Else {

        Renormalize to the bounded exponent with the original infinite precision value...

            ...and round that value to the destination precision using the current rounding mode.

        If (the just computed value differs from the value computed in step (3) in exponent or mantissa) {

            Set the Underflow Exception flag to 1.

        } // Note: Underflow is not set if the tiny result is the same as when computed with an unbounded exponent.

        Output = rounded result using the destination precision and destination exponent range;

    }

}

Else { // Not a tiny number.

    Output = rounded number;

}

Inexact Exception

An Inexact Exception occurs when the internal unrounded result, with infinite precision and unbounded exponent range, differs from the generated result after format conversion, normalizing or denormalizing, and rounding. An Inexact Exception occurs irrespective of any saturation to exact zero or exact +1.0. The

Inexact Exception is normal and may occur more often than not. For example, the calculation 1.0 / 3.0 is inexact in any binary floating-point format. These rules determine whether a result is inexact:

- Infinities and NaNs are never inexact.
- Flushing a denorm internal result to zero (always for Half Float and if the appropriate Denorm Mode is 0 for Float and Double Float) is always inexact.
- If any rounding occurs using the current Rounding Mode, so the rounded result differs from the internal unrounded result, the result is inexact. However explicit round to integral using any of the rounding instructions (*rndd*, *rnde*, *rndu*, and *rndz*) is never inexact.

Floating-Point Compare Operations

Four mutually exclusive relations are possible between two floating-point values, src0 and src1:

Less than. src0 < src1 and neither source is NaN.

Equal. src0 = src1 and neither source is NaN.

Greater than. src0 > src1 and neither source is NaN.

Unordered. Any source is NaN.

Any NaN compares unordered to any value, including itself.

Infinities of the same sign compare as equal.

Zeros compare as equal regardless of sign: -0 = +0.

Floating-Point Compare Relations

src1 / src0	-inf	-fin	-denorm	-0	+0	+denorm	+fin	+inf	NaN
-inf	E	L	L	L	L	L	L	L	U
-fin	G	*	L	L	L	L	L	L	U
-denorm	G	G	*^	L^	L^	L^	L	L	U
-0	G	G	G^	E	E	L^	L	L	U
+0	G	G	G^	E	E	L^	L	L	U
+denorm	G	G	G^	G^	G^	*^	L	L	U
+fin	G	G	G	G	G	G	*	L	U
+inf	G	G	G	G	G	G	G	E	U
NaN	U	U	U	U	U	U	U	U	U

Notes	
*	Relation can be L, E, or G.
^	When denorms are flushed to zero then all denorms and zeros compare as E.

The next six tables show the results of six specific comparisons, corresponding to the **.g**, **.l**, **.e**, **.ne**, **.ge**, and **.le** conditional modifiers used with the *cmp* instruction and a floating-point source type. Any NaN source produces a false comparison result for these modifiers other than **.ne** and produces a true comparison result for the **.ne** modifier.

Results of Greater Than Comparison -- *cmp*.g

src1 / src0	-inf	-fin	-denorm	-0	+0	+denorm	+fin	+inf	NaN
-inf	F	F	F	F	F	F	F	F	F
-fin	T	*	F	F	F	F	F	F	F
-denorm	T	T	*^	F^	F^	F^	F	F	F
-0	T	T	T^	F	F	F^	F	F	F
+0	T	T	T^	F	F	F^	F	F	F
+denorm	T	T	T^	T^	T^	*^	F	F	F
+fin	T	T	T	T	T	T	*	F	F
+inf	T	T	T	T	T	T	T	F	F
NaN	F	F	F	F	F	F	F	F	F

Notes	
*	Result can be T or F.
^	When denorms are flushed to zero then all denorms and zeros compare as equal, making the **.g** comparison result F.

Results of Less Than Comparison -- *cmp*.l

src1 / src0	-inf	-fin	-denorm	-0	+0	+denorm	+fin	+inf	NaN
-inf	F	T	T	T	T	T	T	T	F
-fin	F	*	T	T	T	T	T	T	F
-denorm	F	F	*^	T^	T^	T^	T	T	F
-0	F	F	F^	F	F	T^	T	T	F
+0	F	F	F^	F	F	T^	T	T	F
+denorm	F	F	F^	F^	F^	*^	T	T	F
+fin	F	F	F	F	F	F	*	T	F
+inf	F	F	F	F	F	F	F	F	F
NaN	F	F	F	F	F	F	F	F	F

Notes	
*	Result can be T or F.

src1 src0	-inf	-fin	-denorm	-0	+0	+denorm	+fin	+inf	NaN
-inf	F	T	T	T	T	T	T	T	F
-fin	F	*	T	T	T	T	T	T	F
-denorm	F	F	*^	T^	T^	T^	T	T	F
-0	F	F	F^	F	F	T^	T	T	F
+0	F	F	F^	F	F	T^	T	T	F
+denorm	F	F	F^	F^	F^	*^	T	T	F
+fin	F	F	F	F	F	F	*	T	F
+inf	F	F	F	F	F	F	F	F	F
NaN	F	F	F	F	F	F	F	F	F
Notes									
^	When denorms are flushed to zero then all denorms and zeros compare as equal, making the **.l** comparison result F.								

Results of Equal Comparison -- *cmp*.e

src1 src0	-inf	-fin	-denorm	-0	+0	+denorm	+fin	+inf	NaN
-inf	T	F	F	F	F	F	F	F	F
-fin	F	*	F	F	F	F	F	F	F
-denorm	F	F	*^	F^	F^	F^	F	F	F
-0	F	F	F^	T	T	F^	F	F	F
+0	F	F	F^	T	T	F^	F	F	F
+denorm	F	F	F^	F^	F^	*^	F	F	F
+fin	F	F	F	F	F	F	*	F	F
+inf	F	F	F	F	F	F	F	T	F
NaN	F	F	F	F	F	F	F	F	F
Notes									
*	Result can be T or F.								
^	When denorms are flushed to zero then all denorms and zeros compare as equal, making the **.e** comparison result T.								

Results of Not Equal Comparison -- *cmp*.ne

src1 src0	-inf	-fin	-denorm	-0	+0	+denorm	+fin	+inf	NaN
-inf	F	T	T	T	T	T	T	T	T
-fin	T	*	T	T	T	T	T	T	T
-denorm	T	T	*^	T^	T^	T^	T	T	T
-0	T	T	T^	F	F	T^	T	T	T
+0	T	T	T^	F	F	T^	T	T	T
+denorm	T	T	T^	T^	T^	*^	T	T	T
+fin	T	T	T	T	T	T	*	T	T
+inf	T	T	T	T	T	T	T	F	T
NaN	T	T	T	T	T	T	T	T	T

Notes	
*	Result can be T or F.
^	When denorms are flushed to zero then all denorms and zeros compare as equal, making the **.ne** comparison result F.

Results of Greater Than or Equal Comparison -- *cmp*.ge

src1 src0	-inf	-fin	-denorm	-0	+0	+denorm	+fin	+inf	NaN
-inf	T	F	F	F	F	F	F	F	F
-fin	T	*	F	F	F	F	F	F	F
-denorm	T	T	*^	F^	F^	F^	F	F	F
-0	T	T	T^	T	T	F^	F	F	F
+0	T	T	T^	T	T	F^	F	F	F
+denorm	T	T	T^	T^	T^	*^	F	F	F
+fin	T	T	T	T	T	T	*	F	F
+inf	T	T	T	T	T	T	T	T	F
NaN	F	F	F	F	F	F	F	F	F

Notes	
*	Result can be T or F.
^	When denorms are flushed to zero then all denorms and zeros compare as equal, making the **.ge** comparison result T.

Results of Less Than or Equal Comparison -- *cmp*.le

src1 / src0	-inf	-fin	-denorm	-0	+0	+denorm	+fin	+inf	NaN
-inf	T	T	T	T	T	T	T	T	F
-fin	F	*	T	T	T	T	T	T	F
-denorm	F	F	*^	T^	T^	T^	T	T	F
-0	F	F	F^	T	T	T^	T	T	F
+0	F	F	F^	T	T	T^	T	T	F
+denorm	F	F	F^	F^	F^	*^	T	T	F
+fin	F	F	F	F	F	F	*	T	F
+inf	F	F	F	F	F	F	F	T	F
NaN	F	F	F	F	F	F	F	F	F

Notes	
*	Result can be T or F.
^	When denorms are flushed to zero then all denorms and zeros compare as equal, making the **.le** comparison result T.

Type Conversion

Float to Integer

Converting from float to integer is based on rounding toward zero(RTZ is for DX, IEEE expects all four rounding modes). If the floating point value is +0, -0, +Denorm, -Denorm, +NaN -r -NaN, the resulting integer value is always 0. If the floating point value is positive infinity (or negative infinity), the conversion result takes the largest (or the smallest) represent-able integer value. If the floating point value is larger (or smaller) than the largest (or the smallest) represent-able integer value, the conversion result takes the largest (or the smallest) represent-able integer value. The following table shows these special cases. The last two rows are just examples. They can be any number outside the represent-able range of the output integer type (UQ, Q, UD, D, UW, W, UB and B).

Input Format			Output Format					
F	UQ	Q	UD	D	UW	W	UB	B
+/- Zero	0000000000000000	0000000000000000	00000000	00000000	00000000	00000000	00000000	00000000
+/- Denorm	0000000000000000	0000000000000000	00000000	00000000	00000000	00000000	00000000	00000000
NAN	0000000000000000	0000000000000000	00000000	00000000	00000000	00000000	00000000	00000000
-NAN	0000000000000000	0000000000000000	00000000	00000000	00000000	00000000	00000000	00000000
INF	FFFFFFFFFFFFFFFF	7FFFFFFFFFFFFFFF	FFFFFFFF	7FFFFFFF	0000FFFF	00007FFF	000000FF	0000007F
-INF	0000000000000000	8000000000000000	00000000	80000000	00000000	00008000	00000000	00000080
$+2^{32}$ (*)	0000000100000000	0000000100000000	FFFFFFFF	7FFFFFFF	0000FFFF	00007FFF	000000FF	0000007F

Input Format			Output Format					
F	UQ	Q	UD	D	UW	W	UB	B
$-2^{32}-1$ (*)	0000000000000000	FFFFFFFEFFFFFFFF	00000000	80000000	00000000	00008000	00000000	00000080
$+2^{64}$ (*)	FFFFFFFFFFFFFFFF	7FFFFFFFFFFFFFFF	FFFFFFFF	7FFFFFFF	0000FFFF	00007FFF	000000FF	0000007F
$-2^{64}-1$(*)	0000000000000000	8000000000000000	00000000	80000000	00000000	00008000	00000000	00000080

Integer to Integer with Same or Higher Precision

Converting an unsigned integer to a signed or an unsigned integer with higher precision is based on zero extension.

Converting an unsigned integer to a signed integer with the same precision is based on modular wrap-around. Without saturation, a larger than represent-able number becomes a negative number. With saturation, a larger than represent-able number is saturated to the largest positive represent-able number.

Converting a signed integer to a signed integer with higher precision is based on sign extension.

Converting a signed integer to an unsigned integer with higher precision is based on sign extension. Without saturation, a negative number becomes a large positive number with the sign bit wrapped-up. With saturation, a negative number is saturated to zero.

Integer to Integer with Lower Precision

Converting a signed or an unsigned integer to a signed or an unsigned integer with lower precision is based on bit truncation. Without saturation, only the lower bits are kept in the output regardless of the sign-ness of input and output. With saturation, a number that is outside the represent-able range is saturated to the closest represent-able value.

Integer to Float

Converting a signed or an unsigned integer to a single precision float number is to round to the closest representable float number. For any integer number with magnitude less than or equal to 24 bits, resulting float number is a precise representation of the input. However, if it is more than 24 bits, by default a "round to nearest even" is performed.

Double Precision Float to Single Precision Float

Double Precision Float	Single Precision Float
-inf	-inf
-finite	-finite/-denorm/-0
-denorm	-0

Double Precision Float	Single Precision Float
-0	-0
+0	+0
+denorm	+0
+finite	+finite/+denorm/+0
+inf	+inf
NaN	NaN

The upper Dword of every Qword will be written with undefined value when converting DF to F.

Single Precision Float to Double Precision Float

Converting a single precision floating-point number to a double precision floating-point number will produce a precise representation of the input.

Single Precision Float	Double Precision Float
-inf	-inf
-finite	-finite
-denorm	-finite
-0	-0
+0	+0
+denorm	+finite
+finite	+finite
+inf	+inf
NaN	NaN

Exceptions

The Architecture defines a basic exception handling mechanism for several exception cases. This mechanism supports both normal operations such as extensions of the mask-stack depth, as well as detecting some illegal conditions.

Exception Types

Type	Trigger / Source	Sync/Async Recognition
Software Exception	Thread code	Synchronous
Breakpoint	• A bit in the instruction word	Synchronous

Type	Trigger / Source	Sync/Async Recognition
	• Breakpoint IP match • Breakpoint Opcode match	
Illegal Opcode	Hardware	Synchronous
Halt	MMIO register write	Asynchronous
Context Save/Restore	Preemption Interrupt	Asynchronous

Threads may choose which exceptions to recognize and which to ignore. This mask information is specified on a per-kernel basis in fixed function state generated by the driver, and delivered to the EU as part of a new thread dispatch. Upon arrival at the EU, the exception-mask information is used to initialize the exception enable fields of that thread's cr0.1 register, which controls exception recognition. This register is instantiated on a per-thread basis, allowing independent control of exception type recognition across hardware threads. The exception enable bits in the cr0.1 register are read/write, and thus can be enabled/disabled via software at any time during thread execution.

The exception handling mechanism relies on the System Routine, a single subroutine that provides common exception handling for all threads on all EUs in the system. This System Routine is defined per-context and is identified via a System IP (SIP) register in context state. At the time of each context switch, the appropriate SIP for that context is loaded into each EU, allowing each context to have custom implementation of exception handling routines if so desired.

The mechanism does not support handling recursive system routine access. This means when a thread cannot be asynchronously interrupted to an exception when executing a SIP.

Example:

An Exception is not supported when hardware is executing a SIP for context save and restore operations.

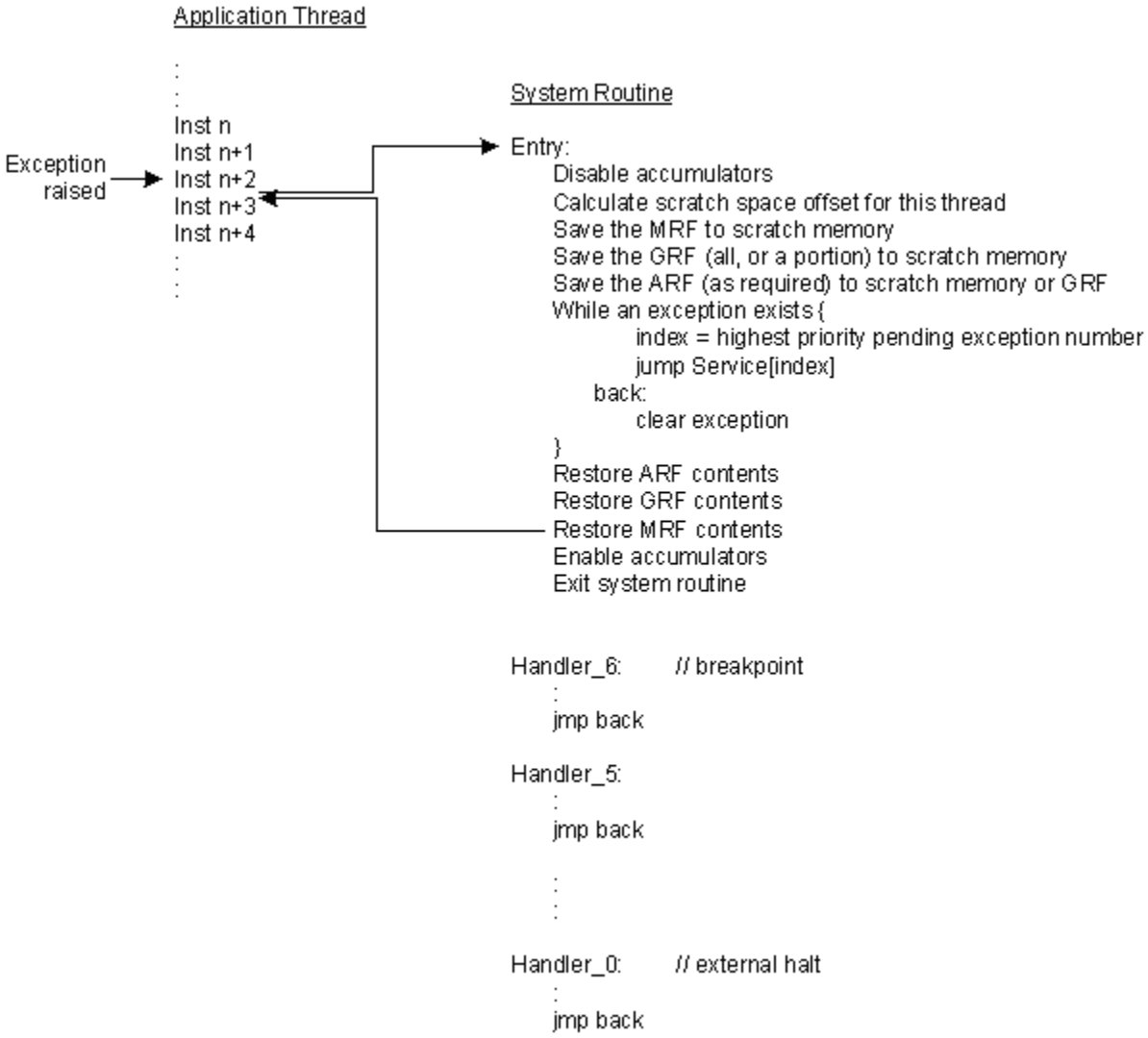
Exception-Related Architecture Registers

Exception-related registers are architecture registers cr0.0 through cr0.2. These registers are instantiated on a per-thread basis providing each hardware thread with unique control over exception recognition and handling. The registers provide the capability to mask exception types, determine the type of raised exception, store the return address, and control exiting from the System Routine back to the application thread.

Many of the bits in these registers are manipulated by both hardware and software. In all cases, the read/write operations by hardware and software occur at exclusive times in a thread's lifetime, thus there is no need for atomic read-modify-write operations when accessing these registers.

System Routine

The following diagram illustrates the basic flow of exception handling and the structure of the System Routine.



Invoking the System Routine

The System Routine is invoked in response to a raised exception. Once an exception is raised, no further instructions from the application thread are issued until the System Routine has executed and returned control back to the application thread.

After an exception is recognized by hardware, the EU saves the thread's IP into its AIP register (cr0.2), and then moves the System Routine offset, SIP, into the thread's IP register. At this point the next instruction issued for that thread is the first instruction of the System Routine.

The System Routine maintains the same execution priority, GRF register space, and thread state as the application thread from which it is invoked.

Due to assuming the same priority, there may be significant absolute time between an exception being raised and invoking the System Routine, as other higher priority threads within the EU continue to execute. From a thread's perspective, once an exception is recognized, the next instruction issued is from the System Routine.

At the time of System Routine invocation, there may still be outstanding registers in-flight from the application thread. Depending on the instruction sequence in the System Routine, an in-flight register may be referenced by the System Routine and cause a register-in-flight dependency. These dependencies are honored by the System Routine and may cause the System Routine to be suspended until the register retires.

Exception processing is not nested within the System Routine. If a future exception is detected while executing the System Routine, the exception is latched into cr0.1, but does not cause a nested re-invocation of the System Routine. The exception recognition hardware recognizes only one outstanding exception of each type; i.e., once a specific exception type is detected and latched in cr0.1, and until the exception is cleared, any further exception of that type is lost.

Accumulators are not natively preserved across the System Routine. To make sure the accumulators are in the identical state once control is returned to the application thread, the System Routine must either set the Accumulator Disable bit of cr0.0 before using any instruction that modifies an accumulator, or save and restore the accumulators (using GRF registers or system thread scratch memory) around the System Routine. Saving and restoring accumulators, including their extended precision bits, can be accomplished by a short series of moves and shifts of the accumulator register. Also note that the state of the Accumulator Disable bit itself must be preserved unless, by convention, the driver software limits its manipulation to only the System Routine.

Further, upon System Routine entry, the execution-related masks (Continue, Loop, If, and Active masks, contained in the Mask Register) will remain set as they were in the application thread. Thus only a subset of channels may be active for execution. To enable execution on all channels, the System Routine may choose to use the instruction option 'NoMask', or may choose to set the mask registers to the desired value so long as it saves/restores the original masks upon System Routine entry/exit.

Similarly, there is no hardware mechanism to preserve flags, mask-stacks, or other architecture registers across the System Routine. The System Routine must ensure that these values are preserved (see the Conditional Instructions Within the System Routine section for related information).

Returning to the Application Thread

Returning to the Application Thread

Prior to returning control to the application thread, the System Routine should clear the proper Exception Status and Control bit in cr0.1. Failure to do so forces the thread's execution to reenter the System Routine before any further instructions are executed from the application thread. (Note that single-stepping functionality is the one exception where the exception's Status and Control bit is not reset before exit.)

The System Routine may choose to loop within a single invocation of the System Routine until all pending exceptions are serviced, or may choose to service exceptions one at a time (a simpler solution, but less efficient).

The System Routine is exited, and control returned to the application thread, via a write to the Primary Exception State and Control bit in cr0.0. Upon clearing this bit, the value of AIP (cr0.2) is restored to the thread's IP register and, with no further exceptions pending, execution resumes at that address. The

System Routine must follow any write to the Primary Exception State and Control bit with at least one SIMD-16 *nop* instruction to allow control to transition. Throughout the System Routine, the AIP register maintains its value at the time the exception was raised unless directly modified by the System Routine. (See the AIP register definition for specifics on the IP value saved to AIP).

System IP (SIP)

The System IP (SIP) is the 16 byte-aligned offset of the first instruction of the System Routine, relative to the General State Base Address. SIP is assigned by the STATE_SIP command to the command streamer which updates SIP in the EU.

The SIP is widened to 48 bits. However, the EU still only uses the low 32 bits (bits 31:4 with bits 3:0 as zero).

When the System Routine is invoked, the application thread's current IP is first saved into the AIP field of cr0.2. The SIP address is then loaded into the thread's IP register and execution continues within the System Routine. Thus, each invocation of the System Routine has a common entry point. Returning from the System Routine loads IP from AIP, continuing thread execution.

System Routine Register Space

The System Routine uses the same GRF space as the thread that invokes it.

As such all of the calling thread's registers and their contents are visible to the System Routine. Further, the System Routine must only use r0..r15 of the GRF, as a minimal thread may have requested and been allocated this few. If the System Routine requires more registers than this, the driver should establish a higher minimum allocation for all threads.

The System Routine may encounter any residual register dependencies of the calling thread until such time that they clear by the return of in-flight writebacks.

Only one 32-bit GRF location, r0.4, is reserved for System Routine use. This location is sufficient to allow the System Routine to calculate the appropriate offset of its private scratch memory in the larger system scratch memory space (as dictated by binding table entry 254). The offset is left as a driver convention, but is likely based on a combination of Thread and EU IDs (see the example system handler in the System Scratch Memory Space) section. Other than the reserved r0.4 register field, there is no explicit GRF register space dedicated to the System Routine, and any GRF needs must be accomplished via (a) convention between the System Routine and application code, or (b) the System Routine temporarily spilling the thread's GRF register contents to scratch memory and restoring those contents before System Routine exit.

No persistent storage is automatically allocated to the System Routine, although a driver implementation may set aside part of system scratch memory for the System Routine.

Any parameter passing to the System Routine (for use by software exceptions) is done via the GRF based on system thread/application thread convention.

System Scratch Memory Space

There is a single unified system scratch memory space per context shared by all EUs. It is anticipated that block is further partitioned into a unique scratch sub-space per thread via conventions implemented in the System Routine, with each hardware thread having a uniform block size at a calculated offset from the base address. The block address for a thread can be based on an offset derived from the thread's execution unit ID and thread ID made available through the TID and EUID field of architecture register sr0.0.

```
Per_Thread_Block_Size = System_Scratch_Block_Size / (EU_Count * Thread_Per_EU);
Offset = (sr0.0.EID * Threads_Per_EU + sr0.0.TID) * Per_Thread_Block_Size;
```

where in GFX:

Threads_Per_EU = 8

EU_Count = 8

System_Scratch_Block_Size is a driver choice.

Access to system scratch memory is performed through the Data Port via linear single register or block-based read/write messages. The driver may choose to use any binding table index for system scratch surface description. As a practical matter, the same index is expected to be used across all binding tables, as the index is typically hard coded in Data Port messages used within the System Routine coupled with the fact that a single System Routine is used for all threads. Read/write messages to the Data Port contain the address of the binding table (provided in r0 of all threads) and an offset, from which the Data Port calculates the final target address.

It is expected that the system scratch memory space is allocated by the driver at context-create time and remains persistent at a constant memory address throughout a context's lifetime.

Conditional Instructions Within the System Routine

It is expected that most, if not all, control flow within the System Routine is scalar in nature. If so, the System Routine should set SPF (Single Program Flow, cr0.0) to enable scalar branching. In this mode, conditional/loop instructions do not update the mask stacks and therefore do not have restrictions on their use nor require the save/restore of hardware mask stack registers.

If SIMD branching is desired within the System Routine, special considerations must be taken. Upon entry to the System Routine, the depth of the mask stacks is unknown at that point, and may be near full. If so, a subsequent conditional instruction and its associated mask 'push' may cause a stack overflow. This would generate an exception within the system routine, an unsupported occurrence. To prevent this, if the System Routine uses SIMD conditional instructions, it must save the mask stacks prior to the first SIMD conditional instruction, and restore them after the last SIMD conditional instruction. As a general solution, it may be easiest to simply implement the save/restore as part of the entry/exit code sequence, using an available GRF register pair as a storage location. Once saved, the stacks should be reset to their empty condition, namely depth = 0 and top of stack = 0xFFFFFFFF.

Exception Descriptions

This section describes conditions that can cause exceptions and transfer control to the System Routine.

Illegal Opcode

The ISA defines a single *illegal* opcode. The byte value of the *illegal* opcode is 0x00 due to it being a likely byte value encountered by a wayward instruction pointer value. The *illegal* instruction signals an exception if exception handling is enabled and invokes the system interrupt routine. If exception handling is NOT enabled, the illegal opcode is executed resulting in undetermined behavior including a system hang. Hardware decodes all legal opcodes supported. Any byte value that is not in the legal opcode list is decoded as an illegal opcode to trigger exception.

Undefined Opcodes

All undefined opcodes in the 8-bit opcode space (which includes instruction bit 7, reserved for future opcode expansion) are detected by hardware. If an undefined opcode is detected, the opcode is overridden by hardware, forcing the opcode value within the pipeline to the defined *illegal* opcode. The offending instruction, should it eventually be issued down the execution unit's pipeline, generates an Illegal Opcode exception as described in the section Illegal Opcode. The memory location of the offending opcode keeps its original value. That location can be queried to determine the opcode value.

Software Exception

A mechanism is provided to allow an application thread to invoke an exception and is triggered using the Software Exception Set and Clear bit of cr0.1. Sub-function determination and parameter passing into and out of the exception handler is left to convention between the system-thread and application-thread. The thread's IP is incremented before saving AIP and entering the System Routine, causing execution to resume at the next application-thread instruction after returning from the System Routine.

Context Save and Restore

The System Routine is also used to save and restore the context of the Execution Unit. This feature is enabled in GPGPU workloads *only*.

When the execution engine receives a preemption or an interrupt, the application thread invokes the System Routine. The System Routine is invoked only when all in-flight registers have retired. The system routine is used to save all the state of the EU to memory. When the sequence is complete, the primary exception control bit is cleared. This action stops all execution for the given thread and invalidates the thread. This means a new thread from a different context may be loaded. When the primary exception control bit is cleared, software must ensure that all outstanding messages from the EU are dispatched out of the execution message pipeline. This is achieved by creating a dependency on the last send that is saving EU state. A dummy instruction before clearing the primary exception control bit ensures that this is achieved.

The System Routine is also invoked on a context restore request. In this case a dummy thread is loaded into the EU which starts with the System Routine. This routine now restores the state of the EU. The

restore sequence used in such a case should be consistent with the save sequence to ensure that state is restored correctly. After completing the restore sequence, the System Routine must clear the primary exception control bit in the Control Register. This enables hardware to switch to the application thread which continues execution.

Programming Note	
Context:	Context Save and Restore
When context save and restore is required to be supported for GPGPU workloads, Stack Overflow exception handling will not be supported. Software will either need to ensure stack is either completely disabled OR used in such a way, an exception will not trigger.	

Events That Do Not Generate Exceptions

The conditions described in this section are either not recognized or do not generate an exception.

Illegal Instruction Format

This condition includes malformed instructions in which the opcode is legal, but the source or destination operands or other instruction attributes do not comply with the instruction specification. There is no direct hardware support to detect these cases and the outcome of issuing a malformed instruction is undefined.

Malformed Message

A message's contents, destination registers, lengths, and descriptors are not interpreted in any way by the execution unit. Errors in specifying message fields do not raise exceptions in the EU but may be detected and reported by the shared functions.

GRF Register Out of Bounds

Unique GRF storage is allocated to each thread which, at a minimum, satisfies the register requirements specified in the thread's declaration. References to GRF register numbers beyond that called for in the thread's declaration do not generate exceptions. Depending on the implementation, out-of-bounds register numbers may be remapped to r0..r15, although this functionality should not be relied upon by the thread. The hardware guarantees the isolation of each thread's register space, thus there is no possibility of direct register manipulation via an out-of-bounds register access.

Hung Thread

There is no hardware mechanism in the EU to detect a hung thread and such a thread may remain hung indefinitely. It is expected that one or more hung threads will eventually cause the driver to recognize a context timeout and take appropriate recovery action.

Instruction Fetch Out of Bounds

The EU implements a full 32-bit instruction address range (with the 4 LSBs don't care), making it possible for a thread to attempt to jump to any 16-byte aligned offset in the 32-bit instruction address range. (Instruction addresses are offsets from the General State Base Address.) The EU does not provide any type of address checking on instruction fetch requests sent to the memory/cache hierarchy, although error conditions for memory addresses are reported via the Page Table Error Register and other memory interface registers.

FPU Math Errors

The EU's floating point units (FPUs) have defined behaviors for traditional floating point errors and do not generate exceptions. There is no support for signaling FPU math errors as exceptions.

Adds the IEEE Exception Trap Enable bit, which enables trapping IEEE exception flags. If enabled, IEEE floating-point exceptions set sticky bits in the IEEE Exceptions field of sr0.1. Note that IEEE floating-point exceptions still do *not* transfer control to any handler.

Computational Overflow

Depending on source operand types and values, destination type, and the operation being performed, overflows may occur in the execution pipelines. Many instructions support the overflow (**.o**) conditional modifier that assigns flag bits based on whether or not an overflow occurs.

The EU never signals exceptions for overflows. Software must provide any overflow handling.

System Routine Example

The following code sequence illustrates some concepts of the System Routine. It is intended to be just a shell, without getting into the specifics of each exception handler.

This example contains code for the message registers in the MRF. Other code in this example is useful for other processor generations.

The example frees enough MRF and GRF space to get the routine started, then jumps to the handler for the specific exception. Many other implementations are also valid, including single exception servicing (as opposed to looping) per invocation, and saving only the GRF or MRF space required by the exception being serviced.

```
#define ACC_DISABLE_MASK 0xFFFFFFFD
#define PRIMARY_EXCP_MASK 0x7FFFFFFF
#define SYSROUTINE_SCRATCH_BLKSIZE 16384 // for example

// Shared function IDs:
#define DPR 0x04000000
#define DPW 0x05000000

// Message lengths:
#define ML5 0x00500000
#define ML9 0x00900000

// Response lengths:
```

```
#define RL0 0x00000000
#define RL4 0x00040000
#define RL8 0x00080000

// Data port block sizes:
#define BS1_LOW  0x0000
#define BS1_HIGH 0x0100
#define BS2      0x0200
#define BS4      0x0300

// Scratch Layout:
#define SCR_OFFSET_MRF 0           //
#define SCR_OFFSET_GRF 512         // + 16 MRF registers
#define SCR_OFFSET_ARF 512 + 4096  // + 16 MRF + 128 GRF registers

// Write data port constants:
// target=dcache, type= oword_block_wr, binding_tbl_offset=0
#define DPW 0x000

// Read data port constants:
// target=dcache, type= oword_block_rd, binding_tbl_offset=0
#define DPR 0x000

Sys_Entry:  // Entry point to the System Routine.

// Disable accumulator for system routine:
and (1) cr0.0 cr0.0 ACC_DISABLE_MASK {NoMask}

// Calc scratch offset for this thread into r0.4:
shr (1) r0.4 sr0.0:uw 6 {NoMask}
add (1) r0.4 r0.4 sr0.0:ub {NoMask}
mul (1) r0.4 r0.4 SYSROUTINE_SCRATCH_BLKSIZE {NoMask}

// Setup m0 with block offset:
mov (8) m0 r0{NoMask}

// Save MRF 7..0 (may choose to save the whole MRF).
add (1) m0.2 r0.4 SCR_OFFSET_MRF {NoMask}
send (8) null m0 null DPW|ML9|RL0 {NoMask}

// Save MRF 8..15 (optional; req'ed if sys-routine stays w/in mrf7-0).
mov (8) m7 r0 {NoMask}
add (1) m7.2 r0.4 (SCR_OFFSET_MRF + 256) {NoMask}
send (8) null m7 null DPW|ML9|RL0 {NoMask}

// Save r0..r1 to system scratch:
// Note: done as a single register to guarantee external visibility
// See
mov (16) m1 r0 {NoMask}
send (8) m0 null null DPW|ML2|RL0 {NoMask}

// Save r2..r3 to free some room:
mov (16) m3 r2 {NoMask}
add (1) m0.2 r0.4 SCR_OFFSET_GRF + 64 {NoMask}
send (8) m0 null null DPW|ML4|RL0 {NoMask}

// Save r4..r7 to free some room (optional, depending on needs):
mov (16) m8 r4 {NoMask}
mov (16) m10 r6 {NoMask}
add (1) m7.2 r0.4 (SCR_OFFSET_GRF + 128) {NoMask}
send (8) m7 null null DPW|ML5|RL0 {NoMask}

// Save r8..r11 to free some room (optional, depending on needs):
mov (16) m1 r8 {NoMask}
```

```
mov (16) m3 r10 {NoMask}
add (1) m0.2 r0.4 (SCR_OFFSET_GRF + 256) {NoMask}
send (8) m0 null null DPW|ML5|RL0 {NoMask}

// Save r12..r15 to free some room (optional, depending on needs):
mov (16) m8 r12 {NoMask}
mov (16) m10 r14 {NoMask}
add (1) m7.2 r0.4 (SCR_OFFSET_GRF + 384) {NoMask}
send (8) m7 null null DPW|ML5|RL0 {NoMask}

// Save selected ARF registers (optional, depending on use):
// flags, others ...
// Save f0.0:
mov (1) r1.0:uw f0.0 {NoMask}

Next:  // Exceptions pending? If not, exit.

cmp.e (1) f0.0 cr0.4:uw 0:uw {NoMask}
(f0.0) mov (1) IP EXIT {NoMask}

// Find highest priority exception:
lzd (1) r1.1:uw cr0.4:uw {NoMask}

// Jump table to service routine:
jmpi (1) r1.1:uw{NoMask}
mov (1) IP CRService_0 {NoMask}
mov (1) IP CRService_1 {NoMask}
mov (1) IP CRService_2 {NoMask}
...
mov (1) IP CRService_15{NoMask}
mov (1) IP Next

Service_0:
// Clear exception from cr0.1.
// Perform service routine.
// Jump to exit (or if looping on exceptions, go to next loop).
...
Service_15:
// Clear exception from cr0.1.
// Perform service routine.
// Jump to exit (or if looping on exceptions, go to next loop).

Exit:
// Restore f0.0.
// Restore other ARF registers (as required).
// Restore r12..r15.
// Restore r8..r11.
// Restore r4..r7.
// Restore r0..r3.
// Restore m8..m15.
// Restore m0..m7.
// Clear Primary Exception State bit in cr0.0:
and (1) cr0.0 cr0.0 PRIMARY_EXCP_MASK
nop (16)
```

Below is a code sequence to programmatically clear the GRF scoreboard in case of a timeout waiting on a register that may never return.

At this point, all we know is we have a hung thread. We'd like to copy the GRF to scratch memory to make it visible, but there may be a register that is hung with an outstanding dependency. To get around

any hung dependency, walk the GRF using NoDDChk, using an execution mask of f0 == 0 so we don't touch the register contents.

```
Clear_Dep:
mov f0 0x00
(f0) mov r0 0x00 {NoDDChk}
(f0) mov r1 0x00 {NoDDChk}
(f0) mov r2 0x00 {NoDDChk}
...
(f0) mov r127 0x00 {NoDDChk}
// GRF scoreboard now cleared.
```

Instruction Set Summary

Instruction Set Characteristics

SIMD Instructions and SIMD Width

Instructions are SIMD (single instruction multiple data) instructions. The number of data elements per instruction, or the execution size, depends on the data type. For example, the execution size for instructions operating on 256-bit wide vectors can be up to 8 for 32-bit data types, and be up to 16 for 16-bit data. The maximum execution size for instructions for 8-bit data types is also limited to 16.

An instruction compression mode is supported for 32-bit instructions (including mixed 32-bit and 16-bit data computation). A compressed instruction works on twice as much SIMD data as that for a non-compressed instruction. A compressed instruction is converted into two native instructions by the instruction dispatcher in the EU.

Instructions are executed on a narrower SIMD execution pipeline. Therefore, native instructions take multiple execution cycles to complete.

Instruction Operands and Register Regions

Most instructions may have up to three operands, two sources and one destination. Each operand is able to address a register region. Source operands support negate and absolute modifier and channel swizzle, and the destination operand supports channel mask.

Dual destination instructions are also supported (four-operand instructions in a general sense): One case is for the implied destination - flag register, where the conditional modifiers and the predicate modifiers may apply. Another case is the message header creation (implied move or implied assembling of the header) in the *send* instruction.

Each execution channel contains an accumulator that is wider than the input data to support back-to-back accumulation operations with increased precision. The added precision (see accumulator register description in Execution Environment chapter) determines the maximum number of accumulations before possible overflow. The accumulator can be pre-loaded through the use of *mov*. It can also be pre-loaded by arithmetic instructions such as *add* or *mul*, since the result of these instructions can go to the accumulator. The accumulator registers are per thread and therefore safe for thread switching.

Register access can be direct or register-indirect. Register-indirect register access uses address registers plus an immediate offset term to compute the register addresses, and only applies to the first source operand (src0) and/or the destination operand.

There is one address register.

There are 16 address sub-registers.

Each sub-register contains a 16-bit unsigned value. The leading two sub-registers form a special doubleword that can be used as the descriptor for the send instruction.

Source operand can also be immediate value (also referred to as inline constants). For instructions with two source operands, only the second operand src1 is allowed to be immediate. For instructions with only one source operand, the source operand src0 is used and it can be an immediate.

An immediate source operand can be a scalar value of specified type up to 32-bit wide, which is replicated to create a vector with length of Execution Size. An immediate operand can also be a special 32-bit vector with 8 elements each of 4-bit signed integer value, or a 32-bit vector with 4 elements each of 8-bit restricted float value.

Instruction Execution

It is implied that all instructions operate across all channels of data unless otherwise specified either via destination mask, predication, execution mask (caused by SIMD branch and loop instructions), or execution size.

Instruction execution size can be specified per instruction, from scalar (*ExecSize* = 1) up to the maximal execution size supported for the data type, with the restriction that execution size can only be in power of 2.

EU Compact Instructions

On receiving an instruction with bit 29 (CmptCtrl) set, HW recognizes it as a 64-bit compact instruction. Hardware then uses the index fields inside the compact instruction to lookup values in the associated compaction tables, then uses the table outputs along with other fields in the compact instruction to reconstruct the 128-bit native-sized instruction.

All flow control instructions use the new offset format, a signed 32-bit offset in units of bytes.

The native 128-bit instruction format provides access to all instruction options. Only some instruction options and combinations of instruction options can be represented in the compact instruction formats.

Which native instructions can be represented as compact instructions and the details of the compact instruction formats and the compaction tables used may change with each processor generation.

In the following instruction format tables, the Mapping Bits and Mapping Description columns describe the mappings into native instruction fields.

EU Compact Instruction Format

This page contains an outline of the compacted (64bit) encodings of various instructions. For the meaning of various fields (e.g. allowed values) consult with the Instruction Fields and EU Instruction Compaction Tables page. Branch Instruction and Send instructions are not compacted.

Compact Instructions Format

<table>
<tr><th>Bit</th><th>2src</th><th>2src-imm</th><th>3src</th></tr>
<tr><td>63</td><td rowspan="7">Src1.Reg</td><td rowspan="11">ImmValue2[11:0]</td><td rowspan="7">Src1.Reg</td></tr>
<tr><td>62</td></tr>
<tr><td>61</td></tr>
<tr><td>60</td></tr>
<tr><td>59</td></tr>
<tr><td>58</td></tr>
<tr><td>57</td></tr>
<tr><td>56</td><td rowspan="5">Src2.Reg</td></tr>
<tr><td>55</td><td rowspan="4">Src1.Index2</td></tr>
<tr><td>54</td></tr>
<tr><td>53</td></tr>
<tr><td>52</td></tr>
<tr><td>51</td><td rowspan="4">Src0.Index2</td><td rowspan="4">Src0.Index2</td></tr>
<tr><td>50</td></tr>
<tr><td>49</td></tr>
<tr><td>48</td></tr>
<tr><td>47</td><td rowspan="7">Src0.Reg</td><td rowspan="7">Src0.Reg</td><td rowspan="7">Src0.Reg</td></tr>
<tr><td>46</td></tr>
<tr><td>45</td></tr>
<tr><td>44</td></tr>
<tr><td>43</td></tr>
<tr><td>42</td></tr>
<tr><td>41</td></tr>
<tr><td>40</td></tr>
<tr><td>39</td><td rowspan="3">SubRegIndex2</td><td rowspan="3">SubRegIndex2</td><td rowspan="3">SubRegIndex3</td></tr>
<tr><td>38</td></tr>
<tr><td>37</td></tr>
</table>

Bit	2src	2src-imm	3src
36			
35			
34	DataTypeIndex2	DataTypeIndex2	SourceIndex3
33			
32			
31			
30			
29	CmptCtrl = 1	CmptCtrl = 1	CmptCtrl = 1
28	ControlIndex2	ControlIndex2	ControlIndex3
27			
26			
25			
24			
23	Dst.Reg	Dst.Reg	Dst.Reg
22			
21			
20			
19			
18			
17			
16			
15	SWSB	SWSB	SWSB
14			
13			
12			
11			
10			
9			
8			
7	DbgCtrl	DbgCtrl	DbgCtrl
6	**Opcode** =(	**Opcode** =(	**Opcode** =(
5	unary/binary/	unary/binary/	ternary op

Bit	2src	2src-imm	3src
4	math/sync op)	math/sync op)	)
3			
2			
1			
0			

EU Instruction Compaction Tables

Unary/Binary Instruction Compaction Tables

The following five tables describe the mappings for the ControlIndex2, DataTypeIndex2, SubRegIndex2, Src0Index2, and Src1Index2 fields in the Unary/Binary Compact Instruction format. The values below are separated with commas and spaces for readability but represent a binary number MSB to LSB.

ControlIndex2 Compact Instruction Field Mappings

ControlIndex2	21-Bit Mapping { CondCtrl[3:0],Saturate,AccWrCtrl,AtomicCtrl,MaskCtrl, PredInv,PredCtrl[3:0], FlagRegNum,FlagSubRegNum, ChOff[2:0],ExecSize[2:0] }	Mapped Meaning
0	0000,0,0,0,0, 0,0000, 0,0, 000,100	(16|M0)
1	0000,0,0,0,0, 0,0000, 0,0, 000,011	(8|M0)
2	0000,0,0,0,1, 0,0000, 0,0, 000,000	(W) (1|M0)
3	0000,0,0,0,1, 0,0000, 0,0, 000,100	(W) (16|M0)
4	0000,0,0,0,1, 0,0000, 0,0, 000,011	(W) (8|M0)
5	0100,0,0,0,0, 0,0000, 0,0, 000,100	(16|M0) (ge)f0.0
6	0000,0,0,0,0, 0,0000, 0,0, 100,100	(16|M16)
7	0101,0,0,0,0, 0,0000, 0,0, 000,100	(16|M0) (lt)f0.0
8	0000,0,0,0,0, 0,0000, 0,0, 000,000	(1|M0)
9	0000,1,0,0,0, 0,0000, 0,0, 000,100	(16|M0) (sat)
10	0000,0,0,0,0, 0,0000, 0,0, 010,011	(8|M8)
11	0011,0,0,0,0, 0,0000, 0,0, 000,100	(16|M0) (gt)f0.0
12	0001,0,0,0,0, 0,0000, 0,0, 000,100	(16|M0) (eq)f0.0
13	0001,0,0,0,1, 0,0000, 0,0, 000,100	(W) (16|M0) (eq)f0.0
14	0010,0,0,0,0, 0,0000, 0,0, 000,100	(16|M0) (ne)f0.0

ControlIndex2	21-Bit Mapping { CondCtrl[3:0],Saturate,AccWrCtrl,AtomicCtrl,MaskCtrl, PredInv,PredCtrl[3:0], FlagRegNum,FlagSubRegNum, ChOff[2:0],ExecSize[2:0] }	Mapped Meaning
15	0000,0,0,0,0, 0,0001, 0,0, 000,100	(f0.0) (16|M0)
16	0101,0,0,0,0, 0,0000, 0,0, 000,011	(8|M0) (lt)f0.0
17	0000,0,0,0,0, 0,0001, 1,0, 000,100	(f1.0) (16|M0)
18	0000,0,0,0,1, 0,0000, 0,0, 000,001	(W) (2|M0)
19	0000,0,0,0,0, 0,0001, 0,1, 000,100	(f0.1) (16|M0)
20	0000,0,0,0,0, 0,0001, 1,1, 000,100	(f1.1) (16|M0)
21	0100,0,0,0,1, 0,0000, 0,0, 000,100	(W) (16|M0) (ge)f0.0
22	0000,0,0,0,0, 0,0000, 0,0, 100,011	(8|M16)
23	0000,0,0,0,0, 0,0000, 0,0, 110,011	(8|M24)
24	0101,0,0,0,1, 0,0000, 0,0, 000,100	(W) (16|M0) (lt)f0.0
25	0100,0,0,0,0, 0,0000, 0,0, 000,011	(8|M0) (ge)f0.0
26	0001,0,0,0,1, 0,0000, 0,0, 000,000	(W) (1|M0) (eq)f0.0
27	0000,1,0,0,0, 0,0000, 0,0, 000,011	(8|M0) (sat)
28	0101,0,0,0,0, 0,0000, 1,0, 000,100	(16|M0) (lt)f1.0
29	0001,0,0,0,0, 0,0000, 0,0, 000,011	(8|M0) (eq)f0.0
30	0000,0,1,0,0, 0,0000, 0,0, 000,011	(8|M0) {AccWrEn}
31	0000,0,0,0,1, 0,0000, 0,0, 100,100	(W) (16|M16)

DataTypeIndex2 Compact Instruction Field Mappings

DataTypeIndex2	20-Bit Mapping { Src1.RegFile[0], Src1.SrcType[3:0], Src0.RegFile[0], Dst.RegFile[0], Dst.HorzStride[1:0], Src1.IsImm, Src0.IsImm, Src0.SrcType[3:0], Dst.DstType[3:0], Dst.AddrMode }	Mapped Meaning { dst, src0, src1 }
0	1,1010,1,1,01, 0,0, 1010,1010,0	grf<1>:f grf:f grf:f
1	0,0000,1,1,01, 0,0, 1010,1010,0	grf<1>:f grf:f arf:ub

DataTypeIndex2	20-Bit Mapping { Src1.RegFile[0], Src1.SrcType[3:0], Src0.RegFile[0], Dst.RegFile[0], Dst.HorzStride[1:0], Src1.IsImm, Src0.IsImm, Src0.SrcType[3:0], Dst.DstType[3:0], Dst.AddrMode }	Mapped Meaning { dst, src0, src1 }
2	0,0000,0,1,01, 0,1, 1010,1010,0	grf<1>:f imm:f
3	0,1010,1,1,01, 1,0, 1010,1010,0	grf<1>:f grf:f imm:f
4	1,1010,1,0,01, 0,0, 1010,1010,0	arf<1>:f grf:f grf:f
5	1,1010,0,1,01, 0,0, 1010,1010,0	grf<1>:f arf:f grf:f
6	0,1010,1,0,01, 1,0, 1010,1010,0	arf<1>:f grf:f imm:f
7	0,0000,0,0,01, 0,0, 0000,0000,0	arf<1>:ub arf:ub arf:ub
8	1,1010,0,0,01, 0,0, 1010,1010,0	arf<1>:f arf:f grf:f
9	0,0101,1,1,01, 1,0, 0110,0110,0	grf<1>:d grf:d imm:w
10	1,0110,1,1,01, 0,0, 0110,0110,0	grf<1>:d grf:d grf:d
11	0,1010,0,1,01, 1,0, 1010,1010,0	grf<1>:f arf:f imm:f
12	1,0010,1,1,01, 0,0, 0010,0010,0	grf<1>:ud grf:ud grf:ud
13	0,1010,0,0,01, 1,0, 1010,1010,0	arf<1>:f arf:f imm:f
14	0,0110,1,1,01, 1,0, 0110,0110,0	grf<1>:d grf:d imm:d
15	0,0010,1,1,01, 1,0, 0010,0010,0	grf<1>:ud grf:ud imm:ud
16	0,0000,1,1,10, 0,0, 1010,1010,0	grf<2>:f grf:f arf:ub
17	0,0101,1,0,01, 1,0, 0110,0110,0	arf<1>:d grf:d imm:w
18	0,0000,0,0,01, 0,0, 0001,0001,0	arf<1>:uw arf:uw arf:ub
19	0,0000,0,1,01, 0,0, 0010,0010,0	grf<1>:ud arf:ud arf:ub
20	0,0100,1,1,01, 1,0, 0001,0101,0	grf<1>:w grf:uw imm:v
21	0,0001,1,1,01, 1,0, 0001,0001,0	grf<1>:uw

DataTypeIndex2	20-Bit Mapping { Src1.RegFile[0], Src1.SrcType[3:0], Src0.RegFile[0], Dst.RegFile[0], Dst.HorzStride[1:0], Src1.IsImm, Src0.IsImm, Src0.SrcType[3:0], Dst.DstType[3:0], Dst.AddrMode }	Mapped Meaning { dst, src0, src1 }
		grf:uw imm:uw
22	1,0010,1,1,10, 0,0, 0010,0010,0	grf<2>:ud grf:ud grf:ud
23	0,0000,1,1,01, 0,0, 1010,0110,0	grf<1>:d grf:f arf:ub
24	1,0001,1,0,01, 0,0, 0110,0110,0	arf<1>:d grf:d grf:uw
25	0,0000,1,1,01, 0,0, 0010,1010,0	grf<1>:f grf:ud arf:ub
26	0,0101,1,1,01, 1,0, 0010,0110,0	grf<1>:d grf:ud imm:w
27	0,0000,0,1,01, 0,0, 0001,0001,0	grf<1>:uw arf:uw arf:ub
28	0,0000,1,1,01, 0,0, 0001,1010,0	grf<1>:f grf:uw arf:ub
29	0,0000,1,1,01, 0,0, 0000,1010,0	grf<1>:f grf:ub arf:ub
30	0,0000,1,1,01, 0,0, 0110,1010,0	grf<1>:f grf:d arf:ub
31	0,0000,0,1,01, 0,0, 1010,1010,0	grf<1>:f arf:f arf:ub

SubRegIndex2 Compact Instruction Field Mappings

SubRegIndex2	15-Bit Mapping { Src1.SubRegNum[4:0], Src0.SubRegNum[4:0], Dst.SubRegNum[4:0] }	Mapped Meaning { dst, src0, src1 }
0	00000,00000,00000	.0 .0 .0
1	10000,00000,00000	.0 .0 .16
2	00100,00000,00000	.0 .0 .4
3	01100,00000,00000	.0 .0 .12
4	00000,00100,00000	.0 .4 .0
5	01000,00000,00000	.0 .0 .8
6	10100,00000,00000	.0 .0 .20
7	00000,00000,01000	.8 .0 .0
8	00000,01000,00000	.0 .8 .0
9	11000,00000,00000	.0 .0 .24
10	11100,00000,00000	.0 .0 .28

SubRegIndex2	15-Bit Mapping { Src1.SubRegNum[4:0], Src0.SubRegNum[4:0], Dst.SubRegNum[4:0] }	Mapped Meaning { dst, src0, src1 }
11	00000,10000,00000	.0 .16 .0
12	00000,00000,00100	.4 .0 .0
13	00000,11000,00000	.0 .24 .0
14	00000,10100,00000	.0 .20 .0
15	00000,01100,00000	.0 .12 .0
16	00000,11100,00000	.0 .28 .0
17	00000,00000,11100	.28 .0 .0
18	00000,00000,10000	.16 .0 .0
19	00000,00000,01100	.12 .0 .0
20	00000,00000,11000	.24 .0 .0
21	00000,00000,10100	.20 .0 .0
22	00000,00000,00010	.2 .0 .0
23	00000,01010,00000	.0 .10 .0
24	00000,00010,00000	.0 .2 .0
25	00000,00100,00100	.4 .4 .0
26	00000,00010,11100	.28 .2 .0
27	00000,00010,00010	.2 .2 .0
28	00000,01100,01100	.12 .12 .0
29	00000,00001,00000	.0 .1 .0
30	00000,00011,00000	.0 .3 .0
31	11000,11000,00000	.0 .24 .24

Src0Index2 Compact Instruction Field Mappings

Src0Index2	12-Bit Mapping { Src0.VertStride[3:0], Src0.Width[2:0], Src0.AddrMode, Src0.HorzStride[1:0], Src0.Mod[1:0] }	Mapped Meaning
0	0100,011,0,01,00	r<8;8,1>
1	0000,000,0,00,00	r<0;1,0>
2	0100,011,0,01,10	-r<8;8,1>
3	0100,011,0,01,01	(abs)r<8;8,1>
4	0000,000,0,00,10	-r<0;1,0>
5	0010,000,0,00,00	r<2;1,0>
6	0010,010,0,00,00	r<2;4,0>
7	0011,010,0,00,00	r<4;4,0>
8	0010,001,0,01,00	r<2;2,1>
9	0011,000,0,00,00	r<4;1,0>

Src0Index2	12-Bit Mapping { Src0.VertStride[3:0], Src0.Width[2:0], Src0.AddrMode, Src0.HorzStride[1:0], Src0.Mod[1:0] }	Mapped Meaning
10	0010,001,0,01,10	-r<2;2,1>
11	0011,010,0,01,00	r<4;4,1>
12	0100,011,0,01,11	-(abs)r<8;8,1>
13	0001,000,0,00,00	r<1;1,0>
14	0000,000,0,00,01	(abs)r<0;1,0>
15	1111,000,1,00,00	r[a]<1,0>

Src1Index2 Compact Instruction Field Mappings

Src1Index2	12-Bit Mapping { Src1.Mod[1:0], Src1.VertStride[3:0], Src1.Width[2:0], Src1.AddrMode, Src1.HorzStride[1:0] }	Mapped Meaning
0	00,0100,011,0,01	r<8;8,1>
1	00,0000,000,0,00	r<0;1,0>
2	10,0100,011,0,01	-r<8;8,1>
3	10,0000,000,0,00	-r<0;1,0>
4	01,0100,011,0,01	(abs)r<8;8,1>
5	10,0011,010,0,00	-r<4;4,0>
6	00,0010,000,0,00	r<2;1,0>
7	00,0010,001,0,01	r<2;2,1>
8	10,0010,001,0,01	-r<2;2,1>
9	00,0011,010,0,00	r<4;4,0>
10	00,0011,010,0,01	r<4;4,1>
11	00,0011,000,0,00	r<4;1,0>
12	11,0100,011,0,01	-(abs)r<8;8,1>
13	01,0000,000,0,00	(abs)r<0;1,0>
14	11,0000,000,0,00	-(abs)r<0;1,0>
15	10,0011,010,0,01	-r<4;4,1>

ImmValue2 Compact Instruction Field

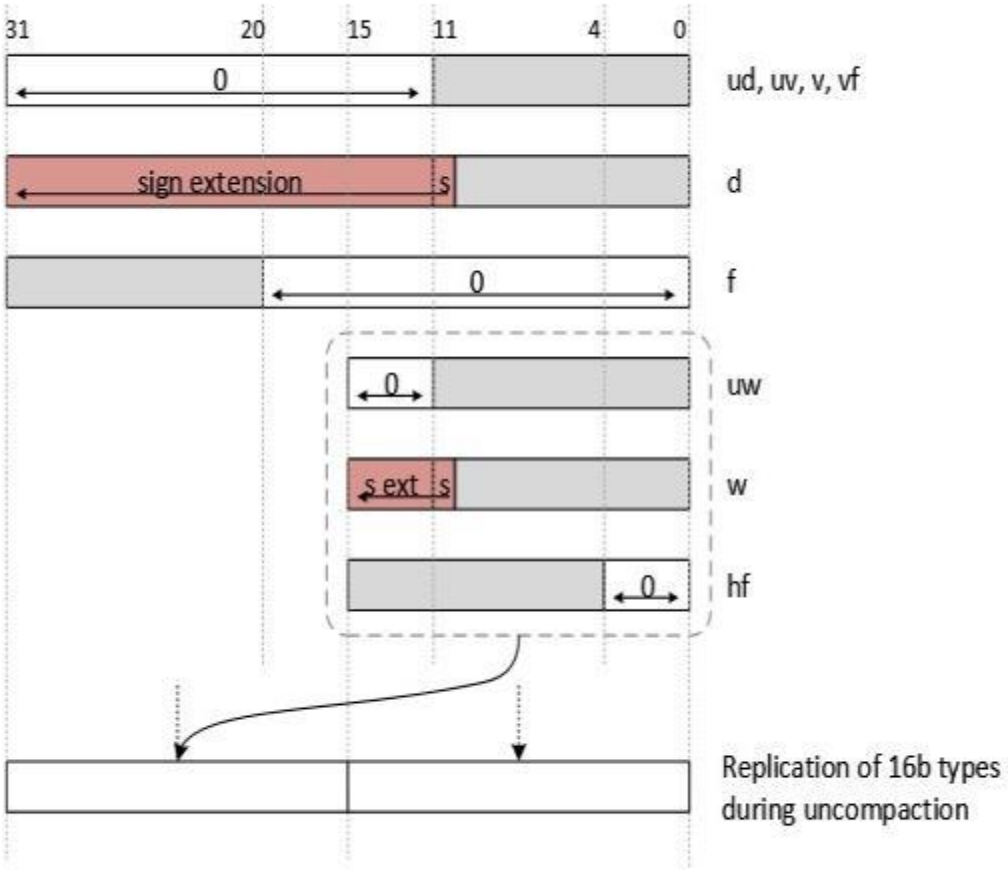
Immediates values of up to 12 bits can be represented in ImmValue2 field of compacted format.

For unsigned integer types if bits [31:12] ([15:12] for 16b type) are all zero's, then bits [11:0] can be compacted as immediate value. Packed datatypes uv, v, vf are treated as 32 unsigned integer in the context of compaction. During uncompaction zero is extended up to the MSB of datatype. 16bit datatypes are replicated to occupy 32bit of uncompacted immediate value.

For signed integer types if bits [31:11] ([15:12] for 16b type) are all zero's or all one's, then bits [11:0] can be compacted as immediate value. During uncompaction sign bit 11 is extended up to the MSB of datatype.16bit datatypes are replicated to occupy 32bit of uncompacted immediate value.

For float types if bits [19:0] ([3:0] for 16b type) are all zero's, then bits [31:20] can be compacted as immediate value. During uncompaction bits [19:0] ([3:0] for 16d type) are set to zero. 16bit datatypes are replicated to occupy 32bit of uncompacted immediate value.

64bit datatypes are not compacted.



Ternary Instruction Compaction Tables

The following three tables describe the mappings for the ControlIndex3, SourceIndex3 and SubRegIndex3 fields in the Ternary Compact Instruction format.

The following three tables describe the mappings for the ControlIndex3, SourceIndex3 and SubRegIndex3 fields in the Ternary Compact Instruction format.

ControlIndex3 Compact Instruction Field Mappings

ControlIndex3	36-Bit Mapping { CondMod[3:0], Src1.SrcType[2:0], Src2.SrcType[2:0], Dst.RegFile[0], Dst.HorzStride[0], Src0.SrcType[2:0], ExecDataType, Dst.DstType[2:0], Saturate, AccWrCtrl, AtomicCtrl, MaskCtrl, PredInv, PredCtrl[3:0], FlagRegNum, FlagSubRegNum, ChOff[2:0], ExecSize[2:0] }	Mapped Meaning
0	0000,010,010,1,0,010,1,010, 0,0,0,0,0,0000,0,0,000,100	(16|M0) grf<1>:f :f :f :f
1	0000,010,010,1,0,010,1,010, 0,0,0,0,0,0000,0,0,000,011	(8|M0) grf<1>:f :f :f :f
2	0000,010,010,0,0,010,1,010, 0,0,0,0,0,0000,0,0,000,011	(8|M0) arf<1>:f :f :f :f
3	0000,010,010,1,0,010,1,010, 0,0,0,1,0,0000,0,0,000,011	(W) (8|M0) grf<1>:f :f :f :f
4	0000,010,010,0,0,010,1,010, 0,0,0,1,0,0000,0,0,000,011	(W) (8|M0) arf<1>:f :f :f :f
5	0000,010,010,0,0,010,1,010, 0,0,0,0,0,0000,0,0,010,011	(8|M8) arf<1>:f :f :f :f
6	0000,010,010,1,0,010,1,010, 0,0,0,0,0,0000,0,0,010,011	(8|M8) grf<1>:f :f :f :f
7	0000,010,010,0,0,010,1,010, 0,0,0,1,0,0000,0,0,010,011	(W) (8|M8) arf<1>:f :f :f :f
8	0000,010,010,1,0,010,1,010, 0,0,0,1,0,0000,0,0,010,011	(W) (8|M8) grf<1>:f :f :f :f
9	0000,010,010,1,0,010,1,010, 0,0,0,1,0,0000,0,0,000,100	(W) (16|M0) grf<1>:f :f :f :f
10	0000,010,010,0,0,010,1,010, 0,0,0,0,0,0000,0,0,000,100	(16|M0) arf<1>:f :f :f :f
11	0000,010,010,1,0,010,1,010, 1,0,0,0,0,0000,0,0,000,100	(16|M0) (sat)grf<1>:f :f :f :f
12	0000,010,010,1,0,010,1,010, 0,0,0,0,0,0000,0,0,100,100	(16|M16) grf<1>:f :f :f :f
13	0000,010,010,0,0,010,1,010, 0,0,0,1,0,0000,0,0,000,100	(W) (16|M0) arf<1>:f :f :f :f
14	0000,010,010,1,0,010,1,010, 0,0,0,1,0,0000,0,0,000,000	(W) (1|M0) grf<1>:f :f :f :f
15	0000,010,010,1,0,010,1,010, 1,0,0,0,0,0000,0,0,000,011	(8|M0) (sat)grf<1>:f :f :f :f
16	0000,010,010,0,0,010,1,010, 0,0,0,1,0,0000,0,0,110,011	(W) (8|M24) arf<1>:f :f :f :f
17	0000,010,010,0,0,010,1,010, 0,0,0,1,0,0000,0,0,100,011	(W) (8|M16) arf<1>:f :f :f :f
18	0000,010,010,1,0,010,1,010, 0,0,0,1,0,0000,0,0,110,011	(W) (8|M24) grf<1>:f :f :f :f
19	0000,010,010,1,0,010,1,010, 0,0,0,1,0,0000,0,0,100,011	(W) (8|M16) grf<1>:f :f :f :f
20	0000,010,010,0,0,010,1,010, 0,0,0,0,0,0000,0,0,100,011	(8|M16) arf<1>:f :f :f :f
21	0000,010,010,0,0,010,1,010, 0,0,0,0,0,0000,0,0,110,011	(8|M24) arf<1>:f :f :f :f
22	0000,010,010,1,0,010,1,010, 0,0,0,0,0,0000,0,0,100,011	(8|M16) grf<1>:f :f :f :f
23	0000,010,010,1,0,010,1,010, 0,0,0,0,0,0000,0,0,110,011	(8|M24) grf<1>:f :f :f :f
24	0000,010,010,0,0,010,1,010, 1,0,0,0,0,0000,0,0,000,100	(16|M0) (sat)arf<1>:f :f :f :f
25	0000,010,010,1,0,010,1,010, 1,0,0,1,0,0000,0,0,000,100	(W) (16|M0) (sat)grf<1>:f :f :f :f
26	0000,010,010,1,0,010,1,010, 0,0,0,1,0,0000,0,0,100,100	(W) (16|M16) grf<1>:f :f :f :f
27	0000,010,010,1,0,010,0,010, 0,0,0,1,0,0000,0,0,000,000	(W) (1|M0) grf<1>:ud :ud :ud :ud
28	0000,010,010,0,0,010,1,010, 0,0,0,0,0,0000,0,0,100,100	(16|M16) arf<1>:f :f :f :f
29	0000,010,010,1,0,010,1,010, 1,0,0,0,0,0000,0,0,100,100	(16|M16)(sat)grf<1>:f :f :f :f
30	0000,010,010,1,0,010,1,010, 0,0,0,1,0,0000,0,0,000,010	(W) (4|M0) grf<1>:f :f :f :f
31	0000,010,010,0,0,010,1,010, 1,0,0,0,0,0000,0,0,000,011	(8|M0) (sat)arf<1>:f :f :f :f

SourceIndex3 Compact Instruction Field Mappings

SourceIndex3	21-Bit Mapping { Src2.RegFile[0], Src2.HorzStride[1:0], Src1.RegFile[0], Src1.HorzStride[1:0], Src1.VertStride[1], Src1.Mod[1:0], Src2.Mod[1:0], Src1.VertStride[0], Src0.RegFile[0], Src0.HorzStride[1:0], Src2.IsImm, Src0.IsImm, Src0.Mod[1:0], Src0.VertStride[1], Src0.VertStride[0] }	Mapped Meaning { src0, src1, src2 }
0	1,00,1,01,1,00,00, 1,1,00,0,0,00,0,0	grf<0;0> grf<8;1> grf<0>
1	1,00,1,01,1,00,00, 1,0,01,0,0,00,1,0	arf<4;1> grf<8;1> grf<0>
2	1,01,1,01,1,00,00, 1,1,01,0,0,00,1,1	grf<8;1> grf<8;1> grf<1>
3	1,00,1,01,1,00,00, 1,1,01,0,0,00,1,1	grf<8;1> grf<8;1> grf<0>
4	1,01,1,00,0,00,00, 0,1,01,0,0,00,1,1	grf<8;1> grf<0;0> grf<1>
5	1,01,1,01,1,00,00, 1,1,01,0,0,10,1,1	-grf<8;1> grf<8;1> grf<1>
6	1,01,0,01,1,00,00, 1,1,01,0,0,00,1,1	grf<8;1> arf<8;1> grf<1>
7	1,00,0,01,1,00,00, 1,1,00,0,0,00,0,0	grf<0;0> arf<8;1> grf<0>
8	1,01,1,01,1,00,00, 1,1,00,0,0,00,0,0	grf<0;0> grf<8;1> grf<1>
9	1,01,1,01,1,00,10, 1,1,01,0,0,00,1,1	grf<8;1> grf<8;1> -grf<1>
10	1,01,1,01,1,10,00, 1,1,01,0,0,00,1,1	grf<8;1> -grf<8;1> grf<1>
11	1,01,1,00,0,00,00, 0,1,00,0,0,00,0,0	grf<0;0> grf<0;0> grf<1>
12	1,00,0,01,1,00,00, 1,1,01,0,0,00,1,1	grf<8;1> arf<8;1> grf<0>
13	1,00,1,01,1,10,00, 1,1,00,0,0,00,0,0	grf<0;0> -grf<8;1> grf<0>
14	1,00,1,01,1,10,00, 1,1,01,0,0,00,1,1	grf<8;1> -grf<8;1> grf<0>
15	1,00,1,01,1,00,00, 1,1,01,0,0,10,1,1	-grf<8;1> grf<8;1> grf<0>
16	1,00,1,00,0,00,00, 0,1,01,0,0,00,1,1	grf<8;1> grf<0;0> grf<0>
17	1,00,1,01,1,00,00, 1,1,00,0,0,10,0,0	-grf<0;0> grf<8;1> grf<0>
18	1,00,1,00,0,00,00, 0,1,00,0,0,00,0,0	grf<0;0> grf<0;0> grf<0>
19	1,01,1,01,1,10,00, 1,1,00,0,0,00,0,0	grf<0;0> -grf<8;1> grf<1>

SourceIndex3	21-Bit Mapping { Src2.RegFile[0], Src2.HorzStride[1:0], Src1.RegFile[0], Src1.HorzStride[1:0], Src1.VertStride[1], Src1.Mod[1:0], Src2.Mod[1:0], Src1.VertStride[0], Src0.RegFile[0], Src0.HorzStride[1:0], Src2.IsImm, Src0.IsImm, Src0.Mod[1:0], Src0.VertStride[1], Src0.VertStride[0] }	Mapped Meaning { src0, src1, src2 }
20	1,00,1,01,1,00,10, 1,1,00,0,0,00,0,0	grf<0;0> grf<8;1> -grf<0>
21	1,01,0,01,1,00,00, 11,00,0,0,00,0,0	grf<0;0> arf<8;1> grf<1>
22	1,00,1,01,1,00,10, 1,1,01,0,0,00,1,1	grf<8;1> grf<8;1> -grf<0>
23	1,01,1,01,1,00,10, 1,1,01,0,0,10,1,1	-grf<8;1> grf<8;1> -grf<1>
24	1,01,0,01,1,00,00, 1,1,01,0,0,10,1,1	-grf<8;1> arf<8;1> grf<1>
25	1,01,1,01,1,10,00, 1,1,01,0,0,10,1,1	-grf<8;1> -grf<8;1> grf<1>
26	1,01,1,00,0,10,00, 0,1,01,0,0,00,1,1	grf<8;1> -grf<0;0> grf<1>
27	1,01,1,00,0,00,10, 0,1,01,0,0,00,1,1	grf<8;1> grf<0;0> -grf<1>
28	1,01,1,01,1,00,00, 1,1,00,0,0,10,0,0	-grf<0;0> grf<8;1> grf<1>
29	1,01,1,01,1,00,10, 1,1,00,0,0,00,0,0	grf<0;0> grf<8;1> -grf<1>
30	1,00,1,00,0,00,10, 0,1,01,0,0,00,1,1	grf<8;1> grf<0;0> -grf<0>
31	1,01,0,01,1,00,10, 1,1,01,0,0,00,1,1	grf<8;1> arf<8;1> -grf<1>

SubRegIndex3 Compact Instruction Field Mappings

SubRegIndex3	20-Bit Mapping { Src2.SubRegNum[4:0], Src1.SubRegNum[4:0], Src0.SubRegNum[4:0], Dst.SubRegNum[4:0] }	Mapped Meaning { dst, src0, src1, src2 }
0	00000,00000,00000,00000	.0 .0 .0 .0
1	00100,00000,00000,00000	.0 .0 .0 .4
2	00000,00000,01100,00000	.0 .12 .0 .0
3	10100,00000,00000,00000	.0 .0 .0 .20
4	10000,00000,11100,00000	.0 .28 .0 .16
5	01100,00000,00000,00000	.0 .0 .0 .12
6	01000,00000,00000,00000	.0 .0 .0 .8
7	00000,01000,00000,00000	.0 .0 .8 .0
8	00000,00100,00000,00000	.0 .0 .4 .0
9	11000,00000,00000,00000	.0 .0 .0 .24
10	10000,00000,00000,00000	.0 .0 .0 .16

SubRegIndex3	20-Bit Mapping { Src2.SubRegNum[4:0], Src1.SubRegNum[4:0], Src0.SubRegNum[4:0], Dst.SubRegNum[4:0] }	Mapped Meaning { dst, src0, src1, src2 }
11	11100,00000,00000,00000	.0 .0 .0 .28
12	00000,11000,00000,00000	.0 .0 .24 .0
13	00000,00000,00100,00000	.0 .4 .0 .0
14	00000,10000,00000,00000	.0 .0 .16 .0
15	00000,01100,00000,00000	.0 .0 .12 .0
16	00000,10100,00000,00000	.0 .0 .20 .0
17	00000,11100,00000,00000	.0 .0 .28 .0
18	00000,00000,01000,00000	.0 .8 .0 .0
19	00000,00000,10000,00000	.0 .16 .0 .0
20	00000,00000,11000,00000	.0 .24 .0 .0
21	00000,00000,10100,00000	.0 .20 .0 .0
22	00000,00000,11100,00000	.0 .28 .0 .0
23	11000,00000,11100,00000	.0 .28 .0 .24
24	00100,00000,01000,00000	.0 .8 .0 .4
25	00100,00000,01100,00000	.0 .12 .0 .4
26	01000,00000,01100,00000	.0 .12 .0 .8
27	10000,00000,11000,00000	.0 .24 .0 .16
28	10000,00000,10100,00000	.0 .20 .0 .16
29	01100,00000,00100,00000	.0 .4 .0 .12
30	10100,00000,11100,00000	.0 .28 .0 .20
31	01000,00000,00100,00000	.0 .4 .0 .8

Opcode Encoding

Byte 0 of the 128-bit instruction word contains the opcode. The opcode uses 7 bits. Bit location 7 in byte 0 is reserved for future opcode extension.

The opcodes are encoded and organized into five groups based on the type of operations: Special instructions, move/logic instructions (opcode=00xxxxxb), flow control instructions (opcode=010xxxxb), miscellaneous instructions (opcode=011xxxxb), parallel arithmetic instructions (opcode=100xxxxb), and vector arithmetic instructions (opcode=101xxxxb). Opcodes 110xxxb are reserved.

Note: Opcodes appear in the overall Instruction Set Summary Table as well. The following subsections still serve the purpose of describing various instruction groups.

Move and Logic Instructions

This instruction group has an opcode format of 11xxxxxb.

- The opcodes for move instructions (***mov***, ***sel*** and ***movi***) share the common 5 MSBs in the form of 11000xxb.

- The opcodes for logic instructions (***not***, ***and***, ***or***, and ***xor***) share the common 5 MSBs in the form of 11001xxb.
- The opcodes for shift instructions (***shr***, ***shl***, ***asr***, **ror**, and **rol**) share the common 4 MSBs in the form of 1101xxxb.
- The opcodes for compare instructions (***cmp*** and ***cmpn***) share the common 6 MSBs in the form of 111000xb. Bit 0 indicates whether it is a normal compare, *cmp*, or a special compare-NaN, *cmpn*.

Move and Logic Instructions

Opcode	Instruction	Description	#src	#dst
0x60	nop	No Operation	0	0
0x61	mov	Component-wise move	1	1
0x62	sel	Component-wise selective move based on predication	2	1
0x63	movi	Fast component-wise indexed move	1	1
0x64	not	Component-wise one's complement (bitwise not)	1	1
0x65	and	Component-wise logical AND (bitwise and)	2	1
0x66	or	Component-wise logical OR (bitwise or)	2	1
0x67	xor	Component-wise logical XOR (bitwise xor)	2	1
0x68	shr	Component-wise logical shift right	2	1
0x69	shl	Component-wise logical shift left	2	1
0x6A	smov	Scattered Move	1	1
0x6C	asr	Component-wise arithmetic shift right	2	1
0x6D	*Reserved*			
0x6E	ror	Component-wise logical rotate right	2	1
0x6F	rol	Component-wise logical rotate left	2	1
0x70	cmp	Component-wise compare, store condition code in destination	2	1
0x71	cmpn	Component-wise compare-NaN, store condition code in destination	2	1
0x72	csel	Component-wise selective move based on result of compare	3	1
0x73-0x76	*Reserved*			
0x77	bfrev	Reverse bits	1	1
0x78	bfe	Bitfield extract	3	1
0x79	bfi1	Bitfield insert macro instruction 1, generate mask	2	1
0x7A	bfi2	Bitfield insert macro instruction 2, insert based on mask	3	1
0x7B-0x7F	*Reserved*			

Flow Control Instructions

This instruction group has an opcode format of 010xxxxb.

Flow Control Instructions

Opcode dec	Opcode hex	Instruction	Description	#src	#dst
32	0x20	jmpi	Jump indexed	1	0
33	0x21	brd	Branch - Diverging	1	0
34	0x22	if	If	0/2	0
35	0x23	brc	Branch - Converging	1	-
36	0x24	else	Else	1	0
37	0x25	endif	End if	0	0
38	0x26	*Reserved*			
39	0x27	while	While	1	0
40	0x28	break	Break	1	0
41	0x29	cont	Continue	1	0
42	0x2A	halt	Halt	1	0
43	0x2B	calla	Subroutine call absolute	1	1
44	0x2C	call	Subroutine call	1	1
45	0x2D	return	Subroutine return	1	1
46	0x2E	goto	Goto	2	0
47	0x2F	join	Join	2	0

Miscellaneous Instructions

This instruction group has an opcode format of 011xxxxb.

Miscellaneous Instructions

Opcode dec	Opcode hex	Instruction	Description	#src	#dst
49	0x31	send	Send	2	1
53-55	0x35-0x37	*Reserved*			
56	0x38	math	Math functions for extended math pipeline	1/2	1/2
57-63	0x39-0x3F	*Reserved*			

Parallel Arithmetic Instructions

This instruction group has an opcode format of 100xxxxb.

Parallel Arithmetic Instructions

Opcode dec	Opcode hex	Instruction	Description	#src	#dst

Opcode		Instruction	Description	#src	#dst
64	0x40	add	Component-wise addition	2	1
65	0x41	mul	Component-wise multiply	2	1
66	0x42	avg	Component-wise average of the two source operands	2	1
67	0x43	frc	Component-wise floating point truncate-to-minus-infinity fraction	1	1
68	0x44	rndu	Component-wise floating point rounding up (ceiling)	1	1
69	0x45	rndd	Component-wise floating point rounding down (floor)	1	1
70	0x46	rnde	Component-wise floating point rounding toward nearest even	1	1
71	0x47	rndz	Component-wise floating point rounding toward zero	1	1
72	0x48	mac	Component-wise multiply accumulate	2	1
73	0x49	mach	multiply accumulate high	2	1
74	0x4A	lzd	leading zero detection	1	1
75	0x4B	fbh	Find first 1 for UD from msb side, or first 1/0 for D.	1	1
76	0x4C	fbl	First first 1 for UD from lsb side	1	1
77	0x4D	cbit	Count bits set	1	1
78	0x4E	addc	Integer add with carry	2	1 + acc.
79	0x4F	subb	integer subtract with borrow	2	1 + acc.

Vector Arithmetic Instructions

This instruction group has an opcode format of 101xxxxb.

Vector Arithmetic Instructions

Opcode		Instruction	Description	#src	#dst
dec	hex				
82	0x52	*Reserved*			
83	0x53	*Reserved*			
88	0x58	dp4a	Dot product accumulate	3	1
91	0x5B	mad (fma)	Component-wise floating point mad computation (a multiple-add)	3	1
92	0x5C	*Reserved*			
93	0x5D	***madm (fmam)***	Component-wise floating point fused multiply and add for macro operations.	3	1
94	0x5E	*Reserved*			
95	0x5F	*Reserved*			

Special Instructions

Instruction
There are two special instructions, *illegal* and *sync*. • *Illegal* instruction may be used for instruction padding in memory outside the normal instruction sequence such as before or after the kernel program as well as between subroutines. • The *Synchronize* instruction is used to interact with the software scoreboard synchronize with hardware barriers or the host.

Opcode	Instruction	Description	#src	#dst
0x00	illegal	Illegal instruction	0	0
0x01	sync	Synchronize	1	0
0x02-0x1F	*Reserved*			

Instruction Set Summary Tables

The columns in the following tables specify instruction mnemonics, hex opcodes, full names, instruction groups, processor generation, the number of source operands, whether the instruction supports predication, any support for source modifiers, an indication of supported data types, whether the instruction supports saturation, and any support for conditional modifiers.

See the separate Accumulator Restrictions table for information about how instructions are allowed to use accumulators.

N and Y indicate No (no support for a feature) and Yes (full support for a feature) respectively.

A SrcMod (source modifier) value of Y indicates that a numeric source modifier is allowed, optionally specifying absolute value, negation, or a forced negative value. The value N indicates no source modifier support.

SrcMod Information
A SrcMod value of ** indicates a logical source modifier is allowed, optionally inverting all source bits (a NOT operation).

In the Src Types and Dst Type columns, Int means any integer type and * means such an extensive list of types that you must refer to the detailed instruction description.

Byte (B, UB) and QWord (Q, UQ) integer types cannot be mixed in the same instruction.

Instruction Set Summary Table A to B (Listed by Instruction Mnemonic)

Mnem.	Hex Opcode	Name	Group	Srcs	Pred?	SrcMod	Src Types	Dst Type	Sat?	CondMod?
add	40	Addition	Parallel Arithmetic	2	Y	Y	*	*	Y	Y
addc	4E	Integer Addition with Carry	Parallel Arithmetic	2	Y	N	UD	UD	N	Y
and	65	Logic And	Move and Logic	2	Y	**	Int	Int	N	Equality only
asr	6C	Arithmetic Shift Right	Move and Logic	2	Y	Y	Int	Int	Y	Y
avg	42	Average	Parallel Arithmetic	2	Y	Y	B, UB W, UW D, UD	B, UB W, UW D, UD	Y	Y
bfe	78	Bit Field Extract	Move and Logic	3	Y	N	UD, D	UD, D	N	N
bfi1	79	Bit Field Insert 1	Move and Logic	2	Y	N	UD, D	UD, D	N	N
bfi2	7A	Bit Field Insert 2	Move and Logic	3	Y	N	UD, D	UD, D	N	N
bfrev	77	Bit Field Reverse	Move and Logic	1	Y	N	UD	UD	N	N
brc	23	Branch Converging	Flow Control	0 or 1	Y	N	D		N	N
brd	21	Branch Diverging	Flow Control	0 or 1	Y	N	D		N	N
break	28	Break	Flow Control	0	Y	N			N	N

Instruction Set Summary Table C to E (Listed by Instruction Mnemonic)

Mnem.	Hex Opcode	Name	Group	Srcs	Pred?	SrcMod	Src Types	Dst Type	Sat?	CondMod?
call	2C	Call	Flow Control	0	Y	N		D, UD	N	N
calla	2B	Call Absolute	Flow Control	0	Y	N		D, UD	N	N
cbit	4D	Count Bits Set	Move and Logic	1	Y	N	UB, UW, UD	UD	N	N
cmp	70	Compare	Move and Logic	2	Y	Y	*	*	N	Y
cmpn	71	Compare NaN	Move and Logic	2	Y	Y	*	*	N	Y

Mnem.	Hex Opcode	Name	Group	Srcs	Pred?	SrcMod	Src Types	Dst Type	Sat?	CondMod?
cont	29	Continue	Flow Control	0	Y	N			N	N
csel	72	Conditional Select	Move and Logic	3	N	Y	F	F	Y	N
dp4a	58	Dot Product 4 Accumulate	Vector Arithmetic	2	Y	Y	F	F	Y	Y
else	24	Else	Flow Control	0	N	N			N	N
endif	25	End If	Flow Control	0	N	N			N	N

Instruction Set Summary Table F to L (Listed by Instruction Mnemonic)

Mnem.	Hex Opcode	Name	Group	Srcs	Pred?	SrcMod	Src Types	Dst Type	Sat?	CondMod?
fbh	4B	Find First Bit from MSB Side	Move and Logic	1	Y	N	D, UD	UD	N	N
fbl	4C	Find First Bit from LSB Side	Move and Logic	1	Y	N	UD	UD	N	N
frc	43	Fraction	Parallel Arithmetic	1	Y	Y	F	F	N	Y
goto	2E	Goto	Flow Control	0	Y	N			N	N
halt	2A	Halt	Flow Control	0	Y	N			N	N
if	22	If	Flow Control	0	Y	N			N	N
illegal	00	Illegal	Special	0	N	N			N	N
jmpi	20	Jump Indexed	Flow Control	1	Y	N	D		N	N
join	2F	Join	Flow Control	0	Y	N			N	N
line	59	Line	Vector Arithmetic	2	Y	Y	F	F	Y	Y
lzd	4A	Leading Zero Detection	Move and Logic	1	Y	Y	D, UD	UD	Y	Y

Instruction Set Summary Table M to P (Listed by Instruction Mnemonic)

Mnem.	Hex Opcode	Name	Group	Srcs	Pred?	SrcMod	Src Types	Dst Type	Sat?	CondMod?
mac	48	Multiply Accumulate	Parallel Arithmetic	2	Y	Y	*	*	Y	Y
mach	49	Multiply Accumulate High	Parallel Arithmetic	2	Y	Y	*	*	Y	Y
mad	5B	Multiply Add	Vector Arithmetic	3	Y	Y	*	*	Y	N
madm	5D	Multiply Add for	Vector	3	Y	Y	*	*	Y	N

Mnem.	Hex Opcode	Name	Group	Srcs	Pred?	SrcMod	Src Types	Dst Type	Sat?	CondMod?
		Macro	Arithmetic							
math	38	Extended Math Function	Miscellaneous	2	Y	N	*	*	Y	N
mov	01	Move	Move and Logic	1	Y	Y	*	*	Y	Y
mov	61	Move	Move and Logic	1	Y	Y	*	*	Y	Y
movi	63	Move Indexed	Move and Logic	1	Y	Y	*	*	Y	N
mul	41	Multiply	Parallel Arithmetic	2	Y	Y	*	*	Y	Y
nop	60	No Operation	Move and Logic	0	N	N			N	N
not	64	Logic Not	Move and Logic	1	Y	**	Int	Int	N	Equality only
or	66	Logic Or	Move and Logic	2	Y	**	Int	Int	N	Equality only

Instruction Set Summary Table R to X (Listed by Instruction Mnemonic)

Mnem.	Hex Opcode	Name	Group	Srcs	Pred?	SrcMod	Src Types	Dst Type	Sat?	CondMod?
ret	2D	Return	Flow Control	1	Y	N	D, UD		N	N
rndd	45	Round Down	Parallel Arithmetic	1	Y	Y	F	F	Y	Y
rnde	46	Round to Nearest or Even	Parallel Arithmetic	1	Y	Y	F	F	Y	Y
rndu	44	Round Up	Parallel Arithmetic	1	Y	Y	F	F	Y	Y
rndz	47	Round to Zero	Parallel Arithmetic	1	Y	Y	F	F	Y	Y
sad2	50	Sum of Absolute Difference 2	Vector Arithmetic	2	Y	Y	B, UB	W, UW	Y	Y
sada2	51	Sum of Absolute Difference Accumulate 2	Vector Arithmetic	2	Y	Y	B, UB	W, UW	Y	Y
sel	62	Select	Move and Logic	2	Y	Y	*	*	Y	Y
send	31	Send Message	Miscellaneous	2	Y	N	*	*	N	N
sendc	32	Send Message Conditional	Miscellaneous	2	Y	N	*	*	N	N

Mnem.	Hex Opcode	Name	Group	Srcs	Pred?	SrcMod	Src Types	Dst Type	Sat?	CondMod?
shl	69	Shift Left	Move and Logic	2	Y	Y	Int	Int	Y	Y
shr	68	Shift Right	Move and Logic	2	Y	Y	Int	Int	Y	Y
smov	6A	Scattered Move	Move and Logic	1	Y	N	*	*	N	N
subb	4F	Integer Subtraction with Borrow	Parallel Arithmetic	2	Y	N	UD	UD	N	Y
sync	01	Synchronize	Special	1	Y	N	*	*	N	N
while	27	While	Flow Control	0	Y	N			N	N
xor	67	Logic Xor	Move and Logic	2	Y	**	Int	Int	N	Equality only

Accumulator Restrictions

This section describes restrictions on accumulator access: general restrictions, restrictions for specific instructions, and how those specific restrictions vary for processor generations. See Accumulator Registers

for a description of the accumulator registers.

Accumulator registers can be accessed as explicit source or destination operands, as an implicit source value when specified for a particular instruction (*sada2* for example), and as an implicit destination when the AccWrEn instruction option is used.

These general rules apply to accumulator access:

1. Flow control, *send*, *sendc*, and *wait* instructions cannot use accumulators.
2. Instructions that specify an implicit accumulator destination (with AccWrEn) cannot specify an explicit accumulator destination operand.
3. Accumulator may be used in one of Src0 or Src1 only for MUL/MAC instruction.
4. Extra accumulator precision for integer is only supported on Src0.
5. Src2 cannot use accumulator for 3 source instructions.
6. INT2FLT format conversion cannot use accumulator as source.

These descriptions are frequently used in this table:

- No restrictions.
- No accumulator access, implicit or explicit.
- Source operands cannot be accumulators.
- Source modifier is not allowed if source is an accumulator.
- Accumulator is an implicit source and thus cannot be an explicit source operand.

- Accumulator cannot be destination, implicit or explicit.
- AccWrEn is required. The accumulator is an implicit destination and thus cannot be an explicit destination operand.

These minor cases occur occasionally in the table:

- Integer source operands cannot be accumulators.
- No explicit accumulator access because this is a three-source instruction. AccWrEn is allowed for implicitly updating the accumulator.
- An accumulator can be a source or destination operand but not both.

A few instructions use more than one of the listed restrictions.

Programming Note
Indirect address access is not allowed when dst is accumulator. Example - mul (16|M0) acc0.0<1>:w -r[a0.12,-256]<8,0>:w r39.10<32;16,0>:w

Accumulator Restrictions

Instructions	Accumulator Restrictions
add	No restrictions.
addc	AccWrEn is required. The accumulator is an implicit destination and thus cannot be an explicit destination operand. Accumulator is not allowed to be explicit source.
and	Source modifier is not allowed if source is an accumulator.
asr *avg*	No restrictions.
bfe *bfi1* *bfi2* *bfrev* *cbit*	No accumulator access, implicit or explicit.
cmp	No restrictions.
cmpn	No restrictions.
csel	No restrictions.
fbh *fbl*	No accumulator access, implicit or explicit.

Instructions	Accumulator Restrictions
frc	No restrictions.
line	Source operands cannot be accumulators.
lzd	Accumulator cannot be source, implicit or explicit.
mac	Integer source operands cannot be explicit accumulators.
mach	Accumulator is an implicit source and thus cannot be an explicit source operand. AccWrEn is required. The accumulator is an implicit destination and thus cannot be an explicit destination operand.
mad	Integer source operands cannot be accumulators.
madm	No explicit accumulator access because this is a three-source instruction. AccWrEn is allowed for implicitly updating the accumulator.
math	No accumulator access, implicit or explicit.
mov	An accumulator can be a source or destination operand but not both.
movi	Source operands cannot be accumulators.
mul	Integer source operands cannot be accumulators.
not *or*	Source modifier is not allowed if source is an accumulator.
rndd *rnde* *rndu* *rndz*	No restrictions.
rol	No restrictions.
ror	No restrictions.
sel	No restrictions.
shl	Accumulator cannot be destination, implicit or explicit.
shr	No restrictions.
smov	No restrictions.
subb	AccWrEn is required. The accumulator is an implicit destination and thus cannot be an explicit destination operand. Accumulator is not allowed to be explicit source.
xor	Source modifier is not allowed if source is an accumulator.

Native Instruction BNF

Describes Syntax supported by IGA.

Grammar

IGA Grammar

The following describes a super-set of the legal inputs to IGA. Many constructs below contain additional restrictions not represented syntactically (e.g. types on instructions).

IGA Grammar			
A program is a sequence of blocks, where the first block may not have a label			
Program	->	(*LabelDefLine*? *Instruction** (*LabelDefLine Instruction*))?	
LabelDefLine	->	*LabelDef LineTerminator*	
LabelRef	->	*Ident*	
LabelDef	->	*Ident* `:`	
InstructionLine	->	*Instruction LineTerminator*	
LineTerminator	->	`\n`	
	|	*EOF*	end of file
The rough layout of an instruction			
Instruction	->	*PredWrEn*? *Mnemonic ExecInfo FlagModifier*? *Operands InstOptions*?	
	|		Confer with the page on for these.
where			
Mnemonic	->	*Ident*	E.g. `mov` or `add`
	|	`math` . *MathFC*	E.g. math.inv
	|	`send` . *SFID*	E.g. send.dc0
e.g. sqt in math.sqt ... math function controls extend the opcode; e.g. math.inv ...			
MathFC	->	`inv`	
	|	`log`	
	|	`exp`	
	|	`sqt`	
	|	`rsqt`	
	|	`pow`	
	|	`sin`	
	|	`cos`	
	|	`idiv`	
	|	`iqot`	
	|	`irem`	

IGA Grammar			
	|	fdiv	
	|	invm	
	|	rsqtm	
e.g. dco in send.dco ... send function id determines what send type it is			
SFID	->	dc0	
	->	dc1	
	->	dc2	
	->	dcr0	
	->	gtwy	
	->	null	
	->	pixi	
	->	rc	
	->	smpl	
	->	ts	
	->	urb	
	->	vme	
	->	cre	
Execution mask part e.g. (16|M16)			
ExecInfo	->	(*ExecWidth* | *EMaskOff*)	
where			
ExecWidth	->	1 | 2 | 4 | 8 | 16 | 32	
the emask			
EMaskOff	->	M0 | M4 | ... | M28	
Instruction write-enable (formly NoMask) as well as predication Examples: (W), (f1.0), or (W~f1.1)			
PredWrEn	->	(*WrEn*) | (*Pred*) | (*WrEn* & *Pred*)	
where			
WrEn	->	W	

IGA Grammar			
Pred	->	*PredNegate*? *FlagRegRef PredMask*?	
PredNegate	->	~	
see PredCtrl enum 3D-Media-GPGPU Engine EU Overview ISA Introduction Instruction Set Reference EUISA Enumerations **PredCtrl**			
PredMask	->	. *PredFunc*	
where			
PredFunc	->	anyv	
	|	allv	
	|	any2h	
	|	all2h	
	|	any4h	
	|	all4h	
	|	any8h	
	|	all8h	
	|	any16h	
	|	all16h	
	|	any32h	
	|	all32h	
A condition modifier indicates the flag register should be updated with respect to the given relational function. E.g. (le)f0.0			
FlagModifier	->	(*Func*) *FlagRegRef*	
where			
Func	->	lt	less than
	|	le	less than than or equal
	|	gt	greater than
	|	ge	greater than or equal
	|	eq	equal
	|	ne	not equal
	|	ov	overflow
	|	un	unordered (NaN)
	|	gz	less than 0; only supported on sel instructions
	|	lz	greather than 0; only supported on sel instructions
Operands	->	*OperandsEmpty*	e.g. nop, illegal, ...

IGA Grammar			
	|	*OperandsUnary*	e.g. mov, not, ...
	|	*OperandsBinary*	e.g. and, add, mul, ..
	|	*OperandsTernary*	e.g. mad, csel, bfn, ...
	|	*OperandsSingleSource*	e.g. wait, jmpi, ...
	|	*OperandsDoubleSource*	e.g. break, cont, if, else (branch control is a sub function)
	|	*OperandsSend*	sends or sendsc
OperandsEmpty	->		
OperandsUnary	->	*DstOperand SrcOperand*	
OperandsBinary	->	*DstOperand SrcOperand SrcOperand*	
OperandsTernary	->	*DstOperand SrcOperand SrcOperand SrcOperand*	
OperandsSingleSource	->	*SrcOperand*	
OperandsDoubleSource	->	*SrcOperand SrcOperand*	
OperandsSend	->	*DstOperand SrcOperand SrcOperand ExDescOp DescOp*	
ExDescOp	->	*AddrRegRef* | *IntLit*	
DescOp	->	*AddrRegRef* | *IntLit*	
Destination operands			
DstOperand	->	*DstOpDirect*	direct register access; e.g. r14.3<1> or sr0.1
	|	*DstOpIndirect*	indirect access; e.g. r[a0.4,16]<1>
where			
DstOpDirect	->	*DstModif? Register DstRegion? DstType?*	
DstOpIndirect	-	*DstModif? r [AddrRegName .] DstRegion?*	

IGA Grammar			
	>		
DstModif	->	(sat)	saturation modifier
DstRegion	->	< *DstHzStride* >	
where			
DstHzStride	->	1 | 2 | 4	
Source operands -- Semantic checks for regioning disallow regions on certain registers (e.g. cannot region ce)			
SrcOperand	->	*SrcOpDirect*	direct register access; e.g. r13.0<8;8,1>
	|	*SrcOpIndirect*	indirect register access; e.g. r[a0.0,16]<1,0>
	|	*SrcOpImm*	source immediate operand; e.g. 13:ud, 0x155:q, 1.2:f
where			
SrcOpDirect	->	*SrcModifier*? *Register SrcRegion*? *SrcRegType*?	
SrcOpIndirect	->	*SrcModifier*? r [*IndexReg* (, *ConstExpr*)] *SrcRegion*? *SrcRegType*?	
SrcOpImm	->	*ConstExpr SrcImmType*	
We semantically enforced that ~ applies only to bitwise ops (e.g. and, or, ...)			
SrcModifier	->	~	bitwise negation for bitwise ops (one's complement)
	|	-	signed negation (two's complement)
	|	(abs)	absolute value
	|	- (abs)	negation of the absolute value
the source region			
SrcRegion	->	*SrcRegionVWH* | *SrcRegionWH* | *SrcRegionVH* | *SrcRegionH*	
SrcRegionVWH	->	< *VtStride* ; *Width* , *HzStride* >	
SrcRegionWH	->	< *Width* , *HzStride* >	for indirect only (uses VxH/Vx1 mode)
SrcRegionVH	->	< *VtStride* ; *HzStride* >	ternary src0 or src1 region
SrcRegionH	->	< *HzStride* >	ternary src2 region
VtStride	->	1	
	|	2	
	|	4	
	|	8	

IGA Grammar			
	|	16	
	|	32	the number of elements to advance the row after stepping Width times
Width	->	1	
	|	2	
	|	4	
	|	8	
	|	16	the number elements per row (before advancing vert stride)
HzStride	->	0	
	|	1	
	|	2	
	|	4	the number of elements to step horizontally between channels
Subregisters are in units of the type of the operand they are applied to. That is 8-byte types only permit subregisters 0-3, 4-byte types permit 0-7, and so forth.			
Register	->	*RegName* (. *SubReg*)?	
where			
See the EU section on ARF registers			
RegName	->	*Names*	
where			
Names	->	*AddrRegName*	address register
	|	acc0	
	|	…	
	|	acc11	
	|	*FlagRegName*	e.g. f0 or f1
	|	fc0	
	|	…	
	|	fc4	flow control registers
	|	ce	channel enable register
	|	cr0	
	|	ip	instruction pointer
	|	n0	
	|	n1	
	|	msg0	
	|	…	
	|	msg31	

IGA Grammar			
	|	null	the null register
	|	r0	general registers
	|	…	
	|	r255	
	|	sr0	
	|	sr1	
	|	tdr	
	|	tm0	
The subregister. Various registers support different number of subregisters. Subregister units are in terms of data type size. E.g. up to: .31 for :b, .15 for :w, .7 for :d, and .3 for :q etc ...			
SubReg	->	*IntLit*	
A reference to a flag register. Examples: f0.0 or f1.1			
FlagRegRef	->	*FlagRegName* (. *FlagSubReg*)?	
where			
FlagSubReg	->	0 | 1	
FlagRegName	->	f0 … f3	
An address register reference; e.g. a0.3 (sometimes called the "index register")			
AddrRegRef	->	*AddrRegName* (. *AddrSubReg*)?	
where			
AddrSubReg	->	0 | … | 15	
AddrRegName	->	a0	
Destination types are the allowable types on registers			
DstType	->	*RegType*	
Source types are the same as the destination types with the inclusion of the packed vector types.			
SrcImmType	->	*ImmType*	
SrcRegType	->	*RegType*	

IGA Grammar			
Register and immediate operand types. RegTypes includes non-vector immediates.			
RegType	->	`:b`	8b signed integer
	|	`:ub`	8b unsigned integer
	|	`:w`	16b signed integer
	|	`:uw`	16b unsigned integer
	|	`:d`	32b signed integer
	|	`:ud`	32b unsigned integer
	|	`:q`	64b signed integer
	|	`:uq`	64b unsigned integer
	|	`:hf`	IEEE-754 16b float
	|	`:f`	IEEE-754 32b float
	|	`:df`	IEEE-754 64b float
Immediate Operand Types			
ImmType	|	`:uv`	8x4b packed unsigned int
	|	`:v`	8x4b packed signed int
	|	`:w`	16b signed integer
	|	`:uw`	16b unsigned integer
	|	`:d`	32b signed integer
	|	`:ud`	32b unsigned integer
	|	`:q`	64b signed integer
	|	`:uq`	64b unsigned integer
	|	`:vf`	4x8b packed float
	|	`:hf`	IEEE 754 16b float
	|	`:f`	IEEE 754 32b float
	|	`:df`	IEEE 754 64b float
Expressions. Constant expressions are evaluated by the assembler with host semantics.			
ConstExpr	->	*IntLit*	integer literal
	|	*FltLit*	floating point literal
	|	*Ident*	reference to a symbolic constant
	|	*UnOp ConstExpr*	unary constant expressions (e.g. ~((1NUM_BITS)-1)
	|	*ConstExpr BinOp ConstExpr*	binary constant expressions e.g. 1 + 2
	|	(*ConstExpr*)	constant grouping expression (1 + 2)
	|	*LabelRef*	a reference to a label definition
Unary operators			
UnOp	->	~	one's complement

IGA Grammar			
	|	-	two's complement
Binary operators (in precedence order)			
BinOp	->	*	multiplicative operators
	|	/	
	|	%	
	|	+	additive operators
	|	-	
	|	«	bit-shift operators
	|	»	
	|	»>	shift with zero-fill
	|	&	bitwise (logical) operators
	|	^	
	|	|	
InstOptions	->	{ (*InstOpt* (, *InstOpt*)*)? }	
where			
InstOpt	->	AccWrEn	
	|	Atomic	
	|	Breakpoint	
	|	Compacted	
	|	EOT	
	|	*DepInfo*	
DepInfo	->	RegDist | SBIDAlloc | SBIDWaitSrc | SBIDWaitDst	
where			
RegDist	->	@*IntLit*	minimum distance on the current pipe
	|	F@*IntLit*	minimum distance on the float pipe
	|	I@*IntLit*	minimum distance on the integer pipe
	|	L@*IntLit*	minimum distance on the long pipe
SBIDAlloc	->	$*IntLit*	allocate a dependency barrier (SBID: (S)core(B)oard (ID)
SBIDWaitSrc	-	$*IntLit*.src	wait until sources are read (on the

IGA Grammar			
	>		instruction that allocated the SBID)
SBIDWaitDst	->	`$IntLit.dst`	wait until writeback completes (of the instruction that allocated the SBID)
Identifiers			
Ident	->	`[A-Za-z_][A-Za-z0-9_]*`	
An integer literal. For each, _ is an optional lexical separator for readability			
IntLit	->	*IntLitDec*	decimal literal e.g. 13, 14, 4000, or 4_000
	|	*IntLitHex*	hex literal e.g. 0xFE or 0xABCD_EFGH
	|	*IntLitBin*	binary literal e.g. 0b1011 would be 11
where			
IntLitDec	->	`[0-9][_0-9]*`	
IntLitHex	->	`0x[0-9a-fA-F]?[_0-9a-fA-F]*`	
IntLitBin	->	`0b[01][_01]*`	
A floating point literal			
FltLit	->	*FltLitDec*	decimal 3.1414
	|	*FltLitSci*	scientific 6.02e23 or 6.67384e-11
	|	`inf`	special symbol for infinity (0x...)
	|	`nan`	special symbol for not a number
where			
FltLitDec	->	`([0-9][_0-9]*)?.[0-9][_0-9]*|([0-9][_0-9]*)?.([0-9][_0-9]*)?`	
FltLitSci	->	`([0-9][_0-9]*)?.[0-9][_0-9]*e([0-9][_0-9]*)`	

Instruction Set Reference

This chapter describes the functions of 3D Media GPGPU Execution Units, listed in alphabetical order according to assembly language mnemonic.

EUISA Instructions List

Conventions

This section describes conventions used in instruction reference pages.

For each instruction that has source or destination types, a table lists the allowed type combinations and may also indicate the processor generations that support certain combinations. A notation like *W indicates that UW and W are both allowed. Multiple types listed together mean that any combination (Cartesian product) of the listed types is allowed.

If a source operand is floating-point, all source operands must have the same floating-point data type.

The Q and UQ types cannot be mixed with the B or UB types, neither as different source types nor as source type and destination type.

Accumulator restrictions are described in the Accumulator Restrictions section and also appear in instruction descriptions.

Pseudo Code Format

Instructions are explained in the following pseudo-code format that resembles the assembly instruction format.

```
[(pred)] opcode (exec_size) dst src0 [src1]
```

Square brackets "[ ]" indicate that a field is optional. Saturation modifiers and instruction options are omitted for simplicity.

General Macros and Definitions

INST_MIN_SIZE is defined as a constant of 8 bytes.

```
#define INST_MIN_SIZE  8  // Instruction minimum size in bytes (for the compact instruction format)
```

The floor function converts a floating point value to an integral floating point value. For a given floating point value, from its closest two integral float values, floor returns the one that is closer to negative infinity. For example, floor(1.3f) = 1.0f and floor(-1.3f) = -2.0f.

```
float floor(float g)
{
    return maximum(any integral float f: f <= g)
}
```

The Condition function takes the conditional signals {SN, ZR, OF, IN, NC} of result, generates a Boolean value according to a conditional evaluation controlled by the conditional modifier cmod, and returns the Boolean.

```
Bool Condition(result, cmod)
```

The ConditionNaN function takes the conditional signals {SN, ZR, OF, IN, NC, NS} of result, generates a Boolean value according to a conditional evaluation controlled by the conditional modifier cmod, and returns the Boolean. The only difference between Condition and ConditionNaN is that ConditionNaN uses the NS (NaN of the second source) signal.

```
Bool ConditionNaN(result, cmod)
```

The Jump function jumps the instruction sequence from the current instruction location by InstCount 8-byte units, where each 16-byte native instruction is two units and each 8-byte compact instruction is one unit. If InstCount is positive and greater than zero, is an unconditional jump forward. If InstCount is negative, is an unconditional jump backward. If InstCount is zero, IP stays on the current instruction in an infinite loop.

```
void Jump(int InstCount)
{
    IP = IP + (InstCount * INST_MIN_SIZE)
}
```

Evaluate Write Enable

The WrEn should be evaluated as below.

Note: MaskCtrl = NoMask (1) skips the check for PcIP[n] == ExIP before enabling a channel.

```
if ( MaskCtrl == 1 ) {
  for ( n = 0; n < exec_size; n++ ) {
    WrEn[n] = 1;
  }
}
else {
  for ( n = 0; n < exec_size; n++ ) {
    if ( PcIP[n] == ExIP ) {
      WrEn[n] = 1;
    }
    else {
      WrEn[n] = 0;
    }
  }
}

if ( PredCtrl != 0000b ) {
  for ( n = 0; n < exec_size; n++ ) {
    WrEn[n] = WrEn[n] & PMask[n];
  }
}

for ( n = exec_size; n < 32; n++ ) {
  WrEn[n] = 0;
}
```

EUISA Instructions

Symbol	Name	Source
add	**Addition**	EUISA
addc	**Addition with Carry**	EUISA
asr	**Arithmetic Shift Right**	EUISA
avg	**Average**	EUISA
bfe	**Bit Field Extract**	EUISA
bfi1	**Bit Field Insert 1**	EUISA
bfi2	**Bit Field Insert 2**	EUISA

Symbol	Name	Source
bfrev	**Bit Field Reverse**	EUISA
brc	**Branch Converging**	EUISA
brd	**Branch Diverging**	EUISA
break	**Break**	EUISA
call	**Call**	EUISA
calla	**Call Absolute**	EUISA
cmp	**Compare**	EUISA
cmpn	**Compare NaN**	EUISA
csel	**Conditional Select**	EUISA
cont	**Continue**	EUISA
cbit	**Count Bits Set**	EUISA
dp4a	**Dot Product 4 Accumulate**	EUISA
else	**Else**	EUISA
endif	**End If**	EUISA
math	**Extended Math Function** • INV - Inverse • LOG - Logarithm • EXP - Exponent • SQRT - Square Root • RSQ - Reciprocal Square Root • POW - Power Function • SIN - SINE • COS - COSINE • INT DIV - Integer Divide • INVM/RSQRTM	EUISA
fbl	**Find First Bit from LSB Side**	EUISA
fbh	**Find First Bit from MSB Side**	EUISA
frc	**Fraction**	EUISA
goto	**Goto**	EUISA
halt	**Halt**	EUISA
if	**If**	EUISA
illegal	**Illegal**	EUISA
join	**Join**	EUISA
jmpi	**Jump Indexed**	EUISA
lzd	**Leading Zero Detection**	EUISA

Symbol	Name	Source
and	**Logic And**	EUISA
not	**Logic Not**	EUISA
or	**Logic Or**	EUISA
xor	**Logic Xor**	EUISA
mov	**Move**	EUISA
movi	**Move Indexed**	EUISA
mul	**Multiply**	EUISA
mac	**Multiply Accumulate**	EUISA
mach	**Multiply Accumulate High**	EUISA
mad	**Multiply Add**	EUISA
nop	**No Operation**	EUISA
ret	**Return**	EUISA
rol	**Rotate Left**	EUISA
ror	**Rotate Right**	EUISA
rndd rnde rndu rndz	▪ **Round Instructions** ▪ **Round Down** ▪ **Round to Nearest or Even** ▪ **Round Up** ▪ **Round to Zero**	EUISA
sel	**Select**	EUISA
send	**Send Message**	EUISA
sendc	**Send Message Conditional**	EUISA
shl	**Shift Left**	EUISA
shr	**Shift Right**	EUISA
subb	**Integer Subtraction with Borrow**	EUISA
sync	**Synchronize**	EUISA
while	**While**	EUISA

Round Instructions

rndd - Round Down

rndu - Round Up

rnde - Round to Nearest or Even

rndz - Round to Zero

rndd - Round Down

Description:

The *rndd* instruction takes component-wise floating point downward rounding (to the integral float number closer to negative infinity) of src0 and storing the rounded integral float results in dst. This is commonly referred to as the floor() function.

Each result follows the rules in the following tables based on the floating-point mode.

Floating-Point Round Down in IEEE mode

src0	-inf	-finite	-denorm	-0	+0	+denorm	+finite	+inf	NaN
dst	-inf	-finite	^	-0	+0	+0	**	+inf	NaN
Notes:									
^	Note: {-1, -0} depending on the Single Precision Denorm Mode.								
**	Result may be {+finite, +0}.								

Floating-Point Round Down in ALT mode

src0	-fmax	-finite	-denorm	-0	+0	+denorm	+finite	+fmax	***
dst	-fmax	-finite	-0	-0	+0	+0	**	+fmax	
Notes:									
**	Result may be {+finite, +0}.								
***	Result is undefined if src0 is {-inf, +inf, NaN}.								

rnde - Round to Nearest or Even

Description:

The *rnde* instruction takes component-wise floating point round-to-even operation of src0 with results in two pieces - a downward rounded integral float results stored in dst and the round-to-even increments stored in the rounding increment bits. The round-to-even increment must be added to the results in dst to create the final round-to-even values to emulate the round-to-even operation, commonly known as the round() function. The final results are the one of the two integral float values that is nearer to the input values. If the neither possibility is nearer, the even alternative is chosen.

Each result follows the rules in the following tables based on the floating-point mode.

Floating-Point Round to Nearest or Even in IEEE mode

src0	-inf	-finite	-denorm	-0	+0	+denorm	+finite	+inf	NaN
dst	-inf	*	-0	-0	+0	+0	**	+inf	NaN
Notes:									
*	Result may be {-finite, -0}.								
**	Result may be {+finite, +0}.								

Floating-Point Round to Nearest or Even in ALT mode

src0	-fmax	-finite	-denorm	-0	+0	+denorm	+finite	+fmax	***
dst	-fmax	*	-0	-0	+0	+0	**	+fmax	
Notes:									
*	Result may be {-finite, -0}.								
**	Result may be {+finite, +0}.								
***	Result is undefined if src0 is {-inf, +inf, NaN}.								

rndu - Round Up

Description:

The *rndu* instruction takes component-wise floating point upward rounding (to the integral float number closer to positive infinity) of src0, commonly known as the ceiling() function.

Each result follows the rules in the following tables based on the floating-point mode.

Floating-Point Round Up in IEEE mode

src0	-inf	-finite	-denorm	-0	+0	+denorm	+finite	+inf	NaN
dst	-inf	*	-0	-0	+0	^	+finite	+inf	NaN
Notes:									
*	Result may be {-finite, -0}.								
^	Note: {+1, +0} depending on the Single Precision Denorm Mode.								

Floating-Point Round Up in ALT mode

src0	-fmax	-finite	-denorm	-0	+0	+denorm	+finite	+fmax	***
dst	-fmax	*	-0	-0	+0	+0	+finite	+fmax	
Notes:									
*	Result may be {-finite, -0}.								
***	Result is undefined if src0 is {-inf, +inf, NaN}.								

rndz - Round to Zero

Description:

The *rndz* instruction takes component-wise floating point round-to-zero operation of src0 with results in two pieces - a downward rounded integral float results stored in dst and the round-to-zero increments stored in the rounding increment bits. The round-to-zero increment must be added to the results in dst to create the final round-to-zero values to emulate the round-to-zero operation, commonly known as

the truncate() function. The final results are the one of the two closest integral float values to the input values that is nearer to zero.

Floating-Point Round to Zero in IEEE mode

src0	-inf	-finite	-denorm	-0	+0	+denorm	+finite	+inf	NaN
dst	-inf	*	-0	-0	+0	+0	**	+inf	NaN
Notes:									
*	Result may be {-finite, -0}.								
**	Result may be {+finite, +0}.								

Floating-Point Round to Zero in ALT mode

src0	-fmax	-finite	-denorm	-0	+0	+denorm	+finite	+fmax	***
dst	-fmax	*	-0	-0	+0	+0	**	+fmax	
Notes:									
*	Result may be {-finite, -0}.								
**	Result may be {+finite, +0}.								
***	Result is undefined if src0 is {-inf, +inf, NaN}.								

math - Extended Math Function

Function
math - Extended Math Function

Description:

The *math* instruction performs extended math function on the components in src0, or src0 and src1, and write the output to the channels of dst. The type of extended math function are based on the FC[3:0] encoding in the table below.

Function Control[3:0]	Function Description
0h	Reserved
1h	INV (reciprocal)
2h	LOG
3h	EXP
4h	SQT
5h	RSQT
6h	SIN
7h	COS
8h	Reserved
Bh	INT DIV - return quotient and remainder
Ch	INT DIV - return quotient only

Function Control[3:0]	Function Description
Dh	INT DIV - return remainder

INV - Inverse

Precision:1 ULP

LOG - Logarithm

Precision: If src0 is [0.5..2], **absolute error** must be no more than 2-21. If src0 is (0..0.5) or (2..+INF], **relative error** must be no more than 2-21

Note: In ALT mode log is computed as Log_2 (abs (src0))

EXP - Exponent

Precision: <= 3ULP

SQT - Square Root

Precision: <= 1ULP

Notes: In ALT mode SQRT is computed as SQRT(abs (src0))

RSQT - Reciprocal Square Root

Precision: <= 3ULP

Notes: In ALT mode RSQ is computed as RSQ(abs (src0))

SIN - SINE

Precision: Absolute error <= 0.0008 for the range of +/- 32767* pi

Outside of the above range the function will remain periodic, producing values between -1 and 1. However, the period of SIN is determined by the internal representation of Pi, meaning that as the magnitude of input increases the absolute error will, in general, also increase.

COS - COSINE

Precision: Absolute error <= 0.0008 for the range of +/- 32767* pi

Outside of the above range the function will remain periodic, producing values between -1 and 1. However, the period of COS is determined by the internal representation of Pi, meaning that as the magnitude of input increases the absolute error will, in general, also increase.

INT DIV - Integer Divide

Precision:32-bit integer

For signed inputs, INT DIV behavior is illustrated by the table below:

Inputs:	**Numerator**	+	+	-	-
	Denominator	+	-	+	-
Outputs:	**Quotient**	+	-	-	+
	Remainder	+	+	-	-

EUISA Structures

Name	Source
AddrSubRegNum	Eulsa
EU_INSTRUCTION_BASIC_ONE_SRC	Eulsa
EU_INSTRUCTION_BASIC_THREE_SRC	Eulsa
EU_INSTRUCTION_BASIC_TWO_SRC	Eulsa
EU_INSTRUCTION_BRANCH_ONE_SRC	Eulsa
EU_INSTRUCTION_BRANCH_TWO_SRC	Eulsa
EU_INSTRUCTION_ILLEGAL	Eulsa
EU_INSTRUCTION_NOP	Eulsa

EUISA Enumerations

Name	Source
AddrMode	Eulsa
ExecSize	Eulsa
FC	Eulsa
HorzStride	Eulsa
PredCtrl	Eulsa
SFID	Eulsa
SrcMod	Eulsa
VertStride	Eulsa
Width	Eulsa

EU Programming Guide

Describes they syntax, gives examples, and provides fulsim extensions.

Fulsim Extenstions

A register can be declared as a symbol using the following form

.declare <symbol>**Base**=RegFile RegBase {.SubRegBase} **ElementSize**=ElementSize {**SrcRegion**=DefaultSrcRegion} {**DstRegion**=DefaultDstRegion} {**Type**=DefaultType}

The register file, the base of the register origin and the element size (in unit of bytes) are the mandatory parameters for a declared register region. Optionally, the base of the sub-register address, the default source region, the default destination region and the default type can be provided in the declaration for the symbol.

For immediate register addressing mode, the declared symbol can be used in the following Cartesian form

<symbol>(RegOff, SubRegOff)<=RegNum = *RegBase*+ **RegOff***;* SubRegNum = *SubRegBase*+ **SubRegOff**

or in the following simplified row-aligned form

<symbol>(RegOff)<=RegNum = *RegBase*+ **RegOff;** SubRegNum = *SubRegBase*

Example Declare:

.declare uwPAKMBA Base=r48 ElementSize=2 Type=uw

.declare wINTRAMB_FLAG_B Base=r8.1 ElementSize=2 Type=w

Example Use:

and (1|M0) (eq)f0.0 wINTRAMB_FLAG_B(0)<1>:w uwPAKMBA(1,0)<0;1,0>:w 0x2000:w

.default_execution_size (EXEC_SIZE)

Where executions size is :1, 4, 8, 16

Example:

.default_execution_size (16)

mov (M0) r45.0<8;8,1>:d 0:d

.default_register_type <type>

Example:

.default_register_type :d

mov (16|M0) r45.0<8;8,1> 0

Usage Examples

Examples of various instructions use.

SEND Instruction

SFID	
cre	Check and Refinement Engine
dc0	Data Cache Data Port 0

SFID	
dc1	Data Cache Data Port 1
dc2	Data Cache Data Port 2
dcr0	Data Cache Read Only Data Port
gtwy	Message Gateway
null	Null
pixi	Pixel Interpolation
rc	Render Cache Data Port
smpl	Sampler
ts	Thread Spawner
ub	URB
vme	Video Motion Estimation
cre	Check and Refinement Engine

SFID	
cre	Check and Refinement Engine
dc0	Data Cache Data Port 0
dc1	Data Cache Data Port 1
dc2	Data Cache Data Port 2
dcr0	Data Cache Read Only Data Port
gtwy	Message Gateway
null	Null
pixi	Pixel Interpolation
rc	Render Cache Data Port
ts	Thread Spawner
ub	URB
vme	Video Motion Estimation
cre	Check and Refinement Engine

send.cre (8|M0) r3 r0 r38 0x82 0x02127019	
send	Send
cre	SFID
(8|M0)	ExecSize: 8. Excution Mask Offset: 0.
r3:f	Destination. GRF Register type. Register Number: 3. Type: Float.
r0	Source 0. GRF Register type. Register Number: 0. Type: None.
r38	Source 1. GRF Register type. Register NUmber: 38. Type: None.
0x82	Extended Message Descriptor. See SEND message for more information.

send.cre (8|M0) r3 r0 r38 0x82 0x02127019	
0x02127019	Message Descriptor. See SEND message for more information.
(W) send.cre (8|M0) null r25 null a0.2 a0.0 {EOT}	
(W)	Write Enable is ON.
send	Send
cre	SFID
(8|M0)	ExecSize: 8. Excution Mask Offset: 0.
null	Destination. ARF NULL Register. Type: UD. This instruction does not return any values.
r25	Source 0.
null	Source 1. Can be null.
a0.2	Extended Message Descriptor.
a0.0	Message Descriptor. Indirect. Value stored in ARF address register a0.0.
EOT	End Of Thread

EU Instructions

Execution Mask	
M0	N1/Q1/H1
M4	N2
M8	N3/Q2
M12	N4
M16	N5/Q3/H2
M20	N6
M24	N7/Q4
M28	N8
(W) add (1|M0) a0.0<1>:ud r1.0<0;1,0>:ud 0x2190000:ud { @2 }	
(W)	Write Enable is ON
add	opcode add
(1|M0)	Execution size: 1. Execution Mask Offset: 0.
a0.0<1>:ud	Architecture Register File (ARF) registers: address. Register Number: 0. SubRegister Number 0. Destination stride: 1. Type: UD.
r1.0<0;1,0>:ud	GRF Register. Register Number: 1. SubRegister Number 0. Scalar access.
@2	Register Dependancy Distance
madm (4|M0) r46.acc3:df r38.noacc:df r36.noacc:df r32.acc2:df Write Enable is OFF	
madm	opcode madm
(4|M0)	Execution size: 4. Execution Mask Offset: 0.
r46.acc3:df	GRF register. Register Number: 46. SubRegister Number: 0. ACC3 is updated to accumulate

Execution Mask	
	precision. See MADM for more information. Type: DF.
r38.noacc:df	GRF register. Register Number: 38. SubRegister Number: 0. No ACC register is upated. See Acumulator register for more information. Type DF.
r32.acc2:df	GRF register. Register Number: 32. SubRegister Number: 0. ACC2 is used for extra precision.
math.invm (4|M4) (eo)f0.0 r8.acc2:df r36.noacc:df r34.noacc:df Write enable is OFF	
math.invm	opcode MATH, Extended Math Function INVM (Inverse with higher precision).
(4|M0)	Execution Size: 4. Execution Mask Offset: 4:
(eo)f0.0	Flag register as implicit destination, with a flag modifier. See math instruction for more information.
cmp (1|M0) (lt)f0.0 null.0<1>:d r35.0<0;1,0>:d r1.7<0;1,0>:d Write enable is OFF	
cmp	Opcode compare
(1|M0)	Execution Size: 1. Execution Mask Offset: 0.
(lt)f0.0	Flag register as implicit destination, with Less Then flag modifier. See cmp instruction for more information.
r35.0<0;1,0>:d	GRF register type. Register Number: 35. SubRegister Number: 0. Region: <0;1,0>, Scalar. vStride: 0. Width: 1. hStride: 0.
sel (1|M0) (lt)f0.0 r2.4<1>:d r2.4<0;1,0>:d r1.2<0;1,0>:d Write enable is OFF	
sel	Opcode select.
(1|M0)	Execution Size: 1. Execution Mask Offset: 0.
(lt)f0.0	Flag register as implicit destination, with Lest Then flag modifier. See sel instruction for more information.
(W&~f0.0) sel (8|M0) r2.0<1>:f r2.0<8;8,1>:f 0xBF800000:f	
(W&~f0.0)	Write Enable is ON. This instruction is predicated. Flag register f0.0 is used, and it's values are negated. See select instruction for more information.
mad (8|M0) r15.0<1>:f r14.0<0;0>:f r3.0<8;1>:f r10.0<0>:f { $2 } Write enable is OFF	
mad	Opcode: mad.
(8|M0)	Execution Size: 8. Execution Mask Offset: 0.
r15.0<1>:f	Destination. Horizontal Stride: 1. Type: Float.
r14.0<0;0>:f	Src0. vStride: 0. hStride: 0. Scalar.
r3.0<8;1>:f	Src1. vStride: 1. hStride: 1. Vector.
r10.0<0>:f	Src2. hStride: 0. Scalar
$2	one of the source register is dependent on a send instruction using token $2
mad (8|M0) r15.0<1>:f r14.0<0;0>:f r3.0<8;1>:f r10.0<1>:f	
r10.0<1>:f	Src2. hStride: 1. Vector.

SEND Instructions

Sends (8|M0) r3:f r0 r38 0x82 0x02127019	
Sends	Split Send. Two Sources.
(8|M0)	ExecSize: 8. Excution Mask Offset: 0.
r3:f	Destination. GRF Register type. Register Number: 3. Type: Float.
r0	Source 0. GRF Register type. Register Number: 0. Type: None.
r38	Source 1. GRF Register type. Register NUmber: 38. Type: None.
0x82	Extended Message Descriptor. See SEND message for more information.
0x02127019	Message Descriptor. See SEND message for more information.
Send (8|M0) r0:f r37:d 0x2 0x08127019	
Send	Send instruction. One Source.
(8|M0)	ExecSize: 8. Excution Mask Offset: 0.
r0:f	Destination. GRF Register Type. Register Number: 0. Type: Float.
r37:d	Source 0. GRF Register Type. Register Number: 0. Type: D.
0x2	Message Descriptor.
0x08127019	Extended Message Descriptor.
Send (16|M0) r3:w r23:ud 0x2 a0.0	
Send	Send instruction. One source.
(16|M0)	ExecSize: 16. Excution Mask Offset: 0.
r3:w	Destination.
r23:ud	Source 0.
0x2	Extended Message Descriptor
a0.0	Message Descriptor. Indirect. Value stored in ARF address register a0.0.
(W) sends (8|M0) null:ud r25 r19 0x10c a0.0	
(W)	Write Enable is ON.
sends	Split Send.
(8|M0)	ExecSize: 8. Excution Mask Offset: 0.
null:ud	Destination. ARF NULL Register. Type: UD. This instruction does not return any values.
r25	Source 0.
r19	Source 1.
0x10C	Extended Message Descriptor.
a0.0	Message Descriptor. Indirect. Value stored in ARF address register a0.0.

Control Flow Instructions

(W) jmpi (1|M0) L1320	
(W)	Write Enable is ON.
jmpi.	Jump Immediate.
(1|M0)	Execution Size: 1. Execution Mask Offset: 0.
L130	Label to jump to.
(W&~f0.1) jmpi (1|M0) L904	
(W&~f0.1)	Write Enable is ON. ARF flag register used as Predicate. SubRegister: 1. Values are inverted.
(~f0.0) if (4|M0) L4744 L4760 // (4|M0) instructions L4744: // (4|M0) instructions L4760: endif (4|M0) L4776 L4776:	
(~f0.0)	ARF Flag register used as a predicate. SubRegister: 0. Values are inverted.
if	IF
(4|M0)	Execution Size: 4. Execution Mask Offset: 0.
L4744	JIP
L4760	UIP
L4776	JIP

Shared Functions

Shared Functions are hardware units that perform operations on behalf a thread running in an EU. Shared Functions are sent a message payload by the thread, and optionally return results to the thread.

The most common shared function operations are memory surface read and write messages performed by the Sampler, the Pixel Port, and the Data Port. The following sections describe each of the shared function operations in detail.

The most common shared function operations are memory surface read and write messages performed by the Pixel Port and the Data Port. The following sections describe each of the shared function operations in detail.

Shared functions are identified by their Shared Function ID (SFID), which is one of the parameters to the EU send instruction.

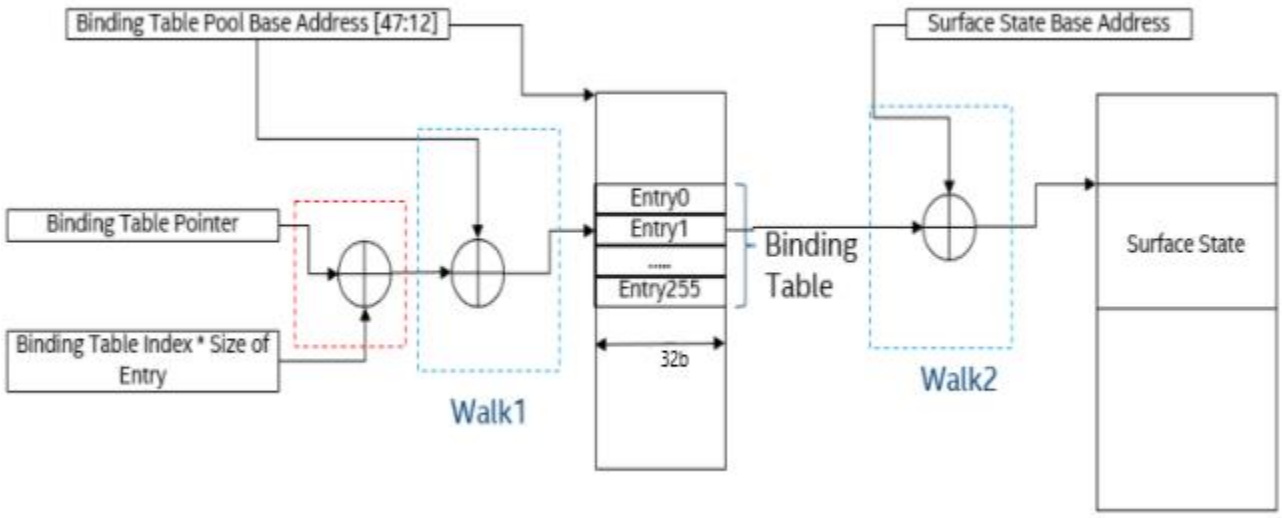
List of Shared Function Identifiers

Programming Note	
Context:	Shared Function ID
SFID_DP_DC1 is an extension of SFID_DP_DC0 to allow for more messages types. They act as a single logical entity.	

Programming Note	
Context:	Shared Function ID
SFID_DP_DC1 , SFID_DP_DC2, and SFID_P_DC3 are extensions of SFID_DP_DC0 to allow for more messages types. They act as a single logical entity.	

Binding Table

The surface state model is intended to be used for constant buffers, render target surfaces, and media surfaces.



Below is a table specifying the different configurations of the Binding Table Pointers and the BINDING_TABLE_STATE. This table applies for render, position and compute engines. Each engine supports a separate address.

Binding Table Entry Format	Binding Table Pointer	Base Address
32 bit size/[31:6]	20:5	Binding Table Pool Address

Below is a table of the different entries and pointer offsets based on the Binding Table Pool and 18:8 Mode:

Render Pipe:

RCS Binding Table Pool Enable	HW Alignment 256B Mode	Binding Table Entry Format	Binding Table Pointer	Base Address
0	0	32 bit size/[31:6]	15:5	RCS Surface State Base Address
1	0	16 bit size/[21:6]	16:6	RCS Binding Table Pool Address
0	1	32 bit size/[31:6]	18:8	RCS Surface State Base Address
1	1	32 bit size/[31:6]	18:8	RCS Binding Table Pool Address

POSH Pipe(Note that the Pool Enable and 256B mode is shared with Render Pipe):

RCS Binding Table Pool Enable*	HW Alignment 256B Mode*	Binding Table Entry Format	Binding Table Pointer	Base Address
0	0	32 bit size/[31:6]	15:5	POCS Surface State Base Address
1	0	16 bit size/[21:6]	16:6	POCS Binding Table Pool Address
0	1	32 bit size/[31:6]	18:8	POCS Surface State Base Address
1	1	32 bit size/[31:6]	18:8	POCS Binding Table Pool Address

* SW must sync between the POCS and RCS prior to changing the binding table pool enable and 256B alignment mode.

Compute Pipe:

Binding Table Pool Enable	Binding Table Entry Format	Binding Table Pointer	Base Address
0	32 bit size/[31:6]	15:5	CCS Surface State Base Address
1	16 bit size/[21:6]	16:6	CCS Binding Table Pool Address

The data port uses the binding table to retrieve surface state when the Binding Table Index is less than 240. See the following for a definition of the binding table:

- **SW Generated BINDING_TABLE_STATE**
- **HW Generated BINDING_TABLE_STATE**

Fused Send Message Handling

Shared Functions receive messages from the EU. The EU kernel executes a SEND instruction to transmit the message payload to the Shared Function. The message payload contains a Message Descriptor, an optional Message Header, and a message-specific Address and Data Payload.

The EU can mark two SEND messages to be combined together as a single operation.

Combine Type	Behavioral Description and Restrictions
Atomic Send Message	A single thread issues back-to-back SEND messages. The two messages are sent as an indivisible pair to the shared function. The shared function processes the pair of messages sequentially. There is no guarantee that the two messages are executed back-to-back by the shared function: another EU's send message may execute between the two messages.
Fused Send Message	A fused-EU pair of threads issues a SEND message. The two messages are sent as an indivisible pair to the shared function. The full extended message descriptor and the Message Header payload are required to be the same for the paired messages. Only the Address and Data payload are independent for the paired messages. The shared function processes the paired messages simultaneously.
Atomic Fused Send Message	When combined with an Atomic Send, the "atomic chained" messages from an EU is sent first, before the messages from other EU of the fused-pair is sent. Atomic Send is only allowed for URB writes messages, and a maximum of 2 such send messages can be chained.

3D Sampler

The 3D Sampling Engine provides the capability of advanced sampling and filtering of surfaces in memory.

The sampling engine function is responsible for providing filtered texture values to the EU in response to sampling engine messages. The sampling engine uses SAMPLER_STATE to control filtering modes, address control modes, and other features of the sampling engine. A pointer to the sampler state is delivered with each message, and an index selects one of 16 states pointed to by the pointer. Some messages do not require SAMPLER_STATE. In addition, the sampling engine uses SURFACE_STATE to define the attributes of the surface being sampled. This includes the location, size, and format of the surface as well as other attributes.

Although data is commonly used for "texturing" of 3D surfaces, the data can be used for any purpose once returned to the execution core. The 3D Sampler can be used to assist the media sampler in specific operations such as video scaling.

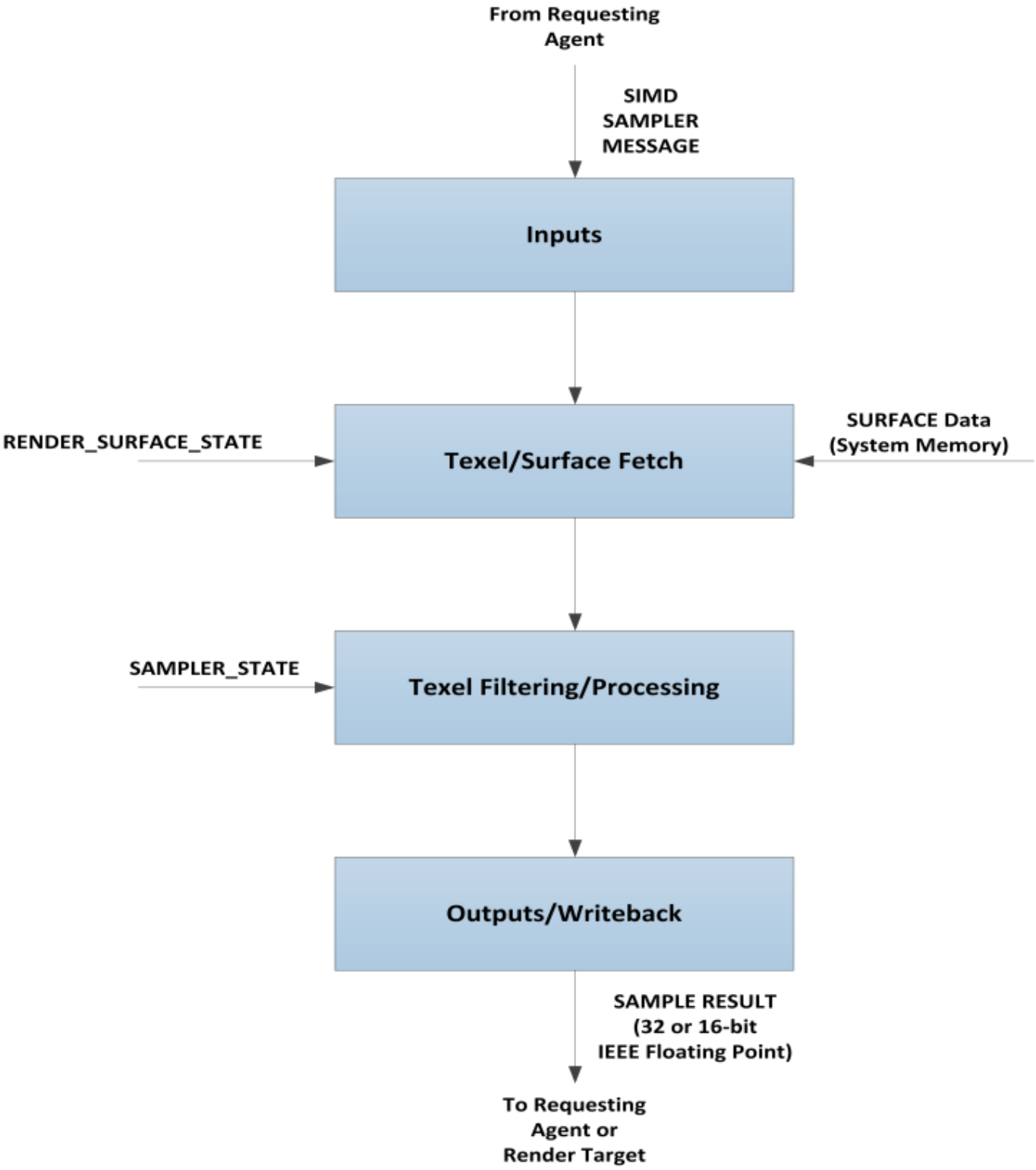
The following table summarizes the various subfunctions provided by the Sampling Engine. After the appropriate subfunctions are complete, the 4-component (reduced to fewer components in some cases) filtered texture value is provided to the EU to complete the *sample* instruction.

Subfunction	Description
Texture Coordinate Processing	Any required operations are performed on the incoming pixel's interpolated internal texture coordinates. These operations may include cube map intersection.
Texel Address Generation	The Sampling Engine determines the required set of texel samples (specific texel values from specific texture maps), as defined by the texture map parameters and filtering modes. This includes coordinate wrap/clamp/mirror control, mipmap LOD computation and sample and/or miplevel weighting factors to be used in the subsequent filtering operations.
Texel Fetch	The required texel samples are read from the texture map. This step may require decompression of texel data. The texel sample data is converted to an internal format.
Texture Palette Lookup	For streams which have "paletted" texture surface formats, this function uses the "index" values read from the texture map to look up texel color data from the texture palette.
Shadow Pre-Filter Compare	For shadow mapping, the texel samples are first compared to the 3rd (R) component of the pixel's texture coordinate. The boolean results are used in the texture filter.
Texel Filtering	Texel samples are combined using the filter weight coefficients computed in the Texture Address Generation function. This "combination" ranges from simply passing through a "nearest" sample to blending the results of anisotropic filters performed on two mipmap levels. The output of this function is a single 4-component texel value.
Texel Color Gamma Linearization	Performs optional gamma decorrection on texel RGB (not A) values.
8x8 Video Scaler	Performs scaling using an 8x8 filter.

Sampling Engine

3D Sampler Theory of Operation

The 3D sampler (sometimes referred to as texture sampler) is a self-contained functional block within the Graphics Core which receives messages from other agents in the Graphics Core, fetches data from external memory sources typically referred to as "surfaces", performs operations on the data and returns the results in standard formats to the requester (or directly to a Render Target is requested). One of the most common applications of the 3D sampler is to return a filtered/blended pixel from a location in a texture map.


From Requesting Agent
SIMD SAMPLER MESSAGE
Inputs
RENDER_SURFACE_STATE
Texel/Surface Fetch
SURFACE Data (System Memory)
SAMPLER_STATE
Texel Filtering/Processing
Outputs/Writeback
SAMPLE RESULT (32 or 16-bit IEEE Floating Point)
To Requesting Agent or Render Target

Sampler Inputs Messages

Input requests to the 3D Sampler are in the form of messages (see Messages sub-section for a description of message types and formats). A pixel shader kernal executing on the Graphics Core is an example of an agent which is capable of sending sample messages to the 3D Sampler.

In its most basic form, the sampler receives coordinates to a location within a field of data (often a texture map or depth map) and returns a value which represents the pixel color or depth which may be filtered/blended as defined by associated surface and sampler state objects. Sampler can also work on un-typed data structures called buffers.

Messages are sent in SIMD (Single Instruction Multiple Data) format where there are 8, 16, 32 or 64 coordinate tuples to be processed (i.e. SIMD8, SIMD16 etc.) in the same manner. Some message types are restricted to SIMD8 and SIMD16 varieties and other are restricted to SIMD32 or SIMD64. See the section on Texture Coordinate Processing more details on texture coordinate requirements.

SIMD8 and SIMD16 messages are further organized into groups of 4 sets of coordinates which generally form a 2x2 "subspan" of texel locations. The spatial locality of the texel locations within a sub-span improves the performance of the sampler and allows the processing of the 4 texel locations in parallel. A SIMD8 message contains two subspans and a SIMD16 contains 4 subspans.

Sampler Data Fetches

The 3D sampler will automatically fetch required data from surfaces in system memory as needed to perform each sample operation. Fetched data may be stored in an internal cache to reduce latency for subsequent fetch operations.

The sampler calculates the address into a surface and uses RENDER_SURFACE_STATE state objects to determine the location within system memory and the format of the surface being fetched. Sampler can also receive or calculate the LOD (Level of Detail) of a surface if the surface supports multiple Mips and will fetch from the correct Mip in this case. See Texture Address Calculation sub-section for more detail on addresses and LOD calculation.

The sampler will also automatically decompress any supported compression format once data has been fetched. See the subsection Surface State for a list of supported surface formats, including compressed formats. Likewise, the sampler can linearize (inverse Gamma) sRGB formats prior to filtering.

Sampler Filtering and Processing

The sampler is capable of performing all basic filtering operations (point, bilinear, trilinear, anisotropic, cube etc.) based on the SAMPLER_STATE state object associated with the sample operation being requested.

In most cases, data returned is in the form of 32-bit or 16-bit IEEE floating-point per channel to ensure maximum precision. See Writeback Message section for a description of the format of returned data. Output Data is only returned to the requesting agent or written to a designated Render Target (RT). Sample results are never cached within the sampler or written to system memory.

Sampler Caching Policies

Sampler State and Texture Caches

The 3D sampler caches both texture and state to improve performance. These read-only caches are referred to as the L1 Texture cache and L1 State cache. The L1 State cache

contains both RENDER_SURFACE_STATE and SAMPLER_STATE entries. The L1 Texture cache contains data from any surface which was sampled. It is assumed that this state and texture data, when fetched, is also cached in the L3 cache for performance reasons.

Cache Replacement and Retention Policy

Both the L1 State and L1 Texture cache will replace a cached entry only when no invalid entries are available. A line is only replaced when no on-going sample operations are using the information of that cache-entry. If all entries are currently being used, the sampler will automatically wait until one or more lines become available. Replaced lines, since they are read-only, are invalidated.

Coherency Mechanisms

Coherency of these caches to memory must be maintained by software. There are 3 mechanisms by which the contents of the L1 State and/or L1 Texture cache are invalidated. The three possible sources for invalidation are:

- Command Streamer: Command stream can issue invalidations for state and texture independently.
- System: In cases where sampler is referencing memory which is coherent with system memory it is possible to receive an invalidation due to a snoop/invalidation from an external agent.

  This will invalidate BOTH L1 State and L1 Texture caches.
- Shader: The CACHE_FLUSH message can be generated by the shader which will invalidate BOTH the L1 State and L1 Texture caches.

From the Command Streamer it is possible to invalidate either L1 State or L1 Texture caches independently. However, invalidating only state may lead to unexpected behavior where old texture data related to an old surface or sampler state are used. It is strongly recommended that a Texture cache invalidation be done whenever a State cache invalidation is done.

See the section on 3D Sampler State for more details on how state is referenced in memory and specific programming notes.

Sampler SW Performance Recommendations

This is a section describing things that SW can do to get better performance out of sampler hardware.

I,L,LA,A formats

These are expanded to RGBA formats in sampler and as result are treated as larger texel sizes in sampler and have worse performance. Most of the time it is better to use the Shader Channel Select signals in RENDER_SURFACE_STATE to emulate them using R or RG formats.

		Channel select			
Base format	New format	R	G	B	A
I*	R*	R	R	R	R
L*	R*	R	R	R	1.0
LA*	RG*	R	R	R	G
A*	R*	0.0	0.0	0.0	R

Issues with this where result can be different. The A channel for LA and A format will be 0.0 rather than 1.0.

The table below shows the mapping requires for all Luminance, Intensity and Transparency formats:

Original Format	Mapped To Format	Channel select Programming (R/G/B/A)
A16_FLOAT	R16_FLOAT	0/0/0/R
A16_UNORM	R16_UNORM	0/0/0/R
A24X8_UNORM	R24_UNORM_X8_TYPELESS	0/0/0/R
A32_FLOAT	R32_FLOAT	0/0/0/R
A32_UNORM	R32_UNORM	0/0/0/R
A8_UNORM	R8_UNORM	0/0/0/R
I16_FLOAT	R16_FLOAT	R/R/R/R
I16_UNORM	R16_UNORM	R/R/R/R
I24X8_UNORM	R24_UNORM_X8_TYPELESS	R/R/R/R
I32_FLOAT	R32_FLOAT	R/R/R/R
I8_SINT	R8_SINT	R/R/R/R
I8_UINT	R8_UINT	R/R/R/R
I8_UNORM	R8_UNORM	R/R/R/R
L16_FLOAT	R16_FLOAT	R/R/R/1
L16_UNORM	R16_UNORM	R/R/R/1
L16A16_FLOAT	R16G16_FLOAT	R/R/R/G
L16A16_UNORM	R16G16_UNORM	R/R/R/G
L24X8_UNORM	R24_UNORM_X8_TYPELESS	R/R/R/1
L32_FLOAT	R32_FLOAT	R/R/R/1
L32_UNORM	R32_UNORM	R/R/R/1
L32A32_FLOAT	R32G32_FLOAT	R/R/R/G
L8_SINT	R8_SINT	R/R/R/1
L8_UINT	R8_UINT	R/R/R/1
L8_UNORM	R8_UNORM	R/R/R/1
L8A8_SINT	R8G8_SINT	R/R/R/G
L8A8_UINT	R8G8_UINT	R/R/R/G
L8A8_UNORM	R8G8_UNORM	R/R/R/G
Y8_UNORM	R8_UNORM	R/R/R/1

Original Format	Mapped To Format	Channel select Programming (R/G/B/A)
A16_FLOAT	R16_FLOAT	0/0/0/R
A16_UNORM	R16_UNORM	0/0/0/R
A24X8_UNORM	R24_UNORM_X8_TYPELESS	0/0/0/R
A32_FLOAT	R32_FLOAT	0/0/0/R
A32_UNORM	R32_UNORM	0/0/0/R
A8_UNORM	R8_UNORM	0/0/0/R
I16_FLOAT	R16_FLOAT	R/R/R/R
I16_UNORM	R16_UNORM	R/R/R/R
I24X8_UNORM	R24_UNORM_X8_TYPELESS	R/R/R/R
I32_FLOAT	R32_FLOAT	R/R/R/R
I8_SINT	R8_SINT	R/R/R/R
I8_UINT	R8_UINT	R/R/R/R
I8_UNORM	R8_UNORM	R/R/R/R
L16_FLOAT	R16_FLOAT	R/R/R/1
L16_UNORM	R16_UNORM	R/R/R/1
L16A16_FLOAT	R16G16_FLOAT	R/R/R/G
L16A16_UNORM	R16G16_UNORM	R/R/R/G
L24X8_UNORM	R24_UNORM_X8_TYPELESS	R/R/R/1
L32_FLOAT	R32_FLOAT	R/R/R/1
L32_UNORM	R32_UNORM	R/R/R/1
L32A32_FLOAT	R32G32_FLOAT	R/R/R/G
L8_SINT	R8_SINT	R/R/R/1
L8_UINT	R8_UINT	R/R/R/1
L8_UNORM	R8_UNORM	R/R/R/1
L8A8_SINT	R8G8_SINT	R/R/R/G
L8A8_UINT	R8G8_UINT	R/R/R/G
L8A8_UNORM	R8G8_UNORM	R/R/R/G
Y8_UNORM	R8_UNORM	R/R/R/1

Tile Mode

If in TILEMODE_LINEAR and the base_address is NOT cacheline aligned there will be a very large performance hit in addition to the one that is already on linear.

Write Channel Mask

A header is no longer needed to specify a channel mask other than all enabled. If head is disabled the response length is used to determine a default channel mask. See **message format** section for more details.

NativeL1 Fill rate improvement SW requirements

Due to hardware restrictions, the L1 fill rate improvements added in by these features fill is affected by Surface Horizontal Alignment. For each texel size in the L1 the following table shows the minimum value for Surface Horizontal Alignment to get full performance. Lower values do not cause a function issue but will slow down performance.

Texel size in L1	Min Horizontal Alignment
16	16
24	8
32	8
48+	4

Texture Coordinate Processing

The Texture Coordinate Processing function of the Sampling Engine performs any operations on the texture coordinates that are required before physical addresses of texel samples can be generated.

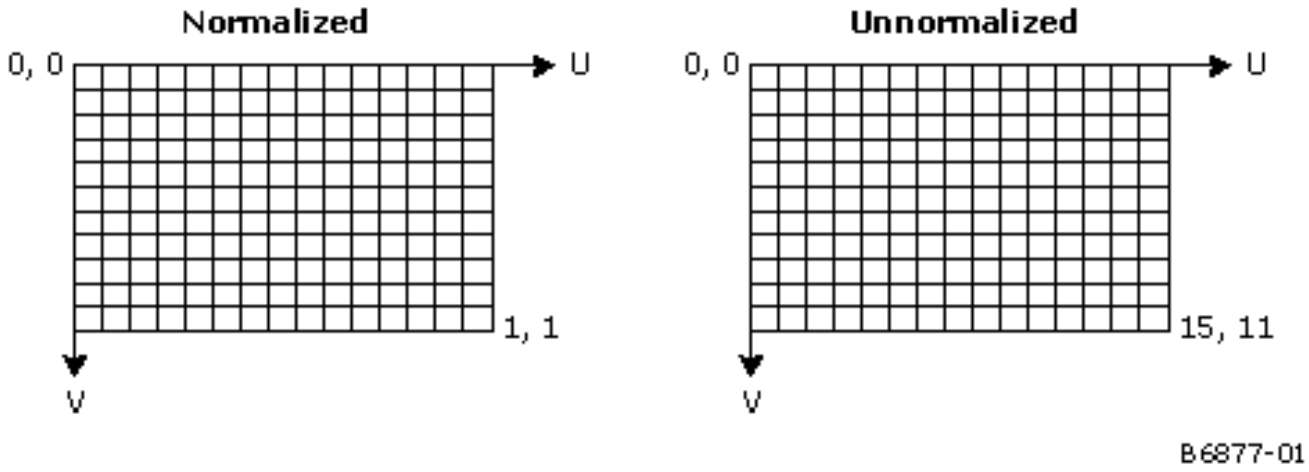
Texture Coordinate Normalization

A texture coordinate may have *normalized* or *unnormalized* values. In this function, unnormalized coordinates are normalized.

Normalized coordinates are specified in units relative to the map dimensions, where the origin is located at the upper/left edge of the upper left texel, and the value 1.0 coincides with the lower/right edge of the lower right texel . 3D rendering typically utilizes normalized coordinates.

Unnormalized coordinates are in units of texels and have not been divided (normalized) by the associated map's height or width. Here the origin is the located at the upper/left edge of the upper left texel of the base texture map.

Normalized vs. Unnormalized Texture Coordinates



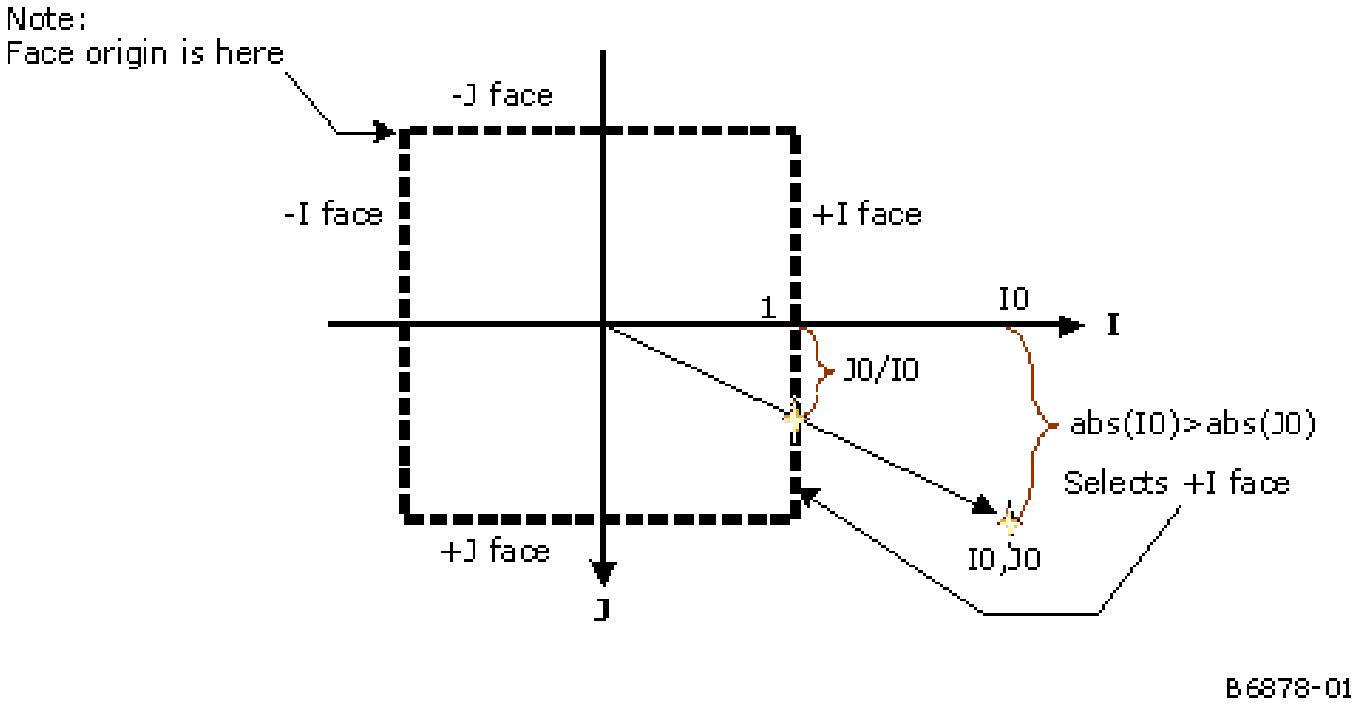
Texture Coordinate Computation

Cartesian (2D) and homogeneous (projected) texture coordinate values are projected from (interpolated) screen space back into texture coordinate space by dividing the pixel's S and T components by the Q component. This operation is done prior to sending sample operations to the 3D sampler.

Vector (cube map) texture coordinates are generated by first determining which of the 6 cube map faces (+X, +Y, +Z, -X, -Y, -Z) the vector intersects. The vector component (X, Y or Z) with the largest absolute value determines the proper (major) axis, and then the sign of that component is used to select between the two faces associated with that axis. The coordinates along the two minor axes are then divided by the coordinate of the major axis, and scaled and translated, to obtain the 2D texture coordinate ([0,1]) within the chosen face. Note that the coordinates delivered to the sampling engine must already have been divided by the component with the largest absolute value.

An illustration of this cube map coordinate computation, simplified to only two dimensions, is provided below:

Cube Map Coordinate Computation Example



Texel Address Generation

To better understand texture mapping, consider the mapping of each object (screen-space) pixel onto the textures images. In texture space, the pixel becomes some arbitrarily sized and aligned quadrilateral. Any given pixel of the object may "cover" multiple texels of the map, or only a fraction of one texel. For each pixel, the usual goal is to sample and filter the texture image in order to best represent the covered texel values, with a minimum of blurring or aliasing artifacts. Per-texture state variables are provided to allow the user to employ quality/performance/footprint tradeoffs in selecting how the particular texture is to be sampled.

The Texel Address Generation function of the Sampling Engine is responsible for determining how the texture maps are to be sampled. Outputs of this function include the number of texel to be fetched, along with the physical addresses of the samples and the filter weights to be applied to the samples after they are read. This information is computed given the incoming texture coordinate and gradient values, and the relevant state variables associated with the sampler and surface. This function also applies the texture coordinate address controls when converting the sample texture coordinates to map addresses.

Level of Detail Computation (Mipmapping)

Due to the specification and processing of texture coordinates at object vertices, and the subsequent object warping due to a perspective projection, the texture image may become *magnified* (where a texel covers more than one pixel) or *minified* (a pixel covers more than one texel) as it is mapped to an object. In the case where an object pixel is found to cover multiple texels (texture minification), merely choosing one (e.g., the texel sample nearest to the pixel's texture coordinate) will likely result in severe aliasing artifacts.

Mipmapping and texture filtering are techniques employed to minimize the effect of undersampling these textures. With mipmapping, software provides *mipmap levels,* a series of pre-filtered texture maps of decreasing resolutions that are stored in a fixed (monolithic) format in memory. When mipmaps are provided and enabled, and an object pixel is found to cover multiple texels (e.g., when a textured object is located a significant distance from the viewer), the device will sample the mipmap level(s) offering a texel/pixel ratio as close to 1.0 as possible.

The device supports up to 14 mipmap levels per map surface, ranging from 16384 x 16384 texels to a 1 X 1 texel. Each successive level has 1/2 the resolution of the previous level in the U and V directions (to a minimum of 1 texel in either direction) until a 1x1 texture map is reached. The dimensions of mipmap levels need not be a power of 2.

Each mipmap level is associated with a *Level of Detail (LOD)* number. LOD is computed as the approximate, $\log_2$ measure of the ratio of texels per pixel. The highest resolution map is considered LOD 0. A larger LOD number corresponds to lower resolution mip level.

The *Sampler[]BaseMipLevel* state variable specifies the LOD value at which the minification filter vs. the magnification filter should be applied.

When the texture map is magnified (a texel covers more than one pixel), the base map (LOD 0) texture map is accessed, and the magnification mode selects between the nearest neighbor texel or bilinear interpolation of the 4 neighboring texels on the base (LOD 0) mipmap.

Base Level Of Detail (LOD)

The per-pixel LOD is computed in an implementation-dependent manner and approximates the $\log_2$ of the texel/pixel ratio at the given pixel. The computation is typically based on the differential texel-space distances associated with a one-pixel differential distance along the screen x- and y-axes. These texel-space distances are computed by evaluating neighboring pixel texture coordinates, these coordinates being in units of texels on the base MIP level (multiplied by the corresponding surface size in texels). The q coordinates represent the third dimension for 3D (volume) surfaces, this coordinate is a constant 0 for 2D surfaces.

The ideal LOD computation is included below.

$$LOD(x,y) = \log_2[\rho(x,y)]$$

where :

$$\rho(x,y) = \max\left\{\sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial q}{\partial x}\right)^2}, \sqrt{\left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial q}{\partial y}\right)^2}\right\},$$

LOD Bias

A biasing offset can be applied to the computed LOD and used to artificially select a higher or lower miplevel and/or affect the weighting of the selected mipmap levels. Selecting a slightly higher mipmap

level will trade off image blurring with possibly increased performance (due to better texture cache reuse). Lowering the LOD tends to sharpen the image, though at the expense of more texture aliasing artifacts.

The LOD bias is defined as sum of the *LODBias* state variable and the *pixLODBias* input from the input message (which can be non-zero only for sample_b messages). The application of LOD Bias is unconditional, therefore these variables must both be set to zero in order to prevent any undesired biasing.

Note that, while the LOD Bias is applied prior to clamping and min/mag determination and therefore can be used to control the min-vs-mag crossover point, its use has the undesired effect of actually changing the LOD used in texture filtering.

LOD Pre-Clamping

The LOD Pre-Clamping function can be enabled or disabled via the *LODPreClampEnable* state variable.

After biasing and/or adjusting of the LOD , the computed LOD value is clamped to a range specified by the (integer and fractional bits of) *MinLOD* and *MaxLOD* state variables prior to use in Min/Mag Determination.

MaxLOD specifies the lowest resolution mip level (maximum LOD value) that can be accessed, even when lower resolution maps may be available. Note that this is the only parameter used to specify the number of valid mip levels that be can be accessed, i.e., there is no explicit "number of levels stored in memory" parameter associated with a mip-mapped texture. All mip levels from the base mip level map through the level specified by the integer bits of *MaxLOD* must be stored in memory, or operation is UNDEFINED.

MinLOD specifies the highest resolution mip level (minimum LOD value) that can be accessed, where LOD==0 corresponds to the base map. This value is primarily used to deny access to high-resolution mip levels that have been evicted from memory when memory availability is low.

MinLOD and *MaxLOD* have both integer and fractional bits. The fractional parts will limit the inter-level filter weighting of the highest or lowest (respectively) resolution map. For example if *MinLOD* is 4.5 and *MipFilter* is LINEAR, LOD 4 can contribute only up to 50% of the final texel color.

Min/Mag Determination

The biased and clamped LOD is used to determine whether the texture is being minified (scaled down) or magnified (scaled up).

The *BaseMipLevel* state variable is subtracted from the biased and clamped LOD. The *BaseMipLevel* state variable therefore has the effect of selecting the "base" mip level used to compute Min/Mag Determination. (This was added to match OpenGL semantics). Setting *BaseMipLevel* to 0 has the effect of using the highest-resolution mip level as the base map.

If the biased and clamped LOD is non-positive, the texture is being magnified, and a single (high-resolution) miplevel will be sampled and filtered using the *MagFilter* state variable. At this point the computed LOD is reset to 0.0. Note that LOD Clamping can restrict access to high-resolution miplevels.

If the biased LOD is positive, the texture is being minified. In this case the *MipFilter* state variable specifies whether one or two mip levels are to be included in the texture filtering, and how that (or those) levels are to be determined as a function of the computed LOD.

LOD Computation Pseudocode

This section illustrates the LOD biasing and clamping computation in pseudocode, encompassing the steps described in the previous sections. The computation of the initial per-pixel LOD value *LOD* is not shown.

```
 Bias:       S4.8
 MinLod:     U4.8
 MaxLod:     U4.8
 Base:       U4.1
 MIPCnt:     U4
 SurfMinLod: U4
 ResMinLod:  U4.8

PerSampleMinLOD: float32

 MinLod        = max(MinLod, PerSampleMinLOD)
 AdjMaxLod     = min(MaxLod, MIPCnt)
 AdjMinLod     = min(MinLod, MIPCnt)
 AdjPR_minLOD  = ResMinLod - SurfMinLod
 AdjMinLod     = max(AdjMinLod, AdjPR_minLOD)
 Out_of_Bounds = AdjPR_minLOD > MIPCnt

 if ( sample_b )
     LOD += Bias + bias_parameter
 else if ( sample_l or ld )
     LOD = Bias + lod_parameter
 else
     LOD += Bias

     PreClamp = LODPreClampMode != PRECLAMP_NONE
     if ( PreClamp )
         if ( PRECLAMP_D3D )
             LOD = min(LOD, AdjMaxLod)
             LOD = max(LOD, AdjMinLod)
         else
             LOD = min(LOD, MaxLod)
             LOD = max(LOD, MinLod)


 MagMode = (LOD - Base <= 0)

MagClampMipNone = LODClampMagnificationMode == MAG_CLAMP_MIPNONE

 if ( (MagMode && MagClampMipNone) or MipFlt == None )
     LOD = 0
     LOD = min(LOD, ceil(AdjMaxLod))
     LOD = max(LOD, floor(AdjMinLod))
 else if ( MipFlt == Nearest )

LOD = min(LOD, ceil(AdjMaxLod))
 LOD = max(LOD, floor(AdjMinLod))
```

```
    LOD += 0.5
    LOD = floor(LOD)
else
    // MipFlt = Linear
    LOD = min(LOD, AdjMaxLod)
    LOD = max(LOD, AdjMinLod)
    TriBeta = frac(LOD)
    LOD0 = floor(LOD)
    LOD1 = LOD0 + 1

if ( ! lod )  // "LOD" message type
    Lod += SurfMinLod
```

If Out_of_Bounds is true, LOD is set to zero and instead of sampling the surface the texels are replaced with zero in all channels, except for surface formats that don't contain alpha, for which the alpha channel is replaced with one. These texels then proceed through the rest of the pipeline.

Intra-Level Filtering Setup

Depending on whether the texture is being minified or magnified, the *MinFilter* or *MagFilter* state variable (respectively) is used to select the sampling filter to be used within a mip level (intra-level, as opposed to any inter-level filter). Note that for volume maps, this selection also applies to filtering between layers.

The processing at this stage is restricted to the selection of the filter type, computation of the number and texture map coordinates of the texture samples, and the computation of any required filter parameters. The filtering of the samples occurs later on in the Sampling Engine function.

The following table summarizes the intra-level filtering modes.

Sampler[]Min/MagFilter value	Description
MAPFILTER_NEAREST	Supported on all surface types. The texel nearest to the pixel's U,V,Q coordinate is read and output from the filter.
MAPFILTER_LINEAR	Not supported on buffer surfaces. The 2, 4, or 8 texels (depending on 1D, 2D/CUBE, or 3D surface, respectively) surrounding the pixel's U,V,Q coordinate are read and a linear filter is applied to produce a single filtered texel value.
MAPFILTER_ANISOTROPIC	Not supported on buffer or 3D surfaces. A projection of the pixel onto the texture map is generated and "subpixel" samples are taken along the major axis of the projection (center axis of the longer dimension). The outermost subpixels are weighted according to closeness to the edge of the projection, inner subpixels are weighted equally. Each subpixel samples a bilinear 2x2 of texels and the results are blended according to weights to produce a filtered texel value.
MAPFILTER_MONO	Supported only on 2D surfaces. This filter is only supported with the monochrome (MONO8) surface format. The monochrome texel block of the specified size surrounding the pixel is selected and filtered.

MAPFILTER_NEAREST

When the MAPFILTER_NEAREST is selected, the texel with coordinates nearest to the pixel's texture coordinate is selected and output as the single texel sample coordinates for the level. This is a form of "Point Sampling".

Corner Texel Mode

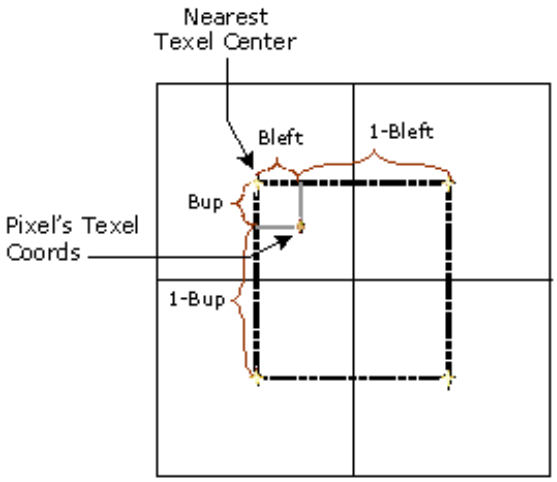
When **Corner Texel Mode** is enabled in the RENDER_SURFACE_STATE the definition of how the texel location is calculated is modifed. Corner Texel Mode shifts the edge of the map to be coincident with the center of the edge texels. The 3D Sampler will handle this shift when sampling the texel.

MAPFILTER_LINEAR

The following description indicates behavior of the MIPFILTER_LINEAR filter for 2D and CUBE surfaces. 1D and 3D surfaces follow a similar method but with a different number of dimensions available.

When the MAPFILTER_LINEAR filter is selected on a 2D surface, the 2x2 region of texels surrounding the pixel's texture coordinate are sampled and later bilinearly filtered. The filter weights each texel's contribution according to its distance from the pixel center. Texels further from the pixel center receive a smaller weight.

Bilinear Filter Sampling

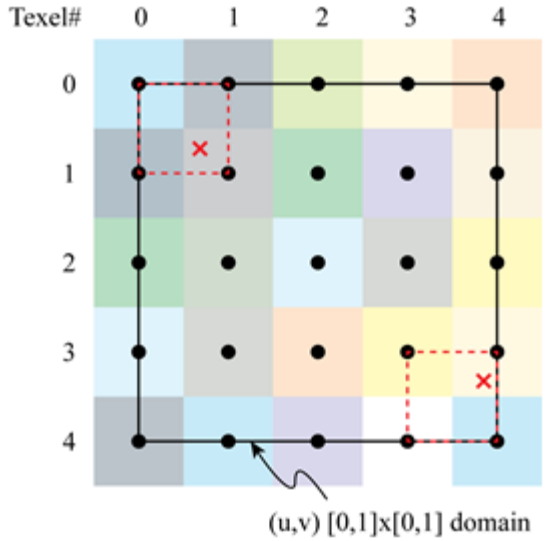


B6879-01

Corner Texel Mode

The Corner Texel Mode bit in surface state changes the linear filtering, by shifting the edges of the map to the center of the texel. Effectively, this means

that the edge of the map no coincides with the center of the texel, hence the name "Corner Texel Mode". The diagram below shows this.



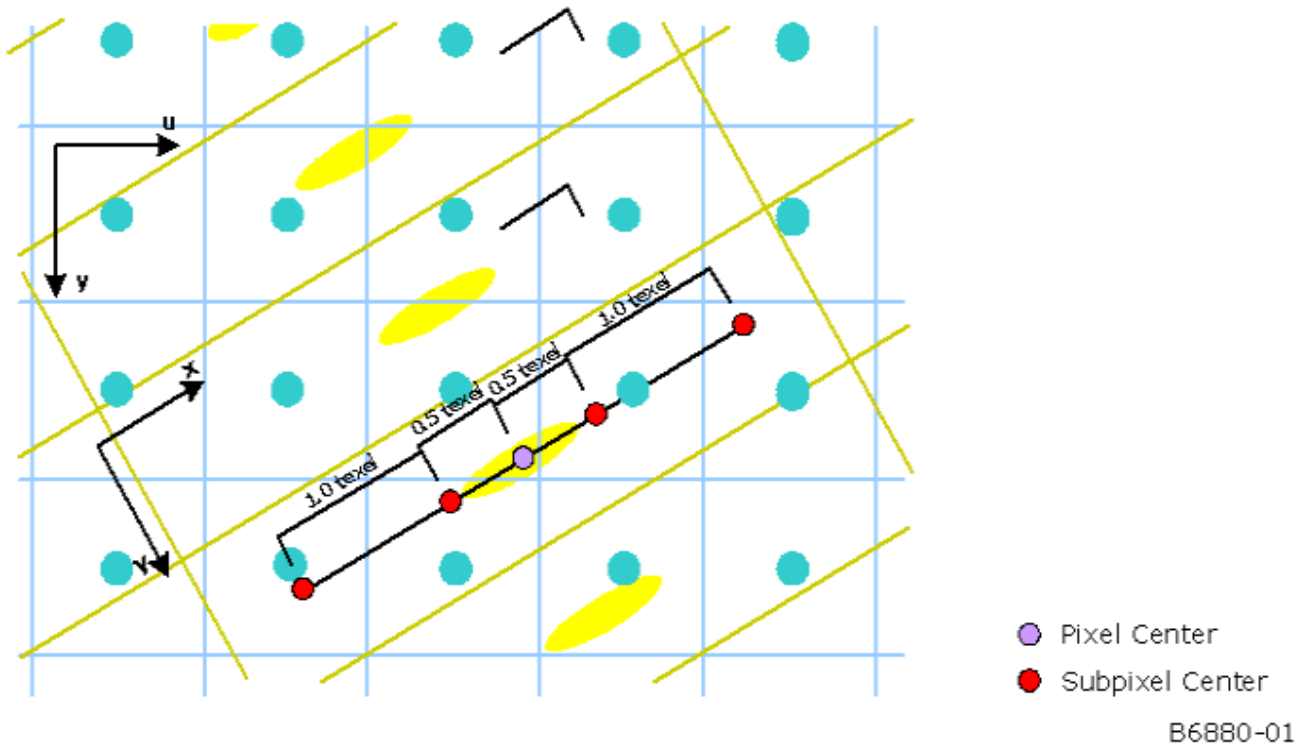
When doing linear filtering, the 3D sampler will treat a surface as being 1 texel less in each dimension when calculating texel location from the sample coordinates.

The benefit of this mode is that it is guaranteed that all the texels which contribute to the linear filter of a sample will be on the map if the sample point is on the map.

MAPFILTER_ANISOTROPIC

The MAPFILTER_ANISOTROPIC texture filter attempts to compensate for the anisotropic mapping of pixels into texture map space. A possibly non-square set of texel sample locations will be sampled and later filtered. The *MaxAnisotropy* state variable is used to select the maximum aspect ratio of the filter employed, up to 16:1.

The algorithm employed first computes the major and minor axes of the pixel projection onto the texture map. LOD is chosen based on the minor axis length in texel space. The anisotropic "ratio" is equal to the ratio between the major axis length and the minor axis length. The next larger even integer above the ratio determines the anisotropic number of "ways", which determines how many subpixels are chosen. A line along the major axis is determined, and "subpixels" are chosen along this line, spaced one texel apart, as shown in the diagram below. In this diagram, the texels are shown in light blue, and the pixels are in yellow.



Each subpixel samples a bilinear 2x2 around it just as if it was a single pixel. The result of each subpixel is then blended together using equal weights on all interior subpixels (not including the two endpoint subpixels). The endpoint subpixels have lesser weight, the value of which depends on how close the "ratio" is to the number of "ways". This is done to ensure continuous behavior in animation.

Inter-Level Filtering Setup

The *MipFilter* state variable determines if and how texture mip maps are to be used and combined. The following table describes the various mip filter modes:

MipFilter Value	Description
MIPFILTER_NONE	Mipmapping is DISABLED. Apply a single filter on the highest resolution map available (after LOD clamping).
MIPFILTER_NEAREST	Choose the nearest mipmap level and apply a single filter to it. Here the biased LOD will be rounded to the nearest integer to obtain the desired miplevel. LOD Clamping may further restrict this miplevel selection.
MIPFILTER_LINEAR	Apply a filter on the two closest mip levels and linear blend the results using the distance between the computed LOD and the level LODs as the blend factor. Again, LOD Clamping may further restrict the selection of miplevels (and the blend factor between them).

When minifying and MIPFILTER_NEAREST is selected, the computed LOD is rounded to the nearest mip level.

When minifying and MIPFILTER_LINEAR is selected, the fractional bits of the computed LOD are used to generate an inter-level blend factor. The LOD is then truncated. The mip level selected by the truncated LOD, and the next higher (lower resolution) mip level are determined.

Regardless of *MipFilter* and the min/mag determination, all computed LOD values (two for MIPFILTER_LINEAR, otherwise one) are then unconditionally clamped to the range specified by the (integer bits of) *MinLOD* and *MaxLOD* state variables.

Texture Address Control

The *[TCX,TCY,TCZ]ControlMode* state variables control the access and/or generation of texel data when the specific texture coordinate component falls *outside* of the normalized texture map coordinate range [0,1).

The table below provides all the supported Address Control modes for each direction.

TC[X,Y,Z] Control	Operation
TEXCOORDMODE_CLAMP	Clamp to the texel value at the edge of the map.
TEXCOORDMODE_CLAMP_BORDER	Use the texture map's border color for any texel samples falling outside the map. The border color is specified via a pointer in SAMPLER_STATE.
TEXCOORDMODE_HALF_BORDER	Similar to CLAMP_BORDER except texels outside of the map are clamped to a value halfway between the edge texel and the border color.
TEXCOORDMODE_WRAP	Upon crossing an edge of the map, repeat at the other side of the map in the same dimension.
TEXCOORDMODE_CUBE	Only used for cube maps. Here texels from adjacent cube faces can be sampled along the edges of faces. This is considered the highest quality mode for cube environment maps.
TEXCOORDMODE_MIRROR	Similar to the wrap mode, though reverse direction through the map each time an edge is crossed. INVALID for use with unnormalized texture coordinates.
TEXCOORDMODE_MIRROR_ONCE	Similar to the wrap mode, though reverse direction through the map each time an edge is crossed. INVALID for use with unnormalized texture coordinates.
TEXCOORDMODE_MIRROR_101	Similar to MIRROR_ONCE in that it only reflects once in each direction, the difference is that it will skip the first pixel of the reflected image.

Separate controls are provided for texture TCX, TCY, TCZ coordinate components so, for example, the TCX coordinate can be wrapped while the TCY coordinate is clamped. Note that there are no controls provided for the TCW component as it is only used to scale the other 3 components before addressing modes are applied.

Programming Note	
Context:	Texture Address Control
TEXCOORDMODE_CUBE can only be used with SURFTYPE_CUBE	

Maximum Wraps/Mirrors

The number of map wraps on a given object is limited to 32. Going beyond this limit is legal, but may result in artifacts due to insufficient internal precision, especially evident with larger surfaces. Precision

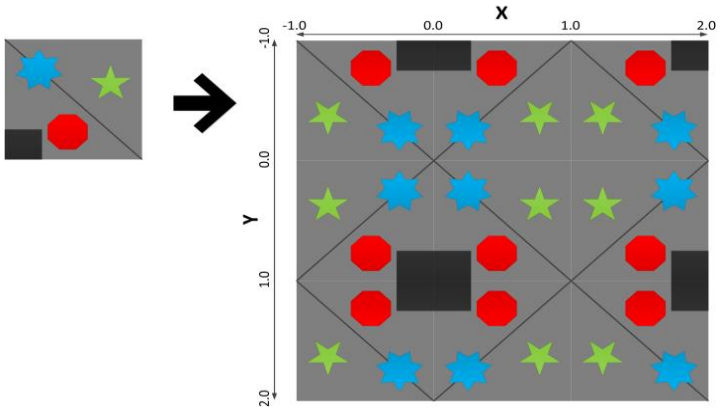
loss starts at the subtexel level (slight color inaccuracies) and eventually reaches the texel level (choosing the wrong texels for filtering).

Note: For **Wrap Shortest** mode, the setup kernel has already taken care of correctly interpolating the texture coordinates. Software needs to specify TEXCOORDMODE_WRAP mode for the sampler that is provided with wrap-shortest texture coordinates, or artifacts may be generated along map edges.

TEXCOORDMODE_MIRROR Mode

TEXCOORDMODE_MIRROR addressing mode is similar to Wrap mode, though here the base map is flipped at every integer junction. For example, for U values between 0 and 1, the texture is addressed normally, between 1 and 2 the texture is flipped (mirrored), between 2 and 3 the texture is normal again, and so on. The second row of pictures in the figure below indicate a map that is mirrored in one direction and then both directions. You can see that in the mirror mode every other integer map wrap the base map is mirrored in either direction.

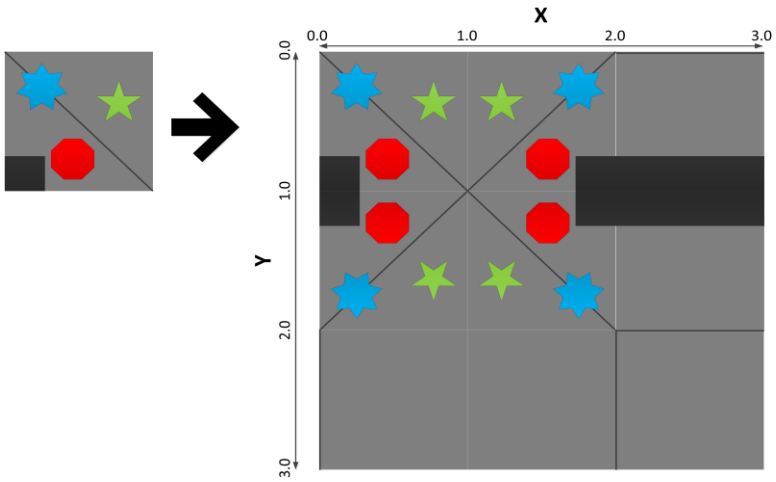
The example below shows how a simple 2D map with TEXCOORDMODE_MIRROR for both TCX and TCY is mapped.



TEXCOORDMODE_MIRROR_ONCE Mode

The TEXCOORDMODE_MIRROR_ONCE addressing mode is a combination of Mirror and Clamp modes. The absolute value of the texture coordinate component is first taken (thus mirroring about 0), and then the result is clamped to 1.0. The map is therefore mirrored once about the origin, and then clamped thereafter. This mode is used to reduce the storage required for symmetric maps.

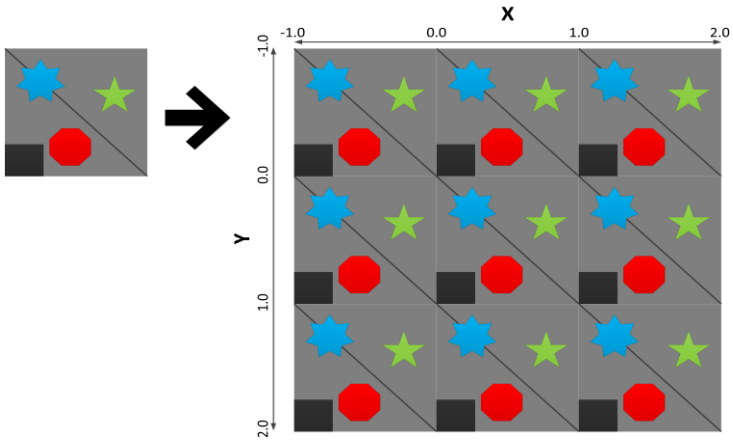
The example below shows how a simple 2D map with TEXCOORDMODE_MIRROR_ONCE for both TCX and TCY is mapped.



TEXCOORDMODE_WRAP Mode

In TEXCOORDMODE_WRAP addressing mode, the integer part of the texture coordinate is discarded, leaving only a fractional coordinate value. This results in the effect of the base map ([0,1)) being continuously repeated in all (axes-aligned) directions. Note that the interpolation between coordinate values 0.1 and 0.9 passes through 0.5 (as opposed to WrapShortest mode which interpolates through 0.0).

The example below shows how a simple 2D map with TEXCOORDMODE_WRAP for both TCX and TCY is mapped.

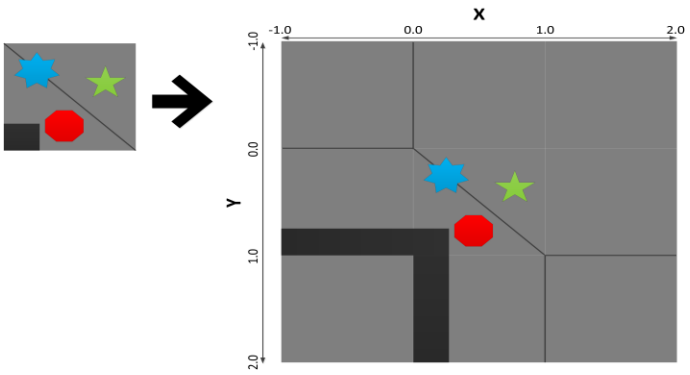


TEXCOORDMODE_CLAMP Mode

The TEXCOORDMODE_CLAMP addressing mode repeats the "edge" texel when the texture coordinate extends outside the [0,1) range of the base texture map. This is contrasted to TEXCOORDMODE_CLAMPBORDER mode which defines a separate texel value for off-map samples. TEXCOORDMODE_CLAMP is also supported for cube maps, where texture samples will only be obtained from the intersecting face (even along edges).

The figure below illustrates the effect of clamp mode. The base texture map is shown, along with a texture mapped object with texture coordinates extending outside of the base map region.

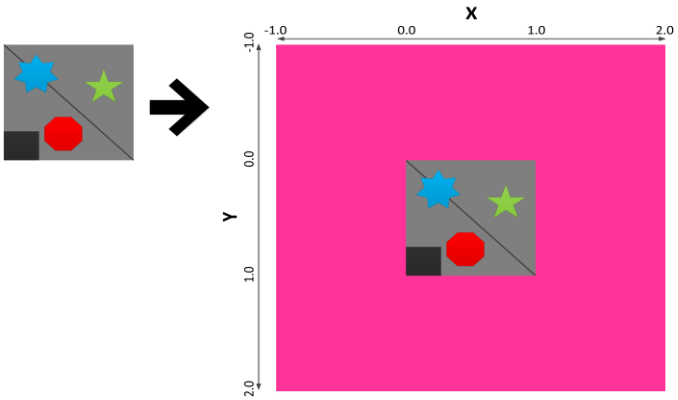
Texture Clamp Mode



TEXCOORDMODE_CLAMPBORDER Mode

For non-cube map textures, TEXCOORDMODE_CLAMPBORDER addressing mode specifies that the texture map's border value *BorderColor* is to be used for any texel samples that fall outside of the base map. The border color is specified via a pointer in SAMPLER_STATE.

The example below shows how a simple 2D map with TEXCOORDMODE_CLAMPBORDER for both TCX and TCY is mapped.



TEXCOORDMODE_HALF_BORDER Mode

For non-cube map textures, TEXCOORDMODE_HALF_BORDER addressing mode is similar to the TEXCOORDMODE_CLAMPBORDER in that it specifies that the texture map's border value *BorderColor*. However this value is blended with the edge texel color and used for any texel samples that fall outside of the base map. The border color is specified via a pointer in SAMPLER_STATE.

TEXCOORDMODE_CUBE Mode

For cube map textures TEXCOORDMODE_CUBE addressing mode can be set to allow inter-face filtering. When texel sample coordinates that extend beyond the selected cube face (e.g., due to intra-level filtering near a cube edge), the correct sample coordinates on the adjoining face will be computed. This will eliminate artifacts along the cube edges, though some artifacts at cube corners may still be present.

TEXCOORDMODE_MIRROR_101

The MIRROR_101 address control mode is similar to the MIRROR_ONCE. The difference is that it will skip the first pixel of the reflected image. Effectively, this means that the reflected images (in both directions) are one pixel smaller in size.

Programming Note	
Context:	MIRROR_101 and Quilted Surfaces
MIRROR_101 is not supported for Quilted Surfaces.	

Programming Note	
Context:	Fast Anisotropic Filtering
MIROR_101 Texture Coordinate Mode cannot be used with Anisotropic Filtering.	

Texel Fetch

The Texel Fetch function of the Sampling Engine reads the texture map contents specified by the texture addresses associated with each texel sample. The texture data is read either directly from the memory-resident texture map, or from internal texture caches. The texture caches can be invalidated by the **Sampler Cache Invalidate** field of the MI_FLUSH instruction or via the **Read Cache Flush Enable** bit of PIPE_CONTROL. Except for consideration of coherency with CPU writes to textures and rendered textures, the texture cache does not affect the functional operation of the Sampling Engine pipeline.

When the surface format of a texture is defined as being a compressed surface, the Sampler will automatically decompress from the stored format into the appropriate [A]RGB values. The compressed texture storage formats and decompression algorithms can be found in the *Memory Data Formats* chapter. When the surface format of a texture is defined as being an index into the texture palette (format names including "Px"), the palette lookup of the index determines the appropriate RGB values.

Texel Chroma Keying

ChromaKey is a term used to describe a method of effectively removing or replacing a specific range of texel values from a map that is applied to a primitive, e.g., in order to define transparent regions in an RGB map. The Texel Chroma Keying function of the Sampling Engine pipeline conditionally tests texel samples against a "key" range, and takes certain actions if any texel samples are found to match the key.

Chroma Key Testing

ChromaKey refers to testing the texel sample components to see if they fall within a range of texel values, as defined by *ChromaKey[][High,Low]* state variables. If each component of a texel sample is found to lie within the respective (inclusive) range and ChromaKey is enabled, then an action will be taken to remove this contribution to the resulting texel stream output. Comparison is done separately on each of the channels and only if all 4 channels are within range the texel will be eliminated.

The Chroma Keying function is enabled on a per-sampler basis by the *ChromaKeyEnable* state variable.

The *ChromaKey[][High,Low]* state variables define the tested color range for a particular texture map.

Chroma Key Effects

There are two operations that can be performed to "remove" matching texel samples from the image. The *ChromaKeyEnable* state variable must first enable the chroma key function. The *ChromaKeyMode* state variable then specifies which operation to perform on a per-sampler basis.

The *ChromaKeyMode* state variable has the following two possible values:

KEYFILTER_KILL_ON_ANY_MATCH: Kill the pixel if any contributing texel sample matches the key.

KEYFILTER_REPLACE_BLACK: Here the sample is replaced with (0,0,0,0).

The Kill Pixel operation has an effect on a pixel only if the associated sampler is referenced by a sample instruction in the pixel shader program. If the sampler is not referenced, the chroma key compare is not done and pixels cannot be killed based on it.

Shadow Prefilter Compare

When a *sample_c* message type is processed, a special shadow-mapping precomparison is performed on the texture sample values prior to filtering. Specifically, each texture sample value is compared to the "ref" component of the input message, using a compare function selected by *ShadowFunction*, and described in the table below. Note that only single-channel texel formats are supported for shadow mapping, and so there is no specific color channel on which the comparison occurs.

ShadowFunction	Result
PREFILTEROP_ALWAYS	0.0
PREFILTEROP_NEVER	1.0
PREFILTEROP_LESS	(texel < ref) ? 0.0 : 1.0
PREFILTEROP_EQUAL	(texel == ref) ? 0.0 : 1.0
PREFILTEROP_LEQUAL	(texel <= ref) ? 0.0 : 1.0
PREFILTEROP_GREATER	(texel > ref) ? 0.0 : 1.0
PREFILTEROP_NOTEQUAL	(texel != ref) ? 0.0 : 1.0
PREFILTEROP_GEQUAL	(texel >= ref) ? 0.0 : 1.0

The binary result of each comparison is fed into the subsequent texture filter operation (in place of the texel's value which would normally be used).

Software is responsible for programming the "ref" component of the input message such that it approximates the same distance metric programmed in the texture map (e.g., distance from a specific light to the object pixel). In this way, the comparison function can be used to generate "in shadow" status for each texture sample, and the filtering operation can be used to provide soft shadow edges.

Texel Filtering

The Texel Filtering function of the Sampling Engine performs any required filtering of multiple texel values on and possibly between texture map layers and levels. The output of this function is a single texel color value.

The state variables *MinFilter*, *MagFilter*, and *MipFilter* are used to control the filtering of texel values. The *MipFilter* state variable specifies how many mipmap levels are included in the filter, and how the results of any filtering on these separate levels are combined to produce a final texel color. The *MinFilter* and *MagFilter* state variables specify how texel samples are filtered within a level.

Texel Color Gamma Linearization

This function is supported to allow pre-gamma-corrected texel RGB (not A) colors to be mapped back into linear (gamma=1.0) gamma space prior to (possible) blending with and writing to the Color Buffer. This permits higher quality image blending by performing the blending on colors in linear gamma space.

This function is enabled on a per-texture basis by use of a surface format with "_SRGB" in its name. If enabled, the pre-filtered texel RGB color to be converted to gamma=1.0 space by applying a ^(2.4) exponential function.

Multisampled Surface Behavior

Multisampled surfaces are sampled using a dedicated point-sample message **ld2dms** or **l2dms_w**. These messages contain Si (sample index) which is used to sample from a specific subsample plane. Each texel of a multi-sampled surface has 1,2,4,8 or 16 "subsamples" per texel.

The sampleinfo message returns specific parameters associated with a multisample surface such as the number of subsamples per texel. The resinfo message returns the height, width, depth (in units of *pixels*, not samples), and MIP count of the surface.

Multisampled surfaces typically have an associated MCS control surface which contains information about which indicates which sample index contains the color data for a specific subample. The control surface can be accessed using the **ld_mcs** message, which returns the information for all subsamples for a given texel.

From an optimization perspective there are two different approaches which can be used to resolver a multisampled surface:

A shader can fetch ld_mcs to determine which subsamples need to be fetched using ld2dms or ld2dms_w. In general, it is expected that most subsamples will reference Sample index zero for it's color by default. It may be optimal for the shader to always fetch to sample index 0 unconditionally, and only fetch to other sample indices based on the fetched MCS. Resolve can then be done with the minimumal number of point-sample operations.

Alternatively, a shader can unconditionally fetch all subsamples to minimize dependencies. In this case the MCS data returned from the ld_mcs is simply passed to the ld2dms message.

Multisample Control Surface

Four messages have been defined for the sampling engine, *ld_mcs*, *ld2dms*, *ld2dms_w* and *ld2dss*. A pixel shader kernel sampling from an multisampled surface using an MCS must first sample from the MCS surface using the *ld_mcs* message. This message behaves like the *ld* message, except that the surface is defined by the MCS fields of SURFACE_STATE rather than the normal fields. The surface format is effectively R8_UINT for 4x surfaces, R32_UINT for 8x surfaces, and two R32_UNIT for x16 surfaces thus

data is returned in unsigned integer format. Following the *ld_mcs*, the kernel issues a *ld2dms* or ld2dms_w message to sample the surface itself. The integer value from the MCS surface is delivered in the mcs parameter of this messages as the sample index.

Since *sample* is no longer supported on multisampled surfaces, the multisample resolve must be done using *ld2dms or ld2dms_w*. For surfaces with **Multisampled Surface Storage Format** set to MSFMT_MSS and **MCS Enable** set to enabled, an optimization is available to enable higher performance for compressed pixels. The *ld2dss* message can be used to sample from a particular sample slice on the surface. By examining the MCS value, software can determine which sample slices to sample from. A simple optimization with probable large return in performance is to compare the MCS value to zero (indicating all samples are on sample slice 0), and sample only from sample slice 0 using *ld2dss* if MCS is zero. Sample slice 0 is the pixel color in this case. If MCS is not zero, each sample is then obtained using *ld2dms* messages and the results are averaged in the kernel after being returned. Refer to the multisample storage format in the GPU Overview volume for more details.

State

SW Generated BINDING_TABLE_STATE

The 3D sampler uses both surface state objects (RENDER_SURFACE_STATE) as well as sampler state objects (SAMPLER_STATE). These objects are cached locally in the sampler state cache

for improved performance as it is assumed that many sampler messages will utilize the same surface and sampler states.

Programming Note	
Context:	Out of Bounds Handling
If a pointer to sampler or surface state goes beyond the end of the sampler or surface state buffer (as defined by the associated size field of the STATE_BASE_ADDRESS command) the sampler will force the address offset to cache-line 0 from the defined Base Address. The result of this state fetch is undefined and depends on how the state buffer has been populated.	

Surface State Fetch

Surface state is fetched from system memory using a Binding Table Pointer (**BTP**). The **BTP** is a 16-bit value provided by the command stream (not directly by the shader) which determines the binding-table to be used. An 8-bit Binding Table Index (**BTI**) is then provided by the shader via the message descriptor, which indicates the offset into the Binding Table. The BTP and BTI are relative to the **Surface State Base Address** and the binding table itself resides in system memory. The contents of the Binding Table is a list of pointers to surface state objects. The pointer from the Binding Table is also relative to the **Sampler State Base Address,** and points directly to a 256-bit **RENDER_SURFACE_STATE** object which sampler will fetch and store in its internal state cache.

Bindless Surface State Fetch

The sampler supports a "Bindless" surface model. Bindless refers to the fact that a Binding table is not required for this mode. Instead, the shader sets the BTI to 253 in the descriptor to indicate Bindless and provides a 20-bit **Surface State Offset (SSO)** from the **Bindless Surface State Base Address .** This SSO

directly selects a specific surface state object which is cached by the 3D Sampler in local State Cache. Both bindless and non-bindless (legacy) modes can be operated at the same time by the sampler.

For Positional Shaing (POSH) there is a separate set of base addresses for Bindless and Non-Bindless Surface State called POSH Surface State Offset and POSH Bindless Surface State Base Address. They work exactly the same, but allow different base addresses for positional shading.

Sampler State Fetch

SAMPLER_STATE objects are fetched independently of surface state and cached locally in the 3D sampler independently (there may one or more SAMPLER_STATE objects associates with one or more RENDER_SURFACE_STATE objects). The sampler state is fetched using the **Sampler State Pointer (SSP)** which is provided either in the message header or directly from the command stream (message headers are not required). The **SSP** is an offset relative to the **Dynamic_State_Base_Address** and selects a table of 16 sampler states. The 4-bit **Sampler Index** (**SI**) in the message descriptor is used to select the specific **SAMPLER_STATE** object to be fetched from system memory and cached locally in the 3D sampler.

Bindless Sampler State

The sampler supports a "Bindless" sampler model. Bindless in this case does not actually refer to the lack of a Binding table since the legacy Sampler State model also did not have a Binding Table. However, the mechanism is similar to bindless surfaces in that the pointer provided directly selects a SAMPLER_STATE object. The sampler uses the same **Sampler State Pointer (SSP)**, but it is relative to the **Bindless_Sampler_Base_Address**. The Sampler Index (SI) in the message descriptor is not used, and can be set to 0. The SAMPLER_STATE object is cached locally in the 3D sampler.

For Positional Shading (POSH) there is a separate set of base addresses available for fetching sampler state called **POSH Bindless Surface State Base Address** and **POSH Dynamic State Base Address.**

State Caching

As mentioned above, the 3D Sampler allows for automatic caching of **RENDER_SURFACE_STATE** objects and **SAMPLER_STATE** objects to provide higher performance. Coherency with system memory in the state cache, like the texture cache is handled partially by software. It is expected that the command stream or shader will issue Cache Flush operation or Cache_Flush sampler message to ensure that the L1 cache remains coherent with system memory.

Programming Note	
Context:	State Cache Coherency
Whenever the value of the Dynamic_State_Base_Addr, Surface_State_Base_Addr, or Binding_Table_Pool_Base_Addr (if Binding Table Pool is enabled) are altered, the L1 state cache and L1 texture cache must be invalidated to ensure the new surface or sampler state is fetched from system memory. Whenever the RENDER_SURFACE_STATE object in memory pointed to by the **Binding Table Pointer (BTP)** and **Binding Table Index (BTI)** is modified or SAMPLER_STATE object pointed to by the **Sampler State Pointer (SSP)** and **Sampler Index (SI)** is modified, the L1 state cache and L1 texture cache must be invalidated to ensure the new surface or sampler state is fetched from system memory.	

Programming Note	
Context:	Bindless Surface State Cache Coherency
Whenever the value of the Bindless_Surface_State_Base_Addr is altered, the L1 state cache, and L1 texture cache must be invalidated to ensure the new surface or sampler state is fetched from system memory. Whenever the RENDER_SURFACE_STATE object in memory pointed to by the **Surface State Offset (SSO)** is modified, the L1 state cache must be invalidated to ensure the new surface or sampler state is fetched from system memory.	

Programming Note	
Context:	Bindless Sampler State Cache Coherency
Whenever the value of the Bindless_Sampler_State_Base_Addr is altered, the L1 state cache and L1 texture cache must be invalidated to ensure the new surface or sampler state is fetched from system memory. Whenever the SAMPLER_STATE object pointed to by the **Sampler State Pointer (SSP)** is modified, the L1 state cache must be invalidated to ensure the new surface or sampler state is fetched from system memory.	

Programming Note	
Context:	State Coherency for indirect values
Any values referenced by pointers within the RENDER_SURFACE_STATE or SAMPLER_STATE (e.g. Clear Color Pointer, Border Color or Indirect State Pointer) are considered to be part of that state and any changes to these referenced values requires an invalidation of the L1 state cache to ensure the new values are being used as part of the state. In the case of surface data pointed to by the Surface Base Address in RENDER SURFACE STATE, the Texture Cache must be invalidated if the surface data changes.	

SURFACE_STATE

The surface state is stored as individual elements, each with its own pointer in the binding table or its own entry in a memory heap in memory. Each surface state element is aligned to a 32-byte boundary.

Surface state defines the state needed for the following objects:

- texture maps (1D, 2D, 3D, cube) read by the sampling engine
- buffers read by the sampling engine
- constant buffers read by the data cache via the data port
- render targets read/written by the render cache via the data port
- streamed vertex buffer output written by the render cache via the data port
- media surfaces read from the texture cache or render cache via the data port
- media surfaces written to the render cache via the data port

The Render Surface State definition can be found in the following section:

Surface Formats

The RENDER_SURFACE_STATE contains a 9-bit field called **Surface Format**, which defines the exact format of the surface being sampled. The definition of the encodings for each supported format, including compressed formats, can be found in the following section:

SURFACE_FORMAT

For ASTC formats, the **ASTC Enable** bit in the RENDER_SURFACE_STATE must be set to 1. When set, the definition of the 9-bit **Surface Format** changes. The following table describes all supported formats for block-based ASTC textures.

SURFACE_FORMAT for All ASTC Formats

Value	[8] LDR/Full [7] 2D/3D [6] U8srgb /FLT16	Width 2D [5:3] 3D [5:4]	Height 2D [2:0] 3D [3:2]	Depth 2D: N/A 3D: [1:0]	Binary Form	Name	(BPE)
000h	000	0	0		000 000 000	ASTC_LDR_2D_4x4_U8sRGB	8.00
008h	000	1	0		000 001 000	ASTC_LDR_2D_5x4_U8sRGB	6.40
009h	000	1	1		000 001 001	ASTC_LDR_2D_5x5_ U8sRGB	5.12
011h	000	2	1		000 010 001	ASTC_LDR_2D_6x5_ U8sRGB	4.27
012h	000	2	2		000 010 010	ASTC_LDR_2D_6x6_ U8sRGB	3.56
021h	000	4	1		000 100 001	ASTC_LDR_2D_8x5_ U8sRGB	3.20
022h	000	4	2		000 100 010	ASTC_LDR_2D_8x6_ U8sRGB	2.67
031h	000	6	1		000 110 001	ASTC_LDR_2D_10x5_ U8sRGB	2.56
032h	000	6	2		000 110 010	ASTC_LDR_2D_10x6_ U8sRGB	2.13
024h	000	4	4		000 100 100	ASTC_LDR_2D_8x8_ U8sRGB	2.00
034h	000	6	4		000 110 100	ASTC_LDR_2D_10x8_ U8sRGB	1.60
036h	000	6	6		000 110 110	ASTC_LDR_2D_10x10_ U8sRGB	1.28
03eh	000	7	6		000 111 110	ASTC_LDR_2D_12x10_ U8sRGB	1.07
03fh	000	7	7		000 111 111	ASTC_LDR_2D_12x12_ U8sRGB	0.89

Value	[8] LDR/Full [7] 2D/3D [6] U8srgb /FLT16	Width 2D [5:3] 3D [5:4]	Height 2D [2:0] 3D [3:2]	Depth 2D: N/A 3D: [1:0]	Binary Form	Name	(BPE)
040h	001	0	0		001 000 000	ASTC_LDR_2D_4x4_FLT16	8.00
048h	001	1	0		001 001 000	ASTC_LDR_2D_5x4_FLT16	6.40
049h	001	1	1		001 001 001	ASTC_LDR_2D_5x5_FLT16	5.12
051h	001	2	1		001 010 001	ASTC_LDR_2D_6x5_FLT16	4.27
052h	001	2	2		001 010 010	ASTC_LDR_2D_6x6_FLT16	3.56
061h	001	4	1		001 100 001	ASTC_LDR_2D_8x5_FLT16	3.20
062h	001	4	2		001 100 010	ASTC_LDR_2D_8x6_FLT16	2.67
071h	001	6	1		001 110 001	ASTC_LDR_2D_10x5_FLT16	2.56
072h	001	6	2		001 110 010	ASTC_LDR_2D_10x6_FLT16	2.13
064h	001	4	4		001 100 100	ASTC_LDR_2D_8x8_FLT16	2.00
074h	001	6	4		001 110 100	ASTC_LDR_2D_10x8_FLT16	1.60
076h	001	6	6		001 110 110	ASTC_LDR_2D_10x10_FLT16	1.28
07eh	001	7	6		001 111 110	ASTC_LDR_2D_12x10_FLT16	1.07
07fh	001	7	7		001 111 111	ASTC_LDR_2D_12x12_FLT16	0.89

Programming Note	
Context:	Luminance, Intensity and Transparency Formats
Luminance, Intensity and Opacity formats can be more efficiently supported with higher performance by mapping them to their equivalent RGB format and programming the Shader Channel selects in the RENDER_SURFACE_STATE. See **SW Performance Hints** for a detailed description of this usage model.	

Sampler-Support Surface Formats

The table below lists all the surface formats which are supported by the Sampler for filtering and any restrictions.

The following table also indicates the mapping of the channels from the surface to the channels output from the sampling engine.

Some formats are supported only in DX10/OGL **Border Color Mode**. Those formats have only that mode indicated. Formats that behave the same way in both **Border Color Modes** are indicated by that column being blank.

Surface Format Name	Shadow Map Support	Chroma Key Support	Border Color Mode	R	G	B	A	Border Color Mode	R	G	B	A
R32G32B32A32_FLOAT				R	G	B	A					
R32G32B32A32_SINT			DX10/OGL	R	G	B	A					
R32G32B32A32_UINT			DX10/OGL	R	G	B	A					
R32G32B32X32_FLOAT				R	G	B	1.0					
R32G32B32_FLOAT				R	G	B	1.0					
R32G32B32_SINT			DX10/OGL	R	G	B	1.0					
R32G32B32_UINT			DX10/OGL	R	G	B	1.0					
R16G16B16A16_UNORM		Yes		R	G	B	A					
R16G16B16A16_SNORM				R	G	B	A					
R16G16B16A16_SINT			DX10/OGL	R	G	B	A					
R16G16B16A16_UINT			DX10/OGL	R	G	B	A					
R16G16B16A16_FLOAT				R	G	B	A					
R32G32_FLOAT			DX10/OGL	R	G	0.0	1.0	OCL	R	G	1.0	1.0
R32G32_SINT			DX10/OGL	R	G	0.0	1.0	OCL				
R32G32_UINT			DX10/OGL	R	G	0.0	1.0	OCL				
R32_FLOAT_X8X24_TYPELESS	Yes		DX10/OGL	R	0.0	0.0	1.0	OCL				
X32_TYPELESS_G8X24_UINT			DX10/OGL	0.0	G	0.0	1.0	OCL				
L32A32_FLOAT			DX10/OGL	L	L	L	A					
R16G16B16X16_UNORM				R	G	B	1.0					
R16G16B16X16_FLOAT				R	G	B	1.0					
A32X32_FLOAT				0.0	0.0	0.0	A					
L32X32_FLOAT				L	L	L	1.0					
I32X32_FLOAT				I	I	I	I					
B8G8R8A8_UNORM		Yes		R	G	B	A					
B8G8R8A8_UNORM_SRGB				R	G	B	A					
R10G10B10A2_UNORM				R	G	B	A					
R10G10B10A2_UINT			DX10/OGL	R	G	B	A					
R10G10B10_SNORM_A2_UNORM				R	G	B	A					
R10G10B10_FLOAT_A2_UNORM			DX10/OGL	R	G	B	A					
R8G8B8A8_UNORM				R	G	B	A					

Surface Format Name	Shadow Map Support	Chroma Key Support	Border Color Mode	R	G	B	A	Border Color Mode	R	G	B	A
R8G8B8A8_UNORM_SRGB				R	G	B	A					
R8G8B8A8_SNORM				R	G	B	A					
R8G8B8A8_SINT			DX10/OGL	R	G	B	A					
R8G8B8A8_UINT			DX10/OGL	R	G	B	A					
R16G16_UNORM		Yes	DX10/OGL	R	G	0.0	1.0	OCL	R	G	1.0	1.0
R16G16_SNORM			DX10/OGL	R	G	0.0	1.0	OCL	R	G	1.0	1.0
R16G16_SINT			DX10/OGL	R	G	0.0	1.0	OCL				
R16G16_UINT			DX10/OGL	R	G	0.0	1.0	OCL				
R16G16_FLOAT			DX10/OGL	R	G	0.0	1.0	DX9	R	G	1.0	1.0
B10G10R10A2_UNORM				R	G	B	A					
R11G11B10_FLOAT				R	G	B	1.0					
R32_SINT			DX10/OGL	R	0.0	0.0	1.0	OCL				
R32_UINT			DX10/OGL	R	0.0	0.0	1.0	OCL				
R32_FLOAT	Yes		DX10/OGL	R	0.0	0.0	1.0	OCL	R	1.0	1.0	1.0
R24_UNORM_X8_TYPELESS	Yes		DX10/OGL	R	0.0	0.0	1.0	OCL				
X24_TYPELESS_G8_UINT			DX10/OGL	0.0	G	0.0	1.0					
L16A16_UNORM				L	L	L	A					
I24X8_UNORM	Yes			I	I	I	I					
L24X8_UNORM	Yes			L	L	L	1.0					
A24X8_UNORM	Yes			0.0	0.0	0.0	A					
I32_FLOAT	Yes			I	I	I	I					
L32_FLOAT	Yes			L	L	L	1.0					
A32_FLOAT	Yes			0.0	0.0	0.0	A					
B8G8R8X8_UNORM		Yes		R	G	B	1.0					
B8G8R8X8_UNORM_SRGB				R	G	B	1.0					
R8G8B8X8_UNORM				R	G	B	1.0					
R8G8B8X8_UNORM_SRGB				R	G	B	1.0					
R9G9B9E5_SHAREDEXP				R	G	B	1.0					
B10G10R10X2_UNORM				R	G	B	1.0					
L16A16_FLOAT				L	L	L	A					
B5G6R5_UNORM		Yes		R	G	B	1.0					
B5G6R5_UNORM_SRGB				R	G	B	1.0					
B5G5R5A1_UNORM		Yes		R	G	B	A					
B5G5R5A1_UNORM_SRGB				R	G	B	A					
B4G4R4A4_UNORM		Yes		R	G	B	A					

Surface Format Name	Shadow Map Support	Chroma Key Support	Border Color Mode	R	G	B	A	Border Color Mode	R	G	B	A
B4G4R4A4_UNORM_SRGB				R	G	B	A					
R8G8_UNORM			DX10/OGL	R	G	0.0	1.0	DX9	R	G	1.0	1.0
R8G8_SNORM		Yes	DX10/OGL	R	G	0.0	1.0	DX9	R	G	1.0	1.0
R8G8_SINT			DX10/OGL	R	G	0.0	1.0					
R8G8_UINT			DX10/OGL	R	G	0.0	1.0					
R16_UNORM	Yes	Yes	DX10/OGL	R	0.0	0.0	1.0	OCL				
R16_SNORM			DX10/OGL	R	0.0	0.0	1.0	OCL				
R16_SINT			DX10/OGL	R	0.0	0.0	1.0	OCL				
R16_UINT			DX10/OGL	R	0.0	0.0	1.0	OCL				
R16_FLOAT			DX10/OGL	R	0.0	0.0	1.0	OCL	R	1.0	1.0	1.0
I16_UNORM	Yes			I	I	I	I					
L16_UNORM	Yes			L	L	L	1.0					
A16_UNORM	Yes			0.0	0.0	0.0	A					
L8A8_UNORM		Yes		L	L	L	A					
I16_FLOAT	Yes			I	I	I	I					
L16_FLOAT	Yes			L	L	L	1.0					
A16_FLOAT	Yes			0.0	0.0	0.0	A					
L8A8_UNORM_SRGB				L	L	L	A					
R5G5_SNORM_B6_UNORM		Yes		R	G	B	1.0					
A1B5G5R5_UNORM				R	G	B	A					
A4B4G4R4_UNORM				R	G	B	A					
R8_UNORM		Yes	DX10/OGL	R	0.0	0.0	1.0					
R8_SNORM			DX10/OGL	R	0.0	0.0	1.0					
R8_SINT			DX10/OGL	R	0.0	0.0	1.0					
R8_UINT			DX10/OGL	R	0.0	0.0	1.0					
A8_UNORM		Yes		0.0	0.0	0.0	A					
I8_UNORM				I	I	I	I					
L8_UNORM		Yes		L	L	L	1.0					
Y8_UNORM		Yes		Y	Y	Y	1.0					
L8_UNORM_SRGB				L	L	L	1.0					
DXT1_RGB_SRGB				R	G	B	1.0					
R1_UNORM				R	0.0	0.0	1.0					
YCRCB_NORMAL		Yes		Cr	Y	Cb	1.0					
YCRCB_SWAPUVY		Yes		Cr	Y	Cb	1.0					
BC1_UNORM				R	G	B	A					

Surface Format Name	Shadow Map Support	Chroma Key Support	Border Color Mode	R	G	B	A	Border Color Mode	R	G	B	A
BC2_UNORM				R	G	B	A					
BC3_UNORM				R	G	B	A					
BC4_UNORM			DX10/OGL	R	0.0	0.0	1.0	OCL				
BC5_UNORM			DX10/OGL	R	G	0.0	1.0	OCL				
BC1_UNORM_SRGB				R	G	B	A					
BC2_UNORM_SRGB				R	G	B	A					
BC3_UNORM_SRGB				R	G	B	A					
YCRCB_SWAPUV				Cr	Y	Cb	1.0					
YCRCB_SWAPY				Cr	Y	Cb	1.0					
DXT1_RGB				R	G	B	1.0					
R8G8B8_UNORM				R	G	B	1.0					
R8G8B8_SNORM				R	G	B	1.0					
BC4_SNORM			DX10/OGL	R	0.0	0.0	1.0	OCL				
BC5_SNORM			DX10/OGL	R	G	0.0	1.0	OCL				
R16G16B16_FLOAT				R	G	B	1.0					
R16G16B16_UNORM				R	G	B	1.0					
R16G16B16_SNORM				R	G	B	1.0					
BC6H_SF16				R	G	B	1.0*					
BC7_UNORM				R	G	B	A					
BC7_UNORM_SRGB				R	G	B	A					
BC6H_UF16				R	G	B	1.0*					
PLANAR_420_8		Yes		Cr	Y	Cb	1.0					
PLANAR_420_16		Yes		Cr	Y	Cb	1.0					
R8G8B8_UNORM_SRGB				R	G	B	1.0					
ETC1_RGB8				R	G	B	1.0					
ETC2_RGB8				R	G	B	1.0					
EAC_R11				R	0.0	0.0	1.0	OCL				
EAC_RG11				R	G	0.0	1.0	OCL				
EAC_SIGNED_R11				R	0.0	0.0	1.0	OCL				
EAC_SIGNED_RG11				R	G	0.0	1.0	OCL				
ETC2_SRGB8				R	G	B	1.0					
R16G16B16_UINT			DX10/OGL	R	G	B	1.0					
R16G16B16_SINT			DX10/OGL	R	G	B	1.0					
ETC2_RGB8_PTA				R	G	B	A					
ETC2_SRGB8_PTA				R	G	B	A					

Surface Format Name	Shadow Map Support	Chroma Key Support	Border Color Mode	R	G	B	A	Border Color Mode	R	G	B	A
ETC2_EAC_RGBA8				R	G	B	A					
ETC2_EAC_SRGB8_A8				R	G	B	A					
R8G8B8_UINT			DX10/OGL	R	G	B	1.0					
R8G8B8_SINT			DX10/OGL	R	G	B	1.0					
ASTC				R	G	B	A					
L8A8_UINT			DX10/OGL	L	L	L	A					
L8A8_SINT			DX10/OGL	L	L	L	A					
L8_UINT			DX10/OGL	L	L	L	1.0					
L8_SINT			DX10/OGL	L	L	L	1.0					
I8_UINT			DX10/OGL	I	I	I	I					
I8_SINT			DX10/OGL	I	I	I	I					

Programming Note	
Context:	SURFACE_STATE
Any surface format not supported by the 3D sampler except Raw Buffer format will return undefined results. The Unsupported Raw Buffer format will be treated as RGBA8888 by default.	

Programming Note	
Context:	1D Surfaces
1D Surfaces must be defined as linear (not tiled).	

Programming Note	
Context:	SURFACE_STATE
Surface formats PLANAR_420_8 and PLANAR_420_16 which require half-pitch chroma planes (e.g. YV12) cannot support fenced tiling.	

Programming Note	
Context:	Chroma Key and sample_unorm
The surface formats PLANAR_420_16, R16_UNORM, R16G16B16A16_UNORM, and R16G16_UNORM are only supported for sample_unorm with Chroma Key enabled.	

It is recommended, for performance reasons, to never use any format of the type L*A*, I* or A*. Instead use R* or RG* in combination with Shader Channel Select.

Programming Note	
Context:	Anisotropic Filtering with 128bpt and DX9 Border Color Mode
Anisotropic Filtering of 128bpt surfaces, when Sampler State **Texture Border Color Mode** is set to DX9, is not supported	

Programming Note	
Context:	NULL Surfaces and Shader Channel Select
Is SURFTYPE_NULL is selected, **Shader Channel Select Alpha** must be programmed to SCS_ZERO.	

Programming Note	
Context:	HiZ Surfaces
HiZ Auxiliary surfaces are not supported for surfaces which are sampled by the 3D sampler. If an Auxiliary surface is programmed into the RENDER_SURFACE_STATE it will be ignored by the 3D sampler and interpreted as AUX_NONE.	

Programming Note	
Context:	Re-described surfaces
If two surface states reference the same texture with different **Surface Format** fields there is a possibility that the sampler will return incorrect pixels unless a texture cache invalidate is done between accesses using different surface state.	

Programming Note	
Context:	Quilted Surfaces
Quilted Surfaces are not supported.	

Programming Note	
Context:	Planar Surfaces with Disabled Sampler
A sample to a YUV Planar surface (e.g. PLANAR_420_8) when the Sampler State has been disabled by setting the Sampler Disable bit, will not return 0's.	

Programming Note	
Context:	R10G10B10_FLOAT_A2_UNORM Point-Sample Precision
When a sampler returns a point-sample to a R10G10B10_FLOAT_A2_NORM surface and the return format is 32-bit Floating Point, the Alpha channel will only have the precision of a 16-bit floating point value (10-bits of fractional precision).	

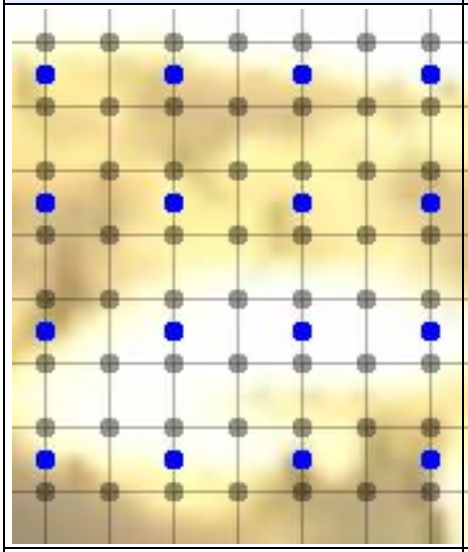
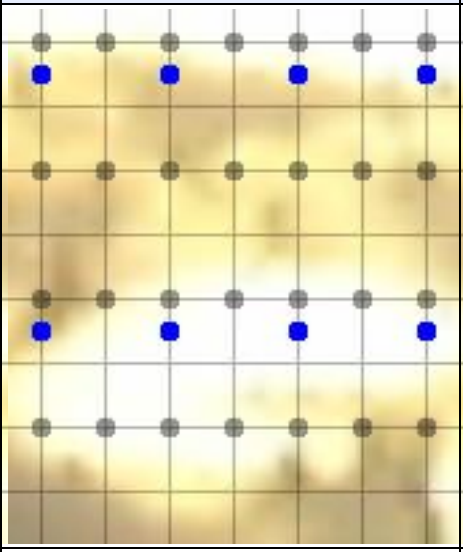
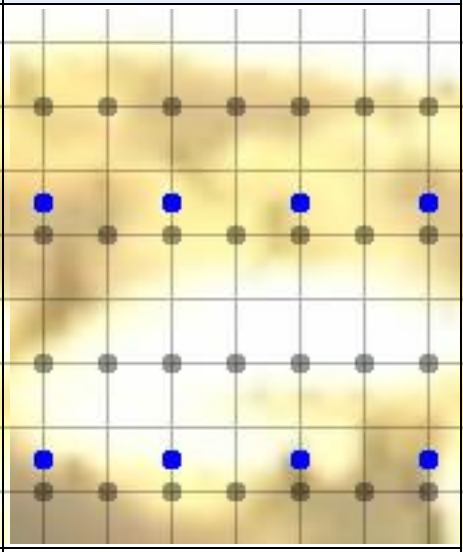
Programming Note	
Context:	Sample Tap Discard and YUV Surfaces
Returned Filter Weight for Sample Tap Discard will only report the filter weight contributing texel values of the Lumina plane for YUV surface such as YCrCb.*	

SURFACE_STATE for Deinterlace sample_8x8 and VME

This section contains media surface state definitions.

Cr(V)/Cb(U) Pixel Offset V Direction

The position of Y is brown and the position of Cr(V)/Cb(U) is blue.

Full Frame	Top Field	Bottom Field
		
V Offset 0.5	V Offset 0.25	V Offset 0.75

MEDIA_SURFACE_STATE

Programming Note	
Context:	SURFACE_STATE for Deinterlace sample_8x
The Faulting modes described in the MEMORY_OBJECT_CONTROL_STATE should be set to the same for the multi-surface Video Analytics functions like "LBP Correlation" and "Correlation Search" for both the surfaces.	

SAMPLER_STATE

SAMPLER_STATE has different formats, depending on the message type used. The sample_8x8 and deinterlace messages use a different format of SAMPLER_STATE as detailed in the corresponding sections.

Restriction: The **Min LOD** and **Max LOD** fields need range increased from [0.0,13.0] to [0.0,14.0] and fractional bits increased from six to eight. This requires a few fields to be moved as indicated in the text.

All 16-byte sampler states in a 64-byte aligned cache-line must be programmed with a valid sampler state.

Chroma Key cannot be used in conjunction with Anisotropic filtering.

Sampler State
SAMPLER STATE
SAMPLER_BORDER_COLOR_STATE

If border color is used, all formats must be provided. Hardware will choose the appropriate format based on **Surface Format** and **Texture Border Color Mode**. The values represented by each format should be the same (other than being subject to range-based clamping and precision) to avoid unexpected behavior.

DWord	Bits	Description
0	31:24	**Border Color Alpha** Format = UNORM8
	23:16	**Border Color Blue** Format = UNORM8
	15:8	**Border Color Green** Format = UNORM8
	7:0	**Border Color Red** Format = UNORM8
1	31:0	**Border Color Red** Format = IEEE_FP
2	31:0	**Border Color Green** Format = IEEE_FP
3	31:0	**Border Color Blue** Format = IEEE_FP
4	31:0	**Border Color Alpha** Format = IEEE_FP
5	31:16	**Border Color Green** Format = FLOAT16
	15:0	**Border Color Red** Format = FLOAT16
6	31:16	**Border Color Alpha** Format = FLOAT16
	15:0	**Border Color Blue** Format = FLOAT16
7	31:16	**Border Color Green** Format = UNORM16
	15:0	**Border Color Red** Format = UNORM16
8	31:16	**Border Color Alpha** Format = UNORM16
	15:0	**Border Color Blue** Format = UNORM16
9	31:16	**Border Color Green** Format = SNORM16

DWord	Bits	Description
	15:0	**Border Color Red** Format = SNORM16
10	31:16	**Border Color Alpha** Format = SNORM16
	15:0	**Border Color Blue** Format = SNORM16
11	31:24	**Border Color Alpha** Format = SNORM8
	23:16	**Border Color Blue** Format = SNORM8
	15:8	**Border Color Green** Format = SNORM8
	7:0	**Border Color Red** Format = SNORM8

Border Color Programming for Integer Surface Formats

For integer formats, there are different possible cases depending on the bits per channel (bpc) and bits per texel (bpt) of the surface format.

Integer Surface Format - Different Types	Surface formats
32 bpc, 128 bpt case (4 types)	R32G32B32A32_UINT R32G32B32_UINT R32G32B32A32_SINT R32G32B32_SINT
16 bpc, 64 bpt case (5 types)	R16G16B16A16_UINT, R10G10B10A2_UINT X32_TYPELESS_G8X24_UINT R16G16B16_UINT R16G16B16A16_SINT R16G16B16_SINT
32 bpc, 64 bpt case (2 types)	R32G32_UINT R32G32_SINT
8 bpc, 32 bpt cases (9 types)	R8G8B8A8_UINT R8G8_UINT R8_UINT X24_TYPELESS_G8_UINT R8G8B8_UINT

Integer Surface Format - Different Types	Surface formats
	R8G8B8A8_SINT R8G8_SINT R8_SINT R8G8B8_SINT
16 bpc, 32 bpt cases (4 types)	R16G16_UINT R16_UINT R16G16_SINT R16_SINT
32 bpc, 32 bpt case (2 types)	R32_UINT R32_SINT

HW supports only 1 index for a given Sampler Border Color state and Sampler State. So, SW will have to program the table in **SAMPLER_BORDER_COLOR_STATE** at DWord offsets 16 to 19, as per the integer surface format type (depends on the bits per channel and bits per texel of the surface format). If any color channel is missing from the surface format, the corresponding border color should be programmed as zero; if the alpha channel is missing, the corresponding Alpha border color should be programmed as 1. Some of the representative cases are listed below:

Case 2: R32G32B32A32_SINT (32 bpc, 128 bpt, 4 channel, SINT)

Each of the values in the above table would have be to programmed as sint32 value.

Case 3: R32G32B32_UINT (32 bpc, 128 bpt, 3 channel)

R/G/B values would be programmed like in Case1. Alpha channel value at DWORDN+3 would have to be programmed as Integer 1.

Sampler State
3DSTATE_CHROMA_KEY
SAMPLER_INDIRECT_STATE

Programming Note	
Context:	Wrap with very small negative coordinates
If the input coordinate is negative and very small (< 2^-25) and the TEXCOORDMODE_WRAP is selected for that coordinate, the sampler will truncate the texel coordinate to 0.0 and return the wrong texel.	

Messages

The sampler receives messages from shader clients. These messages contain information to allow the sampler to perform sample operations and return results. A message consists of four components:

- Execution Mask: Indicates, for a given SIMD, which pixels are valid.
- Message Descriptor: Required information including length of the message, and the length of the response.

- Message Header: Optional information which may be required for certain operations (e.g. Direct Write to Render Target).
- Message Payload: Specific data for each sampler operation including coordinates and other message parameters.

Messages from the EU can be "fused", meaning that a pair of threads will simultaneously issue identical sample operations (for different pixels) to a single 3D sampler. The 3D sampler will process these two messages in parallel. Two fused SIMD8 messages effectively become SIMD16 and two fused SIMD16 messages effectively become a SIMD32 message. It is required that the two messages utilize the exact same state (surface and sampler) because only one state will be fetched for both messages. In addition, if a header is present, it must be present on both messages and the contents of the header must be identical. See the section on Fused Send Message Handling for additional details.

Programming Notes

Programming Note	
Context:	Shader restrictions
Messages which require the calculation of LOD in the sampler must send pixels coordinates for each group of 4 pixels (known as a sub-span) even if all pixels are not enabled for return to the shader. The LOD calculation and the subsequent return of pixels will not be correct if all pixels in a sub-span do not have valid coordinates or gradients to allow calculation of LOD.	

Programming Note	
Context:	3D Sampler Messages
Write back messages can return erroneous information in the Pixel Null Mask field for sample_c messages on a 64-bit surface format (e.g. R16G16B16A16_UNORM)	

Programming Note	
Context:	Message Headers and Mid-Thread Pre-Emption
A message header is required for GPGPU kernels for all sample_8x8 and sample_unorm messages to allow save/restore mechanisms to work correctly.	

Procedural Textures and Texel Shading

Evaluate Messages

The 3D sampler supports usage by Pixel Shaders which utilize the evaluate() type messages for Procedural Textures. The format of these messages to sampler are identical to normal sample() messages except in two respects:

- The return length of the message (as defined in the Sampler Message Descriptor) must be set to zero to indicate this is an evaluate operation
- The RENDER_SURFACE_STATE **Procedural Texture** bit field must be set.

If both of the above conditions hold for an incoming sampler message, the 3D sampler will interpret them as an evaluate message, and will not perform the sample operation, but will send the requested texels to the AMFS block to spawn the required Texel Shaders.

The table below shows the behavior of 3D sampler for incoming messages for all possible states of message return length and **Procedural Texture** field in RENDER_SURFACE_STATE.

Return Length	Procedural Texture state bit	Behavior
0	0	Message Will be Dropped, this is an unsupported operation
0	1	Message is interpreted as an evaluate message and all required texels will be sent to AMFS for spawning of a Texel Shader
>0	0	Legacy Behavior: Sampler will perform sample operations as normal and return sample result to shader
>0	1	Sampler will perform sample operations as normal and return sample result to shader

Restrictions

Programming Note	
Context:	Tiling
Procedural Textures and associated evaluate messages to 3D sampler are only supported for TileYs surfaces.	

Programming Note	
Context:	Bindless Surfaces
Procedural Textures and associated evaluate messages to 3D sampler are only supported for Bindless Surface and Bindless Sampler State	

Programming Note	
Context:	Non-Eval Messages
AMFS Evaluation Shaders must not use the same surface state as a Non-Evaluation Shader to the same surface.	

Each 4K page fits in a 16byte value and within the 16 bytes, the below math will help get us to the right nibble –

Tile Format	SURF_TYPE	Byte selector	Nibble selector
Tile Y	2D,3D,CUBE	Main_Surface_Adr[13, 12, 11, 10, 8, 7]	Main_Surface_Adr[6]
Tile Ys 8bpp	2D,3D,CUBE	Main_Surface_Adr[13, 12, 11, 10, 8, 7]	Main_Surface_Adr[6]
Tile Ys 16/32 bpp	2D,CUBE	Main_Surface_Adr[13, 12, 11, 10, 9, 8]	Main_Surface_Adr[6]

Tile Ys 32 bpp	3D	Main_Surface_Adr[13, 12, 11, 10, 8, 7]	Main_Surface_Adr[6]
Tile Ys 16	3D	Main_Surface_Adr[13, 11, 10, 9, 8, 7]	Main_Surface_Adr[6]
Tile Ys 64/128 bpp	2D,CUBE	Main_Surface_Adr[13, 12, 11, 10, 9, 8]	Main_Surface_Adr[7]
Linear Untyped	BUFFER	Main_Surface_Adr[13, 12, 11, 10, 9,8]	Main_Surface_Adr[7]

Programming Note	
Context:	non-compute shaders doing evaluate

SW has to send a CS stalling flush before issuing AMFS flush to the 3D pipe.

All shaders that perform evaluates must send a Cache Flush message to the sampler with a non-zero read-length after all evaluates are issued and before End-Of-Thread

Programming Note	
Context:	Evaluate in compute context

1. A pipe control flush with "AMFS flush Enable" set and "DC flush enable set" must be sent down the pipe before a context switch, when compute shaders do evaluate.
2. if compute shader does evaluates, and SW needs to flush the AMFS pipe, it has to first send a pipecontrol flush to the compute pipe and then switch to 3D pipe before sending a pipecontrol with "Command Streamer Stall Enable", AMFS flush Enable, and DC flush enable set on it

Programming Note	
Context:	Evaluate in compute context & pre-emption

1) If shaders do evaluate (3d/compute), pre-emption must be disabled, until AMFS data is flushed out of all the caches.

2) All shaders that perform evaluates must send a Cache Flush message to the sampler with a non-zero read-length after all evaluates are issued and before End-Of-Thread

Programming Note	
Context:	Evaluate in compute context

Compute shaders run on a CCS context must not issue AMFS evaluates. All AMFS evaluates must run in an RCS context

Workaround	
Context:	3D and compute

Any kernel that contains AMFS evaluate (WriteSamplerFeedback) operations must issue two back to back sampler

Workaround	
Context:	3D and compute
cache flush messages after all evaluate operations are sent and before the kernel EOT message. 1. The first sampler cache flush message must have a zero-length return. This is used to signal EOT to the AMFS unit. 2. The second sampler cache flush message must have a non-zero-length return. This is used to block the kernel EOT until all AMFS operations are flushed out of the sampler. Failure to do both sampler cache flush messages can result in HW hangs and/or spurious page faults.	

Compression is done on a 128B byte cacheline. On the render path, the pairing bit determines which two 64B fragments constitute a 128B render cacheline.

The pairing bit is defined in the following table

Surface Type	TileY	Tiled Ys	Tiled Ys	Tiled Ys	Tiled Ys	Other
	All Bpe	64 & 128	32	16	8	
2D/Cube	9	6	7	7	9	6
3D	9	8	9	12	9	6
Other	6	6	6	6	6	6

Message Descriptor and Execution Mask

Execution Mask

SIMD16. The 16-bit execution mask forms the valid pixel signals. This determines which pixels are sampled and results returned to the GRF registers. Samples for invalid pixels are not overwritten in the GRF. However, if LOD needs to be computed for a subspan based on the message type and MIP filter mode and at least one pixel in the subspan being valid, the sampling engine assumes that the parameters for the upper left, upper right, and lower left pixels in the subspan are valid regardless of the execution mask, as these are needed for the LOD computation.

SIMD8. The low 8 bits of the execution mask form the valid pixel signals. If LOD needs to be computed based on MIP filter mode and at least one pixel in the subspan being valid, the sampling engine assumes that the parameters for the upper left, upper right, and lower left pixels in the subspan are valid regardless of the execution mask, since these are needed for the LOD computation.

SIMD32. The execution mask is ignored, all pixels are considered valid, and all channels are returned regardless of the execution mask.

Message Descriptor

The Sampler Message Descriptor is composed of a 32-bit Message Descriptor and a 32-bit Extended Message Descriptor (used for support Bindless resources, CPS etc.). The lower 11 bits of the Extended Message Descriptor are defined by the Extended Message Descriptor for the Execution Unit.

The message descriptor is sent in parallel with the message payload (and header if used).

Descriptor
Message Descriptor - Sampling Engine
Extended Message Descriptor - Sampling Engine
Extended Message Descriptor - Execution Unit

Restrictions for Message Descriptors

Use of any message to the Sampling Engine function with the **End of Thread** bit set in the message descriptor is not allowed.

Message Header

The message header for the sampling engine is the same regardless of the message type. The message header is optional. If the header is not present, the behavior is as if the message was sent with all fields in the header set to zero and the write channel masks are all enabled and offsets are zero. However, if the header is not included in the message, the Sampler State Pointer will be obtained from the command stream input for the given thread.

For fused messages, the two messages must contain the same content in the message header or both messages must not have a header.

The exact definition of the Sampler Message Header can be found in the section below:

Sampler Message Header

Message Types

3D Sampler Message Types

The 3D sampler supports multiple message types with different types of behaviors being supported. Each message can be supported with multiple SIMD forms (e.g. SIMD8, SIMD16 etc). See the section Message Formats for which SIMD forms are supported as well as the specific parameters and order of parameters for each message.

Below is a complete list of supported 3D Sampler message types:

Message Types
sample_*
ld_*
gather4_*
LOD

Message Types
sampleinfo
resinfo
cache_flush
sample_unorm1
sample_8x81

[1] For SIMD32/SIMD64 Media-type messages such as sample_unorm, sample_8X8 and deinterlace see the Media Sampler SIMD32/64 section for a description of the payloads of these messages and the expected behavior.

Common Message Variants

Many message types have multiple variants which provide for different sampling behaviors. The variants of a message type are named by appending a suffix to the base message.

Variant Suffix	Definition
_l	LOD Override: The LOD is provided in the message rather than being calculated from the gradients of the sub-span pixel coordinates.
_b	LOD Bias: A floating-point value between +16.0 and -16.0 is added to the LOD based on gradients of the sub-span pixel coordinates.
_c	Compare: Returns a white or black result depending on the comparison of a Ref parameter to the resulting red-channel of the sample. Comparison type is defined by the **Shadow Function** field in the SAMPLER_STATE
_lz	LOD=0 Override: LOD is forced to 0. No LOD is calculated or provided in the message
_d	Gradient: Rather than receiving absolute texel coordinates for all pixels of sub-span, a single pixel coordinate tuple is provided and a set of floating-point gradient values with respect to those pixel coordinates. This is then used to calculate other pixel coodinates and the LOD
_killpix	Kill Pixel: Used in conjunction with the Chroma Key mode enabled by the Sampler State Field CHROMA_KEY_ENABLE. It causes the associated sampler result to include a "Kill Pixel Mask" where 0's indicate pixels which matched a particular Chroma Key Mode.

See the subsections for each message type for a description of the behaviors and programming restrictions.

Restrictions and Programming Notes for All Message Types

Programming Note	
Context:	Message Types
For surfaces of type SURFTYPE_CUBE, the sampling engine requires u, v, and r parameters that have already been divided by the absolute value of the parameter (u, v, or r) with the largest absolute value.	

Programming Note	
Context:	Reduction Filter and *_c message variants.
Programming restrictions defined for *_c variants of all message types will not apply if the reduction filter is enabled via the Reduction Type Enable field in the SAMPLER_STATE and a Reduction Type that is **NOT** COMPARISON is selected. The restrictions of the non *_c variant will apply instead.	

Sample Message Types

Sample Message Definition

Message Type	Description
sample_*	The surface is sampled using the indicated sampler state. LOD is computed using calculated gradients between adjacent pixels. One, two, or three floating-point coordinate parameters may be specified depending on how many dimensions the indicated surface type uses. Extra parameters specified are ignored. Missing parameters are defaulted to 0. The sample message enables filtering/blending operations (e.g. MAP_LINEAR) and addressing modes (e.g. Mirror) which are controlled by SAMPLER_STATE.

Supported Variants

Message	Description
sample	Basic sample operation as described above.
sample_l	Basic sample with LOD in message, not computed
sample_b	Basic sample with Bias in messaged added to computed LOD.
sample_c	Basic sample with Reference value in message and comparison against Red channel of the sampled surface returned. Used primarily for shadow maps. Comparison type is defined by the **Shadow Function** field in the SAMPLER_STATE.
sample_lz	Basic sample with LOD forced to 0. Possibly higher performance than **sample_l** for cases where the surface has MIP_COUNT=1 because the LOD does not need to be computed or sent in message.
sample_l_c	sample_c with LOD override in message, not computed.
sample_b_c	sample_c with LOD Bias
sample_c_lz	Similar to sample_c with LOD forced to 0. Possibly higher performance than **sample_c** or **sample_l_c** for cases where the surface has MIP_COUNT=1 because the LOD does not need to be computed.
sample_d	The surface is sampled using the indicated sampler state. LOD is computed using the gradients present in the message. The *r* coordinate and its gradients are required only for surface types that use the third coordinate. Usage of this message type on cube surfaces assumes that the u, v, and gradients have already been transformed onto the appropriate face, but still in [-1,+1] range. The *r* coordinate contains the faceid, and the r gradients are ignored by hardware. Previously known as sample_g.

Message	Description
sample_d_c	Same as sample_d, but returns comparison against Red Channel of sampled surface like sample_c. Previously known as sample_g_c.
sample_killpix	Basic sample, but returns a Kill Pixel Mask based on Chroma Key sampler comparison results. An additional register is returned after the sample results which contains the kill pixel mask. This message type is required to allow the result of a chroma key enabled sampler in KEYFILTER_KILL_ON_ANY_MATCH mode to affect the final pixel mask.

Restrictions and Programming Notes for Sample

Programming Note
sample: The Surface Type of the associated surface must be SURFTYPE_1D, SURFTYPE_2D, SURFTYPE_3D, or SURFTYPE_CUBE.
sample: Number of Multisamples on the associated surface must be MULTISAMPLECOUNT_1.
sample: The *mlod* parameter specifies the per-pixel MinLOD.

Restrictions and Programming Notes for sample_b

Programming Note
sample_b: The Surface Type of the associated surface must be SURFTYPE_1D, SURFTYPE_2D, SURFTYPE_3D, or SURFTYPE_CUBE. Number of Multisamples on the associated surface must be MULTISAMPLECOUNT_1. The LOD bias delivered in the **bias** parameter is restricted to a range of [-16.0, +16.0). Values outside this range produce undefined results.
sample_b: The *mlod* parameter specifies the per-pixel MinLOD.

Restrictions and Programming Notes for sample_l and sample_lz

Programming Note
sample_l and sample_lz: The Surface Type of the associated surface must be SURFTYPE_1D, SURFTYPE_2D, SURFTYPE_3D, or SURFTYPE_CUBE. Number of Multisamples on the associated surface must be MULTISAMPLECOUNT_1.
sample_l and sample_lz: The ld_lz message is the same as ld except the lod parameter is set to zero instead of being delivered.

Restrictions and Programming Notes for sample_c and sample_c_lz

Programming Note
sample_c and sample_c_lz: The Surface Type of the associated surface must be SURFTYPE_1D, SURFTYPE_2D, or SURFTYPE_CUBE. The Surface Format of the associated surface must be indicated as supporting shadow mapping as indicated in the surface format table. MIPFILTER_LINEAR, MAPFILTER_LINEAR, MAPFILTER_ANISOTROPIC are allowed even for surface formats that are listed as not supporting filtering in the surface formats table. Use of the SIMD4x2 form of *sample_c* without **Force LOD to Zero** enabled in the message header is not allowed, as it is not possible for the hardware to compute LOD for SIMD4x2 messages. Using SURFTYPE_CUBE surfaces is undefined with the following surface formats: I24X8_UNORM, L24X8_UNORM, A24X8_UNORM, I32_FLOAT, L32_FLOAT, and A32_FLOAT. Number of Multisamples on the associated surface must be MULTISAMPLECOUNT_1. Using DX9 **Texture Border Color Mode** and either of the following is undefined: • Any applicable Address Control Mode (depending on Surface Type) is set to TEXCOORDMODE_CLAMP_BORDER or TEXCOORDMODE_HALF_BORDER. • Surface Type is SURFTYPE_CUBE and any Cube Face Enable is disabled.
sample_c and sample_c_lz: Must use the DX10 BORDER_COLOR_MODE (selected in sampler state) when using CLAMP_BORDER addressing mode.
sample_c: The *mlod* parameter specifies the per-pixel MinLOD.

Restrictions and Programming Notes for sample_l_c

Programming Note
sample_l_c: All restrictions applying to both sample_l and sample_c must be honored.
sample_l_c: Must use the DX10 BORDER_COLOR_MODE (selected in sampler state) when using CLAMP_BORDER addressing mode.

Restrictions and Programming Notes for sample_b_c

Programming Note
sample_b_c: All restrictions applying to both sample_bl and sample_c must be honored.
sample_b_c: All variations of the sample_c, must use the DX10 BORDER_COLOR_MODE (selected in sampler state) when using CLAMP_BORDER addressing mode.

Restrictions and Programming Notes for sample_d

Programming Note
sample_d: The Surface Type of the associated surface must be SURFTYPE_1D, SURFTYPE_2D, SURFTYPE_3D, or SURFTYPE_CUBE. Number of Multisamples on the associated surface must be MULTISAMPLECOUNT_1.
sample_d: The mlod parameter specifies the per-pixel MinLOD.

Restrictions and Programming Notes for sample_d_c

Programming Note
sample_d_c: All restrictions applying to both sample_d and sample_c must be honored.

Restrictions and Programming Notes for sample_killpix

Programming Note
sample_killpix: The Surface Type of the associated surface must be SURFTYPE_1D, SURFTYPE_2D, SURFTYPE_3D, or SURFTYPE_CUBE. sample_killpix is supported only in SIMD8 mode. Number of Multisamples on the associated surface must be MULTISAMPLECOUNT_1.
sample_killpix: If the sampler is disable via the **Sampler Disable** bit in the SAMPLER_STATE, the Kill Pixel Mask returned will be undefined.

Restrictions and Programming Notes for sample_min, sample_max

Programming Note
The Surface Type of the associated surface must be SURFTYPE_2D and **Surface Array** must be disabled. Number of Multisamples on the associated surface must be MULTISAMPLECOUNT_1
If sampling on a 3D surface (Volumetric),the following two conditions cannot be supported on a sample_min or sample_max at the same time: • Sampler state addresse mode for q direction specifies D3D11_TEXADDRESS_BORDER • Address mode for U and V do not specify D3D11_TEXADDRESS_BORDER
If Null and/or Border Texels are being excluded by using the Return Filter Weight fields in the SAMPLER_STATE, and the surface type includes an Alpha Channel, the Alpha channel must be obtained separately by having the Return Filter Weight fields cleared, and the resulting

Programming Note
Alpha will still contain contributions from NULL or Border Texels.

Restrictions and Programming Notes for sample_lz

Programming Note
sample_lz: The Surface Type of the associated surface must be SURFTYPE_1D, SURFTYPE_2D, SURFTYPE_3D

ld Message Types

ld Message Definition

Message Type	Description
ld_*	The surface is sampled using a default sampler state. It performs a "point sample" with the LOD provided in the message rather than being calculated by gradients and the coordinates are non-normalized integers rather than normalized floating-point values. If the message doesn't include an LOD parameter, the message samples from LOD 0. For **ld_*** message types, the sampler state is defaulted as follows: • min, mag, and mip filter modes are "nearest". • All address control modes are "zero", a special mode in which any texel off the map or outside the MIP range of the surface has a value of zero in all channels, except for surface formats without an alpha channel, which return a value of one in the alpha channel. The ld* family of messages do not perform any blending operation. Additional parameters are needed for the variants of ld defined below. Potentially lower power than using **sample_l** with Filter Mode=MAP_NEAREST because coordinates do not need to be converted from floating-point.

Supported Variants

Message	Description
ld	Basic ld operation as described above.
ld_lz	Basic ld with LOD forced to 0. Reduces the size of the message with potential performance and power advantage because LOD is not sent as part of the message.
ld2dms	Basic ld operation but to a multi-sampled surface. Includes additional parameters for MCS values (see ld_mcs below) and Si (sample index) for each pixel in the message. Can be used for all multi-sampled surfaces up to X8 MSAA amd for X16 MSAA in cases where the upper half of the sub-samples (planes 8 thru 15) all reference plane0 (see ld_mcs to see how determining X16 MSAA viability is determined).
ld2dms_w	Same as ld2dms but provides more bits for MCS to support multi-sampled surfaces which are X16 MSAA. Therefore, the ld2dms_w message has two MCS parameters, named mcsl and mcsh. These

Message	Description
	parameters contain the low and high halves of the 64-bit MCS value, respectively. However, use of the ld2dms_w message is allowed with Number of Multisamples set to 2, 4, 8, or 16. With this message, if Number of Multisamples is 2, 4 or 8, the ld2dms_w message behaves the same as ld2dms, with mcsl used for mcs, and mcsh ignored. For SIMD8/SIMD16 the mcsh parameters is only used for X16 MSAA, and should be all 0's for X8 MSAA or lower depth.
ld2dss	Same as ld2dms, but does not require MCS, and uses only ssi parameter to pick same sub-slice for all pixels. The ssi parameter defines the sample slice that will be sampled from. Refer to the multisample storage format in the GPU Overview volume for more details.
ld_mcs	Basic ld operation, but used to fetch the MCS auxiliary surface data rather than pixels from the surface itself. The returned MCS data is used to construct the ld2dms/ld2dms_w/ld2dss message to sample from the multi-sampled surface.

Restrictions and Programming Notes for ld

Programming Note
ld: The Surface Type of the associated surface must be SURFTYPE_1D, SURFTYPE_2D, SURFTYPE_3D, or SURFTYPE_BUFFER for the ld message. Number of Multisamples on the associated surface must be MULTISAMPLECOUNT_1

Restrictions and Programming Notes for ld_lz

Programming Note
ld_lz: The Surface Type of the associated surface must be SURFTYPE_1D, SURFTYPE_2D, SURFTYPE_3D, or SURFTYPE_BUFFER for the ld message.

Restrictions and Programming Notes for ld_mcs

Programming Note
ld_mcs: The Surface Type of the surface associated with the auxiliary surface must be SURFTYPE_2D.
ld_mcs: The ld_mcs message uses the Auxiliary Surface Base Address and Auxiliary Surface Pitch fields in SURFACE_STATE to determine the base address and pitch of the surface. Surface Format is overridden to R8_UINT if Number of Multisamples is 2 or 4, R32_UINT if Number of Multisamples is 8, or R32G32_UINT if Number of Multisamples is 16. This message cannot be used on a non-multisampled surface. Otherwise, ld_mcs behaves like the ld message. If ld_mcs is issued on a surface with Auxiliary Surface Mode not set to AUX_MCS, this message returns zeros in all channels.

Programming Note
ld_mcs: All LOD parameters must be the same value within a single message (i.e. all samples returned for message must be from a single MIP).
ld_mcs: If ld_mcs is issued on a surface with Auxiliary Surface Mode not set to AUX_CCS_D, this message returns the value which indicates that all planes are uncompressed. This value can be used directly in the ld2dms_w message to ensure the correct sub-pixels are fetched. ld_mcs always returns 512 bits of data for SIMD8 and 1024 bits of data for SIMD16 regardless of the MSAA depth of the surface. If ld_mcs is issued on an auxiliary surface associated with a X8 MSAA or lower depth, the upper half of the return data will be 0's.

Restrictions and Programming Notes for ld2dms

Programming Note
ld2dms: The sampled surface type must not be MULTISAMPLECOUNT_1 The sample surface type must not be MULTISAMPLECOUNT_16 any of the upper 8 planes contain valid data. The Surface Type of the associated surface must be SURFTYPE_2D.
ld_2dms: All LOD parameters must be the same value within a single message (i.e. all samples returned for message must be from a single MIP).

Restrictions and Programming Notes for ld_2dms_w

Programming Note
ld_2dms_w: The Surface type must not be MULTISAMPLECOUNT_1. The Surface Type of the associated surface must be SURFTYPE_2D.
ld_2dms_w: All LOD parameters must be the same value within a single message (i.e. all samples returned for message must be from a single MIP).
ld_2dms_w: The MCS parameter is ignored for UMS (uncompressed MSAA) surfaces which do not have an Auxiliary MCS surface. In this case, the ssi parameter provided in the message is used to determine which sub-pixel to fetch.

Restrictions and Programming Notes for ld_2dss_w

Programming Note
ld_2dss: All LOD parameters must be the same value within a single message (i.e. all samples returned for message must be from a single MIP).

Restrictions and Programming Notes for ld

Programming Note
ld: The Surface Type of the associated surface must be SURFTYPE_1D, SURFTYPE_2D, SURFTYPE_3D, or SURFTYPE_BUFFER for the ld message. Number of Multisamples on the associated surface must be MULTISAMPLECOUNT_1

Restrictions and Programming Notes for ld_lz

Programming Note
ld_lz: The Surface Type of the associated surface must be SURFTYPE_1D, SURFTYPE_2D, SURFTYPE_3D, or SURFTYPE_BUFFER for the ld message.

Restrictions and Programming Notes for ld_mcs

Programming Note
ld_mcs: The Surface Type of the surface associated with the auxiliary surface must be SURFTYPE_2D.
ld_mcs: The ld_mcs message uses the Auxiliary Surface Base Address and Auxiliary Surface Pitch fields in SURFACE_STATE to determine the base address and pitch of the surface. Surface Format is overridden to R8_UINT if Number of Multisamples is 2 or 4, R32_UINT if Number of Multisamples is 8, or R32G32_UINT if Number of Multisamples is 16. This message cannot be used on a non-multisampled surface. Otherwise, ld_mcs behaves like the ld message. If ld_mcs is issued on a surface with Auxiliary Surface Mode not set to AUX_MCS, this message returns zeros in all channels.
ld_mcs: All LOD parameters must be the same value within a single message (i.e. all samples returned for message must be from a single MIP).
ld_mcs: If ld_mcs is issued on a surface with Auxiliary Surface Mode not set to AUX_CCS_D, this message returns the value which indicates that all planes are uncompressed. This value can be used directly in the ld2dms_w message to

Programming Note
ensure the correct sub-pixels are fetched. ld_mcs always returns 512 bits of data for SIMD8 and 1024 bits of data for SIMD16 regardless of the MSAA depth of the surface. If ld_mcs is issued on an auxiliary surface associated with a X8 MSAA or lower depth, the upper half of the return data will be 0's.

Restrictions and Programming Notes for ld2dms

Programming Note
ld2dms: The sampled surface type must not be MULTISAMPLECOUNT_1 The sample surface type must not be MULTISAMPLECOUNT_16 any of the upper 8 planes contain valid data. The Surface Type of the associated surface must be SURFTYPE_2D.
ld_2dms: All LOD parameters must be the same value within a single message (i.e. all samples returned for message must be from a single MIP).

Restrictions and Programming Notes for ld_2dms_w

Programming Note
ld_2dms_w: The Surface type must not be MULTISAMPLECOUNT_1. The Surface Type of the associated surface must be SURFTYPE_2D.
ld_2dms_w: All LOD parameters must be the same value within a single message (i.e. all samples returned for message must be from a single MIP).
ld_2dms_w: The MCS parameter is ignored for UMS (uncompressed MSAA) surfaces which do not have an Auxiliary MCS surface. In this case, the ssi parameter provided in the message is used to determine which sub-pixel to fetch.

Restrictions and Programming Notes for ld_2dss_w

Programming Note
ld_2dss: All LOD parameters must be the same value within a single message (i.e. all samples returned for message must be from a single MIP).

gather4 Message Types

Definitions

<table>
<tr><th>Message Type</th><th>Description</th></tr>
<tr><td>gather4_*</td><td>gather4* messages ignore min/mag/mip filter modes and instead does four point samples using (u,v) texture coordinate with deltas at the following locations: (-,+),(+,+),(+,-),(-,-), where the magnitude of the deltas are always half a texel. For gather4_i/b where there is an implicit LOD, it is calcuated the same as mip_filter_nearest. For gather4, the LOD is forced to zero.

The selected color channel the four contributing texels are returned directly in the sample's corresponding four channels as follows:

<table><tr><td>upper left sample = alpha channel</td><td>upper right sample = blue channel</td></tr><tr><td>lower left sample = red channel</td><td>lower right sample = green channel</td></tr></table>
Returned color channel is selected through a message field: Gather4 Source Channel Select

Two or three floating-point coordinates may be specified depending on how many coordinate dimensions the indicated surface type uses. Extra parameters specified are ignored. Missing parameters default to 0.</td></tr>
<tr><td>gather4_*</td><td>If the Return Filter Weight fields in SAMPLER_STATE are set, any texel which is NULL and/or Border will be set to 0.0 in the returned result.</td></tr>
</table>

Supported Variants:

Message	Description
gather4	Basic gather4 behavior as described above.
gather4_c	Similar to basic gather4 but performs a compare between a Reference parameters and the gathered pixels and returns a white or black (1 or 0). In addition, like **sample_c**, it only returns data on the Red channel, so the **Gather4 Source Channel Select** must be set to Red.
gather4_po	Similar to gather4, but includes offu and offv parameters, which contain texel-space offsets that override the U/V Offset fields in the message header. Unlike the message header fields however, these offsets have a wider range [-32,+31], and can differ per pixel or sample. The format of the data is 32-bit 2's complement signed integer, but hardware only interprets the least significant 6 bits of each value, treating it as a 6-bit 2's complement signed integer
gather4_po_c	This is a combination of gather4_c and gather4_po, and will return data only on the Red Channel (**Gather4 Source Channel Select** must be set to Red), but uses offu and offv to shift the position on the surface.

Restrictions and Programming Notes for gather4:

Programming Note
gather4: The Surface Type of the associated surface must be SURFTYPE_2D or SURFTYPE_CUBE. Number of Multisamples on the associated surface must be MULTISAMPLECOUNT_1. Using DX9 Border Color Mode when either of the following is true will yield undefined results: • Any applicable Address Control Mode (depending on Surface Type) is set to TEXCOORDMODE_CLAMP_BORDER or TEXCOORDMODE_HALF_BORDER. • Surface Type is SURFTYPE_CUBE and any Cube Face Enable is disabled.
gather4: If Surface Format is R10G10B10_SNORM_A2_UNORM and Gather4 Source Channel Select is alpha channel, the returned value may be incorrect.
gather4: When using gather4 type messages on CUBEMAP surfaces with SINT* surface formats the ChromaKeyEnable state bit should be set and the ChromaKeyMode state set to KEYFILTER_KILL_ON_ANY_MATCH.

Restrictions and Programming Notes for gather4_c:

Programming Note
gather4_c: The Surface Type of the associated surface must be SURFTYPE_2D or SURFTYPE_CUBE. The Surface Format of the associated surface must be one of the following: R32_FLOAT_X8X24_TYPELESS, R32_FLOAT, R24_UNORM_X8_TYPELESS, or R16_UNORM. The channel selected by the Gather4 Source Channel Select field in the message header must be set to **Red**. Number of Multisamples on the associated surface must be MULTISAMPLECOUNT_1. Using DX9 Border Color Mode and either of the following is undefined: • Any applicable Address Control Mode (depending on Surface Type) is set to TEXCOORDMODE_CLAMP_BORDER or TEXCOORDMODE_HALF_BORDER. • Surface Type is SURFTYPE_CUBE and any Cube Face Enable is disabled.
gather4_c: When using gather4 type messages on CUBEMAP surfaces with SINT* surface formats the ChromaKeyEnable state bit should be set and the ChromaKeyMode state set to KEYFILTER_KILL_ON_ANY_MATCH.

Restrictions and Programming Notes for gather4_po:

Programming Note
gather4_po: The Surface Type of the associated surface must be SURFTYPE_2D. Number of Multisamples on the associated surface must be MULTISAMPLECOUNT_1. Using DX9 Border Color Mode and either of the following is undefined: • Any applicable Address Control Mode (depending on Surface Type) is set to

Programming Note
TEXCOORDMODE_CLAMP_BORDER or TEXCOORDMODE_HALF_BORDER. Surface Type is SURFTYPE_CUBE and any Cube Face Enable is disabled.
gather4_po: When using gather4 type messages on CUBEMAP surfaces with SINT* surface formats the ChromaKeyEnable state bit should be set and the ChromaKeyMode state set to KEYFILTER_KILL_ON_ANY_MATCH.

Restrictions and Programming Notes for gather4_po_c:

Programming Note
gather4_po_c: The Surface Type of the associated surface must be SURFTYPE_2D. The Surface Format of the associated surface must be one of the following: R32_FLOAT_X8X24_TYPELESS, R32_FLOAT, R24_UNORM_X8_TYPELESS, or R16_UNORM The channel selected by the Gather4 Source Channel Select field in the message header must be set to **Red**. Number of Multisamples on the associated surface must be MULTISAMPLECOUNT_1. Using DX9 Border Color Mode and either of the following is undefined: • Any applicable Address Control Mode (depending on Surface Type) is set to TEXCOORDMODE_CLAMP_BORDER or TEXCOORDMODE_HALF_BORDER. • Surface Type is SURFTYPE_CUBE and any Cube Face Enable is disabled.
gather4_po_c: When using gather4 type messages on CUBEMAP surfaces with SINT* surface formats the ChromaKeyEnable state bit should be set and the ChromaKeyMode state set to KEYFILTER_KILL_ON_ANY_MATCH.

sampleinfo Message Type

sampleinfo Message Definition

Message Type	Description
sampleinfo	The surface indicated in the surface state is not sampled. Instead, the number of samples (UINT32) and the sample position palette index (UINT32) for the surface are returned in the red and alpha channels respectively as UINT32 values. The sample position palette index returned in alpha is incremented by one from its value in the surface state. The **Sampler State Pointer** and **Sampler Index** are ignored. The Surface Type of the associated surface must be SURFTYPE_2D or SURFTYPE_NULL.

Supported Variants:

None

Restrictions and Programming Notes for sampleinfo:

Programming Note
sampleinfo: The return format for sampleinfo message must be 32-bits.

LOD Message Type

LOD Message Definition

Message Type	Description
LOD	The surface indicated in the surface state is not sampled. Instead, LOD is computed as if the surface will be sampled, using the indicated sampler state, and the clamped and unclamped LOD values are returned in the red and green channels, respectively, in FLOAT32 format. The blue and alpha channels are undefined, and can be masked to avoid returning them. LOD is computed using gradients between adjacent pixels. Three parameters are always specified, with extra parameters not needed for the surface being ignored.

Supported Variants:

None

Restrictions and Programming Notes for LOD:

Programming Note
LOD: The Surface Type of the associated surface must be SURFTYPE_1D, SURFTYPE_2D, SURFTYPE_3D, or SURFTYPE_CUBE. Number of Multisamples on the associated surface must be MULTISAMPLECOUNT_1.
LOD: The return format for LOD message must be 32-bits.

resinfo Message Type

resinfo Message Definition

Message Type	Description
resinfo	The surface indicated in the surface state is not sampled. Instead, the width, height, depth, and MIP

Message Type	Description
	count of the surface are returned as indicated in the table below. The format of the returned data is UINT32. In the case of a 1D or 2D surfaces, the depth value returned is the number of array slices. For non-arrayed 1D and 2D surfaces the value returned will be 1. The width, height, and depth may be scaled by the LOD parameter provided in the message to give the correct dimensions of the specified mip level. The LOD parameter is an unsigned 32-bit integer in this mode (note that negative values are out-of-range when interpreted as unsigned integers). The Sampler State Pointer and Sampler Index are always ignored. For SURFTYPE_1D, 2D, 3D, and CUBE surfaces, if the delivered LOD is outside of the range [0..MipCount-1], the returned values in the red, green, and blue channels are 0s.
resinfo	The surface indicated in the surface state is not sampled. Instead, the width, height, and depth of the surface are returned as indicated in the table below. The format of the returned data is UINT32. The Sampler State Pointer and Sampler Index are always ignored.

Returned Parameters for **resinfo:**

For the resinfo message the Red, Green and Blue channels returned contain the following specialized information.

Surface Type	Red	Green	Blue	Alpha
SURFTYPE_1D	(Width+1)»LOD	Surface Array ? Depth+1 : 0	0	MIP Count
SURFTYPE_2D	(Width+1)»LOD) * (QuiltWidth+1)	((Height+1)»LOD)* (QuiltHeight+1)	Surface Array ? Depth+1 : 0	MIP Count
SURFTYPE_3D	(Width+1)»LOD	(Height+1)»LOD	(Depth+1)»LOD	MIP Count
SURFTYPE_CUBE	(Width+1)»LOD	(Height+1)»LOD	Surface Array ? Depth+1 : 0	MIP Count
SURFTYPE_BUFFER SURFTYPE_SCRATCH	Buffer size (from combined Depth/Height/Width) If buffer size is exactly 2^32, zero is returned in this field.	Undefined	Undefined	Undefined
SURFTYPE_NULL	0	0	0	0

Supported Variants:

None

Restrictions and Programming Notes for resinfo:

Programming Note
resinfo: The return format of resinfo message must be 32-bits.

cache_flush Message Type

cache_flush Message Definition

Message Type	Description
cache_flush	The texture caches in the sampling engine are invalidated. This includes all levels of texture cache, however the state caches are not invalidated. Any outstanding sample operations which arrive at the texture sampler prior to the cache_flush message are completed normally. Any sample operations which arrive after the cache_flush message will cause texture cache-misses and sampler will fetch textures from memory.

Supported Variants:

None

Restrictions and Programming Notes for cache_flush:

Programming Note
cache_flush: Software must ensure that this message is ordered appropriately with other messages. Hardware ensures only that this message is processed in the same order as the messages are received. The message causes a write back message to indicate that the flush has been completed. This message must be sent as a SIMD32/64 type message with header only (no additional message contents)
cache_flush: The cache_flush message must be sent at the end of any AMFS first-pass shader where the last instruction is not a send to sampler.

Media Message Types

The 3D sampler supports specific message types which are intended for use by the Media sampler. These messages include:

Message
sample_unorm
sample_8X8

Media message types are described in more detail, with payload information defined in the Media Sampler Section.

sample_unorm Message Types

sample_unorm Message Definition

Message Type	Description
sample_unorm_*	The surface is sampled using the indicated sampler state. 32 contiguous pixels in a 8-wide by 4-high block are sampled. The U and V addresses for the upper left pixel are delivered in this message along with Delta U and Delta V floating-point parameters and . Given a pixel at (x,y) relative to the upper left pixel (where (0,0) is the upper left pixel), the U and V for that pixel are computed as follows: U(x,y) = U(0,0) + DeltaU * x V(x,y) = V(0,0) + DeltaV * y Typically this message and its variants are used for media applications.

Supported Variants:

Message	Description
sample_unorm	Basic sample_unorm behavior as described above.
sample_unorm_RG	Similar to sample_unorm, but only the Red and Green Channels are returned because there is an implied write mask.
sample_unorm_killpix	This message is identical to the sample_unorm message except it returns a kill pixel mask in addition to the selected channels in the writeback message. This message type is required to allow the result of a chroma key enabled sampler in KEYFILTER_KILL_ON_ANY_MATCH mode to affect the final pixel mask. All restrictions of the sample_unorm message apply to this message also.
sample_unorm_RG_killpix	This message is identical to the sample_unorm_RG message except it returns a kill pixel mask in addition to the selected channels in the writeback message. This message type is required to allow the result of a chroma key enabled sampler in KEYFILTER_KILL_ON_ANY_MATCH mode to affect the final pixel mask. All restrictions of the sample_unorm message apply to this message also.

Restrictions and Programming Notes for sample_unorm, sample_unorm_RG, sample_unorm_killpix, sample_unorm_RG_killpix:

Programming Note
The Surface Type of the associated surface must be SURFTYPE_2D. The Surface Format of the associated surface must be UNORM with <= 8 bits per channel. The MIP Count, Depth, Surface Min LOD, Resource Min LOD, and Min Array Element of the associated surface must

Programming Note
be 0. The Min and Mag Mode Filter must be MAPFILTER_NEAREST or MAPFILTER_LINEAR. The Mip Mode Filter must be MIPFILTER_NONE. The TCX and TCY Address Control Mode cannot be any of: • TEXCOORDMODE_CLAMP_BORDER • TEXCOORDMODE_HALF_BORDER • TEXCOORDMODE_MIRROR_ONCE • TEXCOORDMODE_WRAP Map Width must be >= 4. Map Height must be >= 4 • Number of Multisamples on the associated surface must be MULTISAMPLECOUNT_1.
Supports additional Surface formats: PLANAR_420_16, R16_UNORM, R16G16_UNORM, R16G16B16A16_UNORM
Map Width must be >=16 Map Height must be >=16

sample_8x8 Message Type

sample_8x8 Message Definition

Message Type	Description
sample_8x8	The surface is sampled using an optional 8x8 filter, using state defined in SAMPLER_STATE and SAMPLER_8x8_STATE. The input consists of 64 contiguous pixels in a 16-wide by 4-high arrangement. The address control mode behaves as clamp mode. The U and V addresses for the upper left pixel are delivered in this message along with a Delta U and Delta V parameter. Given a pixel at (x,y) relative to the upper left pixel (where (0,0) is the upper left pixel), the U and V for that pixel are computed as follows: U(x,y) = U(0,0) + DeltaU * x + U_2nd_Derivative * x * (x - 1)/2 V(x,y) = V(0,0) + DeltaV * y + V_2nd_Derivative * y * (y - 1)/2

Supported Variants:

None.

Restrictions and Programming Notes for sample_8x8:

Programming Note
The Surface Type of the associated surface must be SURFTYPE_2D. The Surface Format of the associated surface must be UNORM with <= 10 bits per channel. *DeltaV * Height* of the associated surface must be less than 16.0. • *Map Width* must be >= 4.
Supports functionality such as convolve, MinMax, MinMaxFilter, Dilate, Erode, BoolCentroid, and Centroid have been added on top of the existing sample_8x8. The convolve, MinMax, and MinMaxFilter are 16-wide X 4-high; the rest have variable size depending on the message. See Media Sampler section for a description of how these modes work.

Message Format

The following section describes how messages are formatted, including how message length and response length are calculated. The sampler supports a variety of different message types which perform different sampling and filtering operations (see Message Types section for a description of these sampler and filtering operations). For each message type, different SIMD modes are supported which determines how many pixels the sampler performs the specified operation on in parallel. The precision of the input parameters and resulting samples can also be controlled by programming the **Return Format** bit in the message descriptor (for sample precision) and using SIMD8H/SIMD8 and SIMD16H/SIMD16 message types (for coordinate precision).

Supported SIMD Types

The table below describes the SIMD modes which are supported. SIMD32 and SIMD64 are used for media-type operations only.

List of Supported Message SIMD Types

SIMD Modes	Description
SIMD8	8 samples with 32-bit Coordinates
SIMD16	16 samples with 32-bit coordinates
SIMD32/64	Media Operations on 32 or 64 pixels

Message Length

The **SIMD Mode** field determines the number of instances (i.e. pixels) to be sampled and the formatting of messages. The **Message Length** field indicates the number of parameters sent with the message. Higher-numbered parameters are optional, and default to a value of 0 if not sent but needed for the surface being sampled. The **Header Present** field determines whether a header is delivered as the first phase of the message or the default header from R0 of the thread's dispatch is used. A header will

increase the length of the message by 1. Note that messages with more than 4 parameters will perform lower due to the additional information being sent to the sampler than messages with 4 or fewer parameters. For example, a sample message using 5 parameters will not be able to sustain the same throughput as a sample message with only 4 valid parameters.

Programming Note
All messages except sampleinfo or cache_flush that had a header must have at least one parameter in addition to the header.

Programming Note	
Context:	3D Sampler Messages - Message Format
The Cache_Flush message is SIMD32, but has no payload, only a header. So, the length is 1 for Cache_Flush	

Response Length

The Response Length field determines the number of 256-bit registers which are used to receive the result of each message type for each SIMD mode. The value of k shown in the response length is 1 if killpix is enabled and 0 otherwise.

SIMD Mode	Return Precision (bits per channel)	Response Length
SIMD8	32	4+k
SIMD8	16	2**
SIMD16	32	8*
SIMD16	16	4*

Notes for Determining Response Lengths Table

Symbol	Note
*	For SIMD16, phases in the response length are reduced by 2 for each channel that is masked. SIMD16 messages with six or more parameters exceed the maximum message length allowed, in which case they are not supported. This includes some forms of sample_b_c, sample_l_c, and gather4_po_c message types. Note that even for these messages, if 5 or fewer parameters are included in the message, the SIMD16 form of the message is allowed. SIMD16 forms of sample_d and sample_d_c are not allowed, regardless of the number of parameters sent.
**	For SIMD8*, phases in the response length are reduced by 1 for each channel that is masked.
***	All SIMD16H messages have the same limitations as the corresponding SIMD16 message would, even though maximum message length is not exceeded. The SIMD16H version of ld2dms_w is an exception. It allows up to 7 parameters.

SIMD32/SIMD64 Special Cases for Response Length

The SIMD32/SIMD64 modes have special response lengths depending on the message type.

Mnemonic	Payload Layout	Response Length
sample_unorm	Pixel Shader	8**
sample_unorm + killpix	Pixel Shader	9**
deinterlace	Pixel Shader	+
sample_unorm	Media	8**
sample_unorm + killpix	Media	9**
sample_8X8	Media	16*
cache_flush	no payload	1

** For sample_unorm, phases in the response length are reduced by 2 for each channel that is masked.

* For sample_8x8, phases in the response length are reduced by 4 for each channel that is masked.

+ For deinterlace, response length depending on certain state fields. Refer to writeback message definition for details.

If **Header Present** is *disabled*, **Response Length** can be set to values in the table below for SIMD8 and SIMD16 messages other than *sample+killpix*. The setting of **Response Length** determines the **Write Channel Mask** fields that are used by the hardware according to the table below. **Response Length** values not indicated in the table are not valid.

SIMD Mode	Return Format	Response Length	Alpha Write Channel Mask	Blue Write Channel Mask	Green Write Channel Mask	Red Write Channel Mask
SIMD8*	32-bit/ 16-bit	1	1	1	1	0
	32-bit/ 16-bit	2	1	1	0	0
	32-bit/ 16-bit	3	1	0	0	0
	32-bit/ 16-bit	4	0	0	0	0
SIMD16*	32-bit	2	1	1	1	0
	16-bit	1	1	1	1	0
	32-bit	4	1	1	0	0
	16-bit	2	1	1	0	0
	32-bit	6	1	0	0	0
	16-bit	3	1	0	0	0
	16-bit	4	0	0	0	0
	32-bit	8	0	0	0	0

If **Pixel Fault Mask Enable** is enabled, response length must be set to a value one larger than that indicated in the tables above.

Programming Note	
Context:	3D Sampler Messages - Message Format
Parameter 0 is required.	

Message Formats

The table below describes the format of each message type for every supported SIMD type.

=The Message Type field in the message descriptor in combination with the Message Length to determine which message is being sent. The table defines all of the parameters sent for each message type. The position of the parameters in the payload is given in the section following corresponding to the *SIMD mode* given in the table. The Mnemonic column specifies the DX10-equivalent shader instructions expected to be translated to each message type.

Message parameters are typically formed in sequential 256-bit registers and sent in order to the sampler to form a complete message. All parameters are of type IEEE_Float, except those in the The ld*, resinfo, and the offu, offv of the gather4_po[_c] instruction message types, which are of type signed integer. Any parameter indicated with a blank entry in the table is unused. A message register containing only unused parameters is not included as part of the message.

The table below shows the supported messages and their formats for SIMD8, SIMD16, SIMD8H and SIMD16H:

Encoding	Mnemonic	Parameters											Comment
		0	1	2	3	4	5	6	7	8	9	10	
00000	sample	u	v	r	ai	mlod							
00001	sample_b	bias	u	v	r	ai	mlod						
00010	sample_l	lod	u	v	r	ai							
00011	sample_c	ref	u	v	r	ai	mlod						
00100	sample_d	u	dudx	dudy	v	dvdx	dvdy	r	drdx	drdy	ai	mlod	
00101	sample_b_c	ref	bias	u	v	r	ai						
00110	sample_l_c	ref	lod	u	v	r	ai						
00111	ld	u	v	lod	r								
01000	gather4	u	v	r	ai								
01001	LOD	u	v	r	ai								
01010	resinfo	lod											
01011	sampleinfo												
01100	sample+killpix	u	v	r									
10000	gather4_c	ref	u	v	r	ai							
10001	gather4_po	u	v	offu	offv	r							

Encoding	Mnemonic	Parameters											Comment
10010	gather4_po_c	ref	u	v	offu	offv	r						
10100	sample_d_c	ref	u	dudx	dudy	v	dvdx	dvdy	r	drdx	drdy	ai	
11000	sample_lz	u	v	r	ai								
11001	sample_c_lz	ref	u	v	r	ai							
11010	ld_lz	u	v	r									
11100	ld2dms_w	si	mcsl	mcsh	u	v	r	lod					SIMD8and SIMD16 Only
11100	ld2dms_w	si	mcs0	mcs1	mcs2	mcs3	u	v	r	lod			SIM8H and SIMD16H Only
11101	ld_mcs	u	v	r	lod								SIMD8 and SIMD16 Only
11101	ld_mcs	u	v	r	lod								SIMD8H and SIMD16H Only
11110	ld2dms	si	mcs	u	v	r	lod						
11111	ld2dss	ssi	u	v	r	lod							

The table below shows the supported messages and their formats for SIMD32/SIMD64 Messages:

Encoding	Mnemonic	Comment
00000	sample_unorm	
00001	sample_unorm+killpix	
01000	deinterlace	
01010	sample_unorm	
01011	sample_unorm + killpix	
01100	sample_8x8	
11111	cache_flush	

Programming Note	
Context:	3D Sampler Messages - Message Format
The ld, ld_mcs, ld2dss, resinfo, gather4_po utilize integer operands and must not be used with SIMD8H SIMD16H which are for 16-bit floating-point operands.	

Programming Note	
Context:	3D Sampler Messages - Message Format
For output control modes other than "16 bit Full", the response lengths are reduced accordingly. Refer to the writeback message format definitions to determine the correct response length for each case.	

Programming Note	
Context:	Min LOD Parameter
To ensure optimal performance, the Min LOD (mlod) parameter should not be included in a *sample* or *sample_b* message unless one or more of the subspans requires a non-zero Min LOD.	

Parameter Types

sample*, LOD, and gather4 messages

For all of the sample*, LOD, and gather4 message types, all parameters are 32-bit or 16-bit floating point, except the 'mcs', 'offu', and 'offv' parameters. Usage of the u, v, and r parameters is as follows based on **Surface Type**. Normalized values range from [0,1] across the surface, with values outside the surface behaving as specified by the **Address Control Mode** in that dimension. Unnormalized values range from [0,n-1] across the surface, where n is the size of the surface in that dimension, with values outside the surface being clamped to the surface.

Surface Type	u	v	r	ai
SURFTYPE1D	normalized 'x' coordinate	unnormalized array index	ignored	ignored
SURFTYPE_2D	normalized 'x' coordinate	normalized 'y' coordinate	unnormalized array index	ignored
SURFTYPE_3D	normalized 'x' coordinate	normalized 'y' coordinate	normalized 'z' coordinate	ignored
SURFTYPE_CUBE	normalized 'x' coordinate	normalized 'y' coordinate	normalized 'z' coordinate	unnormalized array index

Ld* messages

For the ld message types, all parameters are 32-bit or 16-bit unsigned integers (Ld must be 32-bit), except the 'mcs' parameter. Usage of the u, v, and r parameters is as follows based on **Surface Type**. Unnormalized values range from [0,n-1] across the surface, where n is the size of the surface in that dimension. Input of any value outside of the range returns zero.

Surface Type	u	v	r
SURFTYPE1D	unnormalized 'x' coordinate	unnormalized array index	ignored
SURFTYPE_2D	unnormalized 'x' coordinate	unnormalized 'y' coordinate	unnormalized array index
SURFTYPE_3D	unnormalized 'x' coordinate	unnormalized 'y' coordinate	unnormalized 'z' coordinate
SURFTYPE_BUFFER	unnormalized 'x' coordinate	ignored	ignored

SIMD Payloads

This section contains the SIMD payload definitions.

SIMD16 Payload

The payload of a SIMD16 message provides addresses for the sampling engine to process 16 entities (examples of an entity are vertex and pixel). The number of parameters required to sample the surface depends on the state that the sampler/surface is in. Each parameter takes two message registers, with 8 entities, each a 32-bit floating point value, being placed in each register. Each parameter always takes a consistent position in the input payload. The length field can be used to send a shorter message, but intermediate parameters cannot be skipped as there is no way to signal this. For example, a 2D map using "sample_b" needs only u, v, and bias, but must send the r parameter as well.

DWord	Bits	Description
M1.7	31:0	**Subspan 1, Pixel 3 (lower right) Parameter 0** Specifies the value of the pixel's parameter 0. The actual parameter that maps to parameter 0 is given in the table in the section. Format = IEEE Float for all sample* message types, U32 for ld and resinfo message types.
M1.6	31:0	**Subspan 1, Pixel 2 (lower left) Parameter 0**
M1.5	31:0	**Subspan 1, Pixel 1 (upper right) Parameter 0**
M1.4	31:0	**Subspan 1, Pixel 0 (upper left) Parameter 0**
M1.3	31:0	**Subspan 0, Pixel 3 (lower right) Parameter 0**
M1.2	31:0	**Subspan 0, Pixel 2 (lower left) Parameter 0**
M1.1	31:0	**Subspan 0, Pixel 1 (upper right) Parameter 0**
M1.0	31:0	**Subspan 0, Pixel 0 (upper left) Parameter 0**
M2.7	31:0	**Subspan 3, Pixel 3 (lower right) Parameter 0**
M2.6	31:0	**Subspan 3, Pixel 2 (lower left) Parameter 0**
M2.5	31:0	**Subspan 3, Pixel 1 (upper right) Parameter 0**
M2.4	31:0	**Subspan 3, Pixel 0 (upper left) Parameter 0**
M2.3	31:0	**Subspan 2, Pixel 3 (lower right) Parameter 0**
M2.2	31:0	**Subspan 2, Pixel 2 (lower left) Parameter 0**
M2.1	31:0	**Subspan 2, Pixel 1 (upper right) Parameter 0**

DWord	Bits	Description
M2.0	31:0	**Subspan 2, Pixel 0 (upper left) Parameter 0**
M3 - Mn		Repeat packets 1 and 2 to cover all required parameters.

SIMD8 Payload

This message is intended to be used in a SIMD8 thread, or in pairs from a SIMD16 thread. Each message contains sample requests for just 8 pixels.

DWord	Bits	Description
M1.7	31:0	**Subspan 1, Pixel 3 (lower right) Parameter 0** Specifies the value of the pixel's parameter 0. The actual parameter that maps to parameter 0 is given in the table in the section. Format = IEEE Float for all sample* message types, U32 for ld and resinfo message types.
M1.6	31:0	**Subspan 1, Pixel 2 (lower left) Parameter 0**
M1.5	31:0	**Subspan 1, Pixel 1 (upper right) Parameter 0**
M1.4	31:0	**Subspan 1, Pixel 0 (upper left) Parameter 0**
M1.3	31:0	**Subspan 0, Pixel 3 (lower right) Parameter 0**
M1.2	31:0	**Subspan 0, Pixel 2 (lower left) Parameter 0**
M1.1	31:0	**Subspan 0, Pixel 1 (upper right) Parameter 0**
M1.0	31:0	**Subspan 0, Pixel 0 (upper left) Parameter 0**
M2 - Mn		Repeat packet 1 to cover all required parameters.

SIMD16H Payload

The payload of a SIMD16H message provides addresses for the sampling engine to process 16 entities (examples of an entity are vertex and pixel). The number of parameters required to sample the surface depends on the state that the sampler/surface is in. Each parameter takes one message register, with 16 entities, each a 16-bit floating point or integer value, being placed in each register. Each parameter always takes a consistent position in the input payload. The length field can be used to send a shorter message, but intermediate parameters cannot be skipped as there is no way to signal this.

DWord	Bits	Description
M1.7	31:16	**Subspan 3, Pixel 3 (lower right) Parameter 0**

DWord	Bits	Description
		Specifies the value of the pixel's parameter 0. The actual parameter that maps to parameter 0 is given in the table in the section. Format = IEEE Half Float for all sample* message types, U16 for ld and resinfo message types.
	15:0	**Subspan 3, Pixel 2 (lower left) Parameter 0**
M1.6	31:16	**Subspan 3, Pixel 1 (upper right) Parameter 0**
	15:0	**Subspan 3, Pixel 0 (upper left) Parameter 0**
M1.5	31:16	**Subspan 2, Pixel 3 (lower right) Parameter 0**
	15:0	**Subspan 2, Pixel 2 (lower left) Parameter 0**
M1.4	31:16	**Subspan 2, Pixel 1 (upper right) Parameter 0**
	15:0	**Subspan 2, Pixel 0 (upper left) Parameter 0**
M1.3	31:16	**Subspan 1, Pixel 3 (lower right) Parameter 0**
	15:0	**Subspan 1, Pixel 2 (lower left) Parameter 0**
M1.2	31:16	**Subspan 1, Pixel 1 (upper right) Parameter 0**
	15:0	**Subspan 1, Pixel 0 (upper left) Parameter 0**
M1.1	31:16	**Subspan 0, Pixel 3 (lower right) Parameter 0**
	15:0	**Subspan 0, Pixel 2 (lower left) Parameter 0**
M1.0	31:16	**Subspan 0, Pixel 1 (upper right) Parameter 0**
	15:0	**Subspan 0, Pixel 0 (upper left) Parameter 0**
M2 - Mn		Repeat packets 1 and 2 to cover all required parameters.

SIMD8H Payload

This message is intended to be used in a SIMD8 thread, or in pairs from a SIMD16 thread. Each message contains sample requests for just 8 pixels.

DWord	Bits	Description
M1.7:4		**Reserved**
M1.3	31:16	**Subspan 1, Pixel 3 (lower right) Parameter 0** Specifies the value of the pixel's parameter 0. The actual parameter that maps to parameter 0 is given in the table in the section. Format = IEEE Half Float for all sample* message types, U16 for ld and resinfo message types.
	15:0	**Subspan 1, Pixel 2 (lower left) Parameter 0**
M1.2	31:16	**Subspan 1, Pixel 1 (upper right) Parameter 0**
	15:0	**Subspan 1, Pixel 0 (upper left) Parameter 0**
M1.1	31:16	**Subspan 0, Pixel 3 (lower right) Parameter 0**
	15:0	**Subspan 0, Pixel 2 (lower left) Parameter 0**
M1.0	31:16	**Subspan 0, Pixel 1 (upper right) Parameter 0**

DWord	Bits	Description
	15:0	**Subspan 0, Pixel 0 (upper left) Parameter 0**
M2 - Mn		Repeat packet 1 to cover all required parameters.

Writeback Message

Corresponding to the four input message definitions are four writeback messages. Each input message generates a corresponding writeback message of the same type.

Programming Note	
Context:	3D Sampler Writeback Message
The Pixel Null Mask field, when enabled via the **Pixel Null Mask Enable** will be incorect for sample_c when applied to a surface with 64-bit per texel format such as R16G16BA16_UNORM. Pixel Null mask Enable may incorrectly report pixels as referencing a Null surface.	

Programming Note	
Context:	Returning Filter Weights
When the **Return Filter Weight for Border Texels** or the **Return Filter Weight for Null Texels** bits in Sampler Message Descriptor are set, the 3D Sampler will return a filter weight in the Alpha Channel.	

SIMD16

Return Format = 32-bit

A SIMD16 writeback message consists of up to 8 destination registers. Which registers are returned is determined by the write channel mask received in the corresponding input message. Each asserted write channel mask results in both destination registers of the corresponding channel being skipped in the writeback message, and all channels with higher numbered registers being dropped down to fill in the space occupied by the masked channel. For example, if only red and alpha are enabled, red is sent to regid+0 and regid+1, and alpha to regid+2 and regid+3. The pixels written within each destination register is determined by the execution mask on the "send" instruction.

If Pixel Null Mask Enable is enabled an additional register is included after the last channel register. Behavior of pixels that had a null texel contribution depends on the setting of **Null Pixel Behavior**.

DWord	Bit	Description
W0.7	31:0	**Subspan 1, Pixel 3 (lower right) Red:** Specifies the value of the pixel's red channel. Format = IEEE Float, S31 signed 2's comp integer, or U32 unsigned integer.
W0.6	31:0	**Subspan 1, Pixel 2 (lower left) Red**
W0.5	31:0	**Subspan 1, Pixel 1 (upper right) Red**
W0.4	31:0	**Supspan 1, Pixel 0 (upper left) Red**
W0.3	31:0	**Subspan 0, Pixel 3 (lower right) Red**
W0.2	31:0	**Subspan 0, Pixel 2 (lower left) Red**
W0.1	31:0	**Subspan 0, Pixel 1 (upper right) Red**
W0.0	31:0	**Supspan 0, Pixel 0 (upper left) Red**
W1.7	31:0	**Subspan 3, Pixel 3 (lower right) Red**
W1.6	31:0	**Subspan 3, Pixel 2 (lower left) Red**
W1.5	31:0	**Subspan 3, Pixel 1 (upper right) Red**
W1.4	31:0	**Supspan 3, Pixel 0 (upper left) Red**
W1.3	31:0	**Subspan 2, Pixel 3 (lower right) Red**
W1.2	31:0	**Subspan 2, Pixel 2 (lower left) Red**
W1.1	31:0	**Subspan 2, Pixel 1 (upper right) Red**
W1.0	31:0	**Supspan 2, Pixel 0 (upper left) Red**
W2		**Subspans 1 and 0 of Green:** See W0 definition for pixel locations
W3		**Subspans 3 and 2 of Green:** See W1 definition for pixel locations
W4		**Subspans 1 and 0 of Blue:** See W0 definition for pixel locations
W5		**Subspans 3 and 2 of Blue:** See W1 definition for pixel locations

DWord	Bit	Description
W6		**Subspans 1 and 0 of Alpha:** See W0 definition for pixel locations
W7		**Subspans 3 and 2 of Alpha:** See W1 definition for pixel locations
W8.7:1		Reserved (not written): W8 is only delivered when **Pixel Fault Mask Enable or Sample Tap Discard** is enabled.
W8.0	31:16	**Border Color Mask:** This field has the bits for all pixels set to 1 except those pixels in which a border color was source for at least one texel.
W8.0	15:0	**Pixel Null Mask:** This field has the bit for all pixels set to 1 except those pixels in which a null page was source for at least one texel.

Return Format = 16-bit

A SIMD16 writeback message with **Return Format** of 16-bit consists of up to 4 destination registers. Which registers are returned is determined by the write channel mask received in the corresponding input message. Each asserted write channel mask results in both destination registers of the corresponding channel being skipped in the writeback message, and all channels with higher numbered registers being dropped down to fill in the space occupied by the masked channel. For example, if only red and alpha are enabled, red is sent to regid+0, and alpha to regid+1. The pixels written within each destination register is determined by the execution mask on the "send" instruction.

If Pixel Null Mask Enable is enabled an additional register is included after the last channel register. Behavior of pixels that had a null texel contribution depends on the setting of **Null Pixel Behavior**.

DWord	Bit	Description
W0.7	31:16	**Subspan 3, Pixel 3 (lower right) Red:** Specifies the value of the pixel's red channel. Format = IEEE Half Float, S15 signed 2's comp integer, or U16 unsigned integer.
	15:0	**Subspan 3, Pixel 2 (lower left) Red**
W0.6	31:16	**Subspan 3, Pixel 1 (upper right) Red**
	15:0	**Supspan 3, Pixel 0 (upper left) Red**
W0.5	31:16	**Subspan 2, Pixel 3 (lower right) Red**
	15:0	**Subspan 2, Pixel 2 (lower left) Red**
W0.4	31:16	**Subspan 2, Pixel 1 (upper right) Red**
	15:0	**Supspan 2, Pixel 0 (upper left) Red**
W0.3	31:16	**Subspan 1, Pixel 3 (lower right) Red**
	15:0	**Subspan 1, Pixel 2 (lower left) Red**
W0.2	31:16	**Subspan 1, Pixel 1 (upper right) Red**
	15:0	**Supspan 1, Pixel 0 (upper left) Red**
W0.1	31:16	**Subspan 0, Pixel 3 (lower right) Red**
	15:0	**Subspan 0, Pixel 2 (lower left) Red**

DWord	Bit	Description
W0.0	31:16	**Subspan 0, Pixel 1 (upper right) Red**
	15:0	**Supspan 0, Pixel 0 (upper left) Red**
W1		**Green:** See W0 definition for pixel locations
W2		**Blue:** See W0 definition for pixel locations
W3		**Alpha:** See W0 definition for pixel locations
W4.7:1		Reserved (not written): W4 is only delivered when **Pixel Fault Mask Enable** is enabled.
W4.0	31:16	**Border Color Mask:** This field has the bits for all pixels set to 1 except those pixels in which a border color was source for at least one texel.
W4.0	15:0	**Pixel Null Mask:** This field has the bit for all pixels set to 1 except those pixels in which a null page was source for at least one texel.

SIMD8

Return Format = 32-bit

A SIMD8* writeback message consists of up to 4 destination registers (5 in the case of sample+killpix). Which registers are returned is determined by the write channel mask received in the corresponding input message. Each asserted write channel mask results in the destination register of the corresponding channel being skipped in the writeback message, and all channels with higher numbered registers being dropped down to fill in the space occupied by the masked channel. For example, if only red and alpha are enabled, red is sent to regid+0, and alpha to regid+1. The pixels written within each destination register is determined by the execution mask on the *send* instruction.

For the sample+killpix message types, an additional register (W4) is included after the last channel register.

If **Pixel Null Mask Enable** is enabled an additional register is included after the last channel register. Behavior of pixels that had a null texel contribution depends on the setting of **Null Pixel Behavior**.

DWord	Bits	Description
W0.7	31:0	**Subspan 1, Pixel 3 (lower right) Red:** Specifies the value of the pixel's red channel. Format = IEEE Float, S31 signed 2's comp integer, or U32 unsigned integer.
W0.6	31:0	**Subspan 1, Pixel 2 (lower left) Red**
W0.5	31:0	**Subspan 1, Pixel 1 (upper right) Red**
W0.4	31:0	**Supspan 1, Pixel 0 (upper left) Red**
W0.3	31:0	**Subspan 0, Pixel 3 (lower right) Red**
W0.2	31:0	**Subspan 0, Pixel 2 (lower left) Red**

DWord	Bits	Description
W0.1	31:0	**Subspan 0, Pixel 1 (upper right) Red**
W0.0	31:0	**Supspan 0, Pixel 0 (upper left) Red**
W1		**Subspans 1 and 0 of Green:** See W0 definition for pixel locations
W2		**Subspans 1 and 0 of Blue:** See W0 definition for pixel locations
W3		**Subspans 1 and 0 of Alpha:** See W0 definition for pixel locations
W4.7:1		Reserved (not written) **:** This W4 is only delivered for the sample+killpix message or when **Pixel Null Mask Enable** in Message Header is set.
W4.0	31:16	**Dispatch Pixel Mask:** For his field is always 0xffff to allow dword-based ANDing with the R0 header in the pixel shader thread for sample+killpix messages.
W4.0	15:0	**Active Pixel Mask:** For sample+killpix messages this field has the bit for all pixels set to 1 except those pixels that have been killed as a result of chroma key with kill pixel mode. Since the SIMD8 message applies to only 8 pixels, only the low 8 bits within this field are used. The high 8 bits are always set to 1.
W4.0	31:24	Reserved: always written as 0xff when **Pixel Null Mask Enable** in Message Header is set.
W4.0	23:16	**Border Color Mask:** This field has the bits for all pixels set to 1 except those pixels in which a border color was source for at least one texel.
W4.0	15:8	Reserved: always written as 0xff when **Pixel Null Mask Enable** in Message Header is set.
W4.0	7:0	**Pixel Null Mask:** This field has the bit for all pixels set to 1 except those pixels in which a null page was source for at least one texel when **Pixel Null Mask Enable** in Message Header is set.

Return Format = 16-bit

A SIMD8* writeback message with **Return Format** of 16-bit consists of up to 4 destination registers). Which registers are returned is determined by the write channel mask received in the corresponding input message. Each asserted write channel mask results in the destination register of the corresponding channel being skipped in the writeback message, and all channels with higher numbered registers being dropped down to fill in the space occupied by the masked channel. For example, if only red and alpha are enabled, red is sent to regid+0, and alpha to regid+1. The pixels written within each destination register is determined by the execution mask on the send instruction.

If Pixel Null Mask Enable is enabled an additional register is included after the last channel register. Behavior of pixels that had a null texel contribution depends on the setting of **Null Pixel Behavior**.

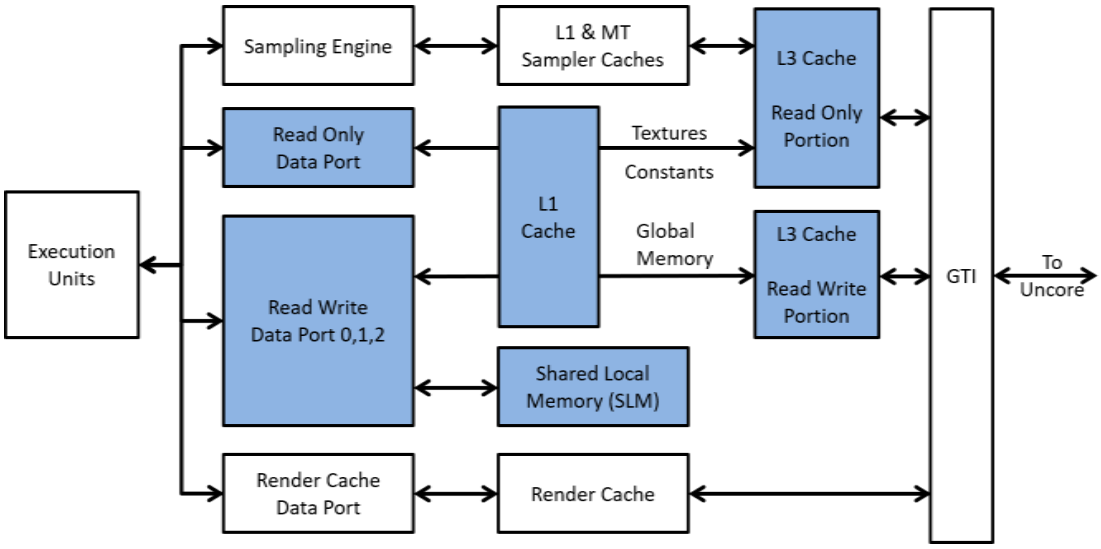
DWord	Bits	Description
W0.7:4		Reserved
W0.3	31:16	**Subspan 1, Pixel 3 (lower right) Red:** Specifies the value of the pixel's red channel. Format = IEEE Half Float, S15 signed 2's comp integer, or U16 unsigned integer.
	15:0	**Subspan 1, Pixel 2 (lower left) Red**
W0.2	31:16	**Subspan 1, Pixel 1 (upper right) Red**
	15:0	**Supspan 1, Pixel 0 (upper left) Red**
W0.1	31:16	**Subspan 0, Pixel 3 (lower right) Red**
	15:0	**Subspan 0, Pixel 2 (lower left) Red**
W0.0	31:16	**Subspan 0, Pixel 1 (upper right) Red**
	15:0	**Supspan 0, Pixel 0 (upper left) Red**
W1		**Subspans 1 and 0 of Green:** See W0 definition for pixel locations
W2		**Subspans 1 and 0 of Blue:** See W0 definition for pixel locations
W3		**Subspans 1 and 0 of Alpha:** See W0 definition for pixel locations
W4.7:1		Reserved (not written): This W4 is only delivered when **Pixel Fault Mask Enable** is enabled.
W4.0	31:24	Reserved: always written as 0xffff
W4.0	23:16	**Border Color Mask:** This field has the bits for all pixels set to 1 except those pixels in which a border color was source for at least one texel.
W4.0	15:8	Reserved: always written as 0xff
W4.0	7:0	**Pixel Null Mask:** This field has the bit for all pixels set to 1 except those pixels in which a null page was source for at least one texel.

Data Port

The Data Port provides all memory accesses for the subsystem other than those provided by the sampling engine. These include constant buffer reads, scratch space reads/writes, and media surface accesses. Render Target accesses to the Render Cache are described in the *Shared Functions Render Target* chapter.

The diagram below shows how the Read-Only and Read/Write Data Ports connect with the caches and memory subsystem. The execution units and sampling engine are shown for clarity.

Data Port Connections to Caches and Memory



The kernel programs running in the execution units communicate with the data port via messages, the same as for the other shared function units. The data ports are considered to be separate shared functions, each with its own shared function identifier.

Read/Write Ordering

Memory access ordering in subsystem programs is managed inside of each thread. Memory access ordering between threads is managed by software because threads can be run in any order, and another thread can run when a thread is blocked waiting for data.

Ordering Conditions
Read, write, and atomic operations issued from the same thread to the same address, using the same data port, and the same cache coherency type, are guaranteed to be processed in the same order as they are issued. Software mechanisms are used to ensure ordering of accesses when issued from different threads, to different addresses, with different data ports, or with different cache coherency types. So within the same thread, memory accesses may not be ordered under some circumstances . For example: • The order that a single SIMD data port read or write message performs its multiple memory accesses is not guaranteed. • If two different virtual addresses map to the same physical memory address, the order of the operations is not guaranteed. • If a read misses a data cache and needs to fetch data from system memory, an access to a different address can complete earlier if it hits a data cache.

When memory access ordering is required and not guaranteed, either in the same thread or with different threads, software must ensure the ordering using fence operations or atomic operations. The Memory Fence message can be used to block a thread until all previous messages issued to that data port cache agent have been globally observed from the point of view of other threads in the system. Different threads in the same thread group can use a Memory Fence message guarantee either global memory ordering or SLM memory ordering at the point when all the threads have reached a thread

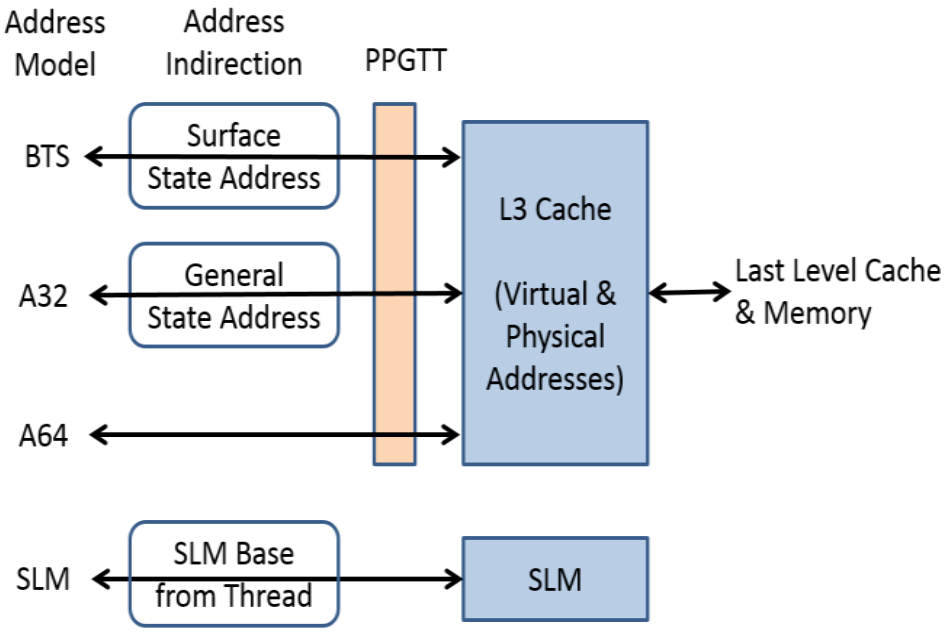
group barrier. Software can use the results of atomic operations to implement in programs a memory ordering between threads using exclusion locks or other protocols.

Address Models

Data structures accessed by the data port are called "surfaces". There are four different Address Models used by the data port to access these surfaces: Binding Table State model (BTS), Shared Local Memory model (SLM), 32-bit Stateless model (A32), and 64-bit Stateless model (A64).

Most messages support only a subset of the Address Models.

Address Models



Surface State Addresses and General State Addresses are translated to virtual Graphics Addresses using the corresponding Base Address in the STATE_BASE_ADDRESS pointers, along with additional base address offset information provided with the Data Port message. SLM addresses are offset by the SLM Base Address that is set with each thread dispatch.

The sections below explain how the Base address is calculated for each Address Model. The Data Port messages then combine that Base address with multiple SIMD offsets for the message's memory operations.

Each calculated memory address is bounds-checked against the Address Model's boundaries. Out-of-bounds read accesses return zero and out-of-bounds write accesses are dropped.

Binding Table Surface State Model (SSM and BTS)

The data port uses the binding table to bind indices to surface state, using the same mechanism used by the sampling engine.

Surface State Address Model Description
The surface state model is used when a Binding Table Index (specified in the message descriptor) of less than 240 is specified, or with the value of 252. When the Binding Table Index is less than 240, the Binding Table Index is used to index into the active binding table, and the binding table entry contains a pointer to the SURFACE_STATE. When the Binding Table Index is 252, the pointer to the SURFACE_STATE is the value of the Surface State Offset (specified in the extended message descriptor).

A Surface State Offset may be used in place of the Binding Table Index and Binding Table Pointer to form what is called a Bindless Surface State address. Throughout this volume, the shorthand terms SSM (Surface State Address Model) and BTS (Binding Table Surface Address Model) are used interchangeably.

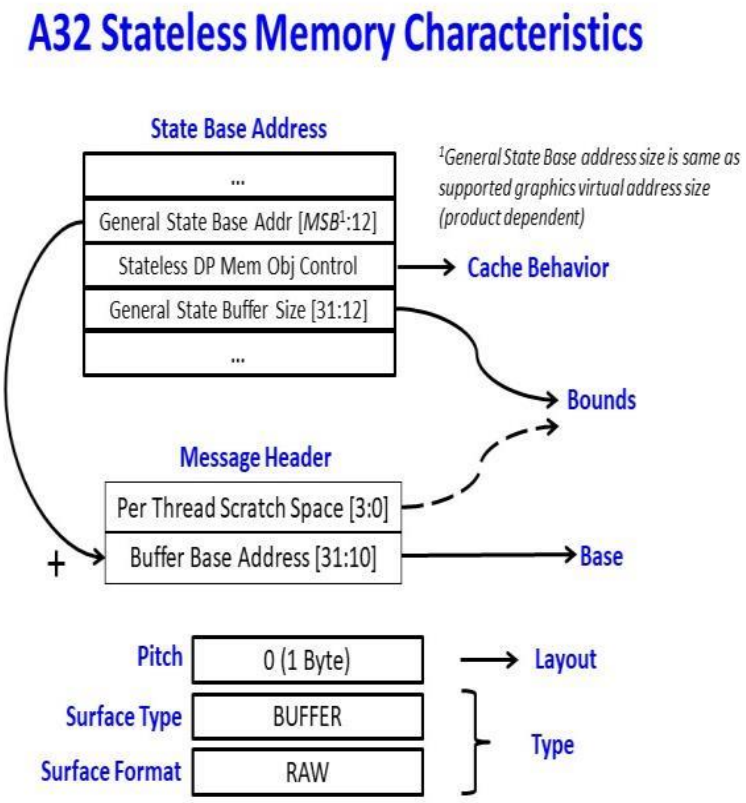
A32 Stateless Model

In the A32 model, the binding table is not accessed, and the parameters that define the surface state are overloaded as follows:

- Surface Type = SURFTYPE_BUFFER
- Surface Format = RAW
- Vertical Line Stride = 0
- Surface Base Address = General State Base Address + (optional) Buffer Base Address
- Buffer Size = compared against General State Buffer Size for bounds checking
- Surface Pitch = 1 byte
- Utilize Fence = false
- Tiled = false

This model is primarily intended for scratch space buffers and for GPGPU global memory accesses.

A32 Stateless Memory Characteristics



Stateless Model Aliases

For products where IA coherency is supported, the stateless aliases 255 and 253 provide a means of SW controlling the coherency properties of an access. The coherency property is ensured for that access only. Typically, SW uses the same coherency type for all access to the same address. Proper fencing is required to ensure that reads and writes are visible. L3UC forces the addressed cache lines out of L3 and the cycles are directly conducted to LLC. This provides a capability for ensuring coherency on a particular location without having to fence all the other cycles.

Binding Table Index	Type	Description
255	IA Coherent	Coherent within GFX and coherent within the entire IA cache memory hierarchy.
253	Non-Coherent	Coherent within GFX for the same cache type (L3, Sampler, or Render).

Programming Note	
Context:	A32 Stateless Model
• The constant, sampler, and render caches are always non-coherent. • A32 messages that are stateless and specified without a BTI (Hword Read/Write and all messages on SFID_DP_DC2) are IA coherent.	

Programming Note
If STATE_COMPUTE_MODE instruction sets Force Non-Coherent Mode, then all A32 accesses by Data Port messages are non-coherent.

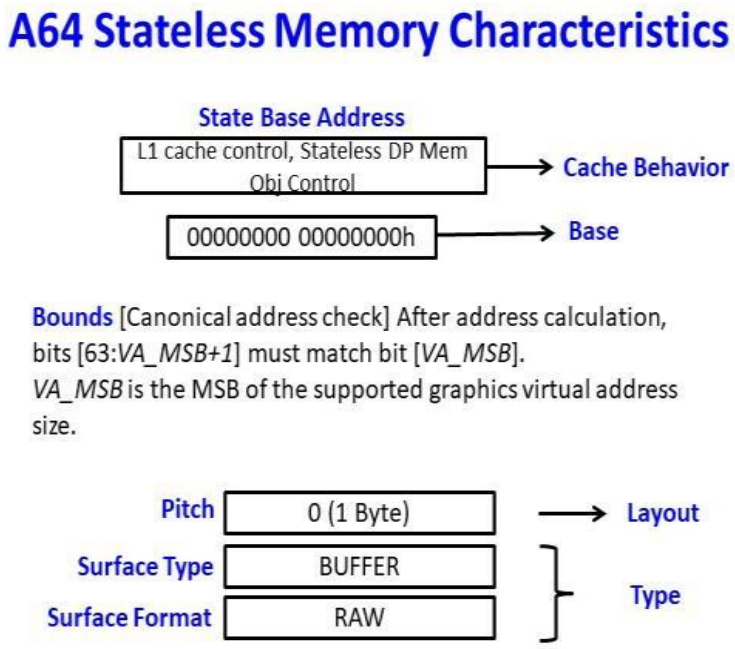
A64 Flat/Stateless Model

In the A64 model, the binding table is not accessed, and the parameters that define the surface state are overloaded as follows:

- Surface Type = SURFTYPE_BUFFER
- Surface Format = RAW
- Vertical Line Stride = 0
- Surface Base Address = Graphics Address
- Buffer Size = not checked
- Surface Pitch = 1 bytes
- Utilize Fence = false
- Tiled = false

This model is primarily intended for programmable shader programs.

A64 Stateless Memory Characteristics



Bounds check for A64 addressing is same making sure the message's 64-bit address offsets satisfies the canonical address check rule: *addr[63:VA_MSB+1]* is same as *addr[VA_MSB]*. The *VA_MSB* value depends on the supported graphics virtual address size in the product (e.g. VA_MSB==47 for 48b virtual address size). The in-bound A64 address ranges based on canonical rule is shown below:

In-bound A64 Address Ranges (canonical rule)

Low Address Range	High Address Range	Notes
0x0 - 0x7fff_ffff_ffff	0xffff_8000_0000_0000 - 0xffff_ffff_ffff_ffff	48b virtual address

If any address offset fails canonical check, all bytes accessed by that offset is treated out-of-bound.

Programming Note	
Context:	A64 Stateless Model
A64 messages that are stateless and specified without a BTI (Hword Read/Write and all messages on SFID_DP_DC2) are IA coherent.	

Programming Note
If STATE_COMPUTE_MODE instruction sets Force Non-Coherent Mode, then all A32 accesses by Data Port messages are non-coherent.

Shared Local Memory (SLM)

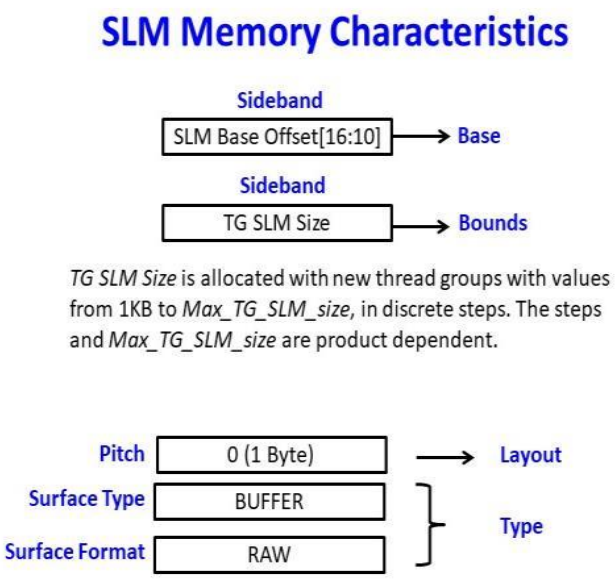
Shared local memory (SLM) is a high bandwidth memory that is not backed up by system memory. The SLM contents are shared between all active threads in the same thread group. Its contents are uninitialized after creation, and its contents disappear when deallocated.

In the SLM model, the binding table is not accessed, and the parameters that define the surface state are overloaded as follows:

- Surface Type = SURFTYPE_BUFFER
- Surface Format = RAW
- Surface Base Address = points to the start of the internal SLM (no memory address is applicable)
- Surface Pitch = 1 byte

Due to the predefined surface state attributes for the SLM, only a subset of the data port messages can be used.

SLM Memory Characteristics



1KB aligned "SLM Base Offset" comes on sideband and is used as the base address for SLM accesses by a message. "TG SLM Size" also comes on sideband and is used for bounds checking of SLM address(es) from the message payload. The allowed "TG SLM Size" values are as follows.

Allowed values of TG SLM Size

TG SLM Size
0KB
1KB
2KB
4KB
8KB
16KB
32KB
64KB

Specifying SLM Access

Programming Note
SLM is accessed for data ports 0 and 1 messages when a Binding Table Index (specified in the message descriptor) of 254 is specified, or by Data Port 2 messages that are not accessing A32 or A64 memory. Only the data cache data port is supported for SLM, the other data ports treat Binding Table Index 254 as a normal surface state access.

Programming Note
Accesses to SLM are bounds checked against the size of the SLM allocation. Addresses beyond the SLM size are out of bounds.
SLM should not be accessed through threads dispatched by the 3D pipe.

Unified Return Buffer Memory

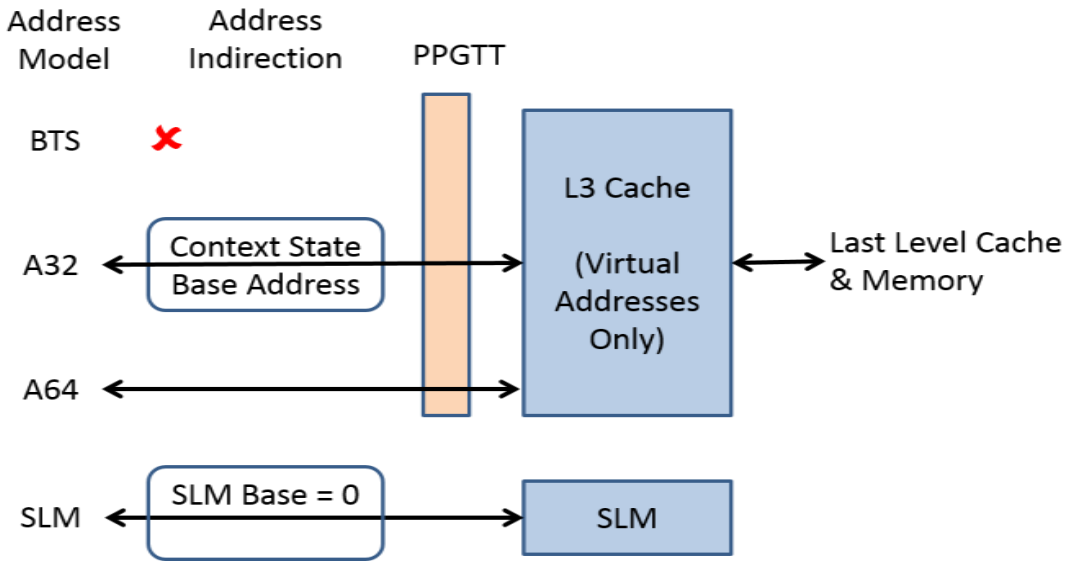
Unified Return Buffer memory (URB) is a high bandwidth memory that is not backed up by system memory. The URB contents are shared between all active threads in the same context. Its contents are uninitialized after creation, and its contents disappear when deallocated.

URB memory is read and written with specialized messages sent to URB Shared Function data port SFID_URB.

Address Models during SIP

When the SIP routine for the thread is running, only the stateless and SLM address models are supported with Data Port messages.

Address Models during SIP



The A32 base address is overridden during the SIP execution to be Dual Subslice's context save area. The SLM base is overridden during the SIP execution to be the Dual Subslice's SLM region.

Stateless A32 and A64 memory accesses are overridden to be non-coherent (L3 cache stores them as virtual addresses).

In addition to the address model restrictions during the SIP execution, only a subset of Data Port messages are defined for use during the SIP execution.

Context Save and Restore Supported Messages for SIP
All A32 stateless messages with BTI =253 override A32 base address with the Context Save area's base address during SIP. This includes the Oword Block and Oword Aligned Block messages, and the Atomic Integer Operation messages. However, Hword Scratch Read/Write messages do not override the A32 base address and should not be used during SIP.

Overview of Memory Accesses

The specific message operation modifies the characteristics of the Address Model to determine the behaviors of each Data Port memory access. An overview of the characteristics is provided in the table below, and then explained in more detail in the following sections.

Characteristic	Overview
Surface Type	Each message has its own requirements for which surface types are supported. Most messages only operate on a restricted subset of surface types. Most Data Port messages only operate on the SURFTYPE_BUFFER surface type, but some specific messages operate on other structured surface types.
Address	Each message has its own requirements for address alignment. Most messages require an address that is aligned on a natural boundary for the width of the data item being accessed.

<table>
<tr><th>Characteristic</th><th>Overview</th></tr>
<tr><td>Alignment</td><td></td></tr>
<tr><td>Address Calculation</td><td>Each message calculates one common Base Offset, and multiple additional Offsets for the SIMD operation.

Depending on the message, the SIMD Offsets can be byte address offsets, or indices that select an address aligned by the message operation's data width.</td></tr>
<tr><td>Bounds Checking</td><td>The per-message Base Address Offset is added to each of the message's additional SIMD Offsets to form offsets. Those offsets are each checked that they are within the surface's and message's boundaries before a memory access is made.

If a memory access is within bounds, each calculated offset is added to the Address model's Base Address to form a virtual, physical, or SLM address that is accessed through the L3 cache structure. If a memory access is out-of-bounds, then the message-specific action occurs (see Bounds Checking in this volume).</td></tr>
<tr><td>Canonical Address Checking</td><td>The calculated virtual memory address, formed from each calculated offset and the Address model's Base Address, could cross the canonical address boundary.

Messages using Binding Table State and A32 stateless address models require the graphics driver to correctly set up the base and bounds parameters so that every in-bounds access does not cross the canonical address boundary.

Messages using A64 Stateless address models detect any calculated offsets that cross the canonical address boundary, either as a fault or as an out-of-bounds access.</td></tr>
<tr><td>Tiled Resources Checking</td><td>The virtual memory address for a data port access may be in the range of addresses that are recognized as Tiled Resources, and that are translated through the TRTT (Tiled Resources Translation Table).

Sparse Tiled Resources support a kind of application-managed virtual memory scheme, where the application tells the driver to map specified 64KB address regions within the surface to memory resources called Tile Pools. Tiles that are not mapped to a Tile Pool are either:
Null tiles, which are treated as Out-of-Bounds
Invalid tiles, which are treated as page faults
Tiles that are mapped to a Tile Pool have a virtual address that is handled like any virtual address that is outside of the Tiled Resources address range.<table><tr><th colspan="2">Programming Note</th></tr><tr><td>Context:</td><td>Cached Read of a Null Tile</td></tr><tr><td colspan="2">If a read from a Null tile gets a cache-hit in a virtually addressed GPU cache, then the read may not return zeroes.</td></tr></table></td></tr>
<tr><td>Fault Handling</td><td>Virtual memory addresses for data ports are translated through the PPGTT. If a fault occurs in either the fault-and-stream or fault-and-halt page fault modes, a fault handler is invoked on the IA processor to potentially correct the fault condition and retry the access. Execution of a faulted</td></tr>
</table>

Characteristic	Overview
	thread is suspended until the IA driver signals the GPU to restart all faulted threads and replay the faulted memory accesses. SLM accesses do not fault. See Read/Write Ordering in this volume for a description of how memory ordering is impacted by a page fault.
Out-of-Bounds handling	If a memory access is marked as out of bounds for the surface or the message, or if a virtual address maps to a Tiled Resource Null page, then a read operation returns 0 and a write operation is dropped.
Cache Behavior	Data port memory values can be cached both inside or outside the GPU. The primary GPU data cache can either be kept coherent with the IA processor with every memory acess, or left non-coherent where software protocols manage the coherency. Each Address Model specifies the coherency model for the read/write data cache ports. • For Surface State accesses: the surface's RENDER_SURFACE_STATE **Coherency Type** field • For A32 and A64 Stateless accesses: the 253 and 255 aliases (see Stateless Model Aliases in A32 Stateless Model in this volume for details) • SLM accesses are localized and not cached The behavior of writing cache data to memory (e.g. write back or write through modes) is specified by the MEMORY_OBJECT_CONTROL_STATE. • For Surface State accesses: the surface's RENDER_SURFACE_STATE **Surface Object Control State** • For A32 and A64 Stateless accesses: the STATE_BASE_ADDRESS **Stateless Data Port Access Memory Object Control** When PPGTT is enabled, the page table entry overrides the LLC/eDRAM cache controls supplied by the MEMORY_OBJECT_CONTROL_STATE.

Surfaces Types

Each message has its own requirements for which surface types are supported. Most messages only operate on a limited subset of surface types. The BUFFER surface type is supported by all the address models. All other surface types are only supported by the BTS address model.

The stateless and SLM address models have a pre-defined surface type, format, and layout. Untyped messages are used with these address models.

The BTS address model is used with either typed or untyped messages. The BTS surface type must be compatible with the expected surface type, format, and layout of the message operation. Untyped messages override the surface format with their specific data type and size. Typed messages require the surface characteristics expected by the message, or the message results are undefined (accesses may be dropped or may return random data).

Surface Types used by Data Port Messages

Message Category	BTS	SLM	A32	A64
Render Target	1D, 2D, 3D, CUBE, NULL	N/A	N/A	N/A
Media	2D, NULL	N/A	N/A	N/A
Typed Surface or Atomic	1D, 2D, 3D, CUBE, BUFFER, NULL	N/A	N/A	N/A
Untyped Surface or Atomic	BUFFER, SCRATCH, NULL	BUFFER	BUFFER	BUFFER
Untyped Scattered or Block	BUFFER, NULL	BUFFER	BUFFER	BUFFER
Scaled Untyped Scattered or Surface	N/A	BUFFER	BUFFER	BUFFER

When a surface state is present, untyped messages ignore the surface format and treat the data as RAW. For example, data is returned from the constant buffer to the GRF without format conversion.

2D, 3D, CUBE and NULL surfaces support YMAJOR tile modes to improve memory locality of nearby accesses. All surface types support LINEAR tile mode. Surfaces with type BUFFER and SCRATCH cannot be tiled. 2D surfaces support XMAJOR tile mode for the Display engine. Stencil format is supported in WMAJOR tile mode for 2D, 3D and CUBE tile formats.

Tile Modes supported by Data Port Surface Types

Surface Type	Linear	Y Major	W Major	X Major
BUFFER	Yes			
SCRATCH	Yes			
1D	Yes			
2D	Yes	Yes	Yes	Yes
CUBE	Yes	Yes	Yes	
3D	Yes	Yes	Yes	
NULL	Yes	Yes	Yes	Yes

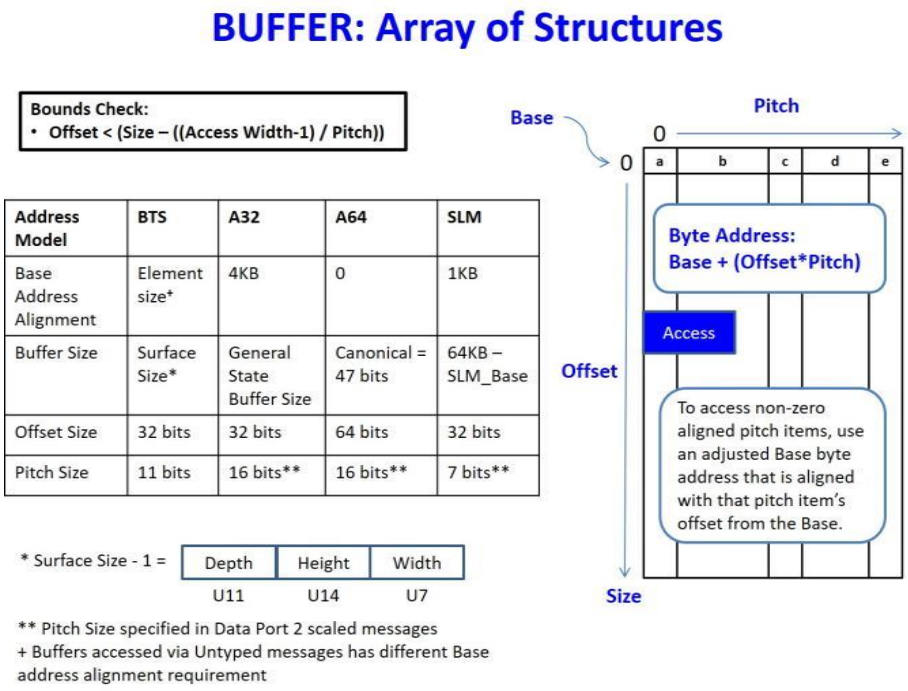
See **Surface Layout and Tiling** in the Memory Views volume for more information about tile layouts.

Programming Note	
Context:	Surfaces Types
For data port messages, compressed surface formats for MSAA and Media surfaces must be Y Major tiled (not Linear, X Major, or W major).	

Addressing Buffers (SURFTYPE_BUFFER)

Most data port messages operate on BUFFER surface types. The figure below illustrates how the base address and bounds calculations are performed on BUFFER surface types.

BUFFER: Array of Structures



Untyped Message U Offset Alignment

For untyped messages accessing SURFTYPE_BUFFER surface types, the U address (byte offset) must be aligned to the data access width (for example, DWords must have the low 2 bits as zeros).

Surface Base Address Alignment for Buffers

For Buffers accessed by Untyped data-port messages, the Base Address must be aligned to MAX(4B, Msg_offset_alignment). I.e. the Surface Base Address has the same alignment requirement as the message's U offsets, but it should also be at least DW aligned.

Surface Size Alignment for Buffers

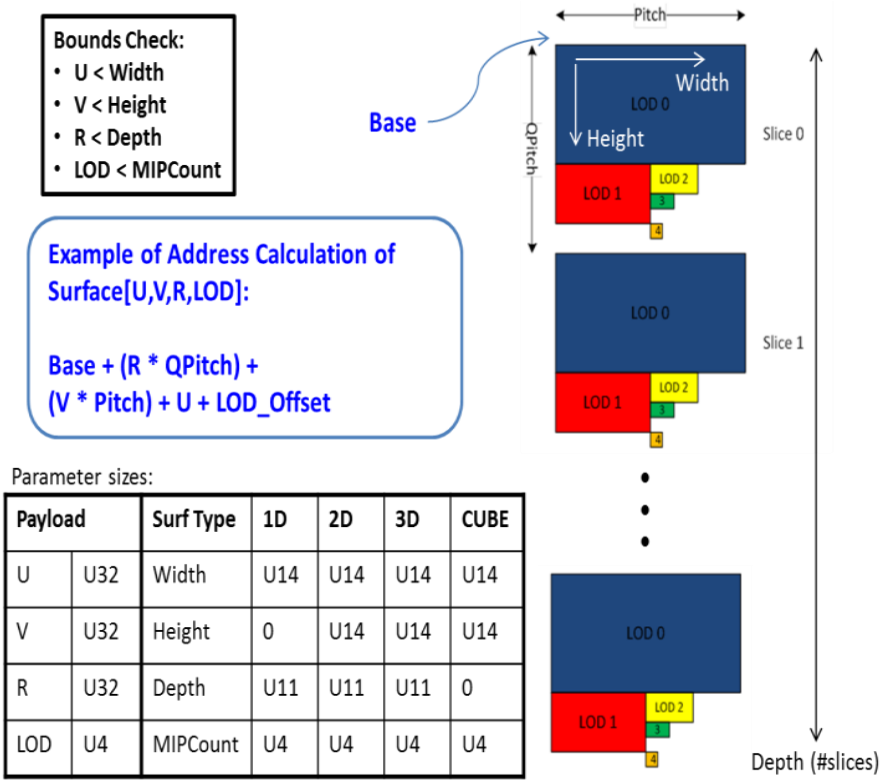
For Buffers accessed by Untyped data-port messages, the Size must be multiple of MAX(4B, Msg_offset_alignment). I.e. the Surface Base Address has the same alignment requirement as the message's U offsets, but it should also be at least DW aligned.

Addressing 1D, 2D, 3D, CUBE Surfaces

Some data port messages operate on various BTS surface types. The addressing and bounds checking for the 1D, 2D, 3D, and CUBE surface types is fully explained in other chapters. See SURFACE_STATE in the Shared Functions volume and **Surface Layout and Tiling** in the GPU Overview volume.

The figure below illustrates the differences in addressing and bound checking between these surface types and the BUFFER and STRBUF surface types.

Multi-Dimensional Surface Buffers



Parameter sizes:

Payload		Surf Type	1D	2D	3D	CUBE
U	U32	Width	U14	U14	U14	U14
V	U32	Height	0	U14	U14	U14
R	U32	Depth	U11	U11	U11	0
LOD	U4	MIPCount	U4	U4	U4	U4

The following table indicates the hardware interpretation of each input address parameter based on surface type. If LOD is specified, then both V and R parameters must also be included, even if they are ignored. If LOD is not specified, then unused V and R parameters are optional in the address payload. If LOD is not present in the address payload, then it has the default value of 0. If the surface state indicates the Number of Multisamples > 1, then the sample number is the first parameter in the address payload.

Address Parameters Used by Surface Type

SURFACE_STATE Fields			Address Payload				
Surface Type	Surface Array	Number of Multisamples	MSAA	U Address	V Address	R Address	LOD
SURFTYPE_1D	disabled	Must be MULTISAMPLECOUNT_1	Never Present	X pixel address	(ignored if present)	(ignored if present)	Optional LOD
	enabled			X pixel address	array index	(ignored if present)	Optional LOD
SURFTYPE_2D	disabled	MULTISAMPLECOUNT_1	Never Present	X pixel address	Y pixel address	(ignored if present)	Optional LOD
	enabled			X pixel address	Y pixel address	array index	Optional LOD
	disabled	Not MULTISAMPLECOUNT_1	Sample Number	X pixel address	Y pixel address	(ignored if present)	Optional LOD
	enabled			X pixel address	Y pixel address	array index	Optional LOD
SURFTYPE_CUBE	disabled	Must be MULTISAMPLECOUNT_1	Never Present	X pixel address	Y pixel address	array index	Optional LOD
	enabled			X pixel address	Y pixel address	array index	Optional LOD
SURFTYPE_3D	disabled	Must be MULTISAMPLECOUNT_1	Never Present	X pixel address	Y pixel address	Z pixel address	Optional LOD
SURFTYPE_BUFFER	disabled	Must be MULTISAMPLECOUNT_1	Never Present	buffer index			
SURFTYPE_STRBUF	disabled			buffer index	byte offset		

Workaround	
Context:	Addressing 1D, 2D, 3D, CUBE Surfaces - Workaround #275

If the surface state indicates the Number of Multisamples > 1, then the LOD parameter is not optional: the R and LOD parameters must be specified along with the MSAA sample number parameter.

Programming Note	
Context:	Addressing 1D, 2D, 3D, CUBE Surfaces

- Vertical stride & Vertical Offset fields of the surface state object are only supported for 2D non-array surfaces.
- Tile W surfaces must be of format R8_UINT and only support SIMD8.

Surface Formats

Untyped messages allow direct read and write access to untyped surfaces. Untyped surfaces are always type BUFFER, and are assumed to be surface format RAW.

Typed messages allow direct read and write accesses to typed surfaces. Typed surfaces are of type SURFTYPE_1D, 2D, 3D, CUBE, or BUFFER and have a supported surface format other than RAW.

Read and write messages for typed surfaces convert the message's data type to/from the selected surface's format. The conversion rules are listed in a table below. Some surface formats are supported for typed surface writes, but are indirectly supported for typed surface reads. Indirectly supported surface formats provide the RAW pixel data in UINT formats, which enables software to complete the conversion to their defined format.

The read conversion of typed surface format to a register data format is always exact. If the destination precision is less than the source precision, a write conversion from a register data format to a typed surface format may be rounded, and denormalized or clamped.

Conversion Rules for Typed Surface Dataport Messages

Register Data Type	Surface Format Type	Read Conversion (Surface to Register)	Write Conversion (Register to Surface)
F32	FLOAT	IEEE float conversion. Normalize denorms if source is narrower. Convert SNan to QNan quietly.	IEEE float conversion. Round to even and denormalize if destination is narrower. Convert SNan to QNan quietly.
F32	SNORM, UNORM	Convert fixed point to IEEE float	Convert IEEE float to fixed point. Round to even and clamp to min/max if destination is narrower.
S31	SINT	Sign extend MSBs if source is narrower	Clamp to min/max if destination is narrower.
U32	UINT	Zero extend MSBs if source is narrower	Clamp to min/max if destination is narrower.
RAW32	Any	32 bit copy	*Not supported*
RAW16	Any	16 bit copy, zero extend MSBs	*Not supported*
RAW8	Any	8 bit copy, zero extend MSBs	*Not supported*

Typed dataport messages support a subset of all surface state formats. The formats supported by typed surface reads and writes are listed in the table below. The table shows what values and formats are returned for the components when read.

Typed surface formats with UINT require the message data in U32 format. Surface formats with SINT require the message data in S32 format. All other typed surface formats require the message data in FLOAT32 format. The surface format for the typed atomic integer operations must be R32_UINT or R32_SINT.

All typed surface formats are supported for writes, as long as the Bits Per Element is a multiple of 8 (byte). This includes all the surface formats excluding SRGB and YCRV formats. The table shows the full list of supported surface formats for writes.

For typed surface writes where the Surface Format has components that are not byte-aligned, all channels must be enabled for writing.

Reads to an unsupported surface format return undefined results. Writes to an unsupported surface format have undefined results.

When an out-of-bounds location is read by typed dataport message, a 0 is returned for any component that is present in the surface format, and the default value is returned for any component that is missing in the surface format. The default value for missing components is 0 for RGB, and either 1 or 1.0 for A.

Supported Typed Surface Read Formats

Surface Format Encoding (Hex)	Format Name	Bits Per Element (BPE)	Data Returned By Typed Read			
			R	G	B	A
000	R32G32B32A32_FLOAT	128	F32	F32	F32	F32
001	R32G32B32A32_SINT	128	S31	S31	S31	S31
002	R32G32B32A32_UINT	128	U32	U32	U32	U32
080	R16G16B16A16_UNORM	64	F32	F32	F32	F32
081	R16G16B16A16_SNORM	64	F32	F32	F32	F32
082	R16G16B16A16_SINT	64	S31	S31	S31	S31
083	R16G16B16A16_UINT	64	U32	U32	U32	U32
084	R16G16B16A16_FLOAT	64	F32	F32	F32	F32
085	R32G32_FLOAT	64	F32	F32	0.0	1.0
086	R32G32_SINT	64	S31	S31	0	1
087	R32G32_UINT	64	U32	U32	0	1
0C0	B8G8R8A8_UNORM	32	*RAW32*	0	0	1
0C2	R10G10B10A2_UNORM	32	*RAW32*	0	0	1
0C7	R8G8B8A8_UNORM	32	F32	F32	F32	F32
0C9	R8G8B8A8_SNORM	32	F32	F32	F32	F32
0CA	R8G8B8A8_SINT	32	S31	S31	S31	S31
0CB	R8G8B8A8_UINT	32	U32	U32	U32	U32
0CC	R16G16_UNORM	32	F32	F32	0.0	1.0
0CD	R16G16_SNORM	32	F32	F32	0.0	1.0
0CE	R16G16_SINT	32	S31	S31	0	1
0CF	R16G16_UINT	32	U32	U32	0	1
0D0	R16G16_FLOAT	32	F32	F32	0.0	1.0
0D1	B10G10R10A2_UNORM	32	*RAW32*	0	0	1
0D3	R11G11B10_FLOAT	32	*RAW32*	0	0	1
0D6	R32_SINT	32	S31	0	0	1
0D7	R32_UINT	32	U32	0	0	1

Surface Format Encoding (Hex)	Format Name	Bits Per Element (BPE)	Data Returned By Typed Read			
0D8	R32_FLOAT	32	F32	0.0	0.0	1.0
100	B5G6R5_UNORM	16	*RAW16*	0	0	1
102	B5G5R5A1_UNORM	16	*RAW16*	0	0	1
104	B4G4R4A4_UNORM	16	*RAW16*	0	0	1
106	R8G8_UNORM	16	F32	F32	0.0	1.0
107	R8G8_SNORM	16	F32	F32	0.0	1.0
108	R8G8_SINT	16	S31	S31	0	1
109	R8G8_UINT	16	U32	U32	0	1
10A	R16_UNORM	16	F32	0.0	0.0	1.0
10B	R16_SNORM	16	F32	0.0	0.0	1.0
10C	R16_SINT	16	S31	0	0	1
10D	R16_UINT	16	U32	0	0	1
10E	R16_FLOAT	16	F32	0.0	0.0	1.0
11A	B5G5R5X1_UNORM	16	*RAW16*	0	0	1
140	R8_UNORM	8	F32	0.0	0.0	1.0
141	R8_SNORM	8	F32	0.0	0.0	1.0
142	R8_SINT	8	S31	0	0	1
143	R8_UINT	8	U32	0	0	1
144	A8_UNORM	8	0.0	0.0	0.0	F32

Supported Typed Surface Write Formats

Surface Format Encoding (Hex)	Format Name	Bits Per Element (BPE)
000	R32G32B32A32_FLOAT	128
001	R32G32B32A32_SINT	128
002	R32G32B32A32_UINT	128
080	R16G16B16A16_UNORM	64
081	R16G16B16A16_SNORM	64
082	R16G16B16A16_SINT	64
083	R16G16B16A16_UINT	64
084	R16G16B16A16_FLOAT	64
085	R32G32_FLOAT	64
086	R32G32_SINT	64
087	R32G32_UINT	64
0C0	B8G8R8A8_UNORM	32

Surface Format Encoding (Hex)	Format Name	Bits Per Element (BPE)
0C2	R10G10B10A2_UNORM	32
0C4	R10G10B10A2_UINT	32
0C7	R8G8B8A8_UNORM	32
0C8	R8G8B8A8_UNORM_SRGB	32
0C9	R8G8B8A8_SNORM	32
0CA	R8G8B8A8_SINT	32
0CB	R8G8B8A8_UINT	32
0CC	R16G16_UNORM	32
0CD	R16G16_SNORM	32
0CE	R16G16_SINT	32
0CF	R16G16_UINT	32
0D0	R16G16_FLOAT	32
0D1	B10G10R10A2_UNORM	32
0D3	R11G11B10_FLOAT	32
0D6	R32_SINT	32
0D7	R32_UINT	32
0D8	R32_FLOAT	32
100	B5G6R5_UNORM	16
102	B5G5R5A1_UNORM	16
104	B4G4R4A4_UNORM	16
106	R8G8_UNORM	16
107	R8G8_SNORM	16
108	R8G8_SINT	16
109	R8G8_UINT	16
10A	R16_UNORM	16
10B	R16_SNORM	16
10C	R16_SINT	16
10D	R16_UINT	16
10E	R16_FLOAT	16
11A	B5G5R5X1_UNORM	16
140	R8_UNORM	8
141	R8_SNORM	8
142	R8_SINT	8
143	R8_UINT	8
144	A8_UNORM	8

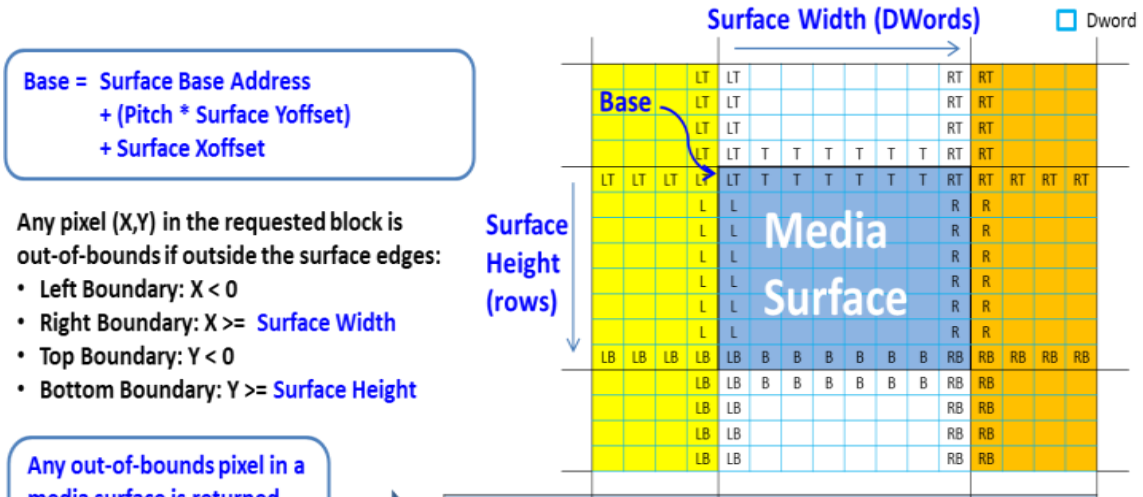
Programming Note	
Context:	Surface Formats
Indirectly supported surface formats are treated as R8_UINT for the 8 BPE formats, R16_UINT for 16 BPE formats, R32_UINT for 32 BPE formats, and R32G32_UINT for 64 BPE formats. And the Channel Mask for surface write messages, and the Shader Channel Select fields from the surface state, are interpreted with these same substitute formats.	

Addressing 2D Media Surfaces

Some data port messages operate on rectangular blocks of 2D surfaces using the BTS address model. These messages are typically used in media image processing. The rectangular block is specified in media messages as a signed (X, Y) offset from the surface's origin, and by the block's width and height.

The rectangular block specified in a media message could be partially (or fully) out-of-bounds of the surface. Some media data port messages perform pixel replication to supply values for out-of-bounds pixels. The figure below illustrates how the boundary pixel values are calculated from an in-bounds pixel value.

Media Surface Boundary Pixel Handling



Surface Format	Left Boundary	Boundary Pixel Value L/R/T/B	Right Boundary
Any 8bpe	B0B0B0B0	B0B1B2B3	B3B3B3B3
YCRCB_NORMAL	B0B1B0B3	B0B1B2B3 (Y0U0Y1V0)	B2B1B2B3
YCRCB_SWAPY	B0B1B2B1	B0B1B2B3 (U0Y0V0Y1)	B0B3B2B3
YCRCB_SWAPUV	B0B1B0B3	B0B1B2B3 (Y0V0Y1U0)	B2B1B2B3
YCRCB_SWAPUVY	B0B1B2B1	B0B1B2B3 (V0Y0U0Y1)	B0B3B2B3
Other 16bpe	B0B1B0B1	B0B1B2B3	B2B3B2B3
Any 32bpe	B0B1B2B3	B0B1B2B3	B0B1B2B3
All Others	*Not supported*	*Not supported*	*Not Supported*

Media accesses are Dwords: all 4 surface boundaries are Dword aligned
(Base is 32B aligned, Pitch is 64B aligned, Xoffset is 4*bpe aligned, Width is Dword aligned).

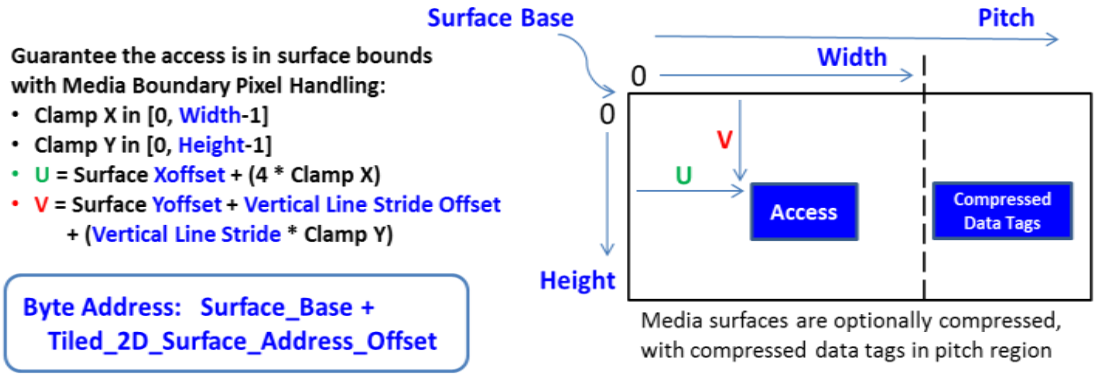
Upper left corner of the requested Block is specified by the signed (X, Y) coordinate.
The block offset and block width are always aligned to size of pixel (BPE) in surface.

The addressing of 2D Media surfaces is subset of the generalized surface state addressing model. For more information, see SURFACE_STATE in the Shared Functions volume and **Surface Layout and Tiling** in the GPU Overview volume.

The Vertical Stride and Vertical Stride Offset fields of the surface state object are used for 2D media surfaces to support interlaced formats in a full frame.

The surface's tile mode specifies how the 2D surface is laid out in the linear virtual address space. The figure below illustrates how pixel's byte address is calculated in a 2D media surface. The table lists how the surface's linear virtual address offset A[38:0] is calculated.

Media Surface Addressing



Bits per Element	Tile Mode	Tile constants		Tiled 2D Surface Address Offset																			
		Cv	Cu	A[38:16]	A[15:12] (64KB Tile)				A[11:6] (4KB Tile)						A[5:0] (Cache Line)							
64 & 128	TileYS	6	10	u[13:10] + (v[13:6]*Pitch[17:10])	u9	v5	u8	v4	u7	v3	u6	v2	u5	u4	v1	v0	u3	u2	u1	u0		
	TileYF	4	8	u[13:8] + (v[13:4]*Pitch[17:8])																		
16 & 32	TileYS	7	9	u[13:9] + (v[13:7]*Pitch[17:9])	u8	v6	u7	v5	u6	v4	u5	v3	u4	v2								
	TileYF	5	7	u[13:7] + (v[13:5]*Pitch[17:7])																		
8	TileYS	8	8	u[13:8] + (v[13:8]*Pitch[17:8])	u7	v7	u6	v6	u5	v5	u4	v4	u3	v2								
	TileYF	6	6	u[13:6] + (v[13:6]*Pitch[17:6])																		
all	TileY	5	7	u[13:7] + (v[13:5]*Pitch[17:7])					u6	u5	u4	v4	u3	v2								
all	TileX	3	9	u[13:9] + (v[13:3]*Pitch[17:9])					v2	v1	v0	u8	u7	u6	u5	u4	u3	u2	u1	u0		
all	Linear	0	0	u[13:0] + (v[13:0]*Pitch[17:0])																		

When a 2D surface is tiled, the Surface Base Address is aligned on the tile size (4KB or 64KB).
NV12 surfaces use X/Y Offset for UV Plane surface state so that Luma 8bpe and Chroma 16bpe both aligned for TileYF/YS.

2D Media surfaces can be set up to be compressible using the Memory Compression Enable surface state field. Some media hardware will write blocks in a compressed format. The media dataport messages will read those compressed formats and automatically decompress a compressed location when read. The tag information needed to interpret and decompress data is stored in the memory region between the surface width and pitch.

Programming Note	
Context:	Addressing 2D Media Surfaces
If the surface state field Memory Compression Enable is set, then the surface state field Coherency Type must be set to GPU Coherent (not IA coherent).	

Programming Note	
Context:	Addressing 2D Media Surfaces
Writing a compressible media surface via a Data Port message will result in storing the target memory block(s) in uncompressed format.	

Programming Note	
Context:	Addressing 2D Media Surfaces

- The surface base address must be 32-byte aligned.
- The surface width must a multiple of DWords.
- Pitch must be a multiple of 64 bytes when the surface is Linear.
- Media block writes to Linear or TileX surfaces must have a height of 16 or less.
- For YUV422 formats, the block width and offset must be pixel pair aligned (i.e. DWord-aligned).
- For media block writes, both X Offset and Block Width must be DWord-aligned.
- The block width and offset must be aligned to the size of pixels stored in the surface. For a surface with 8bpe pixels for example, the block width and offset can be byte-aligned. For a surface with 16bpe pixels, it is word-aligned.

2D Media Surface Formats

Media dataport messages allow direct read and write accesses to media surfaces. Media surfaces are only type SURFTYPE_2D.

Read and write messages for media surfaces never convert the data type of the selected surface's format.

If the media surface state is set up as not compressible, the formats supported are listed in the table below. Reads or writes to an unsupported surface format have undefined results. The surface format is only used to determine out-of-bounds pixel replication.

If the media surface state is set up as compressible, then only a small subset of surface state formats are supported by the media data port message. The formats supported by compressible media surface reads are listed in the table below. Reads to an unsupported surface format return undefined results. Writes to a compressible surface format have undefined results. The surface format is used to determine out-of-bounds pixel replication and to determine compression type.

Supported Media Surface Formats

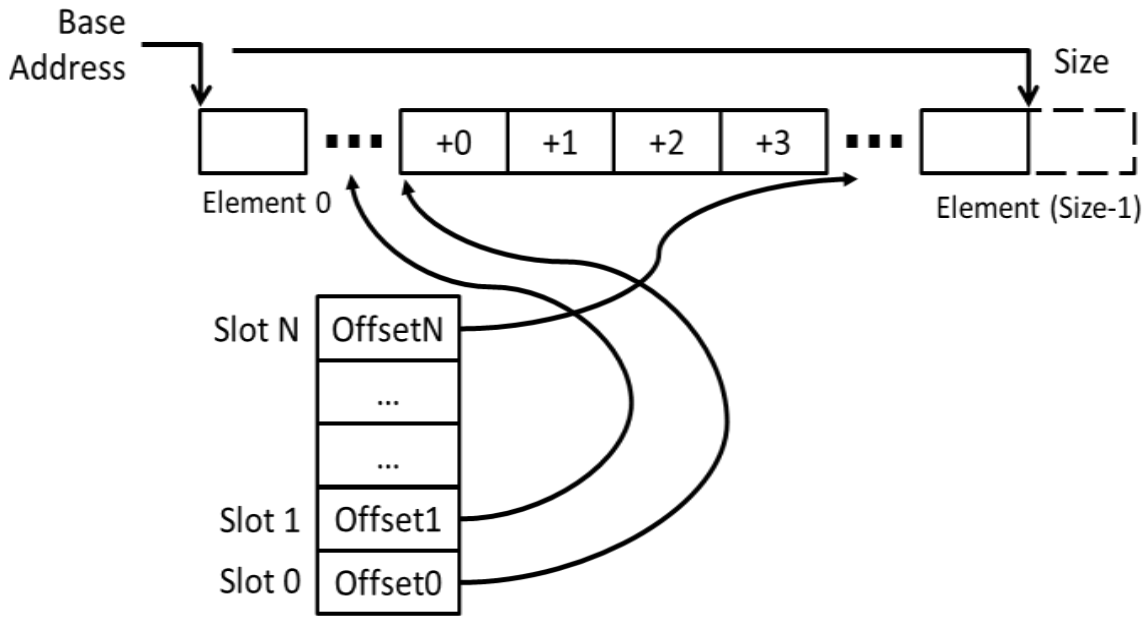
Surface Format Encoding (Hex)	Format Name	Bits Per Element(BPE)	Supported Compressible Media Surface Formats
0C0	B8G8R8A8_UNORM	32	*Not supported*
0C2	R10G10B10A2_UNORM	32	*Not supported*
0C4	R10G10B10A2_UINT	32	*Not supported*
0C7	R8G8B8A8_UNORM	32	*Not supported*
0C9	R8G8B8A8_SNORM	32	*Not supported*
0CA	R8G8B8A8_SINT	32	*Not supported*
0CB	R8G8B8A8_UINT	32	*Not supported*
0CC	R16G16_UNORM	32	*Not supported*
0CD	R16G16_SNORM	32	*Not supported*
0CE	R16G16_SINT	32	*Not supported*
0CF	R16G16_UINT	32	*Not supported*
0D0	R16G16_FLOAT	32	*Not supported*
0D1	B10G10R10A2_UNORM	32	*Not supported*
0D3	R11G11B10_FLOAT	32	*Not supported*
0D5	R10G10B10_FLOAT_A2_UNORM	32	*Not supported*
0D6	R32_SINT	32	*Not supported*
0D7	R32_UINT	32	*Not supported*
0D8	R32_FLOAT	32	*Not supported*
100	B5G6R5_UNORM	16	*Not supported*
102	B5G5R5A1_UNORM	16	*Not supported*
104	B4G4R4A4_UNORM	16	*Not supported*
106	R8G8_UNORM	16	NV12 (UV 8bpe)
107	R8G8_SNORM	16	*Not supported*
108	R8G8_SINT	16	*Not supported*
109	R8G8_UINT	16	NV12 (UV 8bpe)
10A	R16_UNORM	16	Y16
10B	R16_SNORM	16	*Not supported*
10C	R16_SINT	16	*Not supported*
10D	R16_UINT	16	Y16
10E	R16_FLOAT	16	*Not supported*
113	A16_UNORM	16	Y16
11A	B5G5R5X1_UNORM	16	*Not supported*
140	R8_UNORM	8	NV12 (Y), Y8, IMC1 (YUV), IMC3 (YUV)
141	R8_SNORM	8	*Not supported*

Surface Format Encoding (Hex)	Format Name	Bits Per Element(BPE)	Supported Compressible Media Surface Formats
142	R8_SINT	8	*Not supported*
143	R8_UINT	8	NV12 (Y), Y8, IMC1 (YUV), IMC3 (YUV)
144	A8_UNORM	8	NV12 (Y), Y8, IMC1 (YUV), IMC3 (YUV)
182	YCRCB_NORMAL	16	YUYV
183	YCRCB_SWAPUVY	16	VYUY
18F	YCRCB_SWAPUV	16	YVYU
190	YCRCB_SWAPY	16	UYVY

Slots and Elements

Data Port messages are SIMD operations: a single instruction operating over multiple data operands. Data operands are addressed either as: consecutive offsets ("elements"), with arbitrary offsets ("slots"), or in a combination of both slots and elements.

Slots and Elements



Each data port message supports a restricted subset of all possible combinations of slots and elements. The details of which combinations are supported are listed for each message in the Messages section of this volume.

The Offset in slots can be in any order and may be duplicates. The data port hardware attempts to minimize memory accesses and optimize for cache line efficiency. The hardware issues in parallel the data operations in parallel for each slot and data element pair, up to the limits of the hardware implementation. Bounds checking is performed independently on each access.

Programming Note	
Context:	Slots and Elements
Writes to overlapping addresses have undefined write ordering.	

For A32 messages, a base address offset is provided in the message header and added to each offset. The base address offset in the message header is always ignored for BTS, SLM, and A64 messages.

Typed messages always use the BTS address model, and can use multiple address components to calculate the address offset for each slot. The Surface Type determines which address components are present and how the address offset is calculated.

Base Address Offsets in SIMD

Slots	SIMD Name	Location of Slot Address Offset	Description
1	Block	Message Header	Message header provides 1 address offset in that is used as the address offset of a block of data elements. Untyped blocks use 1 address. Typed blocks such as Media Block use X/Y coordinates.
2	Dual Block or SIMD4x2	Address Payload	Message address payload provides 2 address offsets which are used as the base address offset of 2 sets of data elements.
8	SIMD8	Address Payload	Message address payload provides 8 address offsets which are used as the address offset of 8 sets of data.
16	SIMD16	Address Payload	Message address payload provides 16 address offsets which are used as the address offset of 16 sets of data. Typed messages process either the upper or lower 8 slots at a time, so two messages must be used for SIMD16 operations.

Programming Note	
Context:	Slots and Elements
The current implementation limits the maximum data payload for a message to 256 bytes. Depending on the number of slots and the data width specified by the operation, some messages restrict the number of elements supported in a single SIMD operation.	

Execution Masks

Every data port message is sent with a 16-bit execution mask. The Execution Mask is used by the data port to enable the read or write operation on the mask's corresponding SIMD channel. The channel is considered active if the mask bit is set.

The execution mask is used to ensure that data associated with an inactive channel are not overwritten in the GRF (read operations) or memory (write operations).

If all execution masks are clear on the SEND instruction, the send message instruction becomes a noop and the message is not sent to the data port.

Data port messages have different semantics on how the execution mask controls the read or write accesses. The table below summarizes the semantics for all the data ports. The details of which semantics are used by each individual message is described in the Messages section of this volume.

Description of SIMD Execution Mask Semantics

Execution Mask Name	SIMD Mode and Data Type	How Execution Mask is Used
SM	SIMD8 or SIMD16	For SIMD16, the 16 bits of the execution mask control the 16 SIMD channels. For SIMD8, the lower 8 bits of the execution mask control the 8 SIMD channels, and the upper 8 bits of the execution mask are ignored.
SMBLK	SIMD8 or SIMD16 With Data Elements	For SIMD16, the 16 bits of the execution mask control the 16 SIMD channels. For SIMD8, the lower 8 bits of the execution mask control the 8 SIMD channels, and the upper 8 bits of the execution mask are ignored. If the date elements > 1, then the execution mask bits are reused for the additional data elements.
SG	SIMD8 Slot Group	Either the lower 8 bits or the upper 8 bits of the execution mask are used to control the 8 SIMD channels, based on the slot group specified in the message descriptor.
BLK	Block	The lower 8 bits of the execution mask control the first 8 DWords, and the upper 8 bits control the second 8 DWords. The DWords are accessed if the corresponding DWord's execution mask bit is set. If the block is 1 OWord size, then either the lower or upper nibbles of the lower 8 bits control the corresponding DWords in that lower or upper OWord. If the block is 8 OWords or 16 OWords, then the execution mask bits are reused for the additional channels.
BLKCM	Block with Channel Mode	The lower 8 bits of the execution mask control the first 8 DWords, and the upper 8 bits control the second 8 DWords. If the block is 1 OWord size, then either the lower or upper nibbles of the lower 8 bits control the corresponding DWords in that lower or upper OWord. If the block is 8 OWords or 16 OWords, then the execution mask bits are reused for the additional blocks.
Ignored	Not Used	The execution mask bits are ignored by this message operation. The data is controlled by other parameters to this message. The execution mask bits must be non-zero for the message to be sent to the data port.

Address Alignment and Data Widths

Data Port messages calculate multiple data operand addresses using offsets from the message's common base address. The details of how each message calculates its address is described in the Messages section.

The address calculations are described using the following notational convention:

(Base + BaseOffset) -> DataType[Offset]

This notation refers to the byte-aligned address (**Base + BaseOffset**), which is used as the address of a linear array of **DataType** operands. The **Offset** is the index into the **DataType** array with that data width.

Address Calculation

Notation	Description
Base	Each Address Model has its own base address source, illustrated in the Surfaces Types section of this volume. The Base address is architecturally defined to always be aligned at a valid address boundary for the message. The notational convention in this chapter labels this calculated base address the **Base**.
Buffer, BaseOffset	The offsets from some messages are specified as byte offsets, and others are DWord (or other DataType) offsets. **Buffer** is shorthand for Buffer Base Address offset that is specified in A32 message headers. The notation in this chapter shows byte offsets as being added to the **Base** on the left side of the pointer (->), and the DataType-sized offsets as being indices to the array on the right of the pointer. Each message defines the address alignment it operates on. Base address offsets are forced to be aligned for the message's DataType by treating any unaligned low address bits as zero. For most messages (except media block messages), the Buffer or BaseOffset value is always an unsigned (positive) value.
B[ ], W[ ], DW[ ], QW[ ], OW[ ], HW[ ]	The **DataType** array is a linear array of Bytes (**B**), Words (**W**), DWords (**DW**), QWords (**QW**), OWords (**OW**), or HWords (**HW**). The index of the array is multiplied by the width of the data type, to form a byte offset to be added to the base address. Each message defines the data type it operates on. The array's byte offset is architecturally defined to always be aligned at a valid address boundary for the message.
Offset{Slot}, U{Slot}, V{Slot}	**Offset{Slot}** is shorthand for Offset0, Offset1, ... in a SIMD8 or SIMD16 address payload. **U{Slot}** and **V{Slot}** are shorthand for saying U.Offset{Slot} and V.Offset{Slot}, where U and V are SIMD8 or SIMD16 address payloads. For most messages (except media block messages), the **Offset** is always an unsigned

Notation	Description
	(positive) value.
Elem, Chan	**Elem** represents the index into an array of consecutive data operands. **Chan** represents the 32-bit Red, Green, Blue, or Alpha channels of color item. These values are sequentially stored in memory as a 4 item DW block.
(Surface[U, V, R, LOD])	The notation **Surface[U, V, R, LOD]** refers to a typed surface which, depending on the surface type, may use the U, V, R, and LOD parameters to calculate the offsets from the surface's state base address for the operands.

Bounds Checking and Faulting

Bounds checking is a programming safety feature of the architecture. If a data port access is outside the boundaries of the surface, that memory operation is dropped before accessing the L3 cache. Out-of-bounds read operations return zero for the data. Each SIMD memory access is individually checked in a message operation, so some accesses could be in-bounds and while others are out-of-bounds.

Out-of-bounds checking is always performed at a DWord granularity. If any part of the DWord is out-of-bounds then the whole DWord is considered out-of-bounds.

The bounds check is performed on the Address Offset portion of the message operation. It is designed to catch user mode programming mistakes. The bounds check assumes that the surface's base address is properly aligned and that the surface's boundary limits are properly configured by the trusted kernel mode graphics driver.

The bounds check does not replace the data security features of the page tables (PPGTT). The page tables are used to protect other memory areas from accesses outside of this program's memory space. Virtual memory addresses for data ports are translated through the PPGTT, and follow that translation table's faulting model. In all page fault modes, either the fault condition is corrected and the memory access is completed, or the entire program is terminated.

Tiled Resources
If a data port access is made to a TRTT Null tile, then that access behaves like an out-of-bounds access.

The bounds checks performed are specific to the address model and surface type of the message. See the figures in the Surfaces Types section of this volume for more information about the boundary conditions for the surfaces.

Description of Boundary Check Semantics

Bounds Check	Address Model	Surface Type	Description
Surface	BTS	BUFFER	Messages using the BTS address model detect any address offsets that, either directly or through the offset calculations, exceed the offset size of the buffer as specified in the SURFACE_STATE. When bounds checks are performed on a BTS BUFFER, Untyped Messages use the data size of the Surface element as the pitch, whereas Typed messages use the programmed surface pitch.
Surface	BTS	1D, 2D, 3D, CUBE	Messages using the BTS address model detect any U, V, or R address components that exceed the Width, Height, and Depth sizes of the surface, as specified in the SURFACE_STATE.
Media	BTS	2D	Some media-oriented messages using the BTS address model detect X or Y address components that exceed the Width and Height of the surface, as specified in the SURFACE_STATE. If a read access is outside of the surface boundaries, the read access is treated as an in-bounds access and returns the value of the nearest pixel. Write accesses outside of the surface boundaries are dropped.
GenState	A32	BUFFER	Messages using the A32 Stateless address model detect any address offsets that, either directly or through the offset calculations, exceed the General State Buffer Size specified in the STATE_BASE_ADDRESS. If General State Buffer Size is zero, then any A32 Stateless access is out-of-bounds.
PTSS	A32	BUFFER	Some A32 messages specify a Per Thread Scratch Space Size in their message header. When that message header is present and Local Thread Checking is enabled in **GT Mode Register**, the A32 message detects any address offsets that, directly or through the offset calculations, exceed the Per Thread Scratch Space Size. These A32 messages must additionally perform the GenState boundary check because the PTSS value is not guaranteed by trusted software to be within the GenState boundary limits.
Canonical	A64	BUFFER	Messages using A64 Stateless address model detect any A64 address offset in the message that cross the canonical address boundary as out-of-bound. Messages using non-A64 address models do not need to check for crossing the canonical address boundary because their surface and boundaries are assumed to be properly configured by the graphics driver.
Shared	SLM	BUFFER	Messages using the SLM address model detect any address offsets that, either directly or through the offset calculations, exceed the SLM memory size allocated to that workgroup. If the offset exceeds the allocated SLM memory size, the offset wraps around in the workgroup's allocated memory boundary and treated as an in-bound access.

Message Formats

Message operations on data ports are described using five standard parts.

Descriptor	Describes the Message Type, the Message Specific Controls, and whether a Message Header is present. The Message Descriptor and the destination Data Port (SFID) is encoded as part of the SEND instruction.
Header	When present, provides additional Message specific controls. Some messages disallow (forbid) a Header, some require a Header, and some permit an optional Header to specify additional non-default-valued parameters.
Address Payload	Provides the slot address offsets for those messages that support scattered operations.
Source Payload	Provides source data for write operations and atomic operations.
Writeback Payload	Returns result data for read operations and for atomic operations.

Most messages do not use (and therefore do not send or receive) all five parts of a message. The specific message encoded in the Message Descriptor determines which of the other four parts of the message sequence are present.

The next sections describe all of the supported formats for the Source and Writeback Data Payloads, the Address Payloads, the Message Descriptors, and their Message Headers.

End of Thread Usage

Read/Write data port messages may not have the End of Thread bit set in the message descriptor other than the following exceptions:

The URB Write message may have End of Thread set for threads dispatched by the 3D Geometry pipe fixed functions.

Message Description Conventions

Message operations with common semantics and limitations are grouped together and described in the same section. These common characteristics are summarized in the first table in each section:

Common Characteristics of a Message Grouping

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask

Although multiple messages are described in the same section, they frequently perform their similar functions using different data ports, address models, and with different SIMD combinations of slots and blocks. The second table in each message section lists all the valid combinations of message parameters and payloads for the messages:

Valid Message Parameters and Payloads

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload

Description of Message Summary Tables

Column Heading	Description
Addr Align	Specifies the required address alignment of the operation. This can be byte (B), DWord (DW), QWord (QW), OWord (OW), or HWord (HW). In most cases, the address must be aligned for the data width of the operation.
Data Width	Specifies the width of the data operation. This can be byte (B), DWord (DW), QWord (QW), OWord (OW), or HWord (OW).
R/W	Specifies if the operation is a Read or Write (R or W). Atomic operations that return data are classified here as reads, and ones that don't are classified here as writes.
Address Model	Specifies the address model used by this message operation: BTS, SLM, A32, or A64. Read-only data port accesses to the constant cache is labeled "CC BTS". Specifies the address model used by this message operation: BTS, SLM, A32, or A64. Read-only data port accesses to the constant cache or sampler cache are labeled "CC BTS" or "SC BTS".
Surface Type	Specifies the supported surface type. See the Surfaces Types section of this volume.
SIMD Slots	In the first table, this summarizes all the SIMD slot configurations supported for this operation and address model. In the second table, this specifies the specific SIMD slot configuration supported by the specific message operation.
Data Elements	Specifies all the sizes of data elements supported by the message operation. For blocks, this is the number of blocks. For surfaces, this is the list of channels that are enabled in the message descriptor.
SIMD Address Calculation	Describes the address calculation performed by this message operation. The calculation is usually related to the others in this section, but the details can change due to the address model used. See Address Alignment and Data Widths for an explanation of the notation used here.
Bounds Check	Specifies the bounds check performed by this message operation. The bounds check is generally determined by the address model used, but some message change the checking. See Bounds Checking and Faulting for an explanation of the notation used here.
Execution Mask	Most message operations perform their operation on a data channel only when the corresponding execution mask is set. See Execution Masks for an explanation of the notation used here.
Message Specific Descriptor	Each data port specifies a unique encoding for its message operations and its parameters. See Message Specific Descriptors for the full list of specific message descriptors used by the data ports. In each section, all the legal combinations of messages and parameters are listed. In some cases, there are multiple message operations that perform the same function.
Message Header	Many messages use a message header to provide additional parameters to control their operation, in addition to the parameters in the message descriptor. Each section's table lists which message headers are legal to use. See Message Headers for a full list of the message headers that are supported by all the messages.
Address Payload	Most messages perform SIMD operations using address offsets for the operation from the address payload. See Message Address Payloads for a full list of the message address payloads that are supported by all the messages. Each section's table lists which message headers are legal to use.

Column Heading	Description
Source Payload	Write messages and atomic messages generally provide data in a source data payload. See Message Data Payloads for a full list of the message data payloads that are supported by all the messages. Each section's table lists which message source payloads are legal to use.
Writeback Payload	Read messages generally return data in a writeback data payload. See Message Data Payloads for a full list of the message data payloads that are supported by all the messages. Each section's table lists which message writeback payloads are used.

Messages

The message operations on data ports are described this section. The operations are organized into subsections by their addressing operation group (blocks, scattered, surfaces, atomic operations), and then by their data width and type. Each subsection describes the messages that have common semantics and limitations. (For example, read and write operations are described in the same subsection.)

Summary of Message Groups

Operation	Addr Align	Data Width	Address Model	Surface Type	SIMD Slots	Data Elements	Bounds Check	Execution Mask
Scattered and Block R/W					1, 2	1, 2, 4, 8		
Byte Scattered R/W	B	B	BTS	BUFFER, NULL	8, 16	1, 2, 4	Surface	SMBLK
	B	B	SLM	BUFFER	8, 16	1, 2, 4	Shared	SMBLK
	B	B	A32	BUFFER	8, 16	1, 2, 4	PTSS	SMBLK
	B	B	A64	BUFFER	8, 16	1, 2, 4	Canonical	SMBLK
DW Scattered R/W	DW	DW	BTS	BUFFER, NULL	8, 16	1	Surface	SM
	DW	DW	A32	BUFFER	8, 16	1	PTSS	SM
	DW	DW	A64	BUFFER	8, 16	1, 2, 4, 8	Canonical	SMBLK
QW Scattered R/W	QW	QW	A64	BUFFER	8, 16	1, 2, 4	Canonical	SMBLK
OW Block R/W	OW	OW	BTS	BUFFER, NULL	1	1, 2, 4, 8	Surface	BLK
	OW	OW	SLM	BUFFER	1	1, 2, 4, 8	Shared	SMBLK
	OW	OW	A32	BUFFER	1	1, 2, 4, 8	PTSS	BLK
	OW	OW	A64	BUFFER	1	1, 2, 4, 8	Canonical	BLK
OW Aligned Block R/W	DW	OW	BTS	BUFFER, NULL	1	1, 2, 4, 8	Surface	Ignored
	DW	OW	A32	BUFFER	1	1, 2, 4, 8	PTSS	Ignored
	DW	OW	A64	BUFFER	1	1, 2, 4, 8	Canonical	Ignored
HW Block R/W	HW	HW	A32	BUFFER	1	1, 2, 4, 8	PTSS	BLKCM
HW Aligned Block R/W	HW	HW	A64	BUFFER	1	1, 2, 4, 8	Canonical	BLKCM
Surface R/W					2, 8, 16	1		
Scattered Untyped	DW	DW	BTS	BUFFER, NULL	8, 16	{Chan 1-4}	Surface	SM

Operation	Addr Align	Data Width	Address Model	Surface Type	SIMD Slots	Data Elements	Bounds Check	Execution Mask
Surface R/W	DW	DW	SLM	BUFFER	8, 16	{Chan 1-4}	Shared	SM
	DW	DW	A32	BUFFER	8, 16	{Chan 1-4}	GenState	SM
	DW	DW	A64	BUFFER	8, 16	{Chan 1-4}	Canonical	SM
Scattered Typed Surface R/W	B	DW	BTS	1D, 2D, 3D, CUBE, BUFFER, NULL	8	{Chan 1-4}	Surface	SG
Scattered MSAA Typed Surface R/W	B	DW	BTS	1D, 2D, 3D, CUBE, BUFFER, NULL	8	{Chan 1-4}	Surface	SG
Atomic Operations					2, 8, 16	1		
W Untyped Atomic Integer	W	W	BTS	BUFFER, NULL	8, 16	1	Surface	SM
	W	W	SLM	BUFFER	8, 16	1	Shared	SM
	W	W	A32	BUFFER	8, 16	1	GenState	SM
	W	W	A64	BUFFER	8	1	Canonical	SM
DW Untyped Atomic Integer	DW	DW	BTS	BUFFER, NULL	8, 16	1	Surface	SM
	DW	DW	SLM	BUFFER	8, 16	1	Shared	SM
	DW	DW	A32	BUFFER	8, 16	1	GenState	SM
	DW	DW	A64	BUFFER	8	1	Canonical	SM
QW Untyped Atomic Integer	QW	QW	BTS	BUFFER, NULL	8, 16	1	Surface	SM
	QW	QW	A32	BUFFER	8, 16	1	GenState	SM
	QW	QW	A64	BUFFER	8	1	Canonical	SM
W Untyped Atomic Float	W	W	BTS	BUFFER, NULL	8, 16	1	Surface	SM
	W	W	SLM	BUFFER	8, 16	1	Shared	SM
	W	W	A32	BUFFER	8, 16	1	GenState	SM
	W	W	A64	BUFFER	8	1	Canonical	SM
DW Untyped Atomic Float	DW	DW	BTS	BUFFER, NULL	8, 16	1	Surface	SM
	DW	DW	SLM	BUFFER	8, 16	1	Shared	SM
	DW	DW	A32	BUFFER	8, 16	1	GenState	SM
	DW	DW	A64	BUFFER	8	1	Canonical	SM
W Typed Atomic Integer	W	W	BTS	1D, 2D, 3D, CUBE, BUFFER, NULL	8	1	Surface	SG
DW Typed Atomic Integer	DW	DW	BTS	1D, 2D, 3D, CUBE, BUFFER, NULL	8	1	Surface	SG
W Atomic Counter	W	W	BTS	Any	8	1	Ignored	SG
DW Atomic Counter	DW	DW	BTS	Any	8	1	Ignored	SG

Operation	Addr Align	Data Width	Address Model	Surface Type	SIMD Slots	Data Elements	Bounds Check	Execution Mask
Media Block R/W								
Media Block R/W	DW	DW	BTS	2D, NULL	1	Height x Width	Media	Ignored
Others								
Read Surface Info	B	B	BTS	Any	1	1	Ignored	Ignored
Memory Fence	N/A	N/A	N/A	N/A	N/A	N/A	Ignored	Ignored

Scattered and Block Read/Write Messages

This section lists the read and write messages that support scattered and block accesses of the standard untyped data: Byte, Word, DWord, QWord, OWords, and HWords.

Byte Scattered ReadWrite Messages

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
B	B	R/W	BTS	BUFFER, NULL	8, 16	1, 2, 4	(Base + GlobalOffset + Offset{Slot})->B[Elem]	Surface	SMBLK
B	B	R/W	SLM	BUFFER	8, 16	1, 2, 4	(Base + Offset{Slot})->B[Elem]	Shared	SMBLK
B	B	R/W	A32	BUFFER	8, 16	1, 2, 4	(Base + Buffer + GlobalOffset + Offset{Slot})->B[Elem]	PTSS	SMBLK
B	B	R/W	A64	BUFFER	8, 16	1, 2, 4	(0 + Offset{Slot})->B[Elem]	Canonical	SMBLK

This message reads or writes 8 or 16 scattered and possibly misaligned Bytes, Words, or DWords, starting at each Offset. The offsets are byte-aligned. The **Global Offset** is added to each of the specific offsets when the message header is present.

A BTS surface is interpreted as a RAW BUFFER regardless of the surface format. The pitch is 1 byte and data is returned from the buffer to the GRF without format conversion.

For BTS BUFFER accesses, the Surface Base address must be aligned to DW.

The execution mask bits may disable accesses on the corresponding SIMD slots. Out-of-bounds accesses are dropped or return zero. The semantics depend on the address model, as listed in the above table.

Applications:

- Byte aligned buffer accesses in programmable shaders

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	BTS	BUFFER, NULL	SIMD8	{1, 2, 4}	MSD0R_BS	{Opt} MH_BTS_GO	MAP32B_SIMD8	{Forbidden}	{1, 1, 1} MDP_DW_SIMD8
W	BTS	BUFFER, NULL	SIMD8	{1, 2, 4}	MSD0W_BS	{Opt} MH_BTS_GO	MAP32B_SIMD8	{1, 1, 1} MDP_DW_SIMD8	{Forbidden}
R	BTS	BUFFER, NULL	SIMD16	{1, 2, 4}	MSD0R_BS	{Opt} MH_BTS_GO	MAP32B_SIMD16	{Forbidden}	{1, 1, 1} MDP_DW_SIMD16
W	BTS	BUFFER, NULL	SIMD16	{1, 2, 4}	MSD0W_BS	{Opt} MH_BTS_GO	MAP32B_SIMD16	{1, 1, 1} MDP_DW_SIMD16	{Forbidden}
R	SLM	BUFFER	SIMD8	{1, 2, 4}	MSD0R_BS	{Forbidden}	MAP32B_SIMD8	{Forbidden}	{1, 1, 1} MDP_DW_SIMD8
W	SLM	BUFFER	SIMD8	{1, 2, 4}	MSD0W_BS	{Forbidden}	MAP32B_SIMD8	{1, 1, 1} MDP_DW_SIMD8	{Forbidden}
R	SLM	BUFFER	SIMD16	{1, 2, 4}	MSD0R_BS	{Forbidden}	MAP32B_SIMD16	{Forbidden}	{1, 1, 1} MDP_DW_SIMD16
W	SLM	BUFFER	SIMD16	{1, 2, 4}	MSD0W_BS	{Forbidden}	MAP32B_SIMD16	{1, 1, 1} MDP_DW_SIMD16	{Forbidden}
R	SLM	BUFFER	SIMD16	{1, 2, 4}	MSD0R_BS	{Forbidden}	MAP16B_SIMD16	{Forbidden}	{1, 1, 1} MDP_DW_SIMD16
W	SLM	BUFFER	SIMD16	{1, 2, 4}	MSD0W_BS	{Forbidden}	MAP16B_SIMD16	{1, 1, 1} MDP_DW_SIMD16	{Forbidden}
R	A32	BUFFER	SIMD8	{1, 2, 4}	MSD0R_BS	{Opt} MH_A32_GO	MAP32B_SIMD8	{Forbidden}	{1, 1, 1} MDP_DW_SIMD8
W	A32	BUFFER	SIMD8	{1, 2, 4}	MSD0W_BS	{Opt} MH_A32_GO	MAP32B_SIMD8	{1, 1, 1} MDP_DW_SIMD8	{Forbidden}
R	A32	BUFFER	SIMD16	{1, 2, 4}	MSD0R_BS	{Opt} MH_A32_GO	MAP32B_SIMD16	{Forbidden}	{1, 1, 1} MDP_DW_SIMD16
W	A32	BUFFER	SIMD16	{1, 2, 4}	MSD0W_BS	{Opt} MH_A32_GO	MAP32B_SIMD16	{1, 1, 1} MDP_DW_SIMD16	{Forbidden}
R	A64	BUFFER	SIMD8	{1, 2, 4}	MSD1R_A64_BS	{Forbidden}	MAP64B_SIMD8	{Forbidden}	{1, 1, 1} MDP_DW_SIMD8
W	A64	BUFFER	SIMD8	{1, 2, 4}	MSD1W_A64_BS	{Forbidden}	MAP64B_SIMD8	{1, 1, 1} MDP_DW_SIMD8	{Forbidden}
R	A64	BUFFER	SIMD16	{1, 2, 4}	MSD1R_A64_BS	{Forbidden}	MAP64B_SIMD16	{Forbidden}	{1, 1, 1} MDP_DW_SIMD16

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
W	A64	BUFFER	SIMD16	{1, 2, 4}	MSD1W_A64_BS	{Forbidden}	MAP64B_SIMD16	{1, 1, 1} MDP_DW_SIMD16	{Forbidden}

Programming Note	
Context:	Byte Scattered ReadWrite Messages
While the hardware does check for and optimize for cases where offsets are equal or contiguous, for optimal performance in some cases an aligned message with aligned data offsets may provide higher performance.	

Programming Note	
Context:	Byte Scattered ReadWrite Messages
For bounds checking the byte scattered read and write messages, the buffer size must be a multiple of 4 bytes.	

DWord Scattered Read/Write Messages

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
DW	DW	R/W	BTS	BUFFER,NULL	8, 16	1	Base->DW[GlobalOffset+Offset{Slot}+0]	Surface	SM
DW	DW	R/W	A32	BUFFER	8, 16	1	(Base+Buffer)->DW[GlobalOffset+Offset{Slot}+0]	PTSS	SM
DW	DW	R/W	A64	BUFFER	8, 16	1, 2, 4	(0+Offset{Slot})->DW[Elem]	Canonical	SMBLK

This message reads or writes 8 or 16 scattered DWords starting at each offset. The **Global Offset** is added to each of the specific offsets when it is provided in the message header.

A BTS surface is interpreted as a RAW BUFFER regardless of the surface format. The pitch is 1 byte and data is returned from the buffer to the GRF without format conversion.

For BTS BUFFER accesses, the Surface Base address must be aligned to DW.

The execution mask bits may disable accesses on the corresponding SIMD slots. Out-of-bounds accesses are dropped or return zero. The semantics depend on the address model, as listed in the above table.

Applications:

- SIMD8/16 constant buffer reads where the indices of each pixel are different (read one channel per message)
- SIMD8/16 scratch space reads/writes where the indices are different (read/write one channel per message)
- General purpose DWord scatter/gathering, used by media

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	BTS	BUFFER,NULL	SIMD8	1	MSD0R_DWS	{Opt} MH_BTS_GO	MAP32B_SIMD8	{Forbidden}	MDP_DW_SIMD8
W	BTS	BUFFER,NULL	SIMD8	1	MSD0W_DWS	{Opt} MH_BTS_GO	MAP32B_SIMD8	MDP_DW_SIMD8	{Forbidden}
R	BTS	BUFFER,NULL	SIMD16	1	MSD0R_DWS	{Opt} MH_BTS_GO	MAP32B_SIMD16	{Forbidden}	MDP_DW_SIMD16
W	BTS	BUFFER,NULL	SIMD16	1	MSD0W_DWS	{Opt} MH_BTS_GO	MAP32B_SIMD16	MDP_DW_SIMD16	{Forbidden}
R	A32	BUFFER	SIMD8	1	MSD0R_DWS	{Opt} MH_A32_GO	MAP32B_SIMD8	{Forbidden}	MDP_DW_SIMD8
W	A32	BUFFER	SIMD8	1	MSD0W_DWS	{Opt} MH_A32_GO	MAP32B_SIMD8	MDP_DW_SIMD8	{Forbidden}
R	A32	BUFFER	SIMD16	1	MSD0R_DWS	{Opt} MH_A32_GO	MAP32B_SIMD16	{Forbidden}	MDP_DW_SIMD16
W	A32	BUFFER	SIMD16	1	MSD0W_DWS	{Opt} MH_A32_GO	MAP32B_SIMD16	MDP_DW_SIMD16	{Forbidden}
R	A64	BUFFER	SIMD8	{1, 2, 4}	MSD1R_A64_DWS	{Forbidden}	MAP64B_SIMD8	{Forbidden}	{1, 2, 4} MDP_DW_SIMD8
W	A64	BUFFER	SIMD8	{1, 2, 4}	MSD1W_A64_DWS	{Forbidden}	MAP64B_SIMD8	{1, 2, 4} MDP_DW_SIMD8	{Forbidden}
R	A64	BUFFER	SIMD16	{1, 2, 4}	MSD1R_A64_DWS	{Forbidden}	MAP64B_SIMD16	{Forbidden}	{1, 2, 4} MDP_DW_SIMD16
W	A64	BUFFER	SIMD16	{1, 2, 4}	MSD1W_A64_DWS	{Forbidden}	MAP64B_SIMD16	{1, 2, 4} MDP_DW_SIMD16	{Forbidden}
R	CC BTS	BUFFER,NULL	SIMD8	1	MSD_CC_DWS	{Opt} MH_BTS_GO	MAP32B_SIMD8	{Forbidden}	MDP_DW_SIMD8
R	CC BTS	BUFFER,NULL	SIMD16	1	MSD_CC_DWS	{Opt} MH_BTS_GO	MAP32B_SIMD16	{Forbidden}	MDP_DW_SIMD16

Programming Note	
Context:	DWord Scattered Read/Write Messages
For BTS and A32 address models, the offsets in the address payload must in the range 0 - 3FFFFFFFh. If the upper 2 bits are non-zero, the result is undefined.	

QWord Scattered Read/Write Messages

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
QW	QW	R/W	A64	BUFFER	8, 16	1, 2, 4	(0+Offset{Slot})->QW[Elem]	Canonical	SMBLK

This message reads or writes 8 or 16 scattered QWords starting at each offset. The **Global Offset** is added to each of the specific offsets when it is provided in the message header.

A BTS surface is interpreted as a RAW BUFFER regardless of the surface format. The pitch is 1 byte and data is returned from the buffer to the GRF without format conversion.

For BTS BUFFER accesses, the Surface Base address must be aligned to QW.

The execution mask bits may disable accesses on the corresponding SIMD slots. Out-of-bounds accesses are dropped or return zero. The semantics depend on the address model, as listed in the above table.

Applications:

SIMD8/16 writes where the indices differ (write one channel per message).

General purpose QWord scatter/gathering, used by double precision compute.

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	A64	BUFFER	SIMD8	{1, 2, 4}	**MSD1R_A64_QWS**	{Forbidden}	**MAP64B_SIMD8**	{Forbidden}	{1, 2, 4} **MDP_QW_SIMD8**
W	A64	BUFFER	SIMD8	{1, 2, 4}	**MSD1W_A64_QWS**	{Forbidden}	**MAP64B_SIMD8**	{1, 2, 4} **MDP_QW_SIMD8**	{Forbidden}
R	A64	BUFFER	SIMD16	{1, 2}	**MSD1R_A64_QWS**	{Forbidden}	**MAP64B_SIMD16**	{Forbidden}	{1, 2} **MDP_QW_SIMD16**
W	A64	BUFFER	SIMD16	{1, 2}	**MSD1W_A64_QWS**	{Forbidden}	**MAP64B_SIMD16**	{1, 2} **MDP_QW_SIMD16**	{Forbidden}

OWord Block Read/Write Messages

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
OW	OW	R/W	BTS	BUFFER,NULL	1	1, 2, 4, 8	Base->OW[GlobalOffset+Elem]	Surface	BLK
OW	OW	R/W	SLM	BUFFER	1	1, 2, 4, 8	Base->OW[GlobalOffset+Elem]	Shared	BLK
OW	OW	R/W	A32	BUFFER	1	1, 2, 4, 8	(Base+Buffer)->OW[GlobalOffset+Elem]	PTSS	BLK
OW	OW	R/W	A64	BUFFER	1	1, 2, 4, 8	(Base+BlockOffset0)->OW[Elem]	Canonical	BLK

This message takes one offset (Global Offset), and reads or writes 1, 2, 4, or 8 contiguous OWords starting at that offset.

A BTS surface is interpreted as a RAW BUFFER regardless of the surface format.

For BTS BUFFER accesses, the pitch is 1 byte and data is returned from the buffer to the GRF without format conversion.

For BTS BUFFER accesses, the Surface Base address must be aligned to OW.

For SCRATCH accesses, the pitch is from the surface state and the data is returned from the buffer to the GRF without format conversion.

Programming Note
The execution mask bits may disable accesses on the corresponding DWs of the message payload.

Out-of-bounds accesses are dropped or return zero. The semantics depend on the address model, as listed in the above table.

Applications:

- Constant buffer reads of a single constant or multiple contiguous constants.
- Scratch space reads/writes.

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	BTS	BUFFER,NULL	1	{1, 2, 4, 8}	**MSD0R_OWB**	**MH_BTS_GO**	{Forbidden}	{Forbidden}	**MDP_OW1L**, **MDP_OW1U**, **MDP_OW2**, **MDP_OW4**
W	BTS	BUFFER,NULL	1	{1, 2, 4, 8}	**MSD0W_OWB**	**MH_BTS_GO**	{Forbidden}	**MDP_OW1L**, **MDP_OW1U**, **MDP_OW2**, **MDP_OW4**	
R	SLM	BUFFER	1	{1, 2, 4, 8}	**MSD0R_OWB**	**MH_SLM_GO**	{Forbidden}	{Forbidden}	**MDP_OW1L**, **MDP_OW1U**, **MDP_OW2**, **MDP_OW4**

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
W	SLM	BUFFER	1	{1, 2, 4, 8}	MSD0W_OWB	MH_SLM_GO	{Forbidden}	MDP_OW1L, MDP_OW1U, MDP_OW2, MDP_OW4	
R	A32	BUFFER	1	{1, 2, 4, 8}	MSD0R_OWB	MH_A32_GO	{Forbidden}	{Forbidden}	MDP_OW1L, MDP_OW1U, MDP_OW2, MDP_OW4
W	A32	BUFFER	1	{1, 2, 4, 8}	MSD0W_OWB	MH_A32_GO	{Forbidden}	MDP_OW1L, MDP_OW1U, MDP_OW2, MDP_OW4	
R	A64	BUFFER	1	{1, 2, 4, 8}	MSD1R_A64_OWB	MH_A64_OWB	{Forbidden}	{Forbidden}	MDP_OW1L, MDP_OW1U, MDP_OW2, MDP_OW4
W	A64	BUFFER	1	{1, 2, 4, 8}	MSD1W_A64_OWB	MH_A64_OWB	{Forbidden}	MDP_OW1L, MDP_OW1U, MDP_OW2, MDP_OW4	
R	CC BTS	BUFFER,NULL	1	{1, 2, 4, 8}	MSD_CC_OWB	MH_BTS_GO	{Forbidden}	{Forbidden}	MDP_OW1L, MDP_OW1U, MDP_OW2, MDP_OW4

For the one-constant case, either the high or low half of the payload register is used depending on the half selected in Block Size, and the other half of the payload register is ignored.

Programming Note	
Context:	OWord Block Read/Write Messages
For the BTS and A32 address models, the offsets in the address payload must be in the range 0 - 0FFFFFFFh. If the upper 4 bits are non-zero, the result is undefined.	

OWord Aligned Block Read/Write Messages

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
DW	OW	R	BTS	BUFFER, NULL	1	1, 2, 4, 8	(Base+GlobalOffset)->OW[Elem]	Surface	Ignored
DW	OW	R	A32	BUFFER	1	1, 2, 4, 8	(Base+Buffer+GlobalOffset)->OW[Elem]	PTSS	Ignored
DW	OW	R	A64	BUFFER	1	1, 2, 4, 8	(0+BlockOffset0)->OW[Elem]	Canonical	Ignored

Functional Description
This message takes one DWord-aligned offset (Global Offset), and reads 1, 2, 4, or 8 contiguous OWords starting at that offset.

A BTS surface is interpreted as a RAW BUFFER regardless of the surface format.

For BTS BUFFER accesses, the pitch is 1 byte and data is returned from the buffer to the GRF without format conversion.

For BTS BUFFER acesses, the Surface Base address must be aligned to DW.

This message is identical to the OWord Block Read/Write message except the offset alignment. The Global Offset is specified in bytes, and must be aligned to the Addr Align entry in table.

The execution mask is ignored for this message. Out-of-bounds accesses are dropped or return zero.

Applications:

Block accesses directly using a byte offset instead of an Oword index.

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	BTS	BUFFER,NULL	1	{1, 2, 4, 8}	MSD0R_OWAB	MH_BTS_GO	{Forbidden}	{Forbidden}	MDP_OW1L, MDP_OW1U, MDP_OW2, MDP_OW4
R	SLM	BUFFER	1	{1, 2, 4, 8}	MSD0R_OWAB	MH_SLM_GO	{Forbidden}	{Forbidden}	MDP_OW1L, MDP_OW1U, MDP_OW2, MDP_OW4
R	A32	BUFFER	1	{1, 2, 4, 8}	MSD0R_OWAB	MH_A32_GO	{Forbidden}	{Forbidden}	MDP_OW1L, MDP_OW1U, MDP_OW2, MDP_OW4
R	A64	BUFFER	1	{1, 2, 4, 8}	MSD1R_A64_OWAB	MH_A64_OWB	{Forbidden}	{Forbidden}	MDP_OW1L, MDP_OW1U, MDP_OW2, MDP_OW4
R	CC BTS	BUFFER,NULL	1	{1, 2, 4, 8}	MSD_CC_OWAB	MH_BTS_GO	{Forbidden}	{Forbidden}	MDP_OW1L, MDP_OW1U, MDP_OW2, MDP_OW4
R	SC BTS	BUFFER,NULL	1	{1, 2, 4, 8}	MSD_SC_OWAB	MH_BTS_GO	{Forbidden}	{Forbidden}	MDP_OW1L, MDP_OW1U, MDP_OW2, MDP_OW4

Programming Note
For read/write cache, only the read path supports the unaligned OWord Block access.

HWord Scratch Block Read/Write Messages

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
HW	HW	R/W	A32	BUFFER	1	1, 2, 4, 8	(Base+Buffer)->HW[SB_Offset+Elem]	PTSS	BLKCM

This message performs a read or write operation of between 1 and 8 registers to an HWord-aligned offset. The required message header payload matches the thread payload R0.

For BTS BUFFER accesses, the Surface Base address must be aligned to HW.

Programming Note
The execution mask bits may disable accesses on the corresponding DWs of the message payload.

Out-of-bounds accesses are dropped or return zero.

Applications:

Scratch space reads/writes for register spill/fill operations.

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	A32	BUFFER	1	{1, 2, 4, 8}	MSD0R_SB	MH_A32_HWB	{Forbidden}	{Forbidden}	MDP_HW1, MDP_HW2, MDP_HW4, MDP_HW8
W	A32	BUFFER	1	{1, 2, 4, 8}	MSD0W_SB	MH_A32_HWB	{Forbidden}	MDP_HW1, MDP_HW2, MDP_HW4, MDP_HW8	{Forbidden}

Channel Mode Interpretation
Channel-enable interpretation is fixed (not programmable): • DWord, which supports a SIMD8 or SIMD16 DWord channel-serial view of a register

Programming Note
For the MSD0R_SB and MSD0W_SB messages, the HWord offset into the A32 memory is provided in the message descriptor and communicated to the data port on the Sideband. This allows a single instruction read or write block operation in a single source instruction. The message descriptor provides 12 bits for the HWord offset, allowing addressing of 4K HWord locations (128KB) based from the A32 general state base address. Scratch Block messages are always used inside a thread's local scratch space, and they are assumed to be non-coherent memory accesses (MDC_STATELESS A32_A64_NC).

HWord Aligned Block Read/Write Messages

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
HW	HW	R	SLM	BUFFER	1	1, 2, 4, 8	(Base+GlobalOffset)->HW[Elem]	Shared	Ignored
DW	HW	R/W	A64	BUFFER	1	1, 2, 4, 8	(0+BlockOffset0)->HW[Elem]	Canonical	BLKCM

Functional Description
This message takes one DWord-aligned offset (Global Offset), and reads 1, 2, 4, or 8 contiguous HWords starting at that offset.

A BTS surface is interpreted as a RAW BUFFER regardless of the surface format.

For BTS BUFFER accesses, the pitch is 1 byte and data is returned from the buffer to the GRF without format conversion, and the Surface Base address must aligned to DW.

For BTS SCRATCH accesses, the pitch is from the surface state and data is returned from the buffer to the GRF without format conversion.

This message is identical to the HWord Block message except the offset alignment. The Global Offset is specified in bytes, and must be aligned to the Addr Align entry in the table.

For address models A32, BTS and SLM, execution mask is ignored for this message. Out-of-bounds accesses are dropped or return zero.

Applications:

Block accesses directly using a byte offset instead of an Hword index.

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	BTS	BUFFER,NULL	1	{1, 2, 4, 8}		MH_SLM_GO	{Forbidden}	{Forbidden}	MDP_HW1, MDP_HW2, MDP_HW4, MDP_HW8
R	BTS	SCRATCH,NULL	1	{1, 2, 4, 8}		MH_SLM_GO	{Forbidden}	{Forbidden}	MDP_HW1, MDP_HW2, MDP_HW4, MDP_HW8
R	SLM	BUFFER	1	{1, 2, 4, 8}		MH_SLM_GO	{Forbidden}	{Forbidden}	MDP_HW1, MDP_HW2, MDP_HW4, MDP_HW8
R	A64	BUFFER	1	{1, 2, 4, 8}	MSD1R_A64_HWB	MH_A64_HWB	{Forbidden}	{Forbidden}	MDP_HW1, MDP_HW2, MDP_HW4, MDP_HW8
W	A64	BUFFER	1	{1, 2, 4, 8}	MSD1W_A64_HWB	MH_A64_HWB	{Forbidden}	MDP_HW1, MDP_HW2,	

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
								MDP_HW4, MDP_HW8	

Channel Mode Interpretation
Channel-enable interpretation is fixed (not programmable): • DWord, which supports a SIMD8 or SIMD16 DWord channel-serial view of a register

Surface ReadWrite Messages

The surface read and write messages allow direct read/write accesses to untyped and typed surfaces.

Untyped surfaces are either SURFTYPE_BUFFER (format RAW, pitch 1) or SURFTYPE_SCRATCH (format RAW, pitch from the surface state). SLM and Stateless messages are always untyped BUFFER accesses.

Typed surfaces are 1D, 2D, 3D, CUBE, or BUFFER types. The surface state specifies all the parameters.There are some limitations on what typed data surface formats are supported by data ports; see the specific messages and Surfaces Types for more details.

Untyped Surface ReadWrite Messages

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
DW	DW	R/W	BTS	BUFFER,NULL	8, 16	{Chan 1-4}	(Base+U{Slot})->DW[Chan]	Surface	SM
DW	DW	R/W	SLM	BUFFER	8, 16	{Chan 1-4}	(Base+U{Slot})->DW[Chan]	Shared	SM
DW	DW	R/W	A32	BUFFER	8, 16	{Chan 1-4}	(Base+Buffer+U{Slot})->DW[Chan]	GenState	SM
DW	DW	R/W	A64	BUFFER	8, 16	{Chan 1-4}	(0+U{Slot})->DW[Chan]	Canonical	SM

This message allows direct read/write accesses to untyped surfaces using 8 or 16 offsets. Up to 4 DWords are accessed beginning at the byte addresses determined. These 4 DWords correspond to the red, green, blue, and alpha channels, in that order and with red mapping to the lowest order DWord.

For BUFFER accesses, if the U offset is not DWord-aligned, the results are undefined.

The execution mask bits may disable accesses on the corresponding SIMD slots. Out-of-bounds accesses are dropped or return zero. The semantics depend on the address model, as listed in the above table.

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	BTS	BUFFER,NULL	SIMD8	{Chan 1-4}	MSD1R_US	{Forbidden}	MAP32b_USU_SIMD8	{Forbidden}	{Chan1-4} MDP_DW_SIMD8
W	BTS	BUFFER,NULL	SIMD8	{Chan 1-4}	MSD1W_US	{Forbidden}	MAP32b_USU_SIMD8	{Chan1-4} MDP_DW_SIMD8	{Forbidden}
R	BTS	BUFFER,NULL	SIMD16	{Chan 1-4}	MSD1R_US	{Forbidden}	MAP32b_USU_SIMD16	{Forbidden}	{Chan1-4} MDP_DW_SIMD16
W	BTS	BUFFER,NULL	SIMD16	{Chan 1-4}	MSD1W_US	{Forbidden}	MAP32b_USU_SIMD16	{Chan1-4} MDP_DW_SIMD16	{Forbidden}
R	SLM	BUFFER	SIMD8	{Chan 1-4}	MSD1R_US	{Forbidden}	MAP32b_USU_SIMD8	{Forbidden}	{Chan1-4} MDP_DW_SIMD8
W	SLM	BUFFER	SIMD8	{Chan 1-4}	MSD1W_US	{Forbidden}	MAP32b_USU_SIMD8	{Chan1-4} MDP_DW_SIMD8	{Forbidden}
R	SLM	BUFFER	SIMD8	{Chan 1-4}		{Forbidden}	MAP32b_USU_SIMD8	{Forbidden}	{Chan1-4} MDP_DW_SIMD8
W	SLM	BUFFER	SIMD8	{Chan 1-4}		{Forbidden}	MAP32b_USU_SIMD8	{Chan1-4} MDP_DW_SIMD8	{Forbidden}
R	SLM	BUFFER	SIMD16	{Chan 1-4}	MSD1R_US	{Forbidden}	MAP32b_USU_SIMD16	{Forbidden}	{Chan1-4} MDP_DW_SIMD16
W	SLM	BUFFER	SIMD16	{Chan 1-4}	MSD1W_US	{Forbidden}	MAP32b_USU_SIMD16	{Chan1-4} MDP_DW_SIMD16	{Forbidden}
R	SLM	BUFFER	SIMD16	{Chan 1-4}	MSD1R_US	{Forbidden}	MAP16b_USU_SIMD16	{Forbidden}	{Chan1-4} MDP_DW_SIMD16
W	SLM	BUFFER	SIMD16	{Chan 1-4}	MSD1W_US	{Forbidden}	MAP16b_USU_SIMD16	{Chan 1-4} MDP_DW_SIMD16	{Forbidden}
R	SLM	BUFFER	SIMD16	{Chan 1-4}		{Forbidden}	MAP32b_USU_SIMD16	{Forbidden}	{Chan1-4} MDP_DW_SIMD16
W	SLM	BUFFER	SIMD16	{Chan 1-4}		{Forbidden}	MAP32b_USU_SIMD16	{Chan1-4} MDP_DW_SIMD16	{Forbidden}
R	A32	BUFFER	SIMD8	{Chan 1-4}	MSD1R_US	{Opt} MH1_A32	MAP32b_USU_SIMD8	{Forbidden}	{Chan1-4} MDP_DW_SIMD8
W	A32	BUFFER	SIMD8	{Chan 1-4}	MSD1W_US	{Opt} MH1_A32	MAP32b_USU_SIMD8	{Chan1-4} MDP_DW_SIMD8	{Forbidden}
R	A32	BUFFER	SIMD16	{Chan 1-4}	MSD1R_US	{Opt} MH1_A32	MAP32b_USU_SIMD16	{Forbidden}	{Chan1-4} MDP_DW_SIMD16
W	A32	BUFFER	SIMD16	{Chan 1-4}	MSD1W_US	{Opt} MH1_A32	MAP32b_USU_SIMD16	{Chan1-4} MDP_DW_SIMD16	{Forbidden}
R	A64	BUFFER	SIMD8	{Chan 1-4}	MSD1R_A64_US	{Forbidden}	MAP64b_USU_SIMD8	{Forbidden}	{Chan1-4} MDP_DW_SIMD8
W	A64	BUFFER	SIMD8	{Chan 1-4}	MSD1W_A64_US	{Forbidden}	MAP64b_USU_SIMD8	{Chan1-4} MDP_DW_SIMD8	{Forbidden}
R	A64	BUFFER	SIMD16	{Chan 1-4}	MSD1R_A64_US	{Forbidden}	MAP64b_USU_SIMD16	{Forbidden}	{Chan1-4} MDP_DW_SIMD16
W	A64	BUFFER	SIMD16	{Chan 1-4}	MSD1W_A64_US	{Forbidden}	MAP64b_USU_SIMD16	{Chan1-4} MDP_DW_SIMD16	{Forbidden}

The number of message registers in the write data payload is determined by the number of channel mask bits that are enabled.

The **Channel Mask** in the message descriptor identifies whether the corresponding channel access is disabled. For BTS surfaces, the surface state **Shader Channel Select** fields identify whether the channels are swizzled for the access.

Typed Surface Read/Write Messages

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
B	DW	R/W	BTS	1D, 2D, 3D, CUBE, BUFFER, NULL	8	{Chan 1-4}	(Surface[U, V, R, LOD])->DW[Chan]	Surface	SG

This message allows direct read/write accesses to typed surfaces using 8 offsets. Up to 4 channels are accessed beginning at the byte address determined: red, green, blue, and alpha channels, in that order.

The execution mask bits may disable accesses on the corresponding SIMD slots. Out-of-bounds accesses are dropped or return zero.

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	BTS	1D, 2D, 3D, CUBE, BUFFER, NULL	SIMD8	{Chan 1-4}	MSD1R_TS	{Forbidden}	MAP32b_TS_SIMD8	{Forbidden}	{Chan1-4} MDP_DW_SIMD8
W	BTS	1D, 2D, 3D, CUBE, BUFFER, NULL	SIMD8	{Chan 1-4}	MSD1W_TS	{Forbidden}	MAP32b_TS_SIMD8	{Chan1-4} MDP_DW_SIMD8	{Forbidden}

The number of message registers in the write data payload is determined by the number of channel mask bits that are enabled.

The **Channel Mask** in the message descriptor identifies whether the corresponding channel access is disabled. The surface state **Shader Channel Select** fields identify whether the channels are swizzled for the access.

See Surface Formats for the description of how the data is formatted for Typed Surface read and write operations.

Atomic Operation Messages

Atomic operation messages cause atomic read-modify-write operations on the destination locations addressed. Atomic operations guarantee that no read or write to the same memory location from this thread or any other thread can occur between the read and the write. There is no guarantee on the write ordering of the individual operations in one SIMD message.

Out-of-bounds atomic accesses are dropped without writing data, and return zero for read data. Atomic accesses to pages in Tiled Resources Translation Tables are required to be IA-coherent. NULL Tiled Resources pages are handled as out-of-bounds accesses.

Untyped atomic messages use the RAW format, with surface type BUFFER or SCRATCH, and perform no type conversion. These messages use the U address parameter, which specifies the byte offset, which must be aligned on the data width (multiple of 8 for QWord, a multiple of 4 for DWord, or a multiple of 2 for Word).

Typed atomic messages use surface types SURFTYPE_1D, 2D, 3D, or BUFFER.

For atomic operations, only the red surface channel is used (the first DWord). The red surface channel select must be set to SCS_RED, in addition to all shader channel selects in the surface state following the rules for the surface format in RENDER_SURFACE_STATE.

The MDC_AOP and MDC_FOP are specified in the Atomic Operation Message Descriptor Control Fields section.

The new value of the destination is computed based on the old value of the destination, and up to two additional sources included in the message (src0 and src1). The sources used by the specific operations are described in the **MDC_AOP** and **MDC_FOP** tables. The AOP operations are subdivided into unary, binary, and trinary operand groupings. Source data payload are only sent for the binary and trinary operations.

The double-width AOP operation CMPWR_2W, also known as CMPRW8B on DWords and CMPWR16B on QWords, require the read-modify-write address offsets to be naturally aligned (QWord and OWord respectively).

The **MDC_RDC** message-specific control field controls whether a value is returned by the atomic message. The value returned depends on the message specific operation (see **MDC_AOP** and **MDC_FOP**).

Word Untyped Atomic Integer Messages

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
W	W	R/W	BTS	BUFFER,NULL	8, 16	1	(Base+U{Slot})->W[0]	Surface	SM
W	W	R/W	SLM	BUFFER	8, 16	1	(Base+U{Slot})->W[0]	Shared	SM
W	W	R/W	A32	BUFFER	8, 16	1	(Base+Buffer+U{Slot})->W[0]	GenState	SM
W	W	R/W	A64	BUFFER	8	1	(0+U{Slot})->W[0]	Canonical	SM

This message performs atomic integer operations on untyped surfaces using 8 or 16 offsets. One Word is accessed beginning at the byte address determined.

For BUFFER accesses, each U offset must be Word-aligned.

The execution mask bits may disable accesses on the corresponding SIMD slots. Out-of-bounds accesses are dropped or return zero. The semantics depend on the address model, as listed in the above table.

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	BTS	BUFFER,NULL	SIMD8	1	MSD1R_WAI	{Forbidden}	MAP32b_USU_SIMD8	{Forbidden}	MDP_W_SIMD8
W	BTS	BUFFER,NULL	SIMD8	1	MSD1W_WAI	{Forbidden}	MAP32b_USU_SIMD8	{Forbidden}	{Forbidden}
R	BTS	BUFFER,NULL	SIMD16	1	MSD1R_WAI	{Forbidden}	MAP32b_USU_SIMD16	{Forbidden}	MDP_W_SIMD16
W	BTS	BUFFER,NULL	SIMD16	1	MSD1W_WAI	{Forbidden}	MAP32b_USU_SIMD16	{Forbidden}	{Forbidden}
R	BTS	BUFFER,NULL	SIMD8	1	MSD1R_WAI		MAP32b_USU_SIMD8	MDP_W_SIMD8	MDP_W_SIMD8
W	BTS	BUFFER,NULL	SIMD8	1	MSD1W_WAI	{Forbidden}	MAP32b_USU_SIMD8	MDP_W_SIMD8	{Forbidden}
R	BTS	BUFFER,NULL	SIMD16	1	MSD1R_WAI		MAP32b_USU_SIMD16	MDP_W_SIMD16	MDP_W_SIMD16
W	BTS	BUFFER,NULL	SIMD16	1	MSD1W_WAI	{Forbidden}	MAP32b_USU_SIMD16	MDP_W_SIMD16	{Forbidden}
R	BTS	BUFFER,NULL	SIMD8	1	MSD1R_WAI		MAP32b_USU_SIMD8	MDP_AOP8_W2	MDP_W_SIMD8

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
W	BTS	BUFFER,NULL	SIMD8	1	MSD1W_WAI	{Forbidden}	MAP32b_USU_SIMD8	MDP_AOP8_W2	{Forbidden}
R	BTS	BUFFER,NULL	SIMD16	1	MSD1R_WAI		MAP32b_USU_SIMD16	MDP_AOP16_W2	MDP_W_SIMD16
W	BTS	BUFFER,NULL	SIMD16	1	MSD1W_WAI	{Forbidden}	MAP32b_USU_SIMD16	MDP_AOP16_W2	{Forbidden}
R	SLM	BUFFER	SIMD8	1	MSD1R_WAI		MAP32b_USU_SIMD8	{Forbidden}	MDP_W_SIMD8
W	SLM	BUFFER	SIMD8	1	MSD1W_WAI	{Forbidden}	MAP32b_USU_SIMD8	{Forbidden}	{Forbidden}
R	SLM	BUFFER	SIMD16	1	MSD1R_WAI	{Forbidden}	MAP32b_USU_SIMD16	{Forbidden}	MDP_W_SIMD16
W	SLM	BUFFER	SIMD16	1	MSD1W_WAI	{Forbidden}	MAP32b_USU_SIMD16	{Forbidden}	{Forbidden}
R	SLM	BUFFER	SIMD16	1	MSD1R_WAI	{Forbidden}	MAP16b_USU_SIMD16	{Forbidden}	MDP_W_SIMD16
W	SLM	BUFFER	SIMD16	1	MSD1W_WAI	{Forbidden}	MAP16b_USU_SIMD16	{Forbidden}	{Forbidden}
R	SLM	BUFFER	SIMD8	1	MSD1R_WAI	{Forbidden}	MAP32b_USU_SIMD8	MDP_W_SIMD8	MDP_W_SIMD8
W	SLM	BUFFER	SIMD8	1	MSD1W_WAI	{Forbidden}	MAP32b_USU_SIMD8	MDP_W_SIMD8	{Forbidden}
R	SLM	BUFFER	SIMD16	1	MSD1R_WAI		MAP32b_USU_SIMD16	MDP_W_SIMD16	MDP_W_SIMD16
W	SLM	BUFFER	SIMD16	1	MSD1W_WAI	{Forbidden}	MAP32b_USU_SIMD16	MDP_W_SIMD16	{Forbidden}
R	SLM	BUFFER	SIMD16	1	MSD1R_WAI		MAP16b_USU_SIMD16	MDP_W_SIMD16	MDP_W_SIMD16
W	SLM	BUFFER	SIMD16	1	MSD1W_WAI	{Forbidden}	MAP16b_USU_SIMD16	MDP_W_SIMD16	{Forbidden}
R	SLM	BUFFER	SIMD8	1	MSD1R_WAI		MAP32b_USU_SIMD8	MDP_AOP8_W2	MDP_W_SIMD8
W	SLM	BUFFER	SIMD8	1	MSD1W_WAI	{Forbidden}	MAP32b_USU_SIMD8	MDP_AOP8_W2	{Forbidden}
R	SLM	BUFFER	SIMD16	1	MSD1R_WAI		MAP32b_USU_SIMD16	MDP_AOP16_W2	MDP_W_SIMD16
W	SLM	BUFFER	SIMD16	1	MSD1W_WAI	{Forbidden}	MAP32b_USU_SIMD16	MDP_AOP16_W2	{Forbidden}
R	SLM	BUFFER	SIMD16	1	MSD1R_WAI		MAP16b_USU_SIMD16	MDP_AOP16_W2	MDP_W_SIMD16
W	SLM	BUFFER	SIMD16	1	MSD1W_WAI	{Forbidden}	MAP16b_USU_SIMD16	MDP_AOP16_W2	{Forbidden}
R	A32	BUFFER	SIMD8	1	MSD1R_WAI	{Opt} MH1_A32	MAP32b_USU_SIMD8	{Forbidden}	MDP_W_SIMD8
W	A32	BUFFER	SIMD8	1	MSD1W_WAI	{Opt} MH1_A32	MAP32b_USU_SIMD8	{Forbidden}	{Forbidden}
R	A32	BUFFER	SIMD16	1	MSD1R_WAI	{Opt} MH1_A32	MAP32b_USU_SIMD16	{Forbidden}	MDP_W_SIMD16
W	A32	BUFFER	SIMD16	1	MSD1W_WAI	{Opt} MH1_A32	MAP32b_USU_SIMD16	{Forbidden}	{Forbidden}
R	A32	BUFFER	SIMD8	1	MSD1R_WAI	{Opt} MH1_A32	MAP32b_USU_SIMD8	MDP_W_SIMD8	MDP_W_SIMD8
W	A32	BUFFER	SIMD8	1	MSD1W_WAI	{Opt} MH1_A32	MAP32b_USU_SIMD8	MDP_W_SIMD8	{Forbidden}
R	A32	BUFFER	SIMD16	1	MSD1R_WAI	{Opt} MH1_A32	MAP32b_USU_SIMD16	MDP_W_SIMD16	MDP_W_SIMD16
W	A32	BUFFER	SIMD16	1	MSD1W_WAI	{Opt} MH1_A32	MAP32b_USU_SIMD16	MDP_W_SIMD16	{Forbidden}
R	A32	BUFFER	SIMD8	1	MSD1R_WAI	{Opt} MH1_A32	MAP32b_USU_SIMD8	MDP_AOP8_W2	MDP_W_SIMD8
W	A32	BUFFER	SIMD8	1	MSD1W_WAI	{Opt} MH1_A32	MAP32b_USU_SIMD8	MDP_AOP8_W2	{Forbidden}
R	A32	BUFFER	SIMD16	1	MSD1R_WAI	{Opt} MH1_A32	MAP32b_USU_SIMD16	MDP_AOP16_W2	MDP_W_SIMD16
W	A32	BUFFER	SIMD16	1	MSD1W_WAI	{Opt} MH1_A32	MAP32b_USU_SIMD16	MDP_AOP16_W2	{Forbidden}
R	A64	BUFFER	SIMD8	1	MSD1R_A64_WAI	{Forbidden}	MAP64b_USU_SIMD8	{Forbidden}	MDP_W_SIMD8
W	A64	BUFFER	SIMD8	1	MSD1W_A64_WAI	{Forbidden}	MAP64b_USU_SIMD8	{Forbidden}	{Forbidden}
R	A64	BUFFER	SIMD8	1	MSD1R_A64_WAI	{Forbidden}	MAP64b_USU_SIMD8	MDP_W_SIMD8	MDP_W_SIMD8
W	A64	BUFFER	SIMD8	1	MSD1W_A64_WAI	{Forbidden}	MAP64b_USU_SIMD8	MDP_W_SIMD8	{Forbidden}
R	A64	BUFFER	SIMD8	1	MSD1R_A64_WAI	{Forbidden}	MAP64b_USU_SIMD8	MDP_AOP8_W2	MDP_W_SIMD8
W	A64	BUFFER	SIMD8	1	MSD1W_A64_WAI	{Forbidden}	MAP64b_USU_SIMD8	MDP_AOP8_W2	{Forbidden}

Programming Note	
Context:	Word Untyped Atomic Integer Messages
AOP imax/imin assume operands are signed 16-bit integers, umax/umin assume operands are unsigned integers. All other operations treat all values as 16-bit unsigned integers. Add and subtract operations wrap without any special indication.	

DWord Untyped Atomic Integer Messages

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
DW	DW	R/W	BTS	BUFFER,NULL	8, 16	1	(Base+U{Slot})->DW[0]	Surface	SM
DW	DW	R/W	SLM	BUFFER	8, 16	1	(Base+U{Slot})->DW[0]	Shared	SM
DW	DW	R/W	A32	BUFFER	8, 16	1	(Base+Buffer+U{Slot})->DW[0]	GenState	SM
DW	DW	R/W	A64	BUFFER	8	1	(0+U{Slot})->DW[0]	Canonical	SM

This message performs atomic integer operations on untyped surfaces using 8 or 16 offsets. One DWord is accessed beginning at the byte address determined.

For BUFFER accesses, each U offset must be DWord-aligned.

The execution mask bits may disable accesses on the corresponding SIMD slots. Out-of-bounds accesses are dropped or return zero. The semantics depend on the address model, as listed in the above table.

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	BTS	BUFFER,NULL	SIMD8	1	MSD1R_DWAI	{Forbidden}	MAP32b_USU_SIMD8	{Forbidden}	MDP_DW_SIMD8
W	BTS	BUFFER,NULL	SIMD8	1	MSD1W_DWAI	{Forbidden}	MAP32b_USU_SIMD8	{Forbidden}	{Forbidden}
R	BTS	BUFFER,NULL	SIMD16	1	MSD1R_DWAI	{Forbidden}	MAP32b_USU_SIMD16	{Forbidden}	MDP_DW_SIMD16
W	BTS	BUFFER,NULL	SIMD16	1	MSD1W_DWAI	{Forbidden}	MAP32b_USU_SIMD16	{Forbidden}	{Forbidden}
R	BTS	BUFFER,NULL	SIMD8	1	MSD1R_DWAI	{Forbidden}	MAP32b_USU_SIMD8	MDP_DW_SIMD8	MDP_DW_SIMD8
W	BTS	BUFFER,NULL	SIMD8	1	MSD1W_DWAI	{Forbidden}	MAP32b_USU_SIMD8	MDP_DW_SIMD8	{Forbidden}
R	BTS	BUFFER,NULL	SIMD16	1	MSD1R_DWAI	{Forbidden}	MAP32b_USU_SIMD16	MDP_DW_SIMD16	MDP_DW_SIMD16
W	BTS	BUFFER,NULL	SIMD16	1	MSD1W_DWAI	{Forbidden}	MAP32b_USU_SIMD16	MDP_DW_SIMD16	{Forbidden}
R	BTS	BUFFER,NULL	SIMD8	1	MSD1R_DWAI	{Forbidden}	MAP32b_USU_SIMD8	MDP_AOP8_DW2	MDP_DW_SIMD8
W	BTS	BUFFER,NULL	SIMD8	1	MSD1W_DWAI	{Forbidden}	MAP32b_USU_SIMD8	MDP_AOP8_DW2	{Forbidden}
R	BTS	BUFFER,NULL	SIMD16	1	MSD1R_DWAI	{Forbidden}	MAP32b_USU_SIMD16	MDP_AOP16_DW2	MDP_DW_SIMD16
W	BTS	BUFFER,NULL	SIMD16	1	MSD1W_DWAI	{Forbidden}	MAP32b_USU_SIMD16	MDP_AOP16_DW2	{Forbidden}
R	SLM	BUFFER	SIMD8	1	MSD1R_DWAI	{Forbidden}	MAP32b_USU_SIMD8	{Forbidden}	MDP_DW_SIMD8
W	SLM	BUFFER	SIMD8	1	MSD1W_DWAI	{Forbidden}	MAP32b_USU_SIMD8	{Forbidden}	{Forbidden}
R	SLM	BUFFER	SIMD16	1	MSD1R_DWAI	{Forbidden}	MAP32b_USU_SIMD16	{Forbidden}	MDP_DW_SIMD16
W	SLM	BUFFER	SIMD16	1	MSD1W_DWAI	{Forbidden}	MAP32b_USU_SIMD16	{Forbidden}	{Forbidden}
R	SLM	BUFFER	SIMD16	1	MSD1R_DWAI	{Forbidden}	MAP16b_USU_SIMD16	{Forbidden}	MDP_DW_SIMD16
W	SLM	BUFFER	SIMD16	1	MSD1W_DWAI	{Forbidden}	MAP16b_USU_SIMD16	{Forbidden}	{Forbidden}
R	SLM	BUFFER	SIMD8	1	MSD1R_DWAI	{Forbidden}	MAP32b_USU_SIMD8	MDP_DW_SIMD8	MDP_DW_SIMD8
W	SLM	BUFFER	SIMD8	1	MSD1W_DWAI	{Forbidden}	MAP32b_USU_SIMD8	MDP_DW_SIMD8	{Forbidden}
R	SLM	BUFFER	SIMD16	1	MSD1R_DWAI	{Forbidden}	MAP32b_USU_SIMD16	MDP_DW_SIMD16	MDP_DW_SIMD16
W	SLM	BUFFER	SIMD16	1	MSD1W_DWAI	{Forbidden}	MAP32b_USU_SIMD16	MDP_DW_SIMD16	{Forbidden}

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	SLM	BUFFER	SIMD16	1	MSD1R_DWAI	{Forbidden}	MAP16b_USU_SIMD16	MDP_DW_SIMD16	MDP_DW_SIMD16
W	SLM	BUFFER	SIMD16	1	MSD1W_DWAI	{Forbidden}	MAP16b_USU_SIMD16	MDP_DW_SIMD16	{Forbidden}
R	SLM	BUFFER	SIMD8	1	MSD1R_DWAI	{Forbidden}	MAP32b_USU_SIMD8	MDP_AOP8_DW2	MDP_DW_SIMD8
W	SLM	BUFFER	SIMD8	1	MSD1W_DWAI	{Forbidden}	MAP32b_USU_SIMD8	MDP_AOP8_DW2	{Forbidden}
R	SLM	BUFFER	SIMD16	1	MSD1R_DWAI	{Forbidden}	MAP32b_USU_SIMD16	MDP_AOP16_DW2	MDP_DW_SIMD16
W	SLM	BUFFER	SIMD16	1	MSD1W_DWAI	{Forbidden}	MAP32b_USU_SIMD16	MDP_AOP16_DW2	{Forbidden}
R	SLM	BUFFER	SIMD16	1	MSD1R_DWAI	{Forbidden}	MAP16b_USU_SIMD16	MDP_AOP16_DW2	MDP_DW_SIMD16
W	SLM	BUFFER	SIMD16	1	MSD1W_DWAI	{Forbidden}	MAP16b_USU_SIMD16	MDP_AOP16_DW2	{Forbidden}
R	A32	BUFFER	SIMD8	1	MSD1R_DWAI	{Opt} MH1_A32	MAP32b_USU_SIMD8	{Forbidden}	MDP_DW_SIMD8
W	A32	BUFFER	SIMD8	1	MSD1W_DWAI	{Opt} MH1_A32	MAP32b_USU_SIMD8	{Forbidden}	{Forbidden}
R	A32	BUFFER	SIMD16	1	MSD1R_DWAI	{Opt} MH1_A32	MAP32b_USU_SIMD16	{Forbidden}	MDP_DW_SIMD16
W	A32	BUFFER	SIMD16	1	MSD1W_DWAI	{Opt} MH1_A32	MAP32b_USU_SIMD16	{Forbidden}	{Forbidden}
R	A32	BUFFER	SIMD8	1	MSD1R_DWAI	{Opt} MH1_A32	MAP32b_USU_SIMD8	MDP_DW_SIMD8	MDP_DW_SIMD8
W	A32	BUFFER	SIMD8	1	MSD1W_DWAI	{Opt} MH1_A32	MAP32b_USU_SIMD8	MDP_DW_SIMD8	{Forbidden}
R	A32	BUFFER	SIMD16	1	MSD1R_DWAI	{Opt} MH1_A32	MAP32b_USU_SIMD16	MDP_DW_SIMD16	MDP_DW_SIMD16
W	A32	BUFFER	SIMD16	1	MSD1W_DWAI	{Opt} MH1_A32	MAP32b_USU_SIMD16	MDP_DW_SIMD16	{Forbidden}
R	A32	BUFFER	SIMD8	1	MSD1R_DWAI	{Opt} MH1_A32	MAP32b_USU_SIMD8	MDP_AOP8_DW2	MDP_DW_SIMD8
W	A32	BUFFER	SIMD8	1	MSD1W_DWAI	{Opt} MH1_A32	MAP32b_USU_SIMD8	MDP_AOP8_DW2	{Forbidden}
R	A32	BUFFER	SIMD16	1	MSD1R_DWAI	{Opt} MH1_A32	MAP32b_USU_SIMD16	MDP_AOP16_DW2	MDP_DW_SIMD16
W	A32	BUFFER	SIMD16	1	MSD1W_DWAI	{Opt} MH1_A32	MAP32b_USU_SIMD16	MDP_AOP16_DW2	{Forbidden}
R	A64	BUFFER	SIMD8	1	MSD1R_A64_DWAI	{Forbidden}	MAP64b_USU_SIMD8	{Forbidden}	MDP_DW_SIMD8
W	A64	BUFFER	SIMD8	1	MSD1W_A64_DWAI	{Forbidden}	MAP64b_USU_SIMD8	{Forbidden}	{Forbidden}
R	A64	BUFFER	SIMD8	1	MSD1R_A64_DWAI	{Forbidden}	MAP64b_USU_SIMD8	MDP_DW_SIMD8	MDP_DW_SIMD8
W	A64	BUFFER	SIMD8	1	MSD1W_A64_DWAI	{Forbidden}	MAP64b_USU_SIMD8	MDP_DW_SIMD8	{Forbidden}
R	A64	BUFFER	SIMD8	1	MSD1R_A64_DWAI	{Forbidden}	MAP64b_USU_SIMD8	MDP_AOP8_DW2	MDP_DW_SIMD8
W	A64	BUFFER	SIMD8	1	MSD1W_A64_DWAI	{Forbidden}	MAP64b_USU_SIMD8	MDP_AOP8_DW2	{Forbidden}

Programming Note	
Context:	DWord Untyped Atomic Integer Messages

AOP imax/imin assume operands are signed 32-bit integers, umax/umin assume operands are unsigned integers. All other operations treat all values as 32-bit unsigned integers. Add and subtract operations wrap without any special indication.

QWord Untyped Atomic Integer Messages

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
QW	QW	R/W	A64	BUFFER	8	1	(0+U{Slot})->QW[0]	Canonical	SM

This message performs QWord atomic integer operations on untyped surfaces using 8 or 16 offsets. The surface format is RAW. One QWord is accessed beginning at the byte address determined.

For BUFFER accesses, each U offset must be QWord-aligned.

The execution mask bits may disable accesses on the corresponding SIMD slots. Out-of-bounds accesses are dropped or return zero. The semantics depend on the address model, as listed in the above table.

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	A64	BUFFER	SIMD8	1	MSD1R_A64_QWAI	{Forbidden}	MAP64b_USU_SIMD8	{Forbidden}	MDP_QW_SIMD8
W	A64	BUFFER	SIMD8	1	MSD1W_A64_QWAI	{Forbidden}	MAP64b_USU_SIMD8	{Forbidden}	{Forbidden}
R	A64	BUFFER	SIMD8	1	MSD1R_A64_QWAI	{Forbidden}	MAP64b_USU_SIMD8	MDP_QW_SIMD8	MDP_QW_SIMD8
W	A64	BUFFER	SIMD8	1	MSD1W_A64_QWAI	{Forbidden}	MAP64b_USU_SIMD8	MDP_QW_SIMD8	{Forbidden}
R	A64	BUFFER	SIMD8	1	MSD1R_A64_QWAI	{Forbidden}	MAP64b_USU_SIMD8	MDP_A64_AOP8_QW2	MDP_QW_SIMD8
W	A64	BUFFER	SIMD8	1	MSD1W_A64_QWAI	{Forbidden}	MAP64b_USU_SIMD8	MDP_A64_AOP8_QW2	{Forbidden}

Programming Note	
Context:	QWord Untyped Atomic Integer Messages

AOP imax/imin assume operands are signed 64-bit integers; umax/umin assume operands are unsigned integers. All other operations treat all values as 64-bit unsigned integers. Add and subtract operations wrap without any special indication.

Word Untyped Atomic Float Messages

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
W	W	R/W	BTS	BUFFER,NULL	8, 16	1	(Base+U{Slot})->W[0]	Surface	SM
W	W	R/W	SLM	BUFFER	8, 16	1	(Base+U{Slot})->W[0]	Shared	SM
W	W	R/W	A32	BUFFER	8, 16	1	(Base+Buffer+U{Slot})->W[0]	GenState	SM
W	W	R/W	A64	BUFFER	8	1	(0+U{Slot})->W[0]	Canonical	SM

This message performs atomic float operations on untyped surfaces using 8 or 16 offsets. One Word is accessed beginning at the byte address determined.

For BUFFER accesses, each U offset must be Word-aligned.

The execution mask bits may disable accesses on the corresponding SIMD slots. Out-of-bounds accesses are dropped or return zero. The semantics depend on the address model, as listed in the above table.

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	BTS	BUFFER,NULL	SIMD8	1	MSD1R_WAF	{Forbidden}	MAP32b_USU_SIMD8	MDP_W_SIMD8	MDP_W_SIMD8
W	BTS	BUFFER,NULL	SIMD8	1	MSD1W_WAF	{Forbidden}	MAP32b_USU_SIMD8	MDP_W_SIMD8	{Forbidden}
R	BTS	BUFFER,NULL	SIMD16	1	MSD1R_WAF	{Forbidden}	MAP32b_USU_SIMD16	MDP_W_SIMD16	MDP_W_SIMD16
W	BTS	BUFFER,NULL	SIMD16	1	MSD1W_WAF	{Forbidden}	MAP32b_USU_SIMD16	MDP_W_SIMD16	{Forbidden}
R	BTS	BUFFER,NULL	SIMD8	1	MSD1R_WAF	{Forbidden}	MAP32b_USU_SIMD8	MDP_AOP8_W2	MDP_W_SIMD8
W	BTS	BUFFER,NULL	SIMD8	1	MSD1W_WAF	{Forbidden}	MAP32b_USU_SIMD8	MDP_AOP8_W2	{Forbidden}
R	BTS	BUFFER,NULL	SIMD16	1	MSD1R_WAF	{Forbidden}	MAP32b_USU_SIMD16	MDP_AOP16_W2	MDP_W_SIMD16
W	BTS	BUFFER,NULL	SIMD16	1	MSD1W_WAF	{Forbidden}	MAP32b_USU_SIMD16	MDP_AOP16_W2	{Forbidden}
R	SLM	BUFFER	SIMD8	1	MSD1R_WAF	{Forbidden}	MAP32b_USU_SIMD8	MDP_W_SIMD8	MDP_W_SIMD8
W	SLM	BUFFER	SIMD8	1	MSD1W_WAF	{Forbidden}	MAP32b_USU_SIMD8	MDP_W_SIMD8	{Forbidden}
R	SLM	BUFFER	SIMD16	1	MSD1R_WAF	{Forbidden}	MAP32b_USU_SIMD16	MDP_W_SIMD16	MDP_W_SIMD16
W	SLM	BUFFER	SIMD16	1	MSD1W_WAF	{Forbidden}	MAP32b_USU_SIMD16	MDP_W_SIMD16	{Forbidden}
R	SLM	BUFFER	SIMD16	1	MSD1R_WAF	{Forbidden}	MAP16b_USU_SIMD16	MDP_W_SIMD16	MDP_W_SIMD16
W	SLM	BUFFER	SIMD16	1	MSD1W_WAF	{Forbidden}	MAP16b_USU_SIMD16	MDP_W_SIMD16	{Forbidden}
R	SLM	BUFFER	SIMD8	1	MSD1R_WAF	{Forbidden}	MAP32b_USU_SIMD8	MDP_AOP8_W2	MDP_W_SIMD8
W	SLM	BUFFER	SIMD8	1	MSD1W_WAF	{Forbidden}	MAP32b_USU_SIMD8	MDP_AOP8_W2	{Forbidden}
R	SLM	BUFFER	SIMD16	1	MSD1R_WAF	{Forbidden}	MAP32b_USU_SIMD16	MDP_AOP16_W2	MDP_W_SIMD16
W	SLM	BUFFER	SIMD16	1	MSD1W_WAF	{Forbidden}	MAP32b_USU_SIMD16	MDP_AOP16_W2	{Forbidden}
R	SLM	BUFFER	SIMD16	1	MSD1R_WAF	{Forbidden}	MAP16b_USU_SIMD16	MDP_AOP16_W2	MDP_W_SIMD16
W	SLM	BUFFER	SIMD16	1	MSD1W_WAF	{Forbidden}	MAP16b_USU_SIMD16	MDP_AOP16_W2	{Forbidden}
R	A32	BUFFER	SIMD8	1	MSD1R_WAF	{Opt} MH1_A32	MAP32b_USU_SIMD8	MDP_W_SIMD8	MDP_W_SIMD8
W	A32	BUFFER	SIMD8	1	MSD1W_WAF	{Opt} MH1_A32	MAP32b_USU_SIMD8	MDP_W_SIMD8	{Forbidden}

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	A32	BUFFER	SIMD16	1	MSD1R_WAF	{Opt} MH1_A32	MAP32b_USU_SIMD16	MDP_W_SIMD16	MDP_W_SIMD16
W	A32	BUFFER	SIMD16	1	MSD1W_WAF	{Opt} MH1_A32	MAP32b_USU_SIMD16	MDP_W_SIMD16	{Forbidden}
R	A32	BUFFER	SIMD8	1	MSD1R_WAF	{Opt} MH1_A32	MAP32b_USU_SIMD8	MDP_AOP8_W2	MDP_W_SIMD8
W	A32	BUFFER	SIMD8	1	MSD1W_WAF	{Opt} MH1_A32	MAP32b_USU_SIMD8	MDP_AOP8_W2	{Forbidden}
R	A32	BUFFER	SIMD16	1	MSD1R_WAF	{Opt} MH1_A32	MAP32b_USU_SIMD16	MDP_AOP16_W2	MDP_W_SIMD16
W	A32	BUFFER	SIMD16	1	MSD1W_WAF	{Opt} MH1_A32	MAP32b_USU_SIMD16	MDP_AOP16_W2	{Forbidden}
R	A64	BUFFER	SIMD8	1	MSD1R_A64_WAF	{Forbidden}	MAP64b_USU_SIMD8	MDP_W_SIMD8	MDP_W_SIMD8
W	A64	BUFFER	SIMD8	1	MSD1W_A64_WAF	{Forbidden}	MAP64b_USU_SIMD8	MDP_W_SIMD8	{Forbidden}
R	A64	BUFFER	SIMD8	1	MSD1R_A64_WAF	{Forbidden}	MAP64b_USU_SIMD8	MDP_AOP8_W2	MDP_W_SIMD8
W	A64	BUFFER	SIMD8	1	MSD1W_A64_WAF	{Forbidden}	MAP64b_USU_SIMD8	MDP_AOP8_W2	{Forbidden}

DWord Untyped Atomic Float Messages

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
DW	DW	R/W	BTS	BUFFER,NULL	8, 16	1	(Base+U{Slot})->DW[0]	Surface	SM
DW	DW	R/W	SLM	BUFFER	8, 16	1	(Base+U{Slot})->DW[0]	Shared	SM
DW	DW	R/W	A32	BUFFER	8, 16	1	(Base+Buffer+U{Slot})->DW[0]	GenState	SM
DW	DW	R/W	A64	BUFFER	8	1	(0+U{Slot})->DW[0]	Canonical	SM

This message performs atomic float operations on untyped surfaces using 8 or 16 offsets. One DWord is accessed beginning at the byte address determined.

For BUFFER accesses, each U offset must be DWord-aligned.

The execution mask bits may disable accesses on the corresponding SIMD slots. Out-of-bounds accesses are dropped or return zero. The semantics depend on the address model, as listed in the above table.

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	BTS	BUFFER,NULL	SIMD8	1	MSD1R_DWAF	{Forbidden}	MAP32b_USU_SIMD8	MDP_DW_SIMD8	MDP_DW_SIMD8
W	BTS	BUFFER,NULL	SIMD8	1	MSD1W_DWAF	{Forbidden}	MAP32b_USU_SIMD8	MDP_DW_SIMD8	{Forbidden}
R	BTS	BUFFER,NULL	SIMD16	1	MSD1R_DWAF	{Forbidden}	MAP32b_USU_SIMD16	MDP_DW_SIMD16	MDP_DW_SIMD16
W	BTS	BUFFER,NUL	SIMD1	1	MSD1W_DWAF	{Forbidden	MAP32b_USU_SIMD1	MDP_DW_SIMD1	{Forbidden}

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
		L	6			}	6	6	
R	BTS	BUFFER,NULL	SIMD8	1	MSD1R_DWAF	{Forbidden}	MAP32b_USU_SIMD8	MDP_AOP8_DW2	MDP_DW_SIMD8
W	BTS	BUFFER,NULL	SIMD8	1	MSD1W_DWAF	{Forbidden}	MAP32b_USU_SIMD8	MDP_AOP8_DW2	{Forbidden}
R	BTS	BUFFER,NULL	SIMD16	1	MSD1R_DWAF	{Forbidden}	MAP32b_USU_SIMD16	MDP_AOP16_DW2	MDP_DW_SIMD16
W	BTS	BUFFER,NULL	SIMD16	1	MSD1W_DWAF	{Forbidden}	MAP32b_USU_SIMD16	MDP_AOP16_DW2	{Forbidden}
R	SLM	BUFFER	SIMD8	1	MSD1R_DWAF	{Forbidden}	MAP32b_USU_SIMD8	MDP_DW_SIMD8	MDP_DW_SIMD8
W	SLM	BUFFER	SIMD8	1	MSD1W_DWAF	{Forbidden}	MAP32b_USU_SIMD8	MDP_DW_SIMD8	{Forbidden}
R	SLM	BUFFER	SIMD16	1	MSD1R_DWAF	{Forbidden}	MAP32b_USU_SIMD16	MDP_DW_SIMD16	MDP_DW_SIMD16
W	SLM	BUFFER	SIMD16	1	MSD1W_DWAF	{Forbidden}	MAP32b_USU_SIMD16	MDP_DW_SIMD16	{Forbidden}
R	SLM	BUFFER	SIMD16	1	MSD1R_DWAF	{Forbidden}	MAP16b_USU_SIMD16	MDP_DW_SIMD16	MDP_DW_SIMD16
W	SLM	BUFFER	SIMD16	1	MSD1W_DWAF	{Forbidden}	MAP16b_USU_SIMD16	MDP_DW_SIMD16	{Forbidden}
R	SLM	BUFFER	SIMD8	1	MSD1R_DWAF	{Forbidden}	MAP32b_USU_SIMD8	MDP_AOP8_DW2	MDP_DW_SIMD8
W	SLM	BUFFER	SIMD8	1	MSD1W_DWAF	{Forbidden}	MAP32b_USU_SIMD8	MDP_AOP8_DW2	{Forbidden}
R	SLM	BUFFER	SIMD16	1	MSD1R_DWAF	{Forbidden}	MAP32b_USU_SIMD16	MDP_AOP16_DW2	MDP_DW_SIMD16
W	SLM	BUFFER	SIMD16	1	MSD1W_DWAF	{Forbidden}	MAP32b_USU_SIMD16	MDP_AOP16_DW2	{Forbidden}
R	SLM	BUFFER	SIMD16	1	MSD1R_DWAF	{Forbidden}	MAP16b_USU_SIMD16	MDP_AOP16_DW2	MDP_DW_SIMD16
W	SLM	BUFFER	SIMD16	1	MSD1W_DWAF	{Forbidden}	MAP16b_USU_SIMD16	MDP_AOP16_DW2	{Forbidden}
R	A32	BUFFER	SIMD8	1	MSD1R_DWAF	{Opt} MH1_A32	MAP32b_USU_SIMD8	MDP_DW_SIMD8	MDP_DW_SIMD8
W	A32	BUFFER	SIMD8	1	MSD1W_DWAF	{Opt} MH1_A32	MAP32b_USU_SIMD8	MDP_DW_SIMD8	{Forbidden}
R	A32	BUFFER	SIMD16	1	MSD1R_DWAF	{Opt} MH1_A32	MAP32b_USU_SIMD16	MDP_DW_SIMD16	MDP_DW_SIMD16
W	A32	BUFFER	SIMD16	1	MSD1W_DWAF	{Opt} MH1_A32	MAP32b_USU_SIMD16	MDP_DW_SIMD16	{Forbidden}
R	A32	BUFFER	SIMD8	1	MSD1R_DWAF	{Opt} MH1_A32	MAP32b_USU_SIMD8	MDP_AOP8_DW2	MDP_DW_SIMD8
W	A32	BUFFER	SIMD8	1	MSD1W_DWAF	{Opt} MH1_A32	MAP32b_USU_SIMD8	MDP_AOP8_DW2	{Forbidden}
R	A32	BUFFER	SIMD16	1	MSD1R_DWAF	{Opt} MH1_A32	MAP32b_USU_SIMD16	MDP_AOP16_DW2	MDP_DW_SIMD16
W	A32	BUFFER	SIMD16	1	MSD1W_DWAF	{Opt} MH1_A32	MAP32b_USU_SIMD16	MDP_AOP16_DW2	{Forbidden}

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	A64	BUFFER	SIMD8	1	MSD1R_A64_DWAF	{Forbidden}	MAP64b_USU_SIMD8	MDP_DW_SIMD8	MDP_DW_SIMD8
W	A64	BUFFER	SIMD8	1	MSD1W_A64_DWAF	{Forbidden}	MAP64b_USU_SIMD8	MDP_DW_SIMD8	{Forbidden}
R	A64	BUFFER	SIMD8	1	MSD1R_A64_DWAF	{Forbidden}	MAP64b_USU_SIMD8	MDP_AOP8_DW2	MDP_DW_SIMD8
W	A64	BUFFER	SIMD8	1	MSD1W_A64_DWAF	{Forbidden}	MAP64b_USU_SIMD8	MDP_AOP8_DW2	{Forbidden}

DWord Typed Atomic Integer Messages

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
DW	DW	R/W	BTS	1D, 2D, 3D, CUBE	8	1	(Surface[U, V, R, LOD])->DW[0]	Surface	SG
DW	DW	R/W	BTS	BUFFER, NULL	8	1	(Surface[U])->DW[0]	Surface	SG

This message performs atomic integer operations on typed surfaces using 8 offsets. The surface format must be one of R32_UINT, R32_SINT or R32_FLOAT, and the surface type must be SURFTYPE_1D, SURFTYPE_2D, SURFTYPE_3D, SURFTYPE_CUBE, or SURFTYPE_BUFFER. One DWord is accessed beginning at the byte address determined by the address payload and the surface format. Each access is a DWord and must be DWord-aligned.

The execution mask bits may disable accesses on the corresponding SIMD slots. Out-of-bounds accesses are dropped or return zero. The semantics depend on the address model, as listed in the above table.

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	BTS	1D, 2D, 3D, CUBE	SIMD8	1	MSD1R_DWTAI	{Forbidden}	MAP32b_TS_SIMD8	{Forbidden}	MDP_DW_SIMD8
W	BTS	1D, 2D, 3D, CUBE	SIMD8	1	MSD1W_DWTAI	{Forbidden}	MAP32b_TS_SIMD8	{Forbidden}	{Forbidden}
R	BTS	BUFFER, NULL	SIMD8	1	MSD1R_DWTAI	{Forbidden}	MAP32b_USU_SIMD8	{Forbidden}	MDP_DW_SIMD8
W	BTS	BUFFER, NULL	SIMD8	1	MSD1W_DWTAI	{Forbidden}	MAP32b_USU_SIMD8	{Forbidden}	{Forbidden}
R	BTS	1D, 2D, 3D, CUBE	SIMD8	1	MSD1R_DWTAI	{Forbidden}	MAP32b_TS_SIMD8	MDP_DW_SIMD8	MDP_DW_SIMD8
W	BTS	1D, 2D, 3D, CUBE	SIMD8	1	MSD1W_DWTAI	{Forbidden}	MAP32b_TS_SIMD8	MDP_DW_SIMD8	{Forbidden}
R	BTS	BUFFER, NULL	SIMD8	1	MSD1R_DWTAI	{Forbidden}	MAP32b_USU_SIMD8	MDP_DW_SIMD8	MDP_DW_SIMD8
W	BTS	BUFFER, NULL	SIMD8	1	MSD1W_DWTAI	{Forbidden}	MAP32b_USU_SIMD8	MDP_DW_SIMD8	{Forbidden}
R	BTS	1D, 2D, 3D, CUBE	SIMD8	1	MSD1R_DWTAI	{Forbidden}	MAP32b_TS_SIMD8	MDP_AOP8_DW2	MDP_DW_SIMD8
W	BTS	1D, 2D,	SIMD8	1	MSD1W_DWTAI	{Forbidden}	MAP32b_TS_SIMD8	MDP_AOP8_DW2	{Forbidden}

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
		3D, CUBE							
R	BTS	BUFFER, NULL	SIMD8	1	MSD1R_DWTAI	{Forbidden}	MAP32b_USU_SIMD8	MDP_AOP8_DW2	MDP_DW_SIMD8
W	BTS	BUFFER, NULL	SIMD8	1	MSD1W_DWTAI	{Forbidden}	MAP32b_USU_SIMD8	MDP_AOP8_DW2	{Forbidden}

Programming Note	
Context:	DWord Typed Atomic Integer Messages
The U address payload specifies a pixel index into the surface, which is multiplied by the **Surface Pitch** from the Surface State to generate the byte address. That final byte address must be Dword aligned.	

Word Atomic Counter Messages

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
W	W	R/W	BTS	Any	8	1	(Surface Append Buffer)->W[0]	Ignored	SG

This message performs atomic integer operations on the "Append Counter" associated with the surface, using 8 offsets. One Word is accessed beginning at the byte address of the surface's auxiliary buffer address, which must be Word-aligned.

For Append Counter Operations there is no address payload: the address is provided by the append counter field in the surface state. The data payloads are the same as untyped atomic integer operations. The counter values in memory are stored as untyped, so no format conversion is needed by the data port when these messages require return data to (the return data from L3 is always treated as untyped, independent of the surface format.

When accessing a surface with this message, if the Auxiliary Surface Mode of the surface state is not AUX_APPEND, the access is treated as out of bounds with the writes being ignored and the reads returning 0.

The execution mask bits may disable accesses on the corresponding SIMD slots.

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	BTS	Any	SIMD8	1	MSD1R_WAC	MH1_BTS_PSM	{Forbidden}	{Forbidden}	MDP_W_SIMD8
W	BTS	Any	SIMD8	1	MSD1W_WAC	MH1_BTS_PSM	{Forbidden}	{Forbidden}	{Forbidden}
R	BTS	Any	SIMD8	1	MSD1R_WAC	{Forbidden}	{Forbidden}	MDP_W_SIMD8	MDP_W_SIMD8
W	BTS	Any	SIMD8	1	MSD1W_WAC	{Forbidden}	{Forbidden}	MDP_W_SIMD8	{Forbidden}

Programming Note	
Context:	Word Atomic Counter Messages
• The unary atomic integer operations require a message header (so that the message data on the OBUS is not	

Programming Note	
Context:	Word Atomic Counter Messages
empty). • The binary atomic integer operations forbids a message header, so the Pixel Sample Mask is defaulted to be fully enabled. • The Append Counter cannot be used with an MSAA auxiliary surface.	

DWord Atomic Counter Messages

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
DW	DW	R/W	BTS	Any	8	1	(Surface Append Buffer)->DW[0]	Ignored	SG

This message performs atomic integer operations on the "Append Counter" associated with the surface, using 8 offsets. One DWord is accessed beginning at the byte address of the surface's auxiliary buffer address, which must be DWord-aligned.

For Append Counter Operations there is no address payload: the address is provided by the append counter field in the surface state. The data payloads are the same as untyped atomic integer operations.

When accessing a surface with this message, if the Auxiliary Surface Mode of the surface state is not AUX_APPEND, the access is treated as out of bounds with the writes being ignored and the reads returning 0.

The execution mask bits may disable accesses on the corresponding SIMD slots.

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	BTS	Any	SIMD8	1	MSD1R_DWAC	MH1_BTS_PSM	{Forbidden}	{Forbidden}	MDP_DW_SIMD8
W	BTS	Any	SIMD8	1	MSD1W_DWAC	MH1_BTS_PSM	{Forbidden}	{Forbidden}	{Forbidden}
R	BTS	Any	SIMD8	1	MSD1R_DWAC	{Forbidden}	{Forbidden}	MDP_DW_SIMD8	MDP_DW_SIMD8
W	BTS	Any	SIMD8	1	MSD1W_DWAC	{Forbidden}	{Forbidden}	MDP_DW_SIMD8	{Forbidden}

Programming Note	
Context:	DWord Atomic Counter Messages
• The unary atomic integer operations require a message header (so that the message data on the OBUS is not empty). • The current implementation of the binary atomic integer operations forbids a message header, so the Pixel Sample Mask is defaulted to be fully enabled. • The Append Counter cannot be used with an MSAA auxiliary surface.	

Media Surface Read/Write Messages

The media block and transpose messages allow direct read/write accesses to media surfaces. These messages operate on rectangular blocks of pixel data.

Media surfaces are 2D types with some media-specific characteristics (for example, vertical offset handling for interlaced frames). The surface state specifies all the parameters.

There are some limitations on what data surface formats are supported by data ports, when compared to the Sampler, Render Cache, and the fixed function Video encode and decode engines. See the specific messages and Addressing 2D Media Surfaces in this volume for more details.

There are some limitations on what data surface formats are supported by data ports, when compared to the Render Cache and the fixed function Video encode and decode engines. See the specific messages and Addressing 2D Media Surfaces in this volume for more details.

Media Block Read/Write Messages

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
DW	DW	R/W	BTS	2D, NULL	1 block	Width x Height	(Surface[X, Y])->DW[0]	Media	Ignored

The read form of this message enables a rectangular block of data samples to be read from the source surface and written into the GRF. The write form enables data from the GRF to be written to a rectangular block. The message specifies as inputs a signed (X, Y) coordinate into the 2D surface and a block size (Width, Height).

The execution mask is ignored. Out-of-bounds writes are dropped. Out-of-bounds reads return a replicated boundary pixel. More details on bounds checking is described in Addressing 2D Media Surfaces in this volume.

Applications:

Block reads/writes for media

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	BTS	2D, NULL	1 block	Height rows x Width bytes	MSD1R_MB	MH_MB	{Forbidden}	{Forbidden}	MDP_HW1, MDP_HW2, MDP_HW4, MDP_HW8
W	BTS	2D, NULL	1 block	Height rows x Width bytes	MSD1W_MB	MH_MB	{Forbidden}	MDP_HW1, MDP_HW2, MDP_HW4, MDP_HW8	{Forbidden}
R	BTS	2D, NULL	1 block	Height rows x Width bytes	MSD_SC_MB	MH_MB	{Forbidden}	{Forbidden}	MDP_HW1, MDP_HW2, MDP_HW4, MDP_HW8

The only surface type allowed is non-arrayed, non-mipmapped SURFTYPE_2D. Accesses are allowed to SURFTYPE_NULL, reads will return 0 and writes will be ignored. Only non-IA-coherent surfaces may be used with compressed media surfaces. More details on 2D surfaces used by this message is described in Addressing 2D Media Surfaces in this volume.

See 2D Media Surface Formats in this volume for limitations on surface formats supported by this message.

Block Width and Block Height

The maximum Block Height is limited by the Block Width. The maximum data size supported in this message is 64 Dwords (256 Bytes). Block reads and writes wider than 32 bytes are supported only with either linear and Tile X surfaces.

Block Width (bytes)	Block Height (rows)	Tile Modes Supported
1-4	1-64	Linear, TileX, TileY/YF/YS
5-8	1-32	Linear, TileX, TileY/YF/YS
9-16	1-16	Linear, TileX, TileY/YF/YS
17-32	1-8	Linear, TileX, TileY/YF/YS
33-64	1-4	Linear, TileX

The layout of the read and write data payload depends on the Block Height and Block Width. The data is aligned to the least significant bits of the first register, and the register pitch is equal to the next power-of-2 that is greater than or equal to the Block Width. The figure below illustrates the layout of the rows in the registers.

Media Block Row Layout in Registers

	Dword	Width = 1-4								Width = 5-8								Width = 9-16								Width = 17-32								Width = 33-64							
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
GRF	0	8	7	6	5	4	3	2	1	4	3	2	1					2	1							1								1							
	1	16	15	14	13	12	11	10	9	8	7	6	5					4	3							2															
	2	24	23	22	21	20	19	18	17	12	11	10	9					6	5							3								2							
	3	32	31	30	29	28	27	26	25	16	15	14	13					8	7							4															
	4	40	39	38	37	36	35	34	33	20	19	18	17					10	9							5								3							
	5	48	47	46	45	44	43	42	41	24	23	22	21					12	11							6															
	6	56	55	54	53	52	51	50	49	28	27	26	25					14	13							7								4							
	7	64	63	62	61	60	59	58	57	32	31	30	29					16	15							8															

Programming Note	
Context:	Media Block Read/Write Messages

Restrictions Media Block Read and Write:

- The surface base address must be 32-byte aligned.
- Pitch must be a multiple of 64 bytes when the surface is linear.
- For YUV422 formats, the block width and offset must be pixel pair aligned (i.e. DWord-aligned).
- The block width and offset must be aligned to the size of pixels stored in the surface. For a surface with 8bpe pixels for example, the block width and offset can be byte-aligned. For a surface with 16bpe pixels, it is

Programming Note	
Context:	Media Block Read/Write Messages

word-aligned.

Restrictions Media Block Read:

- For media block reads, the Block Width must be 32 or less.

Restrictions Media Block Write:

- For media block writes, both X Offset and Block Width must be DWord-aligned.
- Media block writes to Linear or TileX surfaces must have a height of 16 or less.
- Media block writes are not supported on compressible media surfaces.

Byte Masked Media Block Write Message

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
DW	DW	W	BTS	2D, NULL	1 block	Width x Height	(Surface[X, Y])->DW[0]	Media	Ignored

This message conditionally writes a rectangular block of pixels from GRF data. The message specifies as inputs a signed (X, Y) coordinate into the 2D surface with block size (Width, Height), and a Byte Mask to enable writing of the individual bytes.

The execution mask is ignored. Out-of-bounds writes are dropped. Media bounds checking is more fully described in Addressing 2D Media Surfaces in this volume.

Applications:

Block writes for media

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
W	BTS	2D, NULL	1 block	Height rows x Width bytes	MSD1W_MB	MH_MBBM	{Forbidden}	MDP_HW1, MDP_HW2, MDP_HW4, MDP_HW8	{Forbidden}

The only surface type allowed is non-arrayed, non-mipmapped SURFTYPE_2D. Accesses are allowed to SURFTYPE_NULL, writes will be ignored. More details on 2D surfaces used by this message is described in Addressing 2D Media Surfaces in this volume.

See 2D Media Surface Formats in this volume for limitations on surface formats supported by this message.

Block Width and Block Height

The maximum Block Height is limited by the Block Width. The maximum data size supported in this message is 64 Dwords (256 Bytes).

Block Width (bytes)	Block Height (rows)	Tile Modes Supported
1-4	1-64	Linear, TileX, TileY/YF/YS
5-8	1-32	Linear, TileX, TileY/YF/YS
9-16	1-16	Linear, TileX, TileY/YF/YS
17-32	1-8	Linear, TileX, TileY/YF/YS

The data payload is aligned with the upper left pixel going into the least significant bits of the first register, and the register pitch is equal to the next power-of-2 that is greater than or equal to the Block Width. The Byte Mask applies horizontally to each row of output. When the Byte Mask bit is set, the byte is written. The figure below illustrates the layout of the row data and corresponding mask bits.

Row and Byte Mask Layout

		Width = 1-4								Width = 5-8								Width = 9-16								Width = 17-32							
Dword		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
GRF	0	8	7	6	5	4	3	2	1	4	3	2	1					2	1							1							
	1	16	15	14	13	12	11	10	9	8	7	6	5					4	3							2							
	2	24	23	22	21	20	19	18	17	12	11	10	9					6	5							3							
	3	32	31	30	29	28	27	26	25	16	15	14	13					8	7							4							
	4	40	39	38	37	36	35	34	33	20	19	18	17					10	9							5							
	5	48	47	46	45	44	43	42	41	24	23	22	21					12	11							6							
	6	56	55	54	53	52	51	50	49	28	27	26	25					14	13							7							
	7	64	63	62	61	60	59	58	57	32	31	30	29					16	15							8							
Byte Mask		Bits 3:0 for each Row								Bits 7:0 for each Row								Bits 15:0 for each Row								Bits 31:0 for each Row							

Programming Note	
Context:	Byte Masked Media Block Write Message

- Linear or TileX surfaces must have a block height <= 16.
- Both X Offset and Block Width must be DWord-aligned.
- The surface base address must be 32-byte aligned.
- Pitch must be a multiple of 64 bytes when the surface is linear.
- Media block writes are not supported on compressible media surfaces.

Other Data Port Messages

This section describes the remaining data port messages that do not fit into one of the other groupings.

Read Surface Info Message

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
B	B	R	BTS	Any	1	1		Ignored	Ignored

This message is used to determine information about a surface MIP level. It returns the surface format, type, width, depth, height of the specified surface address.

The LOD is in-bounds if address payload's LOD < surface's MIPCount and if the surface's MinLOD + address payload's LOD < 15. If the LOD is not in-bounds then 0 is returned for the width, height, and depth values.

The execution mask is ignored. No bounds checking is performed on the U, V, or R address components in the address payload.

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	BTS	Any	1	1	MSD_RSI	{Forbidden}	MAP32b_RSI	{Forbidden}	MDP_RSI

There is no message header for this message type.

The 64-bit Instruction Base Address returned in the Writeback Payload is specified as a 48-bit state base address and is extended to 64 bits in HW, so SW can read it for conversion of 64-bit instruction pointers.

Memory Fence Message

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
N/A	N/A	N/A	BTS	N/A	N/A	N/A	N/A	Ignored	Ignored
N/A	N/A	N/A	SLM	N/A	N/A	N/A	N/A	Ignored	Ignored

A memory fence message issued by a thread causes further messages issued by the thread to be blocked until all previous data port messages have completed, or the results can be globally observed from the point of view of other threads in the system. This includes both read and write messages.

The execution mask is ignored. No bounds checking is performed.

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
N/A	BTS	N/A	N/A	N/A	MSD_MEMFENCE	MH_IGNORE	{Forbidden}	{Forbidden}	MDP_DW_SIMD8
N/A	SLM	N/A	N/A	N/A	MSD_MEMFENCE	MH_IGNORE	{Forbidden}	{Forbidden}	MDP_DW_SIMD8

The memory fence message signals completion by returning data into the writeback register. The data returned in the writeback register is undefined. When an instruction reads the writeback register value, then this thread is blocked until all previous data port messages are globally observable. The writeback register must be read before this thread sends another data port message.

A memory fence memory is typically performed prior to a Gateway Barrier message (which synchronizes threads in a group) so that when the barrier releases the threads, each thread is ensured its next data port message is ordered after the memory fence. See Read/Write Ordering in this volume for more details.

Global observability of SLM memory means the SLM memory is visible to all the threads in the thread group that have access to that memory. Global observability of the coherent L3 cache means that those portions of the L3 cache are observable both to graphics threads and to the rest of the system. Global observability of non-coherent L3 cache means those portions of the L3 cache are observable to graphics threads but not necessarily to the rest of the system.

The memory fence message descriptor also has controls to flush (invalidate) various caches. These controls are generally not used.

Programming Note	
Context:	Memory Fence Message
• The L3 has a few different partitioning schemes. In some cases RW data, Constant data and/or texture data can be mixed in the same partition. If a flush is needed for any data type in the partition the entire partition is flushed. • When a cache flush control is specified to the L3 cache, the completion of the memory fence message does not mean that the cache flush has completed. When a thread needs the flush to be completed before continuing execution, the thread should issue a data port read or atomic operation (to any global address) after the cache flush. The cache flush will be completed when this global memory read or atomic operation completes.	

URB Read/Write Messages

The URB read and write messages allow direct read/write accesses to the Unified Return Buffer used by the 3D pipeline.

DWord URB Read/Write Messages

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
OW	DW	R/W	URB	BUFFER	8	{Chan 1-16}	Handle->OW[GlobalOffset+Offset{Slot}].DW[chan]	NA	SM

For each SIMD8 slot, this message reads or writes up to 8 or 16 contiguous DWords starting at each slot's offset.

The number of DWords to read is specified by RLEN in the message descriptor. The number of DWords to write is specified by the MLEN in the message descriptor, after accounting for the message header and address payloads.

The per-slot message offset address payload is optional. If present, it is added to the **Global Offset** in the message descriptor to calculate the offset from the handle specified in the message header.

Masked write messages supply an additional address payload: the per-slot Channel Masks that control which URB Dwords are written. The unmasked read and write messages do not supply a Channel Mask address payload and access all the channels in the message.

The execution mask bits may disable disable all channel accesses on the corresponding SIMD slots.

Applications:

- Write 3D shader kernel results to URB for next phase of 3D pipeline
- Read 3D attributes from URB for this 3D shader kernel

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	URB	BUFFER	SIMD8	{Chan 1-8}	MSDUR_DWS	MH_URB_HANDLE	{opt} MAPU_SIMD8	{Forbidden}	{Chan 1-8} MDP_DW_SIMD8
W	URB	BUFFER	SIMD8	{Chan 1-8}	MSDUR_DWS	MH_URB_HANDLE	{opt} MAPU_SIMD8	{Chan 1-8} MDP_DW_SIMD8	{Forbidden}
W	URB	BUFFER	SIMD8	{Chan 1-8)	MSDUW_MDWS	MH_URB_HANDLE	{opt} MAPU_SIMD8 + MAPU_CMASK_SIMD8	{Chan 1-8} MDP_DW_SIMD8	{Forbidden}

Message-Specific Descriptors

All the operations supported on the 4 data ports (Data Port 0, 1, 2, and RO), and over the 4 address models (BTS, SLM, A32, A64) are described by the Message Descriptors.

All the message descriptors indicate whether a Message Header is present, and the Message Type. The decoding of each message descriptor is specific to the message type and the data port it is sent to.

Data Port 0, 1, and RO usually provide a Binding Table Index (BTI) to specify a surface in the BTS address model. The special entry 255 is used to reference Stateless A32 or A64 address model, and the special entry 254 is used to reference the SLM address model.

The special entry 252 is used to reference bindless resource operation. When 252 is used, the surface is identified by the SSO (Surface State Offset) field. That field is encoded in the SENDS instruction's extended descriptor using indirect register addressing.

Data Port Bindless Surface Extended Message Descriptor

Entries 240-251 are reserved for future use.

Data Port 0 Message Specific Descriptors

This section contains the message specific descriptors for Data Port 0.

MT_DP0 - Data Port 0 Message Types

Scattered Read/Write Messages

Message Descriptor
MSD0R_BS - Byte Scattered Read MSD
MSD0W_BS - Byte Scattered Write MSD

Message Descriptor
MSD0R_DWS - Dword Scattered Read MSD
MSD0W_DWS - Dword Scattered Write MSD

Block Read/Write Messages

Message Descriptor
MSD0R_OWB - Oword Block Read MSD
MSD0W_OWB - Oword Block Write MSD
MSD0R_OWAB - Oword Aligned Block Read MSD
MSD0R_HWB - Scratch Block Read MSD
MSD0W_HWB - Scratch Block Write MSD

Other Data Port Messages

Message Descriptor
MSD_MEMFENCE - Memory Fence MSD

Data Port 1 Message Specific Descriptors

This section contains the message specific descriptors for Data Port 1.

MT_DP1 - Data Port 1 Message Types

A64 Scattered Read/Write

Descriptor
MSD1R_A64_BS - A64 Byte Scattered Read MSD
MSD1W_A64_BS - A64 Byte Scattered Write MSD
MSD1R_A64_DWS - A64 Dword Scattered Read MSD
MSD1W_A64_DWS - A64 Dword Scattered Write MSD
MSD1R_A64_QWS - A64 Qword Scattered Read MSD
MSD1W_A64_QWS - A64 Qword Scattered Write MSD

A64 Block Read/Write

Descriptor
MSD1R_A64_OWB - A64 Oword Block Read MSD
MSD1W_A64_OWB - A64 Oword Block Write MSD
MSD1R_A64_OWUB - A64 Oword Unaligned Block Read MSD
MSD1R_A64_HWB - A64 Hword Block Read MSD
MSD1W_A64_HWB - A64 Hword Block Write MSD

Surface Read/Write

Descriptor
MSD1R_US - Untyped Surface Read MSD
MSD1W_US - Untyped Surface Write MSD
MSD1R_A64_US - A64 Untyped Surface Read MSD
MSD1W_A64_US - A64 Untyped Surface Write MSD
MSD1R_TS - Typed Surface Read MSD
MSD1W_TS - Typed Surface Write MSD

Media Messages

Descriptor
MSD1R_MB - Media Block Read MSD
MSD1W_MB - Media Block Write MSD

A32 Word Untyped Atomic Operations

A32 Dword Untyped Atomic Operations

A32 Dword Untyped Atomic Operations

A64 Dword Untyped Atomic Operations

A64 Qword Untyped Atomic Operations

Word Typed Atomic Operations

Dword Typed Atomic Operations

Word Atomic Counter Operations

Dword Atomic Counter Operations

Read-Only Data Port Message Specific Descriptors

This section contains the message specific descriptors for the Read-Only Data Port .

MT_DP_RO - Read-Only Data Port Message Types

Block Read/WriteBlock Read/WriteBlock Read/Write

Message Descriptor
MSD_CC_DWS - Constant Cache Dword Scattered Read MSD
MSD_CC_OWB - Constant Cache Oword Block Read MSD
MSD_CC_OWUB - Constant Cache Oword Unaligned Block Read MSD
MSD_SC_OWUB - Sampler Cache Oword Unaligned Block Read MSD
MSD_SC_MB - Sampler Cache Media Block Read MSD

Other Messages

Message Descriptor
MSD_RSI - Read Surface Info MSD

URB Data Port Message Specific Descriptors

This section contains the message specific descriptors for the URB Data Port SFID_URB.

URB Messages

Message Descriptor
MSDUR_DWS - URB Dword Read
MSDUW_DWS - URB Dword Write
MSDUW_MDWS - URB Masked Dword Write

Message Headers

Message Headers provide additional parameters used by the message. When present, the Header is 1 register and specifies a common set of parameters used by most of the data port's messages. The specific combinations supported are described in the Messages section.

Data Port 0 supports various block and scattered operations for the Read/Write data cache. The Data Port 0 Message Header is programmed to provide:

Buffer Base Address	A base address offset used for A32 stateless address models. The default value is 0 when the message header is not present.
Global Offset	A Byte, DWord, or QWord offset that is added to base address offset to address the data operands. The default value is 0 when the message header is not present.
Per Thread Scratch Space	This is used for bounds checking by the data port operation, to guarantee that the address calculation does not go outside of the range expected for this thread. The default value is the maximum value (11) when the message header is not present.

The A64 Read/Write Block operation provides A64 Stateless operations on the Data Port 1 Read/Write data cache. This special Data Port 1 Message Header is programmed to provide:

Block Offset 0 **Block Offset 1**	This specifies the 1 or 2 64-bit addresses used for the A64 stateless address model.
HWord Channel Mode	This controls how channel execution masks are interpreted for HWord operations.

Surface and atomic operations, both untyped and typed, are provided on Data Port 1 for the Read/Write data cache. This Message Header is programmed to provide:

Message Header Fields for Data Port 1	
Buffer Base Address	A base address used for SLM and A32 stateless address models. The default value is 0 when the message header is not present.

Data Port 0 Message Headers

This section contains the message headers for Data Port 0.

Message Header
MH_BTS_GO - Block Message Header
MH_A32_GO - Stateless Block Message Header
MH_A32_HWB - Scratch Hword Block Message Header
MH_IGNORE - Ignored Message Header

Data Port 1 Message Headers

This section contains the message headers for Data Port 1.

Message Header
MH_A64_OWB - A64 Oword Block Message Header
MH_A64_HWB - A64 Hword Block Message Header
MH1_A32 - Stateless Surface Message Header
MH_MB - Normal Media Block Message Header
MH_MBBM - Byte Masked Media Block Message Header

URB Data Port Message Headers

This section contains the message headers for the URB Data Port.

Message Header
MH_URB_HANDLE - URB Handle Message Header

Message Address Payloads

The next sections describe all the supported formats of the Address Payloads.

Message Address Payloads are packed into registers based on the address model, surface type, and the SIMD size. The number of registers used by an address payload is 1, 2, 4, or 5.

32 Bit Address Payloads

Address Payload
MAP32b_SIMD8 - SIMD8 32-Bit Address Payload
MAP32b_SIMD16 - SIMD16 32-Bit Address Payload
MAP16b SIMD16 - SIMD16 16-Bit Address Payload

64 Bit Address Payloads

Address Payload
MAP64b_SIMD8 - SIMD8 64-Bit Address Payload
MAP64b_SIMD16 - SIMD16 64-Bit Address Payload

32 Bit Untyped Surface Address Payloads

Address Payload
MAP32b_USU_SIMD8 - SIMD8 Untyped BUFFER Surface 32-Bit Address Payload
MAP32b_USU_SIMD16 - SIMD16 Untyped BUFFER Surface 32-Bit Address Payload
MAP16b USU SIMD16 - SIMD16 Untyped BUFFER Surface 16-Bit Address Payload
MAP32b_USUV_SIMD8 - SIMD8 Untyped STRBUF Surface 32-Bit Address Payload
MAP32b_USUV_SIMD16 - SIMD16 Untyped STRBUF Surface 32-Bit Address Payload
MAP16b USUV SIMD16 - SIMD16 Untyped STRBUF Surface 16-Bit Address Payload

64 Bit Untyped Surface Address Payloads

Address Payload
MAP64b_USU_SIMD8 - SIMD8 Untyped BUFFER Surface 64-Bit Address Payload
MAP64b_USU_SIMD16 - SIMD16 Untyped BUFFER Surface 64-Bit Address Payload

32 Bit Typed Surface Address Payloads

Address Payload
MAP32b_TS_SIMD8 - SIMD8 Typed Surface 32-Bit Address Payload
MAP32b_RSI - Read Surface Info 32-Bit Address Payload

URB Address Payloads

Address Payload
MAPU_SIMD8 - SIMD8 URB Offset Message Address Payload
MAPU_CMASK_SIMD8 - SIMD8 URB Channel Mask Message Address Payload

Message Data Payloads

The next sections describe the all the supported formats of Data Payloads.

Message Data Payloads are packed into registers based on the width of the data and on the SIMD size. The supported data payloads are 1, 2, 4, or 8 registers.

Oword Block Data Payloads

Payload
MDP_OW1L - Lower Oword Block Data Payload
MDP_OW1U - Upper Oword Block Data Payload
MDP_OW2 - Oword 2 Block Data Payload
MDP_OW4 - Oword 4 Block Data Payload
MDP_OW8 - Oword 8 Block Data Payload

Hword Block Data Payloads

Payload
MDP_HW1 - Hword 1 Block Data Payload
MDP_HW2 - Hword 2 Block Data Payload
MDP_HW4 - Hword 4 Block Data Payload
MDP_HW8 - Hword 8 Block Data Payload

Word SIMD Data Payloads

Payload
MDP_W_SIMD8 - Word SIMD8 Data Payload
MDP_W_SIMD16 - Word SIMD16 Data Payload

Dword SIMD Data Payloads

Payload
MDP_DW_SIMD8 - Dword SIMD8 Data Payload
MDP_DW_SIMD16 - Dword SIMD16 Data Payload

Qword SIMD Data Payloads

Payload
MDP_QW_SIMD8 - Qword SIMD8 Data Payload
MDP_QW_SIMD16 - Qword SIMD16 Data Payload

SIMD Atomic Operation Data Payloads

Payload
MDP_AOP8_W2 - Word SIMD8 Atomic Operation CMPWR Message Data Payload
MDP_AOP16_W2 - Word SIMD16 Atomic Operation CMPWR Message Data Payload
MDP_AOP8_DW2 - Dword SIMD8 Atomic Operation CMPWR Message Data Payload
MDP_AOP16_DW2 - Dword SIMD16 Atomic Operation CMPWR Message Data Payload
MDP_AOP8_QW1 - Qword SIMD8 Atomic Operation Return Data Message Data Payload
MDP_AOP16_QW1 - Qword SIMD16 Atomic Operation Return Data Message Data Payload
MDP_AOP8_QW2 - Qword SIMD8 Atomic Operation CMPWR8B Message Data Payload
MDP_AOP16_QW2 - Qword SIMD16 Atomic Operation CMPWR8B Message Data Payload
MDP_A64_AOP8_QW2 - Qword SIMD8 Atomic Operation CMPWR Message Data Payload
MDP_A64_AOP8_OW2 - Oword A64 SIMD8 Atomic Operation CMPWR16B Message Data Payload

Other Data Payloads

Payload
MDP_RSI - Read Surface Info Data Payload

Common Message Descriptor Controls

Many messages are related and share a common set of control fields. For example, many read and write operations use the same Message Specific Controls in their message descriptor. These common fields are described in the next sections.

Binding Table Index Message Descriptor Control Fields

Field
MDC_BTS - Surface Binding Table Index Message Descriptor Control Field
MDC_STATELESS - Stateless Binding Table Index Message Descriptor Control Field
MDC_BTS_A32 - Surface or Stateless Binding Table Index Message Descriptor Control Field
MDC_BTS_SLM_A32 - Any Binding Table Index Message Descriptor Control Field

Header Present Message Descriptor Control Fields

Field
MDC_MHP - Header Present Message Descriptor Control Field
MDC_MHR - Header Required Message Descriptor Control Field
MDC_MHF - Header Forbidden Message Descriptor Control Field
MDC_A32_MHP - A32 Scaled Header Present Message Descriptor Control Field
MDC_A64_MHP - A64 Scaled Header Present Message Descriptor Control Field

Data Blocks Message Descriptor Control Fields

Field
MDC_DB_OW - Oword Data Blocks Message Descriptor Control Field
MDC_DB_OWD - Oword Dual Data Blocks Message Descriptor Control Field
MDC_DB_HW - Hword Register Blocks Message Descriptor Control Field
MDC_A64_DB_OW - A64 Oword Data Blocks Message Descriptor Control Field
MDC_A64_DB_HW - A64 Hword Data Blocks Message Descriptor Control Field
MDC_DS - Data Size Message Descriptor Control Field
MDC_A64_DS - A64 Data Size Message Descriptor Control Field

SIMD Mode Message Descriptor Control Fields

Field
MDC_SM3 - SIMD Mode 3 Message Descriptor Control Field
MDC_SM3S - Subset SIMD Mode 3 Message Descriptor Control Field
MDC_SM2 - SIMD Mode 2 Message Descriptor Control Field
MDC_SM2R - Reversed SIMD Mode 2 Message Descriptor Control Field
MDC_SM2S - Subset SIMD Mode 2 Message Descriptor Control Field
MDC_SG3 - Slot Group 3 Message Descriptor Control Field
MDC_SG2 - Slot Group 2 Message Descriptor Control Field

Atomic Operation Message Descriptor Control Fields

Field
MDC_AOP1 - Atomic Integer Unary Operation Message Descriptor Control Field
MDC_AOP2 - Atomic Integer Binary Operation Message Descriptor Control Field
MDC_AOP3 - Atomic Integer Ternary Operation Message Descriptor Control Field
MDC_AOP3S - Subset Atomic Integer Ternary Operation Message Descriptor Control Field
MDC_FOP2 - Atomic Float Binary Operation Message Descriptor Control Field
MDC_FOP3 - Atomic Float Ternary Operation Message Descriptor Control Field

Other Message Descriptor Control Fields

Field
MDC_IAR - Invalidate After Read Message Descriptor Control Field
MDC_CMODE - Channel Mode Message Descriptor Control Field
MDC_CMASK - Channel Mask Message Descriptor Control Field
MDC_UW_CMASK - Untyped Write Channel Mask Message Descriptor Control Field
MDC_VLSO - Vertical Line Stride Override Message Descriptor Control Field

Common Message Payload Controls

Some message headers and payloads are related and share a common set of control fields. These common payload fields are described below.

Header Controls

Control
MHC_FFTID - FFTID Message Header Control
MHC_A32_BBA - A32 Buffer Base Address Message Header Control
MHC_A64_CMODE - Hword Channel Mode Message Header Control
MHC_BDIM - Block Dimensions Message Header Control
MHC_MB_CONTROL - Normal Media Block Message Header Control
MHC_MBBM_CONTROL - Byte Masked Media Block Message Header Control
MHC_PTSS - Per Thread Scratch Space Message Header Control

Address Controls

Control
MACR_32b - SIMD 32-Bit Address Payload Control
MACR_64b - SIMD 64-Bit Address Payload Control
MACD_LOD - LOD Message Address Payload Control
MACR_LOD_SIMD8 - SIMD8 LOD Message Address Payload Control
MACD_MSAA_SN - MSAA Sample Number Message Address Control

Data Controls

Control
MDCR_W - Word Data Payload Register
MDCR_DW - Dword Data Payload Register
MDCR_QW - Qword Data Payload Register
MDCR_OW - Oword Data Payload Register

URB Controls

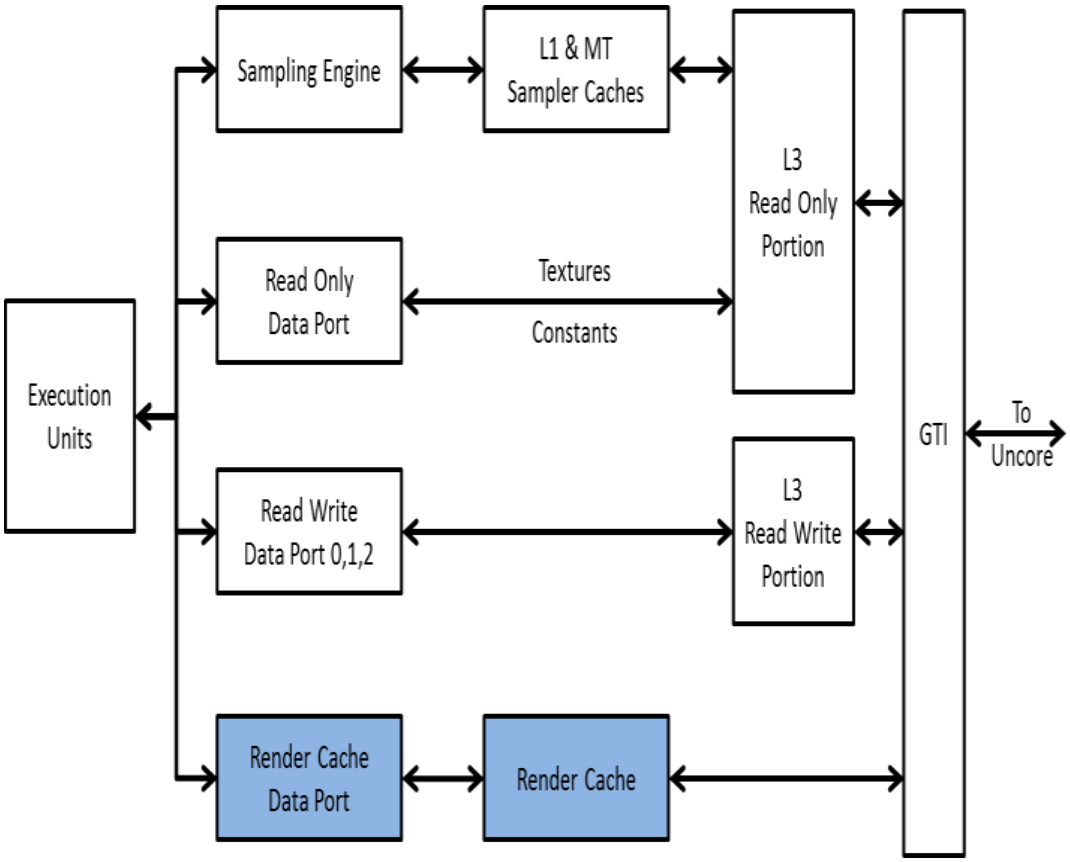
Control
MACD_URB_CMASK - URB Channel Mask Payload Control
MHC_URB_HANDLE - URB Handle Message Header Control

Pixel Data Port

The pixel data port allows read/write accesses to render targets. Render target is cached in Render Color Cache (RCC). Auxiliary Surafce (a.k.a Control Surface) i.e. MCS buffer corresponding to a render target is cached in MSC. Messages on O-bus with Shared Function ID = Render Target (RCC) are routed to Pixel Dataport.

The diagram below shows how the Pixel Data Port connects with the caches and memory subsystem. The execution units and sampling engine are shown for clarity.

Pixel Data Port Connection to Memory



Render Target

This section describes the Render Target as pixel output from the Pixel Shader stage in the 3D Pipeline.

End of Thread Usage

Render Target data port messages may not have the End of Thread bit set in the message descriptor other than the following exceptions:

The Render Target Write message may have End of Thread set for pixel shader threads dispatched by the windower in non-contiguous dispatch mode.

Programming Note	
Context:	End of Thread Usage
When Dispatch Rate is Coarse and the Shader converts to Pixel Phases, the EOT message is required with masks being zero after all the phases.	

Bounds Checking

Read and write accesses to pixels outside of the surface are considered out-of-bounds. However, if only the **Render Target Array Index** is out of bounds and the rest of the access is in-bounds, then the index is set to zero and the read or write surface access is considered in-bounds.

Writes that are out-of-bounds are suppressed and do not modify memory contents. If an in-bounds pixel component is missing due to the Surface Format, the write is suppressed. If an in-bounds pixel component is masked off (Pixel Sample Mask bit is zero), the write is suppressed.

Reads that are out-of-bounds are suppressed and return zero for all components. If an in-bounds pixel component is missing due to the Surface Format, a zero is returned for RGB and a 1.0 is returned for Alpha. A surface read is performed on any in-bounds pixel where the component is present in the Surface Format, regardless of whether it is masked off (Pixel Sample Mask bit is zero). All render target reads, in-bounds or out-of-bounds, modify the GRF.

Render Target Fast Clear

Fast clear of the render target is performed by setting the **Render Target Fast Clear Enable** field in**3DSTATE_PS** and rendering a rectangle The size of the rectangle is related to the size of the MCS.

The following is required when performing a render target fast clear:

- The render target(s) is/are bound as they normally would be, with the MCS surface defined in SURFACE_STATE.
- A rectangle primitive of the same size as the MCS surface is delivered. Please make sure to read MCS/CCS Buffers for Render Targets page for clearing rectangle size guidelines.
- **Depth Test Enable, Depth Buffer Write Enable, Stencil Test Enable, Stencil Buffer Write Enable,** and **Alpha Test Enable** must all be disabled.
- Pixel Shader is not required for fast clearing or resolving render target.

Color Fast Clear Synchronization

Preamble pre fast clear synchronization	Postamble post fast clear synchronization	Additional global programming
PIPE_CONTROL: Depth Stall = 1 Tile Cache Flush = 1 RT Write Flush = 1 SW may choose Range Flush instead of Tile Cache Flush for performance reasons	**PIPE_CONTROL**: Depth Stall = 1 Tile Cache Flush = 1 RT Write Flush = 1 SW may choose Range Flush instead of Tile Cache Flush for performance reasons	Wait on Depth Stall Done Enable = 1

Programming Note	
Context:	Clear Value initialization requirement

In **Render Surface State**

- If Clear Value Address Enable = 1, than the pointer must point at initialized memory, and all 16B need to be initialized, 0's if Clear Value is not explicitly programmed (like at surface creation before Fast Clear operation)
- If Clear Value Address Enable = 0, than clear value address itself is treated as clear value and that should also be initialized to 0's if not explicitly programmed.

Programming Note	
Context:	Workaround

SW must try one of the following two options

1. Before fast clearing any resource, SW must partially resolve the resource i.e. corresponding CCS for the resource MUST NOT be in CLEAR state
2. Clear Color address is based on clear color used for clear DDI

Programming Note	
Context:	Render Target Fast Clear

SW needs to disable Render Target Fast clear for surface type = 2D, surface format = 8 bpp, tile format = TYS pr TY, Mip is not aligned to 32x4 pixels

HwManaged FastClear allows SW to store FastClearValue in separate graphics allocation, instead of keeping them in RENDER_SURFACE_STATE. This behavior can be enabled by setting **ClearValueAddressEnable** in RENDER_SURFACE_STATE.

Proper sequence of commands is as follows:

1. Storing clear color to allocation.
2. Ensuring that step 1. is finished and visible for TextureCache.

3. Performing FastClear.

Step 2. is required on products with ClearColorConversion feature. This feature is enabled by setting **ClearColorConversionEnable.** This causes HW to read stored color from ClearColorAllocation and write back with the native format or RenderTarget - and clear color needs to be present and visible. Reading is done from TextureCache, writing is done to RenderCache.

How to handle steps 1&2 properly:

If ClearValueAllocation was previously read (either with sampler reading this surface with **ClearValueAddressEnable** or with FastClear operation using Clear Color Conversion feature) SW must invalidate TextureCache with proper PIPE_CONTROL. When certain L3/Tile Cache configuration are used, there might be also TileCacheFlush required.

SW must store clear color using MI_STORE_DATA_IMM with **ForceWriteCompletionCheck** bit set.

Render Target Resolve

If the MCS is enabled on a non-multisampled render target, the render target must be resolved before being used for other purposes (display, texture, CPU lock) The clear value from SURFACE_STATE is written into pixels in the render target indicated as clear in the MCS. This is done by setting the **Render Target Resolve Enable** field in 3DSTATE_PS and rendering a full render target sized rectangle. Once this is complete, the render target will contain the same contents as it would have had the rendering been performed with MCS surface disabled. In a typical usage model, the render target(s) need to be resolved after rendering and before using it as a source for any consecutive operation.

When performing a render target resolve, PIPE_CONTROL with end of pipe sync must be delivered.

The Resolve Rectangle size is same as Clear Rectangle size

Programming Note	
Context:	Render Target Resolve
• **Depth Test Enable, Depth Buffer Write Enable, Stencil Test Enable, Stencil Buffer Write Enable,** and **Alpha Test Enable** must all be disabled. • This render target resolve procedure is not supported on multisampled render targets. Unresolved multisampled render targets are directly supported by the sampling engine, which resolves clear values in addition to decompressing the surface This applies to both *ld2dms* and *sample2dms* messages.	

Execution Mask

For Render Target Write messages, either the message header **Pixel Sample Mask** or the channel execution mask is used to control the pixel write operations. If the Message Header is present, then the **Pixel Sample Mask** is used and the channel execution mask is ignored. If the Message Header is not present, then the channel execution mask is used in place of the **Pixel Sample Mask**.

For Render Target Read messages, the channel execution mask is used to control GRF write operations. The message returns zeroes for samples not covered by current pixel coverage mask or samples outside of MSAA index range.

Flushing the Render Cache

The render cache can be flushed via PIPE_CONTROL command with "Render Target Cache Flush" bit set. Typically, end of pipe synchronization that requires render targets to be made visible to Sampler or CPU or Display should use this mechanism to flush render cache. With the Render Target Cache Flush feature, some cases do require flushing render cache as mentioned in the respective feature's sections.

Multiple Render Targets (MRT)

Pixel Shader can output more than one color. Each color is accessed with a separate Render Target Write message, each with a different surface indicated (different binding table index). In this Multiple Render Target (MRT) case, depth buffer and stencil buffer are updated only by the message(s) to the last render target, indicated by the Last Render Target Select bit set to clear the pixel scoreboard bits.

MRT is not supported when one or more RTs have these surface formats: YCRCB_SWAPUVY, YCRCB_SWAPUV, YCRCB_SWAPY, or YCRCB_NORMAL.

MCS/CCS Buffers for Render Target(s)

Lossless Color Compression on Render target can be enabled for the purposes described below:

1. MMIO bit Cache Mode 1 (0x7004) register bit 5
2. MMIO bit Cache Mode 1 (0x7004) register bit 15
3. RT surface state (Auxiliary Surface Mode[2:0])

Note: Lossless Color Compression can only be applied to Surfaces which are TileY, TileYs, or TileYf.

The following table summarizes modes of operation related to the Lossless Color Compression on Render target:

Cache Mode 1 MMIO Bit 5 (Please refer to Vol 1c)	Auxiliary Surface Mode Surface State	Cache Mode 1 MMIO Bit 15 (Please refer to Vol 1c)	Operation
1	X	X	Normal mode of operation i.e. no MSAA compression and no color clear
0	AUX_NONE (0h)	X	Normal mode of operation i.e. no MSAA compression and no color clear
0	AUX_CCS_D (1h)	X	Depending on the Number of multi-samples, either MCS or CCS is enabled. No MSAA or Color Compression is enabled.
0	AUX_CCS_E (5h)	1	Depending on the Number of multi-samples, either MCS or CCS is enabled. For Number of multi-samples > 1 MSAA Compression is enabled. For Number of multi-samples = 1 Color Compression is disabled.
0	AUX_CCS_E (5h)	0	Depending on the Number of multi-samples, either MCS or CCS is enabled. For Number of multi-samples > 1 MSAA Compression is

Cache Mode 1 MMIO Bit 5 (Please refer to Vol 1c)	Auxiliary Surface Mode Surface State	Cache Mode 1 MMIO Bit 15 (Please refer to Vol 1c)	Operation
			enabled. For Number of multi-samples = 1 Color Compression is enabled.
0	AUX_MCS_LCE	0	For Number of multi-samples > 1, MSAA compression and Color Compression, both are enabled. For, this case, SW must allocate both CCS and MCS. CCS is per each plane of the multi-sample PLANAR surface.

MSAA	Width of Clear Rect	Height of Clear Rect
2x	Ceil(1/8*width)	Ceil(1/2*height)
4X	Ceil(1/8*width)	Ceil(1/2*height)
8X	Ceil(1/2*width)	Ceil(1/2*height)
16X	width	Ceil(1/2*height)

- **MSAA Compression:** Multi-sample render target is bound to the pipeline and MSAA compression feature is enabled. In this case, MCS buffer stores the information required for MSAA compression algorithm. The size and layout of the MCS buffer is based on per-pixel RT. For 4X and 8X MSAA, MCS buffer element is 8bpp and 32bpp respectively. Height, width, and layout of MCS buffer in this case must match with Render Target height, width, and layout. MCS buffer is tiledY. When MCS buffer is enabled and bound to MSRT, it is required that it is cleared prior to any rendering. A clear value can be specified optionally in the surface state of the corresponding RT. Clear pass for this case requires that scaled down primitive is sent down with upper left coordinate to coincide with actual rectangle being cleared. For MSAA, clear rectangle's height and width need to as show in the following table in terms of (width,height) of the RT. MCS Buffer only Supports TileY mode.
- **Fast Color Clear:** This feature has a dedicated page. When non multi-sample render target is bound to the pipeline and MCS buffer is enabled, MCS buffer is used as an intermediate (coarse granular) buffer per RT. Hence, MCS buffer is used to improve render target clear. When MCS is buffer is used for color clear of non-multisampler render target, the following restrictions apply:

Color Clear of Non-MultiSampler Render Target Restrictions

Restrictions
Support is limited to tiled render targets.
They are supported with MCS buffer layout with these alignments in the RT space: Horizontal Alignment = 128 and Vertical Alignment = 64.
MCS and Lossless compression is supported for TiledY/TileYs/TileYf non-MSRTs only.

Restrictions
Clear is supported only on the full RT; i.e., no partial clear or overlapping clears.
In-place full resolve of a 3D/Volume surface is not supported. In order to fully resolve 3D/volume surface, copy operation must be performed to a new destination (declared as uncompressed) using the compressed 3D surface as a source.
Fast clear to 0 is not supported for MSAA > 1x
The following table describes the RT alignment:

TiledY RT CL	Pixels	Lines
bpp		
8	32	4
16	16	4
32	8	4
64	4	4
128	2	4
TiledX RT CL		
bpp		
32	16	2
64	8	2
128	4	2

CCS buffer (non-MSRT) is supported only for RT formats 8bpp, 16bpp, 32bpp, 64bpp, and 128bpp.
Clear pass must have a clear rectangle that must follow alignment rules in terms of pixels and lines as shown in the table below. Further, the clear-rectangle height and width must be multiple of the following dimensions. If the height and width of the render target being cleared do not meet these requirements, an MCS buffer can be created such that it follows the requirement and covers the RT.

Programming Note	
Context:	CCS surface initialization
CCS surface does not require initialization. Illegal CCS vales are treated as uncompressed memory. Surface with compression enabled may be written to without prior fast clear call.	

RT Pixel Blocks Per CCS CL: This is supposed to be used for sizing the CCS Surface.

Tile-YF RT	Pixels	Lines
bpp		
8	256	64
16	256	32
32	128	32
64	128	16
128	64	16

Tile-YF RT	Pixels	Lines
Tile-YS RT	Pixels	Lines
bpp		
8	128	128
16	128	64
32	64	64
64	64	32
128	32	32

Tile-Y RT	Pixels	Lines
bpp		
8	512	32
16	256	32
32	128	32
64	64	32
128	32	32

To optimize the performance CCS buffer (when bound to 1X RT) clear similarly to CCS buffer clear for MSRT case, clear rect is required to be scaled by the following factors in the horizontal and vertical directions:

CCS CL for Tiled-YF RCC	Horizontal Scale Down Factor	Vertical Scale Down Factor
bpp		
8	128	32
16	128	16
32	64	16
64	64	8
128	32	8

CCS CL for Tiled-YS RCC	Horizontal Scale Down Factor	Vertical Scale Down Factor
bpp		
8	64	64
16	64	32
32	32	32
64	32	16
128	16	16

CCS CL for Tiled-Y RCC	Horizontal Scale Down Factor	Vertical Scale Down Factor

CCS CL for Tiled-Y RCC	Horizontal Scale Down Factor	Vertical Scale Down Factor
bpp		
8	256	16
16	128	16
32	64	16
64	32	16
128	16	16

The following SW requirements for MCS buffer clear functionality apply in addition to the general SW requirements listed below:

- For non-MSRTs, loss less compression of render targets is supported for 32, 64, and 128 bpp surfaces as described in the Shared Functions Data Port under Render Target Write. Lossless RT compression can be enabled for each RT surface with auxiliary surface control bits as described in the Surface State. This feature changes the MSC layout for various tiled and BPP formats as described in above sections. Fast clear, render, and resolve operations are fundamentally the same. Since Sampler support reading this auxiliary (MCS) buffer for non-MSRTs, resolve passes can be avoided in the cases when fast cleared and possibly compressed RTs are consumed by the sampler.
- SW does not need to compile any PS for clear and resolve passes but must ensure that PS dispatch enable bit is set.

Any transition from any value in {Clear, Render, Resolve} to a different value in {Clear, Render, Resolve} requires end of pipe synchronization.

The following are the general SW requirements for MCS buffer clear functionality:

- At the time of Render Target creation, SW needs to create clear-buffer, i.e., MCS buffer.
- At the clear time, clear value for that RT must be programmed and clear enable bit must be set in the surface state of the corresponding RT.
- SW must clear the RT with setting a RT clear bit set in the PS state during the clear pass as described in the following sub-section.
- Since only one RT is bound with a clear pass, only one RT can be cleared at a time. To clear multiple RTs, multiple clear passes are required.
- If Software wants to enable Color Compression without Fast clear, Software needs to initialize MCS with zeros.
- Lossless compression and CCS initialized to all F (using HW Fast Clear or SW direct Clear) on the same surface is not supported.
- Before binding the "cleared" RT to texture OR honoring a CPU lock OR submitting for flip, SW must ensure a resolve pass. Such a resolve pass is described in the following sub-section.

Programming Note
CCS is a linear buffer created for storing meta-data (AUX data) for lossless compression. This buffer related information is mentioned in Render Surface State. CCS buffer's size is based on the padded main surface (after

Programming Note
following Halign and Valign requirements mentioned in the Render Surface State). CCS_Buffer_Size = Padded_Main_Surface_Size/256

Accessing Render Target

The final results of pixel shaders are written to either UAV surfaces or to Render target surfaces. Render targets support a large set of surface formats (refer to *Render Target Surfaces* for details) with hardware conversion from the format delivered by the thread. The render target message processing in HW includes format conversion, alpha-test, alpha-to-coverage, depth/stencil tests, blending and logop depending on various states.

The render target read/write messages are specifically for the use of pixel shader threads spawned by the Pixel Shader Dispatch(PSD), and may not be used by any other thread types.

Message Common Control Fields

The following data tables describe common control fields used in the Render Target message descriptors.

MDC_RT_SGS - Slot Group Select Render Cache Message Descriptor Control Field

Message Headers

The render target messages use a two-register message header.

If the header is not present, the behavior is as if the message was sent with most fields set to the same value that was delivered in R0 and R1 on the pixel shader thread dispatch. The following fields, which are not delivered in the pixel shader dispatch, behave as if they are set to zero:

Render Target Index

Source0 Alpha Present to Render Target

Additionally, **Render Target Read** messages must include a header.

MH_RT - Render Target Message Header

Render Target Specific Extended Message Descriptor

All Render Target Read and Write messages use same format of Extended Message Descriptor:

Extended Message Descriptor Render Target [SKL+]

Render Target Message Specific Descriptors

This section contains various registers for message specific descriptors.

MT_DP_RT - Render Data Port Message Types

MSD_RTW_SIMD16 - SIMD16 Render Target Write MSD

MSD_RTW_REP16 - REP16 Render Target Write MSD

MSD_RTW_HI8DS - HI8DS Render Target Write MSD

MSD_RTW_LO8DS - LO8DS Render Target Write MSD

MSD_RTW_SIMD8 - SIMD8 Render Target Write MSD

MSD_RTR_SIMD16 - SIMD16 Render Target Read MSD

MSD_RTR_SIMD8 - SIMD8 Render Target Read MSD

MSD_RTWH_SIMD16 - Half Precision SIMD16 Render Target Write MSD

MSD_RTWH_REP16 - Half Precision REP16 Render Target Write MSD

MSD_RTWH_HI8DS - Half Precision HI8DS Render Target Write MSD

MSD_RTWH_LO8DS - Half Precision LO8DS Render Target Write MSD

MSD_RTWH_SIMD8 - Half Precision SIMD8 Render Target Write MSD

Render Target Surfaces

This section contains information on render target surface types and surface formats.

Render Target Surface Types

All surface types, except SURFTYPE_SCRATCH, are allowed.

For SURFTYPE_BUFFER and SURFTYPE_1D surfaces, only the X coordinate is used to index into the surface. The Y coordinate must be zero.

For SURFTYPE_1D, 2D, 3D, and CUBE surfaces, a Render Target Array Index is included in the input message to provide an additional coordinate. The Render Target Array Index must be zero for SURFTYPE_BUFFER.

The surface format is restricted to the set supported as render target. If source/dest color blend is enabled on Render Target Write message, the surface format is further restricted to the set supported as alpha blend render target.

Cannot be used on a surface in field mode (Vertical Line Stride = 1).

Render Target Write Surface Formats

The following table indicates the surface formats supported by the Render Target Write message. All Render Target formats supported for Render Target Writes are also supported for Render Target Reads, except YCRCB formats.

The table lists the Surface Formats supported for reading and/or writing of Render Targets. For blending operations, the named Projects support color blending on the listed Surface Formats.

Surface Formats for Render Target Messages

Surface Format Name	Color Blend Support	Message Type Support	Lossless Compression Support
R32G32B32A32_FLOAT	All	Read/Write	All
R32G32B32A32_SINT	*Not Supported*	Read/Write	All
R32G32B32A32_UINT	*Not Supported*	Read/Write	All

Surface Format Name	Color Blend Support	Message Type Support	Lossless Compression Support
R32G32B32X32_FLOAT		Read/Write	
R16G16B16A16_UNORM	All	Read/Write	All
R16G16B16A16_SNORM	All	Read/Write	All
R16G16B16A16_SINT	*Not Supported*	Read/Write	All
R16G16B16A16_UINT	*Not Supported*	Read/Write	All
R16G16B16A16_FLOAT	All	Read/Write	All
R32G32_FLOAT	All	Read/Write	All
R32G32_SINT	*Not Supported*	Read/Write	All
R32G32_UINT	*Not Supported*	Read/Write	All
R16G16B16X16_FLOAT	All	Read/Write	All
B8G8R8A8_UNORM	All	Read/Write	All
B8G8R8A8_UNORM_SRGB	All	Read/Write	
R10G10B10A2_UNORM	All	Read/Write	
R10G10B10A2_UINT	*Not Supported*	Read/Write	
R8G8B8A8_UNORM	All	Read/Write	All
R8G8B8A8_UNORM_SRGB	All	Read/Write	
R8G8B8A8_SNORM	All	Read/Write	All
R8G8B8A8_SINT	*Not Supported*	Read/Write	All
R8G8B8A8_UINT	*Not Supported*	Read/Write	All
R16G16_UNORM	All	Read/Write	All
R16G16_SNORM	All	Read/Write	All
R16G16_SINT	*Not Supported*	Read/Write	All
R16G16_UINT	*Not Supported*	Read/Write	All
R16G16_FLOAT	All	Read/Write	All
B10G10R10A2_UNORM	All	Read/Write	
B10G10R10A2_UNORM_SRGB	All	Read/Write	
R10G10B10_FLOAT_A2_UNORM		Read/Write	
R11G11B10_FLOAT	All	Read/Write	
R32_SINT	*Not Supported*	Read/Write	All
R32_UINT	*Not Supported*	Read/Write	All
R32_FLOAT	All	Read/Write	All
B8G8R8X8_UNORM	All	Read/Write	All
B8G8R8X8_UNORM_SRGB	All	Read/Write	
B5G6R5_UNORM	All	Read/Write	
B5G6R5_UNORM_SRGB	All	Read/Write	
B5G5R5A1_UNORM	All	Read/Write	
B5G5R5A1_UNORM_SRGB	All	Read/Write	

Surface Format Name	Color Blend Support	Message Type Support	Lossless Compression Support
B4G4R4A4_UNORM	All	Read/Write	
B4G4R4A4_UNORM_SRGB	All	Read/Write	
R8G8_UNORM	All	Read/Write	
R8G8_SNORM	All	Read/Write	
R8G8_SINT	*Not Supported*	Read/Write	
R8G8_UINT	*Not Supported*	Read/Write	
R16_UNORM	All	Read/Write	
R16_SNORM	All	Read/Write	
R16_SINT	*Not Supported*	Read/Write	
R16_UINT	*Not Supported*	Read/Write	
R16_FLOAT	All	Read/Write	
B5G5R5X1_UNORM	All	Read/Write	
B5G5R5X1_UNORM_SRGB	All	Read/Write	
A1B5G5R5_UNORM	All	Read/Write	
A4B4G4R4_UNORM	All	Read/Write	
R8_UNORM	All	Read/Write	
R8_SNORM	All	Read/Write	
R8_SINT	*Not Supported*	Read/Write	
R8_UINT	*Not Supported*	Read/Write	
A8_UNORM	All	Read/Write	
YCRCB_NORMAL	*Not Supported*	Write	*Not Supported*
YCRCB_SWAPUVY	*Not Supported*	Write	*Not Supported*
YCRCB_SWAPUV	*Not Supported*	Write	*Not Supported*
YCRCB_SWAPY	*Not Supported*	Write	*Not Supported*

Subspan/Pixel to Slot Mapping

Pixels are numbered as follows within a subspan:

0 = upper left

1 = upper right

2 = lower left

3 = lower right

Programming Note	
Context:	Render Target Surfaces
• When SIMD32 or SIMD16 PS threads send render target writes with multiple SIMD8 and SIMD16 messages, the following must hold: • All the slots (as described above) must have a corresponding render target write irrespective of the slot's validity. A slot is considered valid when at least one sample is enabled. For example, a SIMD16 PS thread must send two SIMD8 render target writes to cover all the slots.	

Render Target Messages

The message operations on Render Cache Data Port use the same interface as the other data port messages. The message's Descriptor is sent to the target data port on a message sideband bus, and the other parts are sequentially transmitted between the data port and EU registers across the OBUS. The total number of registers sent and received by the data port is provided to the data port on the message sideband bus.

Descriptor	Describes the Message Type, the Message Specific Controls, and whether a Message Header is present. The Message Descriptor and the destination Data Port (SFID) are encoded as part of the *send* instruction.
Header	When present, provides additional Message specific controls. The message header is optional for some Render Target messages. When present, it is a 2-register payload.
Source Payload	Provides source data for write operations.
Writeback Payload	Returns result data for read operations.

The next sections describe all of the supported formats for the Source and Writeback Data Payloads, the Message Descriptors, and their Message Headers.

Message Summary tables describe the parameters and characteristics of the Render Target data port messages, following the notational conventions used in Message Formats.

Message-Specific Descriptors

The Render Cache data port has only 2 message types: Render Target Write and Render Target Read. The Render Target Write message type has 5 subtype operations. The Render Target Read message type has 2 subtype operations. The specific message descriptors for these messages are below.

Render Target Write Message

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
DW	DW	W	BTS	1D, 2D, 3D, CUBE, BUFFER	8, 16	1	(Surface[U, V, R, LOD])->SurfaceFormat[Slot]	Surface	RTPSM

This message writes 8 or 16 pixels to a render target. Depending on parameters contained in the message and state, it may also perform a depth and stencil buffer write and/or a render target read for a color blend operation. Additional operations enabled in the Color Calculator state are also initiated as a result of issuing this message (depth test, alpha test, logic ops, etc.). This message is intended only for use by pixel shader kernels for writing results to render targets.

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_LO8DS	{Opt} MH_RT	{Forbidden}	MDP_RTW_8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_LO8DS	MH_RT_ZM	{Forbidden}	MDP_RTW_ZM8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_LO8DS	MH_RT_Z	{Forbidden}	MDP_RTW_Z8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_LO8DS	MH_RT_M	{Forbidden}	MDP_RTW_M8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_HI8DS	{Opt} MH_RT	{Forbidden}	MDP_RTW_8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_HI8DS	MH_RT_ZM	{Forbidden}	MDP_RTW_ZM8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_HI8DS	MH_RT_Z	{Forbidden}	MDP_RTW_Z8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_HI8DS	MH_RT_M	{Forbidden}	MDP_RTW_M8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_SIMD8	{Opt} MH_RT	{Forbidden}	MDP_RTW_8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_SIMD8	MH_RT_ZMA	{Forbidden}	MDP_RTW_ZMA8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_SIMD8	MH_RT_ZA	{Forbidden}	MDP_RTW_ZA8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_SIMD8	MH_RT_MA	{Forbidden}	MDP_RTW_MA8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_SIMD8	MH_RT_ZM	{Forbidden}	MDP_RTW_ZM8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_SIMD8	MH_RT_Z	{Forbidden}	MDP_RTW_Z8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_SIMD8	MH_RT_A	{Forbidden}	MDP_RTW_A8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_SIMD8	MH_RT_M	{Forbidden}	MDP_RTW_M8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_LO8DS	{Opt} MH_RT	{Forbidden}	MDP_RTW_S8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_LO8DS	MH_RT_ZM	{Forbidden}	MDP_RTW_SZM8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_LO8DS	MH_RT_Z	{Forbidden}	MDP_RTW_SZ8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_LO8DS	MH_RT_M	{Forbidden}	MDP_RTW_SM8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_HI8DS	{Opt} MH_RT	{Forbidden}	MDP_RTW_S8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_HI8DS	MH_RT_ZM	{Forbidden}	MDP_RTW_SZM8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_HI8DS	MH_RT_Z	{Forbidden}	MDP_RTW_SZ8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_HI8DS	MH_RT_M	{Forbidden}	MDP_RTW_SM8DS	{Forbidden}
W	BTS	1D, 2D, 3D,	SIMD8	1	MSD_RTW_SIMD8	{Opt} MH_RT	{Forbidden}	MDP_RTW_S8	{Forbidden}

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
		CUBE, BUFFER							
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_SIMD8	MH_RT_ZMA	{Forbidden}	MDP_RTW_SZMA8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_SIMD8	MH_RT_ZA	{Forbidden}	MDP_RTW_SZA8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_SIMD8	MH_RT_MA	{Forbidden}	MDP_RTW_SMA8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_SIMD8	MH_RT_ZM	{Forbidden}	MDP_RTW_SZM8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_SIMD8	MH_RT_Z	{Forbidden}	MDP_RTW_SZ8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_SIMD8	MH_RT_A	{Forbidden}	MDP_RTW_SA8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTW_SIMD8	MH_RT_M	{Forbidden}	MDP_RTW_SM8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD16	1	MSD_RTW_SIMD16	{Opt} MH_RT	{Forbidden}	MDP_RTW_16	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD16	1	MSD_RTW_SIMD16	MH_RT_ZMA	{Forbidden}	MDP_RTW_ZMA16	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD16	1	MSD_RTW_SIMD16	MH_RT_ZA	{Forbidden}	MDP_RTW_ZA16	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD16	1	MSD_RTW_SIMD16	MH_RT_MA	{Forbidden}	MDP_RTW_MA16	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD16	1	MSD_RTW_SIMD16	MH_RT_ZM	{Forbidden}	MDP_RTW_ZM16	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD16	1	MSD_RTW_SIMD16	MH_RT_Z	{Forbidden}	MDP_RTW_Z16	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD16	1	MSD_RTW_SIMD16	MH_RT_A	{Forbidden}	MDP_RTW_A16	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD16	1	MSD_RTW_SIMD16	MH_RT_M	{Forbidden}	MDP_RTW_M16	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD16	1	MSD_RTW_REP16	{Opt} MH_RT	{Forbidden}	MDP_RTW_16REP	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD16	1	MSD_RTW_REP16	MH_RT_M	{Forbidden}	MDP_RTW_M16REP	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_LO8DS	{Opt} MH_RT	{Forbidden}	MDP_RTWH_8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_LO8DS	MH_RT_ZM	{Forbidden}	MDP_RTWH_ZM8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_LO8DS	MH_RT_Z	{Forbidden}	MDP_RTWH_Z8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_LO8DS	MH_RT_M	{Forbidden}	MDP_RTWH_M8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_HI8DS	{Opt} MH_RT	{Forbidden}	MDP_RTWH_8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_HI8DS	MH_RT_ZM	{Forbidden}	MDP_RTWH_ZM8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_HI8DS	MH_RT_Z	{Forbidden}	MDP_RTWH_Z8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_HI8DS	MH_RT_M	{Forbidden}	MDP_RTWH_M8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_SIMD8	{Opt} MH_RT	{Forbidden}	MDP_RTWH_8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_SIMD8	MH_RT_ZMA	{Forbidden}	MDP_RTWH_ZMA8	{Forbidden}

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_SIMD8	MH_RT_ZA	{Forbidden}	MDP_RTWH_ZA8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_SIMD8	MH_RT_MA	{Forbidden}	MDP_RTWH_MA8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_SIMD8	MH_RT_ZM	{Forbidden}	MDP_RTWH_ZM8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_SIMD8	MH_RT_Z	{Forbidden}	MDP_RTWH_Z8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_SIMD8	MH_RT_A	{Forbidden}	MDP_RTWH_A8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_SIMD8	MH_RT_M	{Forbidden}	MDP_RTWH_M8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_LO8DS	{Opt} MH_RT	{Forbidden}	MDP_RTWH_S8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_LO8DS	MH_RT_ZM	{Forbidden}	MDP_RTWH_SZM8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_LO8DS	MH_RT_Z	{Forbidden}	MDP_RTWH_SZ8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_LO8DS	MH_RT_M	{Forbidden}	MDP_RTWH_SM8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_HI8DS	{Opt} MH_RT	{Forbidden}	MDP_RTWH_S8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_HI8DS	MH_RT_ZM	{Forbidden}	MDP_RTWH_SZM8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_HI8DS	MH_RT_Z	{Forbidden}	MDP_RTWH_SZ8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_HI8DS	MH_RT_M	{Forbidden}	MDP_RTWH_SM8DS	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_SIMD8	{Opt} MH_RT	{Forbidden}	MDP_RTWH_S8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_SIMD8	MH_RT_ZMA	{Forbidden}	MDP_RTWH_SZMA8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_SIMD8	MH_RT_ZA	{Forbidden}	MDP_RTWH_SZA8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_SIMD8	MH_RT_MA	{Forbidden}	MDP_RTWH_SMA8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_SIMD8	MH_RT_ZM	{Forbidden}	MDP_RTWH_SZM8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_SIMD8	MH_RT_Z	{Forbidden}	MDP_RTWH_SZ8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_SIMD8	MH_RT_A	{Forbidden}	MDP_RTWH_SA8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD8	1	MSD_RTWH_SIMD8	MH_RT_M	{Forbidden}	MDP_RTWH_SM8	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD16	1	MSD_RTWH_SIMD16	{Opt} MH_RT	{Forbidden}	MDP_RTWH_16	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD16	1	MSD_RTWH_SIMD16	MH_RT_ZMA	{Forbidden}	MDP_RTWH_ZMA16	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD16	1	MSD_RTWH_SIMD16	MH_RT_ZA	{Forbidden}	MDP_RTWH_ZA16	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD16	1	MSD_RTWH_SIMD16	MH_RT_MA	{Forbidden}	MDP_RTWH_MA16	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD16	1	MSD_RTWH_SIMD16	MH_RT_ZM	{Forbidden}	MDP_RTWH_ZM16	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD16	1	MSD_RTWH_SIMD16	MH_RT_Z	{Forbidden}	MDP_RTWH_Z16	{Forbidden}

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD16	1	MSD_RTWH_SIMD16	MH_RT_A	{Forbidden}	MDP_RTWH_A16	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD16	1	MSD_RTWH_SIMD16	MH_RT_M	{Forbidden}	MDP_RTWH_M16	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD16	1	MSD_RTWH_REP16	{Opt} MH_RT	{Forbidden}	MDP_RTWH_16REP	{Forbidden}
W	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD16	1	MSD_RTWH_REP16	MH_RT_M	{Forbidden}	MDP_RTWH_M16REP	{Forbidden}

The sample for a pixel is killed (not written to the render target or depth buffer) if the corresponding oMask bit is zero. Bits in oMask larger than the number of multisamples are ignored.

The color payload is included if the Message Type is SIMD8 single source or SIMD8 Image Write.

The SIMD16 Replicated Data Color payload is included if the Message Type specifies single source message with replicated data. One set of R/G/B/A data is included in the message, and this data is replicated to all 16 pixels. The Replicated SIMD16 message is legal with color data and oMask, but the registers for depth, stencil, and antialias alpha data cannot be included with this message, and the corresponding bits in the message header must indicate that these registers are not present.

The following table enumerates the order that the data payload is packed in registers: Source 0 Alpha (s0A), Output Mask (oM), Pixel components (R, G, B, A), Source Depth (sZ), and Output Stencil (oS).

Summary of Data Payload Register Order for Render Target Write Messages

Message Type	Present?				Message Register												
	S0A	oM	sZ	oS	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14
SIMD16	0	0	0	0	1/0R	3/2R	1/0G	3/2G	1/0B	3/2B	1/0A	3/2A					
SIMD16	1	0	0	0	1/0s0A	3/2s0A	1/0R	3/2R	1/0G	3/2G	1/0B	3/2B	1/0A	3/2A			
SIMD16	0	0	1	0	1/0R	3/2R	1/0G	3/2G	1/0B	3/2B	1/0A	3/2A	1/0sZ	3/2sZ			
SIMD16	1	0	1	0	1/0s0A	3/2s0A	1/0R	3/2R	1/0G	3/2G	1/0B	3/2B	1/0A	3/2A	1/0sZ	3/2sZ	
SIMD16	0	1	0	0	oM	1/0R	3/2R	1/0G	3/2G	1/0B	3/2B	1/0A	3/2A				
SIMD16	1	1	0	0	1/0soA	3/2s0A	oM	1/0R	3/2R	1/0G	3/2G	1/0B	3/2B	1/0A	3/2A		
SIMD16	0	1	1	0	oM	1/0R	3/2R	1/0G	3/2G	1/0B	3/2B	1/0A	3/2A	1/0sZ	3/2sZ		
SIMD16	1	1	1	0	1/0s0A	3/2s0A	oM	1/0R	3/2R	1/0G	3/2G	1/0B	3/2B	1/0A	3/2A	1/0sZ	3/2sZ
REP16	0	0	0	0	RGBA												
LO8DS	0	0	0	0	1/0s0R	1/0s0G	1/0s0B	1/0s0A	1/0s1R	1/0s1G	1/0s1B	1/0s1A					
LO8DS	0	0	1	0	1/0s0R	1/0s0G	1/0s0B	1/0s0A	1/0s1R	1/0s1G	1/0s1B	1/0s1A	1/0sZ				
LO8DS	0	1	0	0	oM	1/0s0R	1/0s0G	1/0s0B	1/0s0A	1/0s1R	1/0s1G	1/0s1B	1/0s1A				
LO8DS	0	1	1	0	oM	1/0s0R	1/0s0G	1/0s0B	1/0s0A	1/0s1R	1/0s1G	1/0s1B	1/0s1A	1/0sZ			
HI8DS	0	0	0	0	3/2s0R	3/2s0G	3/2s0B	3/2s0A	3/2s1R	3/2s1G	3/2s1B	3/2s1A					
HI8DS	0	0	1	0	3/2s0R	3/2s0G	3/2s0B	3/2s0A	3/2s1R	3/2s1G	3/2s1B	3/2s1A	3/2sZ				
HI8DS	0	1	0	0	oM	3/2s0R	3/2s0G	3/2s0B	3/2s0A	3/2s1R	3/2s1G	3/2s1B	3/2s1A				
HI8DS	0	1	1	0	oM	3/2s0R	3/2s0G	3/2s0B	3/2s0A	3/2s1R	3/2s1G	3/2s1B	3/2s1A	3/2sZ			
SIMD8	0	0	0	0	R	G	B	A									
SIMD8	1	0	0	0	s0A	R	G	B	A								
SIMD8	0	0	1	0	R	G	B	A	sZ								
SIMD8	1	0	1	0	s0A	R	G	B	A	sZ							
SIMD8	0	1	0	0	oM	R	G	B	A								
SIMD8	1	1	0	0	s0A	oM	R	G	B	A							

SIMD8	0	1	1	0	oM	R	G	B	A	sZ							
SIMD8	1	1	1	0	s0A	oM	R	G	B	A	sZ						
LO8DS	0	0	0	1	1/0s0R	1/0s0G	1/0s0B	1/0s0A	1/0s1R	1/0s1G	1/0s1B	1/0s1A	1/0oS				
LO8DS	0	0	1	1	1/0s0R	1/0s0G	1/0s0B	1/0s0A	1/0s1R	1/0s1G	1/0s1B	1/0s1A	1/0sZ	1/0OoS			
LO8DS	0	1	0	1	oM	1/0s0R	1/0s0G	1/0s0B	1/0s0A	1/0s1R	1/0s1G	1/0s1B	1/0s1A	1/0oS			
LO8DS	0	1	1	1	oM	1/0s0R	1/0s0G	1/0s0B	1/0s0A	1/0s1R	1/0s1G	1/0s1B	1/0s1A	1/0sZ	1/0oS		
HI8DS	0	0	0	1	3/2s0R	3/2s0G	3/2s0B	3/2s0A	3/2s1R	3/2s1G	3/2s1B	3/2s1A	3/2oS				
HI8DS	0	0	1	1	3/2s0R	3/2s0G	3/2s0B	3/2s0A	3/2s1R	3/2s1G	3/2s1B	3/2s1A	3/2sZ	3/2oS			
HI8DS	0	1	0	1	oM	3/2s0R	3/2s0G	3/2s0B	3/2s0A	3/2s1R	3/2s1G	3/2s1B	3/2s1A	3/2oS			
HI8DS	0	1	1	1	oM	3/2s0R	3/2s0G	3/2s0B	3/2s0A	3/2s1R	3/2s1G	3/2s1B	3/2s1A	3/2sZ	3/2oS		
SIMD8	0	0	0	1	R	G	B	A	oS								
SIMD8	1	0	0	1	s0A	R	G	B	A	oS							
SIMD8	0	0	1	1	R	G	B	A	sZ	oS							
SIMD8	1	0	1	1	s0A	R	G	B	A	sZ	oS						
SIMD8	0	1	0	1	oM	R	G	B	A	oS							
SIMD8	1	1	0	1	s0A	oM	R	G	B	A	oS						
SIMD8	0	1	1	1	oM	R	G	B	A	sZ	oS						
SIMD8	1	1	1	1	s0A	oM	R	G	B	A	sZ	oS					
HP SIMD16	0	0	0	0	R	G	B	A									
HP SIMD16	1	0	0	0	s0A	R	G	B	A								
HP SIMD16	0	0	1	0	R	G	B	A	sZ								
HP SIMD16	1	0	1	0	s0A	R	G	B	A	sZ							
HP SIMD16	0	1	0	0	oM	R	G	B	A								
HP SIMD16	1	1	0	0	s0A	oM	R	G	B	A							
HP SIMD16	0	1	1	0	oM	R	G	B	A	sZ							
HP SIMD16	1	1	1	0	s0A	oM	R	G	B	A	sZ						
HP REP16	0	0	0	0	RGBA												
HP LO8DS	0	0	0	0	1/0s0R	1/0s0G	1/0s0B	1/0s0A	1/0s1R	1/0s1G	1/0s1B	1/0s1A					
HP LO8DS	0	0	1	0	1/0s0R	1/0s0G	1/0s0B	1/0s0A	1/0s1R	1/0s1G	1/0s1B	1/0s1A	1/0sZ				
HP LO8DS	0	1	0	0	oM	1/0s0R	1/0s0G	1/0s0B	1/0s0A	1/0s1R	1/0s1G	1/0s1B	1/0s1A				
HP LO8DS	0	1	1	0	oM	1/0s0R	1/0s0G	1/0s0B	1/0s0A	1/0s1R	1/0s1G	1/0s1B	1/0s1A	1/0sZ			
HP HI8DS	0	0	0	0	3/2s0R	3/2s0G	3/2s0B	3/2s0A	3/2s1R	3/2s1G	3/2s1B	3/2s1A					
HP HI8DS	0	0	1	0	3/2s0R	3/2s0G	3/2s0B	3/2s0A	3/2s1R	3/2s1G	3/2s1B	3/2s1A	3/2sZ				
HP HI8DS	0	1	0	0	oM	3/2s0R	3/2s0G	3/2s0B	3/2s0A	3/2s1R	3/2s1G	3/2s1B	3/2s1A				
HP HI8DS	0	1	1	0	oM	3/2s0R	3/2s0G	3/2s0B	3/2s0A	3/2s1R	3/2s1G	3/2s1B	3/2s1A	3/2sZ			
HP SIMD8	0	0	0	0	R	G	B	A									
HP SIMD8	1	0	0	0	s0A	R	G	B	A								
HP SIMD8	0	0	1	0	R	G	B	A	sZ								
HP SIMD8	1	0	1	0	s0A	R	G	B	A	sZ							
HP SIMD8	0	1	0	0	oM	R	G	B	A								
HP SIMD8	1	1	0	0	s0A	oM	R	G	B	A							
HP SIMD8	0	1	1	0	oM	R	G	B	A	sZ							
HP SIMD8	1	1	1	0	s0A	oM	R	G	B	A	sZ						
HP LO8DS	0	0	0	1	1/0s0R	1/0s0G	1/0s0B	1/0s0A	1/0s1R	1/0s1G	1/0s1B	1/0s1A	1/0oS				
HP LO8DS	0	0	1	1	1/0s0R	1/0s0G	1/0s0B	1/0s0A	1/0s1R	1/0s1G	1/0s1B	1/0s1A	1/0sZ	1/0OoS			
HP LO8DS	0	1	0	1	oM	1/0s0R	1/0s0G	1/0s0B	1/0s0A	1/0s1R	1/0s1G	1/0s1B	1/0s1A	1/0oS			
HP LO8DS	0	1	1	1	oM	1/0s0R	1/0s0G	1/0s0B	1/0s0A	1/0s1R	1/0s1G	1/0s1B	1/0s1A	1/0sZ	1/0oS		
HP HI8DS	0	0	0	1	3/2s0R	3/2s0G	3/2s0B	3/2s0A	3/2s1R	3/2s1G	3/2s1B	3/2s1A	3/2oS				
HP HI8DS	0	0	1	1	3/2s0R	3/2s0G	3/2s0B	3/2s0A	3/2s1R	3/2s1G	3/2s1B	3/2s1A	3/2sZ	3/2oS			

HP HI8DS	0	1	0	1	oM	3/2s0R	3/2s0G	3/2s0B	3/2s0A	3/2s1R	3/2s1G	3/2s1B	3/2s1A	3/2oS			
HP HI8DS	0	1	1	1	oM	3/2s0R	3/2s0G	3/2s0B	3/2s0A	3/2s1R	3/2s1G	3/2s1B	3/2s1A	3/2sZ	3/2oS		
HP SIMD8	0	0	0	1	R	G	B	A	oS								
HP SIMD8	1	0	0	1	s0A	R	G	B	A	oS							
HP SIMD8	0	0	1	1	R	G	B	A	sZ	oS							
HP SIMD8	1	0	1	1	s0A	R	G	B	A	sZ	oS						
HP SIMD8	0	1	0	1	oM	R	G	B	A	oS							
HP SIMD8	1	1	0	1	s0A	oM	R	G	B	A	oS						
HP SIMD8	0	1	1	1	oM	R	G	B	A	sZ	oS						
HP SIMD8	1	1	1	1	s0A	oM	R	G	B	A	sZ	oS					

Programming Note	
Context:	Render Target Write Message
Typically, the last message in a pixel shader is a Render Target Write message, with EOT set.	

Programming Note	
Context:	Render Target Write Message

- The dual source message cannot be used if the Render Target Rotation field in SURFACE_STATE is set to anything other than RTROTATE_0DEG.
- If multiple SIMD8 Dual Source messages are delivered by the pixel shader thread, each SIMD8_DUALSRC_LO message must be issued *before* the SIMD8_DUALSRC_HI message with the same Slot Group Select setting.
- Output Stencil is not supported with SIMD16 Render Target Write Messages.

Programming Note	
Context:	Render Target Writes in Multirate Shaders

- In multirate shaders, render target writes that can potentially kill pixels (via oMask, oDepth, or oStencil) cannot modify the oMask, oDepth, or oStencil value after writing to a render target. For instance if a pixel-rate render target is written, an associated sample-rate render target cannot modify these values or undefined behavior can occur.
- In multirate shaders that write to render targets in multiple stages, render target 0 must be written to before any other render targets.
- Pixel shaders dispatched at the coarse rate that contain a SIMD16 pixel-rate shader, cannot use SIMD8 messages, preventing a SIMD16 pixel-rate (or sample-rate) shader from using Dual Source messages.

Replicate Data

The replicate data render target message is used for clearing Render Target. This message performs better than the other messages due to its smaller message length. This message does not support depth, stencil, or antialias alpha data being sent with it.

The pixel scoreboard bits corresponding to the dispatched pixel mask are cleared only if the Last Render Target Select bit is set in the message descriptor.

Programming Note	
Context:	Replicate Data Render Target Write Message
Replicate Data Render Target Write message is not supported with AA alpha i.e. when primitive has Anti-Aliased alpha enabled e.g. AA lines and points.	

Single Source

The "normal" render target messages are single source. There are two forms, SIMD16 and SIMD8, intended for the equivalent-sized pixel shader threads. A single source (4 channels) is delivered for each of the 16 or 8 pixels in the message payload. Optional depth, stencil, and antialias alpha information can also be delivered with these messages.

The pixel scoreboard bits corresponding to the dispatched pixel mask (or half of the mask in the case of SIMD8 messages) are cleared only if the Last Render Target Select bit is set in the message descriptor. However, if Last Render Target Select is set, the message still causes pixel scoreboard clear and depth/stencil buffer updates if enabled.

Dual Source

The dual source render target messages only have SIMD8 forms due to maximum message length limitations. SIMD16 pixel shaders must send two of these messages to cover all of the pixels. Each message contains two colors (4 channels each) for each pixel in the message payload. In addition to the first source, the second source can be selected as a blend factor (BLENDFACTOR_*_SRC1_* options in the blend factor fields of COLOR_CALC_STATE or BLEND_STATE). Optional depth, stencil, and antialias alpha information can also be delivered with these messages.

For SIMD16 PS thread with two output colors with SRC1 as one of the blend factors with blending enabled, SW must send messages in the following sequence for each RT: SIMD8 dual source (SRC0 and SRC1) RTW message (low); SIMD8 dual source RTW message (high); SIMD16 single src (SRC1) RTW message with second color. If blending is not enabled OR if blending is enabled without SRC1 as a blend factor, SW continues to send regular SIMD16 SRC0 and SRC1 messages.

Render Target Read Message

Addr Align	Data Width	R/W	Address Model	Surface Type	SIMD Slots	Data Elements	SIMD Address Calculation	Bounds Check	Execution Mask
DW	DW	R	BTS	1D, 2D, 3D, CUBE, BUFFER	8, 16	4	(Surface[U, V, R, LOD])->DW[Chan]	Surface	RTPSM

This message takes 8 or 16 pixels for reads to a render target. This message is intended only for use by pixel shader kernels for reading data from render targets.

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
R	BTS	1D, 2D, 3D,	SIMD8	{4}	MSD_RTR_SIMD8	MH_RT	{Forbidden}	{Forbidden}	{4} MDP_DW_SIMD16

R/W	Address Model	Surface Type	SIMD Slots	Data Elements	Message Specific Descriptor	Message Header	Address Payload	Source Payload	Writeback Payload
		CUBE, BUFFER							
R	BTS	1D, 2D, 3D, CUBE, BUFFER	SIMD16	{4}	MSD_RTR_SIMD16	MH_RT	{Forbidden}	{Forbidden}	{4} MDP_DW_SIMD8

A SIMD8 writeback message consists of 4 destination registers. A SIMD16 writeback message consists of 8 destination registers.

Restrictions:

Must not have End-of-Thread bit set in message.

This message can only be issued from a kernel specified in WM_STATE or 3DSTATE_WM (pixel shader kernel), dispatched in non-contiguous mode. Any other kernel issuing this message causes undefined behavior.

Render Target read on an MSRT is supported via per-sample pixel shader.

Message Data Payloads

The data payloads for Render Target Read messages depend on the SIMD message subtype and the channel enables in the message descriptor. The writeback data payloads are 4 channels of red, green, blue, and alpha data. Each channel is one or two message registers laid out in the standard 8 or 16 DWord format, depending whether it is a SIMD8 or SIMD16 data operation.

The data payloads for Render Target Write messages depend on the Message Subtype from the message descriptor, and the settings of the oMask, Source 0 Alpha, and Source Depth presence control bits in the message header. The following data tables list all the valid source payloads combinations for the Render Target Write message.

The half precision Render Target Write messages have data payloads that can pack a full SIMD16 payload into 1 register instead of two. The half-precision packed format is used for RGBA and Source 0 Alpha, but Source Depth data payload is always supplied in full precision.

Render Target Data Payloads

This section contains various registers for the Render Target Data Payloads.

MDP_RTW_16 - SIMD16 Render Target Data Payload

MDP_RTW_ZMA16 - SZ OM S0A SIMD16 Render Target Data Payload

MDP_RTW_ZA16 - SZ S0A SIMD16 Render Target Data Payload

MDP_RTW_MA16 - OM S0A SIMD16 Render Target Data Payload

MDP_RTW_ZM16 - SZ OM SIMD16 Render Target Data Payload

MDP_RTW_Z16 - SZ SIMD16 Render Target Data Payload

MDP_RTW_A16 - S0A SIMD16 Render Target Data Payload

MDP_RTW_M16 - OM SIMD16 Render Target Data Payload

MDP_RTW_16REP - Replicated SIMD16 Render Target Data Payload

MDP_RTW_M16REP - OM Replicated SIMD16 Render Target Data Payload

MDP_RTW_8 - SIMD8 Render Target Data Payload

MDP_RTW_ZMA8 - SZ OM S0A SIMD8 Render Target Data Payload

MDP_RTW_ZM8 - SZ OM SIMD8 Render Target Data Payload

MDP_RTW_ZA8 - SZ S0A SIMD8 Render Target Data Payload

MDP_RTW_MA8 - OM S0A SIMD8 Render Target Data Payload

MDP_RTW_Z8 - SZ SIMD8 Render Target Data Payload

MDP_RTW_A8 - S0A SIMD8 Render Target Data Payload

MDP_RTW_M8 - OM SIMD8 Render Target Data Payload

MDP_RTW_S8 - OS SIMD8 Render Target Data Payload

MDP_RTW_SZMA8 - OS SZ OM S0A SIMD8 Render Target Data Payload

MDP_RTW_SZM8 - OS SZ OM SIMD8 Render Target Data Payload

MDP_RTW_SZA8 - OS SZ S0A SIMD8 Render Target Data Payload

MDP_RTW_SMA8 - OS OM S0A SIMD8 Render Target Data Payload

MDP_RTW_SZ8 - OS SZ SIMD8 Render Target Data Payload

MDP_RTW_SA8 - OS S0A SIMD8 Render Target Data Payload

MDP_RTW_SM8 - OS OM SIMD8 Render Target Data Payload

MDP_RTW_8DS - SIMD8 Dual Source Render Target Data Payload

MDP_RTW_8DS - SIMD8 Dual Source Render Target Data Payload

MDP_RTW_ZM8DS - SZ OM SIMD8 Dual Source Render Target Data Payload

MDP_RTW_Z8DS - SZ SIMD8 Dual Source Render Target Data Payload

MDP_RTW_M8DS - OM SIMD8 Dual Source Render Target Data Payload

MDP_RTW_S8DS - OS SIMD8 Dual Source Render Target Data Payload

MDP_RTW_M8DS - OM SIMD8 Dual Source Render Target Data Payload

MDP_RTW_SZM8DS - OS SZ OM SIMD8 Dual Source Render Target Data Payload

MDP_RTW_SZ8DS - OS SZ SIMD8 Dual Source Render Target Data Payload

MDP_RTWH_16 - Half Precision SIMD16 Render Target Data Payload

MDP_RTW_SZ8DS - OS SZ SIMD8 Dual Source Render Target Data Payload

MDP_RTWH_ZM16 - Half Precision SZ OM SIMD16 Render Target Data Payload

MDP_RTWH_ZMA16 - Half Precision SZ OM S0A SIMD16 Render Target Data Payload

MDP_RTWH_ZA16 - Half Precision SZ S0A SIMD16 Render Target Data Payload

MDP_RTWH_MA16 - Half Precision OM S0A SIMD16 Render Target Data Payload

MDP_RTWH_Z16 - Half Precision SZ SIMD16 Render Target Data Payload

MDP_RTWH_A16 - Half Precision S0A SIMD16 Render Target Data Payload

MDP_RTWH_M16 - Half Precision OM SIMD16 Render Target Data Payload

MDP_RTWH_16REP - Half Precision Replicated SIMD16 Render Target Data Payload

MDP_RTWH_M16REP - Half Precision OM Replicated SIMD16 Render Target Data Payload

MDP_RTWH_8 - Half Precision SIMD8 Render Target Data Payload

MDP_RTWH_ZMA8 - Half Precision SZ OM S0A SIMD8 Render Target Data Payload

MDP_RTWH_ZM8 - Half Precision SZ OM SIMD8 Render Target Data Payload

MDP_RTWH_ZA8 - Half Precision SZ S0A SIMD8 Render Target Data Payload

MDP_RTWH_MA8 - Half Precision OM S0A SIMD8 Render Target Data Payload

MDP_RTWH_Z8 - Half Precision SZ SIMD8 Render Target Data Payload

MDP_RTWH_A8 - Half Precision S0A SIMD8 Render Target Data Payload

MDP_RTWH_M8 - Half Precision OM SIMD8 Render Target Data Payload

MDP_RTWH_S8 - Half Precision OS SIMD8 Render Target Data Payload

MDP_RTWH_SZMA8 - Half Precision OS SZ OM S0A SIMD8 Render Target Data Payload

MDP_RTWH_SZM8 - Half Precision OS SZ OM SIMD8 Render Target Data Payload

MDP_RTWH_SZA8 - Half Precision OS SZ S0A SIMD8 Render Target Data Payload

MDP_RTWH_SMA8 - Half Precision OS OM S0A SIMD8 Render Target Data Payload

MDP_RTWH_SZ8 - Half Precision OS SZ SIMD8 Render Target Data Payload

MDP_RTWH_SA8 - Half Precision OS S0A SIMD8 Render Target Data Payload

MDP_RTWH_SM8 - Half Precision OS OM SIMD8 Render Target Data Payload

MDP_RTWH_8DS - Half Precision SIMD8 Dual Source Render Target Data Payload

MDP_RTWH_ZM8 - Half Precision SZ OM SIMD8 Render Target Data Payload

MDP_RTWH_M8DS - Half Precision OM SIMD8 Dual Source Render Target Data Payload

MDP_RTWH_S8DS - Half Precision OS SIMD8 Dual Source Render Target Data Payload

MDP_RTWH_SZM8DS - Half Precision OS SZ OM SIMD8 Dual Source Render Target Data Payload

MDP_RTWH_SZ8DS - Half Precision OS SZ SIMD8 Dual Source Render Target Data Payload

MDP_RTWH_SM8DS - Half Precision OS OM SIMD8 Dual Source Render Target Data Payload

Shared Functions Pixel Interpolater

The Pixel Interpolator provides barycentric parameters at various offsets relative to the pixel location. These barycentric parameters are in the same format and layout as those received in the pixel shader dispatch. Please refer to the "Windower" chapter in the "3D Pipeline" volume for more details on barycentric parameters.

Barycentric parameters delivered in the pixel shader payload are at pre-defined positions based on **Barycentric Interpolation Mode** bits selected in 3DSTATE_WM. The pixel interpolator allows barycentric parameters to be computed at additional locations.

Messages

The following is the message definition for the Pixel Interpolator shared function.

Programming Note	
Context:	Messages
Pixel Interpolator messages can only be delivered by pixel shader kernels.	

Execution Mask. Each bit in the execution mask enables the corresponding slot's barycentric parameter return to the destination registers.

Initiating Message

Message Descriptor

Bits	Description
19	**Header Present:** Specifies whether the message includes a header phase. Must be zero for all *Pixel Interpolator* messages. Format = Enable
18:17	Ignored
16	**SIMD Mode.** Specifies the SIMD mode of the message being sent. Format = U1 0: SIMD8 mode 1: SIMD16 mode
15	**Shading Rate for Attribute Evaluation** This field indicates how to interpret the message type (bits 13:12). Format: U1 0: Evaluate at pixel shading rate. 1: Evaluate at coarse pixel shading rate.
14	**Interpolation Mode.** Specifies which interpolation mode is used. Format = U1 0: Perspective Interpolation 1: Linear Interpolation **Programming Note** This field cannot be set to "Linear Interpolation" unless **Non-Perspective Barycentric Enable** in 3DSTATE_CLIP is enabled. **Programming Note** **Context:** Message Descriptor

Bits	Description
	This field is ignored when Message Type is set to Coarse To Pixel Mapping message.
13:12	**Message Type.** Specifies the type of message being sent when pixel-rate evaluation is requested. Format = U2 0: Per Message Offset (eval_snapped with immediate offset) 1: Sample Position Offset (eval_sindex) 2: Centroid Position Offset (eval_centroid) 3: Per Slot Offset (eval_snapped with register offset) **Message Type.** Specifies the type of message being sent when coarse-rate evaluation is requested. Format = U2 0: Coarse to Pixel Mapping Message (internal message) 1: Reserved 2: Coarse Centroid Position (eval_centroid) 3: Per Slot Coarse Pixel Offset (eval_snapped with register offset) **Programming Note** **Context:** Message Descriptor When Message Type is Sample Position, requesting an attribute at sample index beyond the range defined by the Forced Sample Count is illegal.
11	**Note:** For shaders dispatched at coarse rate, this bit has a different meaning and is part of the **Pixel Shader Phase** field. **Slot Group Select.** This field selects whether slots 15:0 or slots 31:16 are used for bypassed data. Bypassed data includes the X/Y addresses and centroid position. For 8- and 16-pixel dispatches, SLOTGRP_LO must be selected on every message. For 32-pixel dispatches, this field must be set correctly for each message based on which slots are currently being processed. 0: SLOTGRP_LO: Choose bypassed data for slots 15:0. 1: SLOTGRP_HI: Choose bypassed data for slots 31:16. **Programming Note** **Context:** Message Descriptor This field must be set to SLOTGRP_LO for SIMD8 messages. SIMD8 messages always use bypassed data for slots 7:0.
11:8	For shaders not dispatched at coarse rate, bit 11 is the **Slot Group Select** bit and bits 10:8 are ignored. **Pixel Shader Phase.** For shaders dispatched at coarse pixel rate, specifies the counter value of the inner PS

Bits	Description
	loop inside monolithic CPS+PS(+S) shader code. The Pixel Interpolator uses this counter value to identify affected pixels for all attributes interpolated at pixel rate and to identify pixels requested for Coarse to Pixel Mapping messages. Format: U4 Range = [0..15] Values other than 0 are valid only for shaders dispatched at coarse shading rate.
7:0	**Message Specific Control.** Refer to the sections below for the definition of these bits based on **Message Type**.

"Per Message Offset" Message Descriptor

Bit	Description
7:4	**Per Message Y Pixel Offset** Specifies the Y Pixel Offset that applies to all slots. Format = S4 2's complement representing units of 1/16 pixel. Range = [-8/16, +7/16]
3:0	**Per Message X Pixel Offset** Specifies the X Pixel Offset that applies to all slots. Format = S4 2's complement representing units of 1/16 pixel. Range = [-8/16, +7/16]

"Sample Position Offset" Message Descriptor

Bits	Description
7:4	**Sample Index** Specifies the sample index that applies to all slots. Sample Index must not exceed the value of NUM_RASTSAMPLES when NUM_RASTSAMPLES > 1. From API, perspective, Forced Sample Count Defines the maximum allowable index in this message. Format = U4 **Range** [0, 15]
3:0	Ignored

"Centroid Position" and "Per Slot Offset" Message Descriptor

Bit	Description
7:0	Ignored

"Coarse to Pixel Mapping" Message Descriptor

Bits	Description
7:0	**Coarse to Pixel Mapping Phase Enable bits** The Coarse to Pixel mapping message may include Output Pixel Coverage in the message payload. This payload is optional, and delivered only when coarse phase of pixel shader computes output coverage mask or discards coarse pixels (output pixel coverage mask is different than input pixel coverage mask). The send instruction requires payload to consist of at least one GRF register. The following bit indicates if payload includes valid Output Pixel Coverage mask, or it is a dummy register to satisfy non-zero payload requirement. bit 5: Output Pixel Coverage Mask Valid in payload The Coarse to Pixel mapping message always returns two phases (two GRF registers) but it can also return optional phases, as enabled by the following bits: bit 4: Perspective Pixel Center Location Barycentric bit 3: Perspective Pixel Centroid Location Barycentric bit 2: Linear Pixel Center Location Barycentric bit 1: Linear Pixel Centroid Location Barycentric bit 0: Pixel Input Coverage Mask

Message Payload for Most Messages

This message payload applies to the following message types:

- Per Message Offset
- Sample Position Offset
- Centroid Position Offset

DWord	Bit	Description
M0.7:0		Ignored

SIMD8 Per Slot Offset Message Payload

This message payload applies only to the SIMD8 Per Slot Offset message type. The message length is 2.

DWord	Bit	Description
M0.7	31:0	**Slot 7 X Pixel Offset** Specifies the X pixel offset for slot 7.

DWord	Bit	Description
		Format = S4 2's complement representing units of 1/16 pixel. The upper 28 bits are ignored. Range = [-8/16, +7/16]
M0.6	31:0	**Slot 6 X Pixel Offset**
M0.5	31:0	**Slot 5 X Pixel Offset**
M0.4	31:0	**Slot 4 X Pixel Offset**
M0.3	31:0	**Slot 3 X Pixel Offset**
M0.2	31:0	**Slot 2 X Pixel Offset**
M0.1	31:0	**Slot 1 X Pixel Offset**
M0.0	31:0	**Slot 0 X Pixel Offset**
M1.7	31:0	**Slot 7 Y Pixel Offset** Specifies the Y pixel offset for slot 7. See **Slot 7 X Pixel Offset** for format
M1.6	31:0	**Slot 6 Y Pixel Offset**
M1.5	31:0	**Slot 5 Y Pixel Offset**
M1.4	31:0	**Slot 4 Y Pixel Offset**
M1.3	31:0	**Slot 3 Y Pixel Offset**
M1.2	31:0	**Slot 2 Y Pixel Offset**
M1.1	31:0	**Slot 1 Y Pixel Offset**
M1.0	31:0	**Slot 0 Y Pixel Offset**

SIMD16 Per Slot Offset Message Payload

This message payload applies only to the SIMD16 Per Slot Offset message type. The message length is 4.

DWord	Bit	Description
M0.6	31:0	**Slot 6 X Pixel Offset**

DWord	Bit	Description
M0.5	31:0	**Slot 5 X Pixel Offset**
M0.4	31:0	**Slot 4 X Pixel Offset**
M0.3	31:0	**Slot 3 X Pixel Offset**
M0.2	31:0	**Slot 2 X Pixel Offset**
M0.1	31:0	**Slot 1 X Pixel Offset**
M0.0	31:0	**Slot 0 X Pixel Offset**
M1.7	31:0	**Slot 15 X Pixel Offset**
M1.6	31:0	**Slot 14 X Pixel Offset**
M1.5	31:0	**Slot 13 X Pixel Offset**
M1.4	31:0	**Slot 12 X Pixel Offset**
M1.3	31:0	**Slot 11 X Pixel Offset**
M1.2	31:0	**Slot 10 X Pixel Offset**
M1.1	31:0	**Slot 9 X Pixel Offset**
M1.0	31:0	**Slot 8 X Pixel Offset**
M2.7	31:0	**Slot 7 Y Pixel Offset** Specifies the Y pixel offset for slot 7. Format = S4 2's complement representing units of 1/16 pixel. The upper 28 bits are ignored. Range = [-8/16, +7/16]
M2.6	31:0	**Slot 6 Y Pixel Offset**
M2.5	31:0	**Slot 5 Y Pixel Offset**
M2.4	31:0	**Slot 4 Y Pixel Offset**
M2.3	31:0	**Slot 3 Y Pixel Offset**

DWord	Bit	Description
M2.2	31:0	**Slot 2 Y Pixel Offset**
M2.1	31:0	**Slot 1 Y Pixel Offset**
M2.0	31:0	**Slot 0 Y Pixel Offset**
M3.7	31:0	**Slot 15 Y Pixel Offset**
M3.6	31:0	**Slot 14 Y Pixel Offset**
M3.5	31:0	**Slot 13 Y Pixel Offset**
M3.4	31:0	**Slot 12 Y Pixel Offset**
M3.3	31:0	**Slot 11 Y Pixel Offset**
M3.2	31:0	**Slot 10 Y Pixel Offset**
M3.1	31:0	**Slot 9 Y Pixel Offset**
M3.0	31:0	**Slot 8 Y Pixel Offset**

Coarse to Pixel Phase Initiating Message (SIMD8 or SIMD16 Message)

The initiating message payload is required, as send message requires at least one GRF register. The payload register may be dummy (to satisfy non-zero payload requirement) or it may include Output Pixel Coverage Mask if coarse pixel shading phase outputs PixelCoverage or discards coarse pixels. The Output Pixel Coverage Mask Valid bit indicates if message payload includes valid data.

Programming Note: This message may be simd8 or simd16 if the thread dispatch rate is simd8. This message MUST be simd16 if the thread dispatch rate is simd16.

M0.7	31:0	Slot 7 Output Pixel Coverage Mask Format = U32
M0.6	31:0	Slot 6 Output Pixel Coverage Mask
M0.5	31:0	Slot 5 Output Pixel Coverage Mask
M0.4	31:0	Slot 4 Output Pixel Coverage Mask
M0.3	31:0	Slot 3 Output Pixel Coverage Mask
M0.2	31:0	Slot 2 Output Pixel Coverage Mask
M0.1	31:0	Slot 1 Output Pixel Coverage Mask
M0.0	31:0	Slot 0 Output Pixel Coverage Mask

M1.x		Slots 15:8 Output Pixel Coverage Mask (delivered only if original dispatch mode was SIMD16)

Writeback Message

SIMD8

The response length for all SIMD8 messages is 2. The data for each slot is written only if its corresponding execution mask bit is set.

DWord	Bit	Description
W0.7	31:0	**Barycentric[1] for Slot 7** Format = IEEE_Float
W0.6	31:0	**Barycentric[1] for Slot 6**
W0.5	31:0	**Barycentric[1] for Slot 5**
W0.4	31:0	**Barycentric[1] for Slot 4**
W0.3	31:0	**Barycentric[1] for Slot 3**
W0.2	31:0	**Barycentric[1] for Slot 2**
W0.1	31:0	**Barycentric[1] for Slot 1**
W0.0	31:0	**Barycentric[1] for Slot 0**
W1.7	31:0	**Barycentric[2] for Slot 7** Format = IEEE_Float
W1.6	31:0	**Barycentric[2] for Slot 6**
W1.5	31:0	**Barycentric[2] for Slot 5**
W1.4	31:0	**Barycentric[2] for Slot 4**
W1.3	31:0	**Barycentric[2] for Slot 3**
W1.2	31:0	**Barycentric[2] for Slot 2**
W1.1	31:0	**Barycentric[2] for Slot 1**
W1.0	31:0	**Barycentric[2] for Slot 0**

SIMD16

The response length for all SIMD16 messages is 4. The data for each slot is written only if its corresponding execution mask bit is set.

DWord	Bit	Description
W0.7	31:0	**Barycentric[1] for Slot 7** Format = IEEE_Float
W0.6	31:0	**Barycentric[1] for Slot 6**
W0.5	31:0	**Barycentric[1] for Slot 5**
W0.4	31:0	**Barycentric[1] for Slot 4**
W0.3	31:0	**Barycentric[1] for Slot 3**
W0.2	31:0	**Barycentric[1] for Slot 2**
W0.1	31:0	**Barycentric[1] for Slot 1**
W0.0	31:0	**Barycentric[1] for Slot 0**
W1.7	31:0	**Barycentric[2] for Slot 7** Format = IEEE_Float
W1.6	31:0	**Barycentric[2] for Slot 6**
W1.5	31:0	**Barycentric[2] for Slot 5**
W1.4	31:0	**Barycentric[2] for Slot 4**
W1.3	31:0	**Barycentric[2] for Slot 3**
W1.2	31:0	**Barycentric[2] for Slot 2**
W1.1	31:0	**Barycentric[2] for Slot 1**
W1.0	31:0	**Barycentric[2] for Slot 0** Format = IEEE_Float
W2.7	31:0	**Barycentric[1] for Slot 15**

DWord	Bit	Description
W2.6	31:0	**Barycentric[1] for Slot 14**
W2.5	31:0	**Barycentric[1] for Slot 13**
W2.4	31:0	**Barycentric[1] for Slot 12**
W2.3	31:0	**Barycentric[1] for Slot 11**
W2.2	31:0	**Barycentric[1] for Slot 10**
W2.1	31:0	**Barycentric[1] for Slot 9**
W2.0	31:0	**Barycentric[1] for Slot 8**
W3.7	31:0	**Barycentric[2] for Slot 15**
W3.6	31:0	**Barycentric[2] for Slot 14**
W3.5	31:0	**Barycentric[2] for Slot 13**
W3.4	31:0	**Barycentric[2] for Slot 12**
W3.3	31:0	**Barycentric[2] for Slot 11**
W3.2	31:0	**Barycentric[2] for Slot 10**
W3.1	31:0	**Barycentric[2] for Slot 9**
W3.0	31:0	**Barycentric[2] for Slot 8**

Coarse to Pixel Mapping Writeback Message (SIMD8 Message)

The writeback message layout is similar to SIMD8 PS thread payload. The first two writeback phases (W0 and W1) are mandatory, and remaining phases are optional as indicated by Coarse to Pixel Mapping Phase Enable bits.

W0.7	31:16	Pixel Mask VMASK (SubSpan[1:0]) : Indicates lit and helper pixels within the two subspans; for each 4-bit group, the OR of the corresponding 4-bit group in the Pixel Mask DMASK. The Pixel Shader kernel should use VMASK as dispatch mask if pixel and/or sample phases compute implicit derivatives. This field must not be modified by the Pixel Shader kernel.
	15:0	Pixel Mask DMASK (SubSpan[1:0]) : Indicates which pixels within the two subspans are lit (rasterized

		and passed early Z/stencil tests if present). The Pixel Shader kernel should use DMASK as dispatch mask if pixel and/or sample phases do not compute implicit derivatives.
W0.6	31:0	Reserved (MBZ)
W0.5	31:0	Reserved (MBZ)
W0.4	31:0	Reserved (MBZ)
W0.3	31:16	Y1 : Y coordinate (screen space) for upper-left pixel of subspan 1 (slot 4) Format = U16
	15:0	X1 : X coordinate (screen space) for upper-left pixel of subspan 1 (slot 4) Format = U16
W0.2	31:16	Y0 : Y coordinate (screen space) for upper-left pixel of subspan 0 (slot 0) Format = U16
	15:0	X0 : X coordinate (screen space) for upper-left pixel of subspan 0 (slot 0) Format = U16
W0.1	31:0	Reserved (MBZ)
W0.0	31:20	Reserved (MBZ)
	19:16	MSAA rate (multisample count) Format: U4 [1..16] This field specifies MSAA sampling rate (required for PS+S monolithic shader).
	15:4	Reserved (MBZ)
	3:0	Next Pixel Shader Phase Format = U4 [0..15] Pixel Interpolator returns identifier of the next pixel phase to be queried in monolithic CPS+PS(+S) implementation. The next phase may be different than requested phase + 1. The value of 0 indicates there are no more pixels available beyond requested phase. The intended usage scenario is 1) ISA kernel sends Coarse to Pixel Mapping requesting data for phase 0 2) If the returned next pixel phase is zero, ISA kernel executes pixel code for phase 0 (one loop pass) and terminates

		3) If the returned next pixel phase is greater than zero, ISA kernel executes pixel code for requested phase and then requests shader input for next phase (until value of zero is returned)
W1.7	31:0	Reserved (MBZ)
W1.6	31:0	Reserved (MBZ)
W1.5	31:0	Reserved (MBZ)
W1.4	31:0	Reserved (MBZ)
W1.3		Parent coarse shader slots for pixel dispatched at slots 7:6
W1.2		Parent coarse shader slots for pixel dispatched at slots 5:4
W1.1		Parent coarse shader slots for pixel dispatched at slots 3:2
W1.0		Parent coarse shader slots for pixel dispatched at slots 1:0
W2.7	31:0	Perspective Pixel Location Barycentric[1] for Slot 7 This phase is included only if the corresponding bit in Coarse to Pixel Mapping message specific control is enabled. Format = IEEE_Float
W2.6	31:0	Perspective Pixel Location Barycentric[1] for Slot 6
W2.5	31:0	Perspective Pixel Location Barycentric[1] for Slot 5
W2.4	31:0	Perspective Pixel Location Barycentric[1] for Slot 4
W2.3	31:0	Perspective Pixel Location Barycentric[1] for Slot 3
W2.2	31:0	Perspective Pixel Location Barycentric[1] for Slot 2
W2.1	31:0	Perspective Pixel Location Barycentric[1] for Slot 1
W2.0	31:0	Perspective Pixel Location Barycentric[1] for Slot 0
W3		Perspective Pixel Location Barycentric[2] for Slots 7:0
W4:5		Perspective Centroid Barycentric, included only if the corresponding bit in Coarse to Pixel Mapping message specific control is enabled.
W6:7		Linear Pixel Location Barycentric, included only if the corresponding bit in Coarse to Pixel Mapping message specific control is enabled.
W8:9		Linear Centroid Barycentric, included only if the corresponding bit in Coarse to Pixel Mapping message specific control is enabled.
W10		Pixel Shader Input Coverage Mask, for slots 7:0 This phase is included only if the corresponding bit in Coarse to Pixel Mapping message specific control is enabled. Format = U32

Coarse to Pixel Mapping Writeback Message (SIMD16 Message)

The writeback message layout is similar to SIMD16 PS thread payload. The first two writeback phases (W0 and W1) are mandatory, and remaining phases are optional as indicated by Coarse to Pixel Mapping Phase Enable bits.

Note: If all optional phases were enabled, the total length of the writeback message would exceed 16 GRFs which is not allowed. In such case, ISA kernel must split request into two PI messages. The following table does not reflect this split, and lists all phases together in the order they are delivered.

W0.7	31:16	Pixel Mask VMASK (SubSpan[1:0]) : Indicates lit and helper pixels within the two subspans; for each 4-bit group, the OR of the corresponding 4-bit group in the Pixel Mask DMASK. The Pixel Shader kernel should use VMASK as dispatch mask if pixel and/or sample phases compute implicit derivatives. This field must not be modified by the Pixel Shader kernel.
	15:0	Pixel Mask DMASK (SubSpan[1:0]) : Indicates which pixels within the two subspans are lit (rasterized and passed early Z/stencil tests if present). The Pixel Shader kernel should use DMASK as dispatch mask if pixel and/or sample phases do not compute implicit derivatives.
W0.6	31:0	Reserved (MBZ)
W0.5	31:16	Y3: Y coordinate (screen space) for upper-left pixel of subspan 3 (slot 12) Format = U16
	15:0	X3: X coordinate (screen space) for upper-left pixel of subspan 3 (slot 12) Format = U16
W0.4	31:16	Y2 : Y coordinate (screen space) for upper-left pixel of subspan 2 (slot 8) Format = U16
	15:0	X2 : X coordinate (screen space) for upper-left pixel of subspan 2 (slot 8) Format = U16
W0.3	31:16	Y1 : Y coordinate (screen space) for upper-left pixel of subspan 1 (slot 4) Format = U16
	15:0	X1 : X coordinate (screen space) for upper-left pixel of subspan 1 (slot 4) Format = U16
W0.2	31:16	Y0 : Y coordinate (screen space) for upper-left pixel of subspan 0 (slot 0) Format = U16

	15:0	X0 : X coordinate (screen space) for upper-left pixel of subspan 0 (slot 0) Format = U16
W0.1	31:0	Reserved (MBZ)
W0.0	31:20	Reserved (MBZ)
	19:16	MSAA rate (multisample count) Format: U4 [1..16] This field specifies MSAA sampling rate (required for PS+S monolithic shader).
	15:4	Reserved (MBZ)
	3:0	Next Pixel Shader Phase Format = U4 [0..15] Pixel Interpolator returns identifier of the next pixel phase to be queried in monolithic CPS+PS(+S) implementation. The next phase may be different than requested phase + 1. The value of 0 indicates there are no more pixels available beyond requested phase. The intended usage scenario is 1) ISA kernel sends Coarse to Pixel Mapping requesting data for phase 0 2) If the returned next pixel phase is zero, ISA kernel executes pixel code for phase 0 (one loop pass) and terminates 3) If the returned next pixel phase is greater than zero, ISA kernel executes pixel code for requested phase and then requests shader input for next phase (until value of zero is returned)
W1.7	32:16	Parent coarse shader slot for pixel dispatched at slot 15 Format: U16 Valid range: 0 - 15
	15:0	Parent coarse shader slot for pixel dispatched at slot 14 Format: U16 Valid range: 0 - 15
W1.6		Parent coarse shader slots for pixel dispatched at slots 13:12
W1.5		Parent coarse shader slots for pixel dispatched at slots 11:10
W1.4		Parent coarse shader slots for pixel dispatched at slots 9:8
W1.3		Parent coarse shader slots for pixel dispatched at slots 7:6
W1.2		Parent coarse shader slots for pixel dispatched at slots 5:4
W1.1		Parent coarse shader slots for pixel dispatched at slots 3:2
W1.0		Parent coarse shader slots for pixel dispatched at slots 1:0

W2.7	31:0	Perspective Pixel Location Barycentric[1] for Slot 7 This phase is included only if the corresponding bit in Coarse to Pixel Mapping message specific control is enabled. Format = IEEE_Float
W2.6	31:0	Perspective Pixel Location Barycentric[1] for Slot 6
W2.5	31:0	Perspective Pixel Location Barycentric[1] for Slot 5
W2.4	31:0	Perspective Pixel Location Barycentric[1] for Slot 4
W2.3	31:0	Perspective Pixel Location Barycentric[1] for Slot 3
W2.2	31:0	Perspective Pixel Location Barycentric[1] for Slot 2
W2.1	31:0	Perspective Pixel Location Barycentric[1] for Slot 1
W2.0	31:0	Perspective Pixel Location Barycentric[1] for Slot 0
W3		Perspective Pixel Location Barycentric[2] for Slots 7:0
W4		Perspective Pixel Location Barycentric[1] for Slots 15:8
W5		Perspective Pixel Location Barycentric[2] for Slots 15:8
W6:9		Perspective Centroid Barycentric, included only if the corresponding bit in Coarse to Pixel Mapping message specific control is enabled.
W10:13		Linear Pixel Location Barycentric, included only if the corresponding bit in Coarse to Pixel Mapping message specific control is enabled.
W14:17		Linear Centroid Barycentric, included only if the corresponding bit in Coarse to Pixel Mapping message specific control is enabled.
W18		Pixel Shader Input Coverage Mask, for slots 7:0 This phase is included only if the corresponding bit in Coarse to Pixel Mapping message specific control is enabled. Format = U32
W19		Pixel Shader Input Coverage Mask, for slots 15:8 Format = U32 This phase is included only if the corresponding bit in Coarse to Pixel Mapping message specific control is enabled.

Message Gateway

The Message Gateway has these functions:

- Barrier messages for thread-to-thread synchronization within a thread group.
- Event messages (Monitor, Wait, and Signal) for thread-to-thread signaling across GPU.

End of Thread messages to exit the GPGPU threads.

Messages

This section describes the Message Gateway messages.

Message Descriptor

The following message descriptor applies to all messages supported by Message Gateway.

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>19</td><td>Header Present. This bit must be 0 for all Message Gateway messages.</td></tr>
<tr><td>18:17</td><td>Ignored.</td></tr>
<tr><td>16:3</td><td>Reserved: MBZ</td></tr>
<tr><td>2:0</td><td>SubFuncID. Identify the supported sub-functions by Message Gateway. Encodings are:
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>001b</td><td>SignalEvent</td><td>Sends an event to all threads that are monitoring that Event ID</td></tr>
<tr><td>010b</td><td>MonitorEvent</td><td>Gateway will remember if this Event ID occurs</td></tr>
<tr><td>011b</td><td>MonitorNoEvent</td><td>Gateway will no longer record any Events for sending thread.</td></tr>
<tr><td>100b</td><td>BarrierMsg</td><td>Record an additional thread reaching the barrier</td></tr>
<tr><td>110b</td><td>WaitForEvent</td><td>Send a writeback if Event ID occured after MonitorEvent</td></tr>
</table>
All other encodings Reserved.</td></tr>
</table>

BarrierMsg Message

The BarrierMsg message gives the ability for multiple threads to synchronize their progress. This is useful when there are data shared between threads. The message consists of a single 256-bit message payload.

Upon receiving one such message, Message Gateway increments the Barrier counter and marks the Barrier requester thread. There is no immediate response from the Message Gateway when the incremented counter is not equal to the terminating thread count. When the incremented counter value does equal the **Barrier Thread Count**, Message Gateway sends a response back to all the Barrier requester threads and resets the Barrier count to zero.

Programming Note	
Context:	BarrierMsg Message
The Message Gateway assumes that the barrier ID sent in barrier message payload is valid and was allocated by TSG. In the event of a programming error specifying an invalid barrier ID, the results are undefined, and may cause the Message Gateway to stop responding to barrier messages for any thread that it services.	

Message Payload

DWord	Bits	Description
M0.5	31:0	**Ignored**
M0.4	31:0	**Ignored**
M0.3	31:0	**Predicate Mask.** This field has a bit set per SIMD channel that passes the predicate. For SIMD8 and SIMD16 the rest of the bits must be 0. This field is ignored for non-predicated barriers.
M0.2	31	**Ignored**
	30:24	**BarrierID.** This field indicates which barrier state is updated. Format: U7 Range = [0, 63]
	23	**Predicate Mask Enable.** This bit indicates that the barrier is a predicated barrier and the SIMD channels passing the predicate should be summed. All threads sending this message to the same barrier should have an identical value for this field, and must specify a response length of 1 for the predicate sum response. Note that Global Barriers must not have the Predicate Mask Enable bit set. **Programming Note:** This control is intended only for GPGPU or Media threads. This control must not be set if the barrier is for a Hull Shader thread. Format: Enable
	22:16	**Ignored**
	15	**Barrier Count Enable.** Allows the message to reprogram the terminating barrier count. If set, the stored value of the terminating barrier count is set to the value of Barrier Count field (below), and used for this barrier operation. If clear, the stored value of the terminating barrier count is not modified and the stored value is used for this barrier operation. **Programming Note:** This control is intended only for Hull Shader threads. This control must not be set if the barrier is allocated by a GPGPU or Media thread. Format: Enable
	14:8	**Barrier Count.** If Barrier Count Enable is set, this field specifies the terminating barrier count. Otherwise this field is ignored. All threads that belong to a single barrier must deliver the same value for this field for a particular barrier iteration. Format: U7

DWord	Bits	Description
	7:0	**Ignored**
M0.1	31:0	**Ignored**
M0.0	31:4	**Ignored**

Writeback Message to Requester Thread

The writeback message is sent only if the **AckReq** bit in the message descriptor is set.

Programming Note	
Context:	Writeback Message to Requester Thread
AckReq must not be used with predicate barriers, because the predicate barrier response is sent to the GRF writeback location instead of the Ack.	

DWord	Bits	Description
W0.7:1		Reserved (not overwritten)
W0.0	31:20	Reserved
	19:16	**Shared Function ID.** Contains the message gateway's shared function ID.
	15:3	Reserved
	2:0	**Error Code** 000: *Successful*. No Error (Normal). Other encodings are reserved.

Broadcast Writeback Message

Description
When the count for a Barrier reaches Barrier.Count, the Message Gateway sends the notification bit N0 to each EU/Thread that reached the barrier. A Barrier Return Byte is not sent.
This response message is sent back only if the Gateway Barrier Message specifies that this is a predicated barrier. This response is written to the GRF writeback location, and the response length specified in the send message to the EU must be 1.

DWord	Bits	Description
W0.7:1		Reserved (not overwritten)
W0.0	31:16	Reserved (not overwritten)
	15:0	**Predicated Barrier Mask Sum.** This field is a sum of the predicate mask bits sent by each thread. This field (and the DW containing it) is not written if the barrier is not marked as a predicated barrier. The kernel should compare this field to 0 for the predicated OR function and compare it to the workgroup size for the predicated AND function.

MonitorEvent

The MonitorEvent message indicates that the Gateway should remember any events of the specified EventID for the messaging thread. No writeback message is sent until a WaitForEvent message arrives, however. Only one event can be monitored at a time per thread, so a new MonitorEvent message will replace the event being monitored by the sending thread with the new EventID. An event recorded for the thread by a previous MonitorEvent is cleared when a new MonitorEvent comes in.

Message Payload

DWord	Bits	Description
M0.7	31:0	Ignored
M0.6	31:0	Ignored
M0.5	31:0	Ignored
M0.4	31:0	Ignored
M0.3	31:0	Ignored
M0.2	31:0	Ignored
M0.1	31:0	Ignored
M0.0	31:24	Ignored
	23:0	**EventID -** Indicates the ID of the event to be monitored.

WaitForEvent

The WaitForEvent message causes the Gateway to send a writeback message to the thread when a monitored event occurs, or to send the writeback immediately if the event occurred before the WaitForEvent and after a MonitorEvent. A timeout value is supplied to cause the Gateway to send the writeback after the timeout occurs. Timeout always happens during pre-emption.

Programming Note
The return of a WaitForEvent writeback message does not mean that that the monitored event occurred. A timeout or preemption may have occurred, or the signal could have been sent by a previous signaling phase, or possibly even by another program that is using the same event number. Therefore, each monitor event should be used in combination with a global memory location that is written and read with atomic messages. That memory location records the signal, and the monitor event becomes a way to know when to poll the global memory location to see if the event has occurred.

Message Payload

DWord	Bits	Description
M0.7	31:0	Ignored
M0.6	31:0	Ignored
M0.5	31:0	Ignored

DWord	Bits	Description
M0.4	31:0	Ignored
M0.3	31:0	Ignored
M0.2	31:0	Ignored
M0.1	31:0	Ignored
M0.0	31:10	Ignored
	9:0	**Timeout-** The amount of time GW should wait before sending a writeback message. This value is in terms of 1024 clocks, so with a 1Ghz clock it would be approximately in terms of uS. 0 and 1 are illegal values, since the actual timeout time can be short by up to 1 increment of the timeout value.

Writeback Message to Requester Thread

DWord	Bit	Description
R0.7	31:0	Ignored
R0.6	31:0	Ignored
R0.5	31:0	Ignored
R0.4	31:0	Ignored
R0.3	31:0	Ignored
R0.2	31:0	Ignored
R0.1	31:0	Ignored
R0.0	31:1	Ignored
	0	**Event Occurred** - This bit is 1 if the event occurred, or 0 if a timeout occurred.

Programming Note
Timing race conditions exist between one thread signaling an event, and another thread waiting for the event. It is possible that the WaitForEvent times out even when a signal has been sent. It is also possible that a WaitForEvent receives an EventOccurred from a prior signaling event that wasn't cleared out. The correct programming model for Monitors uses atomic operations on a memory location shared by the threads for synchronization and monitors to reduce polling overhead on the memory location. The signaling thread first specifies the event with an atomic operation and then signals the event. The receiving thread waits for the event (which might be correct, might be a timeout or might be an old signal), and then verifies the event by checking the shared memory location.

SignalEvent

This message indicates to the rest of the system that the specified event has occurred. This event will be recorded for all threads monitoring it, and messages will be sent to all threads with an outstanding WaitForEvent.

Message Payload

DWord	Bits	Description
M0.7	31:0	Ignored
M0.6	31:0	Ignored
M0.5	31:0	Ignored
M0.4	31:0	Ignored
M0.3	31:0	Ignored
M0.2	31:0	Ignored
M0.1	31:0	Ignored
M0.0	31:24	Ignored
	23:0	**EventID -** Indicates the ID of the event to be signaled.

MonitorNoEvent

This message indicates that the thread no longer wishes to monitor any events. The EOT message at the end of the thread does the same thing, so this message is optional for most usages. It is not needed when switching between different events - just a new MonitorEvent with the new Event ID suffices.

Message Payload

DWord	Bits	Description
M0.7	31:0	Ignored
M0.6	31:0	Ignored
M0.5	31:0	Ignored
M0.4	31:0	Ignored
M0.3	31:0	Ignored
M0.2	31:0	Ignored
M0.1	31:0	Ignored
M0.0	31:0	Ignored

Barrier Messages

Barriers are used to synchronize the execution of two or more concurrently running threads in a threadgroup.

Each thread executes the barrier in two parts:

- send a signal_barrier message to Gateway port
- use sync.bar instruction to wait for a barrier completion notification from Gateway port

When all the threads sharing the same barrier have signaled the barrier, then the Gateway port notifies the threads that the barrier is complete.

The barrier is marked inactive when it is allocated, and whenever a barrier completion notification is sent.

A barrier is allocated when a threadgroup is dispatched, and released when all the threads in the threadgroup exit. A thread that signals a barrier must wait for the completion notification before the thread exits.

Event Messages

Event Monitors are used by threads to be notified of an event signaled by another running threads in the GPU.

When a thread sends a **signal_event** message to Gateway port, the event is broadcast to all running threads in the GPU that have previously requested to be notified of the event.

To receive an event, a thread:

1. Requests to be notified of an Event by sending a monitor_event message to the Gateway port. This registers the thread as expecting the event.
2. Waits until the event is signaled (or a timeout occurs) by sending a wait_event message to the Gateway port.
3. Checks a memory-based semaphore to ensure that the signal was received.

Events are not, by themselves, a thread synchronization mechanism. Typically, independently running threads synchronize using atomic operations on memory-based semaphores. Events are a mechanism to reduce polling operations on memory-based semaphores.

Sample code to send and receive events with a semaphore:

Signaling Thread	Receiving Thread

extern int semaphore = 0;
// do work, produce results ...
atomic_store(&semaphore, 1);
signal_event(&semaphore);
	extern int semaphore;
	monitor_event(&semaphore);
	while ((atomic_cmpxchg(&semaphore, 1, 0) == 0) {
	  wait_event( );
	}
	monitor_no_event( );
	// consume other thread's results ...

When a thread exits, it is automatically unregistered from any events (implicit monitor_no_event).

End of Thread (EOT) Message

A thread exits by executing a send instruction with the EOT bit set. Only a small number of send messages support EOT. The Gateway EOT message is one of those few messages. (Which message to use for a thread depends on which fixed function was used to dispatch the thread.)

The Gateway EOT message removes the barrier and event tracking for the thread, and decrements the use count on the thread's barriers and shared local memory. When the use count for a barrier or shared local memory reaches zero, then that resource is available for allocation by another threadgroup.

Intel® Iris® Xe and UHD Graphics Open Source

Programmer's Reference Manual

For the 2020-2021 11th Generation Intel Xeon®, Core™, Celeron®, Pentium® Gold Processors based on the "Tiger Lake" Platform

Volume 10: Copy Engine

December 2021, Revision 1.0

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Table of Contents

Copy Engine 1

Blitter (BLT) Command Streamer 1

Blitter Engine Command Streamer (BCS) 1

Software Control Bit Definitions 7

Registers for Blitter Engine Command Streamer 7

BLT Engine 8

Classical BLT Engine Functional Description 8

Fast Copy Engine 35

BLT Instruction Overview 36

BLT Engine State 36

Device Cache Coherency: Render & Texture Caches 37

Verbatim Copy operation 37

BLT Engine Instructions 38

2D (XY) BLT Instructions 40

BLT Engine Instruction Field Definitions 44

Copy Engine

Copy Engine or BLT Engine is an engine that runs in parallel with Render Engine, Compute Engine and Media Engine. It is capable of moving blocks of data from one location (source) in the memory to another location (destination) in the memory. It can also fill up a specified location in the memory with fixed data. Copy Engine can perform pre-defined logical or bitwise operation on source, destination and another fixed data called pattern data and write to the destination confined by a clip rectangle. Copy Engine can be divided into two categories:

1. Classical BLT Engine - Can perform basic Copy and Fill operations as well as bit-wise operations. It is low performance.
2. Fast Copy Engine - performs basic copy and fill operations. It is high performance. It can also support compute specific copy operations other than the regular 2D operations.

Fast Copy Engine supports only few high performance commands, whereas Classical BLT Engine, which is also known as Legacy Blitter, supports all others command, including some very complex commands involving bit-wise/logical operations.

Blitter (BLT) Command Streamer

Blitter Command Streamer is the BLT engine command streamer where scheduler submits workloads for the BLT engine. Blitter Command streamer supports execution of the generic functions plus BLT engine commands. Each BLT Engine has its own independent command streamer.

Blitter Engine Command Streamer (BCS)

The BCS (Blitter Command Streamer) unit primarily serves as the software programming interface between the O/S driver and the Blitter Engine. It is responsible for fetching, decoding, and dispatching of data packets (Blitter Commands) to the front-end interface module of Blitter Engine.

Logic Functions Included
• MMIO register programming interface. • DMA action for fetching of ring data from memory. • Management of the Head pointer for the Ring Buffer. • Decode of ring data and sending it to the blit engine. • Handling of user interrupts. • Flushing the Blitter Engine. • Handle NOP.
• DMA action for fetching of execlists from memory. • Handling of ring context switch interrupt.

The BCS unit only claims memory mapped I/O cycles that are targeted to its range of 0x22000 to 0x224FF. The Blitter, Render and Media Engines use semaphore to synchronize their operations.

BCS operates completely independent of the other render and media command streams.

The simple sequence of events is as follows: a ring (say PRB0) is programmed by a memory-mapped register write cycle. The DMA inside BCS is kicked off. The DMA fetches commands from memory based on the starting address and head pointer. The DMA requests cache lines from memory (one cacheline CL at a time). There is guaranteed space in the DMA FIFO (8 CL deep) for data coming back from memory. The DMA control logic has copies of the head pointer and the tail pointer. The DMA increments the head pointer after making requests for ring commands. Once the DMA copy of the head pointer becomes equal to the tail pointer, the DMA stops requesting.

The parser starts executing once the DMA FIFO has valid commands. All the commands have a header DWord packet. Based on the encoding in the header packet, the command may be targeted towards Blit Engine or the command parser. After execution of every command, the actual head pointer is updated. The ring is considered empty when the head pointer becomes equal to the tail pointer.

Context Management

Copy Engine Register State Context

Register State Context

EXECLIST CONTEXT
EXECLIST CONTEXT(PPGTT Base)
ENGINE CONTEXT

Description	MMIO Offset/Command	Unit	Dw Count	DW Offset
CSFE Execlist Context		BCSFE	192	**0**
MI_BATCH_BUFFER_END		CSEND	1	**00C0**
NOOP		CSEND	127	**00C1**
			DW	320
			K Bytes	1.28

Copy Engine Power Context

This section lists the power context image of Copy Engine across generations.

Blitter Engine Power Context

The table below captures the data from BCS power context save/restored by PM. Address offset in the below table is relative to the starting location of BCS in the overall power context image managed by PM.

BCS Power Context Image

Description			# of DW	Address Offset(PWR)	CSFE/CSBE
CSFE Power Context with Display			192	**0**	**CSFE**
NOOP		BCS	1	**00D0**	**CSBE**
Load_Register_Immediate header	0x1100_1003	BCS	1	**00D1**	**CSBE**
GAB MODE REGISTER	0x220a0	BCS	2	**00D4**	**CSBE**
NOOP		BCS	8	**00D6**	**CSBE**
NOOP		BCS	1	**00DE**	**CSBE**
MI_BATCH_BUFFER_END		BCS	1	**00DF**	**CSBE**

Blitter Command Formats

2D Commands

The 2D commands include various flavors of BLT operations, along with commands to set up BLT engine state without actually performing a BLT. Most commands are of fixed length, though there are a few commands that include a variable amount of "inline" data at the end of the command (in case of legacy Blitter command).

All the following commands are defined in *Blitter Instructions*.

2D Command Map

Opcode (28:22)	Command
00h	Reserved
01h	XY_SETUP_BLT
02h	Reserved
03h	XY_SETUP_CLIP_BLT
04h-10h	Reserved
11h	XY_SETUP_MONO_PATTERN_SL_BLT
12h-23h	Reserved
24h	XY_PIXEL_BLT
25h	XY_SCANLINES_BLT
26h	XY_TEXT_BLT
27h-30h	ReservedReserved
31h	XY_TEXT_IMMEDIATE_BLT
32h-3Fh	Reserved
40h	COLOR_BLT

Opcode (28:22)	Command
41h	XY_BLOCK_COPY_BLT
42h	XY_FAST_COPY_BLT
43h	SRC_COPY_BLT
44h	XY_FAST_COLOR_BLT
45h-47h	Reserved
49h-4Fh	Reserved
50h	XY_COLOR_BLT
51h	XY_PAT_BLT
52h	XY_MONO_PAT_BLT
53h	XY_SRC_COPY_BLT
54h	XY_MONO_SRC_COPY_BLT
55h	XY_FULL_BLT
56h	XY_FULL_MONO_SRC_BLT
57h	XY_FULL_MONO_PATTERN_BLT
58h	XY_FULL_MONO_PATTERN_MONO_SRC_BLT
59h	XY_MONO_PAT_FIXED_BLT
71h	XY_MONO_SRC_COPY_IMMEDIATE_BLT
72h	XY_PAT_BLT_IMMEDIATE
73h	XY_SRC_COPY_CHROMA_BLT
74h	XY_FULL_IMMEDIATE_PATTERN_BLT
75h	XY_FULL_MONO_SRC_IMMEDIATE_PATTERN_BL
76h	XY_PAT_CHROMA_BLT
77h	XY_PAT_CHROMA_BLT_IMMEDIATE
78h-7Fh	Reserved

Blitter Command Header Format

Type	Bits				
	31:29	28:24	23	22	21:0
Memory Interface (MI)	000	Opcode 00h - NOP 0Xh - Single DWord Commands 1Xh - Two+ DWord Commands 2Xh - Store Data Commands 3Xh - Ring/Batch Buffer Cmds		Identification No./DWord Count Command Dependent Data 5:0 - DWord Count 5:0 - DWord Count 5:0 - DWord Count	
Reserved	001				
Reserved	011				

Type	Bits			
	31:29	28:22	21:9	8:0
Blitter (2D)	010	Command Opcode	Command Dependent Data	Dword Count

Logical Context Support

The following are the Logical Context Support Registers:

Register
BB_ADDR - Batch Buffer Head Pointer Register
BB_ADDR_UDW - Batch Buffer Upper Head Pointer Register
SBB_ADDR - Second Level Batch Buffer Head Pointer Register
SBB_ADDR_UDW - Second Level Batch Buffer Upper Head Pointer Register
SYNC_FLIP_STATUS - Wait For Event and Display Flip Flags Register
SYNC_FLIP_STATUS_1 - Wait For Event and Display Flip Flags Register 1
SYNC_FLIP_STATUS_2 - Wait For Event and Display Flip Flags Register 2
CXT_EL_OFFSET - Exec-List Context Offset
BB_START_ADDR_UDW - Batch Buffer Start Upper Head Pointer Register
BB_ADDR_DIFF - Batch Address Difference Register
WAIT_FOR_RC6_EXIT - Control Register for Power Management
SBB_STATE - Second Level Batch Buffer State Register
BB_OFFSET - Batch Offset Register
RING_BUFFER_HEAD_PREEMPT_REG - RING_BUFFER_HEAD_PREEMPT_REG
BB_PREEMPT_ADDR - Batch Buffer Head Pointer Preemption Register
BB_PREEMPT_ADDR_UDW - Batch Buffer Upper Head Pointer Preemption Register
SBB_PREEMPT_ADDR - Second Level Batch Buffer Head Pointer Preemption Register
SBB_PREEMPT_ADDR_UDW - Second Level Batch Buffer Upper Head Pointer Preemption Register
MI_PREDICATE_RESULT_1 - Predicate Rendering Data Result 1
MI_PREDICATE_RESULT_2 - Predicate Rendering Data Result 2
INDIRECT_CTX - Indirect Context Pointer
INDIRECT_CTX_OFFSET - Indirect Context Offset Pointer
BB_PER_CTX_PTR - Batch Buffer Per Context Pointer

Mode Registers

The following table describes the Mode Registers.

Registers
BCS_CXT_SIZE - BCS Context Sizes
MI_MODE - Mode Register for Software Interface

Mode Registers (continued)

Reisters
INSTPM - Instruction Parser Mode Register
EXCC - Execute Condition Code Register
IDLEDLY - Idle Switch Delay
SEMA_WAIT_POLL - Semaphore Polling Interval on Wait
RESET_CTRL - Reset Control Register
PREEMPTION_HINT - Preemption Hint
PREEMPTION_HINT_UDW - Preemption Hint Upper DWord

Register
HWS_PGA - Hardware Status Page Address Register

MI Commands for Blitter Engine

This chapter describes the formats of the "Memory Interface" commands, including brief descriptions of their use. The functions performed by these commands are discussed fully in the *Memory Interface Functions* Device Programming Environment chapter.

This chapter describes MI Commands for the blitter graphics processing engine. The term "for Blitter Engine" in the title has been added to differentiate this chapter from a similar one describing the MI commands for the Media Decode Engine and the Rendering Engine.

The commands detailed in this chapter are used across products. However, slight changes may be present in some commands (i.e., for features added or removed), or some commands may be removed entirely. Refer to the *Preface* chapter for product specific summary.

Commands
MI_NOOP
MI_ARB_ON_OFF
MI_BATCH_BUFFER_START

The following table lists the non-privileged registers that can be written to from a non-secure batch buffer executed from Render Command Streamer.

User Mode Non-Privileged Registers

MMIO Name	MMIO Offset	Size in DWords
BCS_GPR	22600h	32
BCS_SWCTRL	22200h	32

Commands
MI_BATCH_BUFFER_END
MI_CONDITIONAL_BATCH_BUFFER_END
MI_DISPLAY_FLIP

Commands
MI_LOAD_SCAN_LINES_EXCL
MI_LOAD_SCAN_LINES_INCL
MI_FLUSH_DW
MI_MATH
MI_REPORT_HEAD
MI_STORE_DATA_IMM
MI_ATOMIC
MI_COPY_MEM_MEM
MI_LOAD_REGISTER_REG
MI_LOAD_REGISTER_MEM
MI_STORE_REGISTER_MEM
MI_SUSPEND_FLUSH
MI_USER_INTERRUPT
MI_WAIT_FOR_EVENT
MI_SEMAPHORE_SIGNAL
MI_SEMAPHORE_WAIT
MI_FORCE_WAKEUP

Software Control Bit Definitions

Registers in the range 22XX are not protected from the load register immediate instruction if the command is executed in the non-secure batch buffer.

BCS_SWCTRL - BCS SW Control

Registers for Blitter Engine Command Streamer

These are the Registers for the Blitter Engine Command Streamer.

Also see the Observability volume for related information

GAB_MODE - Mode Register for GAB

BLT Engine

Introduction

2D Rendering can be divided into 2 categories: classical BLTs, described here, and 3D BLTs. 3D BLTs are operations which can take advantage of the 3D drawing engine's functionality and access patterns.

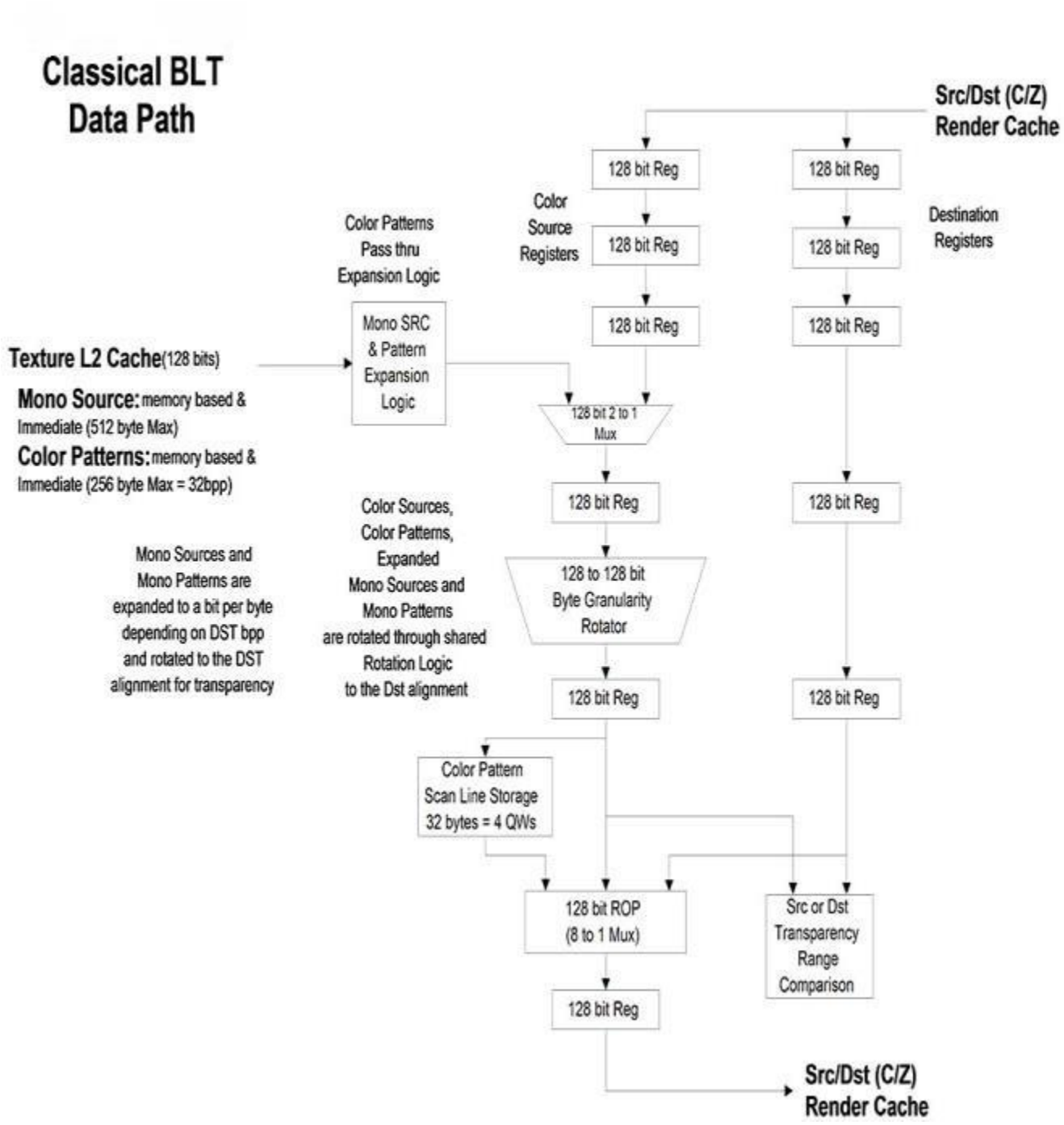
Functions such as Alpha BLTs, arithmetic (bilinear) stretch BLTs, rotations, transposing pixel maps, color space conversion, and DIBs are all considered 3D BLTs and are covered in the 3D rendering section. DIBs can be thought of as an indexed texture which uses the texture palette for performing the data translation. All drawing engines have swappable context. The same hardware can be used by multiple driver threads where the current state of the hardware is saved to memory and the appropriate state is loaded from memory on thread switches.

All operands for both 3D and classical BLTs can be in graphics aperture or cacheable system memory. Some operands can be immediates which are sent through the command stream. Immediate operands are: patterns, monochrome sources, DIB palettes, and DIB source operands. All non-monochrome operands which are not tiled have a stride granularity of a double-word (4 bytes).

The classical BLT commands support both linear addressing and X, Y coordinates with and without clipping. All X1 and Y1 destination and clipping coordinates are inclusive, while X2 and Y2 are exclusive. Currently, only destination coordinates can be negative. The source and clipping coordinates must be positive. If clipping is disabled, but a negative destination coordinate is specified, the negative coordinate is clipped to 0. Linear address BLT commands must supply a non-zero height and width. If either height or width = 0, then no accesses occur.

Classical BLT Engine Functional Description

The graphics controller provides a hardware-based BLT engine to off load the work of moving blocks of graphics data from the host CPU. Although the BLT engine is often used simply to copy a block of graphics data from the source to the destination, it also has the ability to perform more complex functions. The BLT engine is capable of receiving three different blocks of graphics data as input as shown in the figure below. The source data may exist in the frame buffer or the Graphics aperture. The pattern data always represents an 8x8 block of pixels that can be located in the frame buffer, Graphics aperture, or passed through a command packet. The pattern data must be located in linear memory. The data already residing at the destination may also be used as an input. The destination data can also be located in the frame buffer or graphics aperture.



Block Diagram and Data Paths of the BLT Engine

The BLT engine may use any combination of these three different blocks of graphics data as operands, in both bit-wise logical operations to generate the actual data to be written to the destination, and in per-pixel write-masking to control the writing of data to the destination. It is intended that the BLT engine will perform these bit-wise and per-pixel operations on color graphics data that is at the same color depth that the rest of the graphics system has been set. However, if either the source or pattern data is monochrome, the BLT engine has the ability to put either block of graphics data through a process called

"color expansion" that converts monochrome graphics data to color. Since the destination is often a location in the on-screen portion of the frame buffer, it is assumed that any data already at the destination will be of the appropriate color depth.

Basic BLT Functional Considerations

Color Depth Configuration and Color Expansion

The graphics system and BLT engine can be configured for color depths of 8, 16, and 32 bits per pixel.

The configuration of the BLT engine for a given color depth dictates the number of bytes of graphics data that the BLT engine will read and write for each pixel while performing a BLT operation. It is assumed that any graphics data already residing at the destination which is used as an input is already at the color depth to which the BLT engine is configured. Similarly, it is assumed that any source or pattern data used as an input has this same color depth, unless one or both is monochrome. If either the source or pattern data is monochrome, the BLT engine performs a process called "color expansion" to convert such monochrome data to color at the color depth to which the BLT engine has been set.

During "color expansion" the individual bits of monochrome source or pattern data that correspond to individual pixels are converted into 1, 2, or 4 bytes (whichever is appropriate for the color depth to which the BLT engine has been set). If a given bit of monochrome source or pattern data carries a value of 1, then the byte(s) of color data resulting from the conversion process are set to carry the value of a specified foreground color. If a given bit of monochrome source or pattern data carries a value of 0, the resulting byte(s) are set to the value of a specified background color or not written if transparency is selected.

The BLT engine is set to a default configuration color depth of 8, 16, or 32 bits per pixel through BLT command packets. Whether the source and pattern data are color or monochrome must be specified using command packets. Foreground and background colors for the color expansion of both monochrome source and pattern data are also specified through the command packets. The source foreground and background colors used in the color expansion of monochrome source data are specified independently of those used for the color expansion of monochrome pattern data.

Graphics Data Size Limitations

The BLT engine is capable of transferring very large quantities of graphics data. Any graphics data read from and written to the destination is permitted to represent a number of pixels that occupies up to 65,536 scan lines and up to 32,768 bytes per scan line at the destination. The maximum number of pixels that may be represented per scan line's worth of graphics data depends on the color depth.

Any source data used as an input must represent the same number of pixels as is represented by any data read from or written to the destination, and it must be organized so as to occupy the same number of scan lines and pixels per scan line.

The actual number of scan lines and bytes per scan line required to accommodate data read from or written to the destination are set in the destination width & height registers or using X and Y coordinates within the command packets. These two values are essential in the programming of the BLT engine, because the engine uses these two values to determine when a given BLT operation has been completed.

Bit-Wise Operations

The BLT engine can perform any one of 256 possible bit-wise operations using various combinations of the three previously described blocks of graphics data that the BLT engine can receive as input.

The choice of bit-wise operation selects which of the three inputs will be used, as well as the particular logical operation to be performed on corresponding bits from each of the selected inputs. The BLT engine automatically foregoes reading any form of graphics data that has not been specified as an input by the choice of bit-wise operation. An 8-bit code written to the raster operation field of the command packets chooses the bit-wise operation. The following table lists the available bit-wise operations and their corresponding 8-bit codes.

Bit-Wise Operations and 8-Bit Codes (00-3F)

Code	Value Written to Bits at Destination	Code	Value Written to Bits at Destination
00	writes all 0's	20	D and (P and (notS))
01	not(D or (P or S)))	21	not(S or(D xor P))
02	D and (not(P or S))	22	D and (notS)
03	not(P or S)	23	not(S or (P and (notD)))
04	S and (not(D or P))	24	(S xor P) and (D xor S)
05	not(D or P)	25	not(P xor (D and (not(S and P))))
06	not(P or (not(D xor S)))	26	S xor (D or (P and S))
07	not(P or (D and S))	27	S xor (D or (not(P xor S)))
08	S and (D and (notP))	28	D and (P xor S)
09	not(P or (D xor S))	29	not(P xor (S xor (D or (P and S))))
0A	D and (notP)	2A	D and (not(P and S))
0B	not(P or (S and (notD)))	2B	not(S xor ((S xor P) and (P xor D)))
0C	S and (notP)	2C	S xor (P and (D or S))
0D	not(P or (D and (notS)))	2D	P xor (S or (notD))
0E	not(P or (not(D or S)))	2E	P xor (S or (D xor P))
0F	notP	2F	not(P and (S or (notD)))
10	P and (not(D or S))	30	P and (notS)
11	not(D or S)	31	not(S or (D and (notP)))
12	not(S or (not(D xor P)))	32	S xor (D or (P or S))
13	not(S or (D and P))	33	notS
14	not(D or (not(P xor S)))	34	S xor (P or (D and S))
15	not(D or (P and S))	35	S xor (P or (not(D xor S)))
16	P xor (S xor (D and (not(P and S))))	36	S xor (D or P)
17	not(S xor ((S xor P) and (D xor S)))	37	not(S and (D or P))
18	(S xor P) and (P xor D)	38	P xor (S and (D or P))
19	not(S xor (D and (not(P and S))))	39	S xor (P or (notD))

Code	Value Written to Bits at Destination	Code	Value Written to Bits at Destination
1A	P xor (D or (S and P))	3A	S xor (P or (D xor S))
1B	not(S xor (D and (P xor S)))	3B	not(S and (P or (notD)))
1C	P xor (S or (D and P))	3C	P xor S
1D	not(D xor (S and (P xor D)))	3D	S xor (P or (not(D or S)))
1E	P xor (D or S)	3E	S xor (P or (D and (notS)))
1F	not(P and (D or S))	3F	not(P and S)

Notes:

S = Source Data
P = Pattern Data
D = Data Already Existing at the Destination

Bit-Wise Operations and 8-bit Codes (40 - 7F)

Code	Value Written to Bits at Destination	Code	Value Written to Bits at Destination
40	P and (S and (notD))	60	P and (D xor S)
41	not(D or (P xor S))	61	not(D xor (S xor (P or (D and S))))
42	(S xor D) and (P xor D)	62	D xor (S and (P or D))
43	not(S xor (P and (not(D and S))))	63	S xor (D or (notP))
44	S and (notD)	64	S xor (D and (P or S))
45	not(D or (P and (notS)))	65	D xor (S or (notP))
46	D xor (S or (P and D))	66	D xor S
47	not(P xor (S and (D xor P)))	67	S xor (D or (not(P or S)))
48	S and (D xor P)	68	not(D xor (S xor (P or (not(D or S)))))
49	not(P xor (D xor (S or (P and D))))	69	not(P xor (D xor S))
4A	D xor (P and (S or D))	6A	D xor (P and S)
4B	P xor (D or (notS))	6B	not(P xor (S xor (D and (P or S))))
4C	S and (not(D and P))	6C	S xor (D and P)
4D	not(S xor ((S xor P) or (D xor S)))	6D	not(P xor (D xor (S and (P or D))))
4E	P xor (D or (S xor P))	6E	S xor (D and (P or (notS)))
4F	not(P and (D or (notS)))	6F	not(P and (not(D xor S)))
50	P and (notD)	70	P and (not(D and S))
51	not(D or (S and (notP)))	71	not(S xor ((S xor D) and (P xor D)))
52	D xor (P or (S and D))	72	S xor (D or (P xor S))
53	not(S xor (P and (D xor S)))	73	not(S and (D or (notP)))
54	not(D or (not(P or S)))	74	D xor (S or (P xor D))
55	notD	75	not(D and (S or (notP)))
56	D xor (P or S)	76	S xor (D or (P and (notS)))

Code	Value Written to Bits at Destination	Code	Value Written to Bits at Destination
57	not(D and (P or S))	77	not(D and S)
58	P xor (D and (S or P))	78	P xor (D and S)
59	D xor (P or (notS))	79	not(D xor (S xor (P and (D or S))))
5A	D xor P	7A	D xor (P and (S or (notD)))
5B	D xor (P or (not(S or D)))	7B	not(S and (not(D xor P)))
5C	D xor (P or (S xor D))	7C	S xor (P and (D or (notS)))
5D	not(D and (P or (notS)))	7D	not(D and (not(P xor S)))
5E	D xor (P or (S and (notD)))	7E	(S xor P) or (D xor S)
5F	not(D and P)	7F	not(D and (P and S))

Notes:

S = Source Data
P = Pattern Data
D = Data Already Existing at the Destination

Bit-Wise Operations and 8-bit Codes (80 - BF)

Code	Value Written to Bits at Destination	Code	Value Written to Bits at Destination
80	D and (P and S)	A0	D and P
81	not((S xor P) or (D xor S))	A1	not(P xor (D or (S and (notP))))
82	D and (not(P xor S))	A2	D and (P or (notS))
83	not(S xor (P and (D or (notS))))	A3	not(D xor (P or (S xor D)))
84	S and (not(D xor P))	A4	not(P xor (D or (not(S or P))))
85	not(P xor (D and (S or (notP))))	A5	not(P xor D)
86	D xor (S xor (P and (D or S)))	A6	D xor (S and (notP))
87	not(P xor (D and S))	A7	not(P xor (D and (S or P)))
88	D and S	A8	D and (P or S)
89	not(S xor (D or (P and (notS))))	A9	not(D xor (P or S))
8A	D and (S or (notP))	AA	D
8B	not(D xor (S or (P xor D)))	AB	D or (not(P or S))
8C	S and (D or (notP))	AC	S xor (P and (D xor S))
8D	not(S xor (D or (P xor S)))	AD	not(D xor (P or (S and D)))
8E	S xor ((S xor D) and (P xor D))	AE	D or (S and (notP))
8F	not(P and (not(D and S)))	AF	D or (notP)
90	P and (not(D xor S))	B0	P and (D or (notS))
91	not(S xor (D and (P or (notS))))	B1	not(P xor (D or (S xor P)))
92	D xor (P xor (S and (D or P)))	B2	S xor ((S xor P) or (D xor S))
93	not(S xor (P and D))	B3	not(S and (not(D and P)))

Code	Value Written to Bits at Destination	Code	Value Written to Bits at Destination
94	P xor (S xor (D and (P or S)))	B4	P xor (S and (notD))
95	not(D xor (P and S))	B5	not(D xor (P and (S or D)))
96	D xor (P xor S)	B6	D xor (P xor (S or (D and P)))
97	P xor (S xor (D or (not(P or S))))	B7	not(S and (D xor P))
98	not(S xor (D or (not(P or S))))	B8	P xor (S and (D xor P))
99	not(D xor S)	B9	not(D xor (S or (P and D)))
9A	D xor (P and (notS))	BA	D or (P and (notS))
9B	not(S xor (D and (P or S)))	BB	D or (notS)
9C	S xor (P and (notD))	BC	S xor (P and (not(D and S)))
9D	not(D xor (S and (P or D)))	BD	not((S xor D) and (P xor D))
9E	D xor (S xor (P or (D and S)))	BE	D or (P xor S)
9F	not(P and (D xor S))	BF	D or (not(P and S))

Notes:

S = Source Data
P = Pattern Data
D = Data Already Existing at the Destination

Bit-Wise Operations and 8-bit Codes (C0 - FF)

Code	Value Written to Bits at Destination	Code	Value Written to Bits at Destination
C0	P and S	E0	P and (D or S)
C1	not(S xor (P or (D and (notS))))	E1	not(P xor (D or S))
C2	not(S xor (P or (not(D or S))))	E2	D xor (S and (P xor D))
C3	not(P xor S)	E3	not(P xor (S or (D and P)))
C4	S and (P or (notD))	E4	S xor (D and (P xor S))
C5	not(S xor (P or (D xor S)))	E5	not(P xor (D or (S and P)))
C6	S xor (D and (notP))	E6	S xor (D and (not(P and S)))
C7	not(P xor (S and (D or P)))	E7	not((S xor P) and (P xor D))
C8	S and (D or P)	E8	S xor ((S xor P) and (D xor S))
C9	not(S xor (P or D))	E9	not(D xor (S xor (P and (not(D and S)))))
CA	D xor (P and (S xor D))	EA	D or (P and S)
CB	not(S xor (P or (D and S)))	EB	D or (not(P xor S))
CC	S	EC	S or (D and P)
CD	S or (not(D or P))	ED	S or (not(D xor P))
CE	S or (D and (notP))	EE	D or S
CF	S or (notP)	EF	S or (D or (notP))
D0	P and (S or (notD))	F0	P

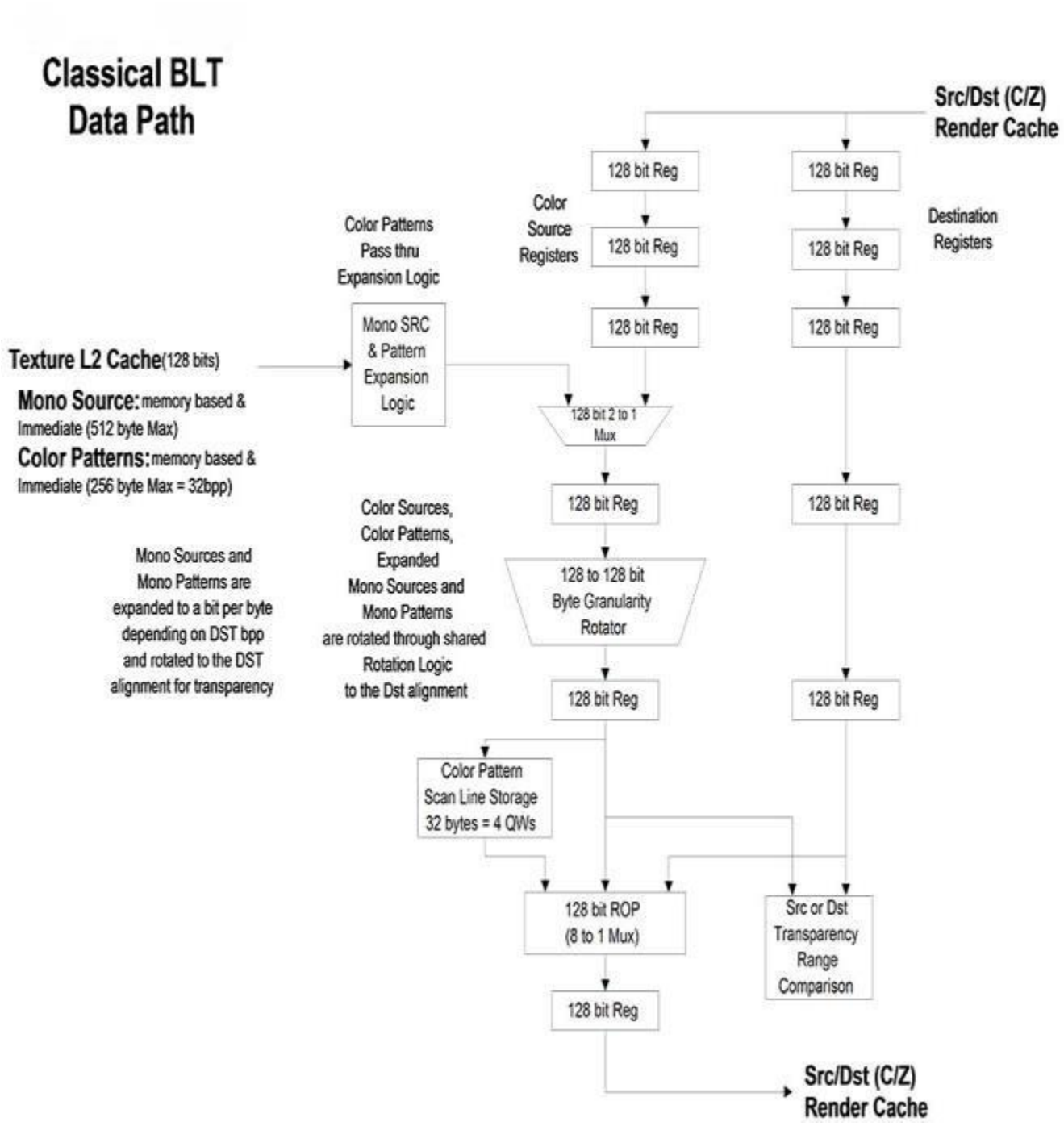
Code	Value Written to Bits at Destination	Code	Value Written to Bits at Destination
D1	not(P xor (S or (D xor P)))	F1	P or (not(D or S))
D2	P xor (D and (notS))	F2	P or (D and (notS))
D3	not(S xor (P and (D or S)))	F3	P or (notS)
D4	S xor ((S xor P) and (P xor D))	F4	P or (S and (notD))
D5	not(D and (not(P and S)))	F5	P or (notD)
D6	P xor (S xor (D or (P and S)))	F6	P or (D xor S)
D7	not(D and (P xor S))	F7	P or (not(D and S))
D8	P xor (D and (S xor P))	F8	P or (D and S)
D9	not(S xor (D or (P and S)))	F9	P or (not(D xor S))
DA	D xor (P and (not(S and D)))	FA	D or P
DB	not((S xor P) and (D xor S))	FB	D or (P or (notS))
DC	S or (P and (notD))	FC	P or S
DD	S or (notD)	FD	P or (S or (notD))
DE	S or (D xor P)	FE	D or (P or S)
DF	S or (not(D and P))	FF	writes all 1's

Notes:

S = Source Data
P = Pattern Data
D = Data Already Existing at the Destination

Per-Pixel Write-Masking Operations

The BLT engine is able to perform per-pixel write-masking with various data sources used as pixel masks to constrain which pixels at the destination are to be written to by the BLT engine. As shown in the figure below, either monochrome source or monochrome pattern data may be used as pixel masks. Color pattern data cannot be used. Another available pixel mask is derived by comparing a particular color range per color channel to either the color already specified for a given pixel at the destination or source.



Block Diagram and Data Paths of the BLT Engine

The command packets can specify the monochrome source or the monochrome pattern data as a pixel mask. When this feature is used, the bits that carry a value of 0 cause the bytes of the corresponding pixel at the destination to not be written to by the BLT engine, thereby preserving whatever data was originally carried within those bytes. This feature can be used in writing characters to the display, while also preserving the pre-existing backgrounds behind those characters. When both operands are in the transparent mode, the logical AND of the 2 operands are used for the write enables per pixel.

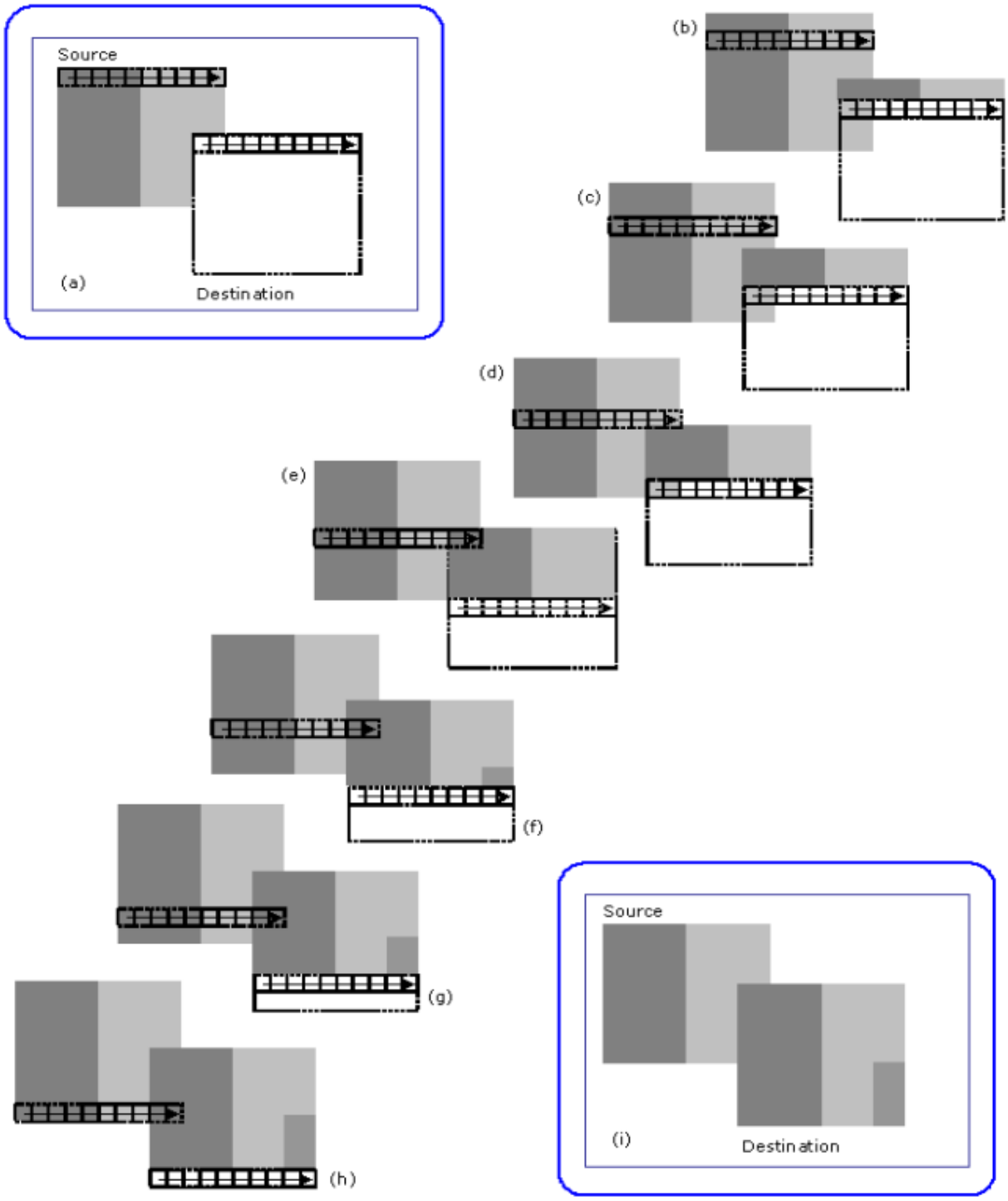
The 3-bit field, destination transparency mode, within the command packets can select per-pixel write-masking with a mask based on the results of color comparisons. The monochrome source background and foreground are range compared with either the bytes for the pixels at the destination or the source operand. This operation is described in the BLT command packet and register descriptions.

When the Source and Destination Locations Overlap

It is possible to have BLT operations in which the locations of the source and destination data overlap. This frequently occurs in BLT operations where a user is shifting the position of a graphical item on the display by only a few pixels. In these situations, the BLT engine must be programmed so that destination data is not written into destination locations that overlap with source locations before the source data at those locations has been read. Otherwise, the source data will become corrupted. The XY commands determine whether there is an overlap and perform the accesses in the proper direction to avoid data corruption.

The following figure shows how the source data can be corrupted when a rectangular block is copied from a source location to an overlapping destination location. The BLT engine typically reads from the source location and writes to the destination location starting with the left-most pixel in the top-most line of both, as shown in step (a). As shown in step (b), corruption of the source data has already started with the copying of the top-most line in step (a) -- part of the source that originally contained lighter-colored pixels has now been overwritten with darker-colored pixels. More source data corruption occurs as steps (b) through (d) are performed. At step (e), another line of the source data is read, but the two right-most pixels of this line are in the region where the source and destination locations overlap, and where the source has already been overwritten as a result of the copying of the top-most line in step (a). Starting in step (f), darker-colored pixels can be seen in the destination where lighter-colored pixels should be. This errant effect occurs repeatedly throughout the remaining steps in this BLT operation. As more lines are copied from the source location to the destination location, it becomes clear that the end result is not what was originally intended.

Source Corruption in BLT with Overlapping Source and Destination Locations

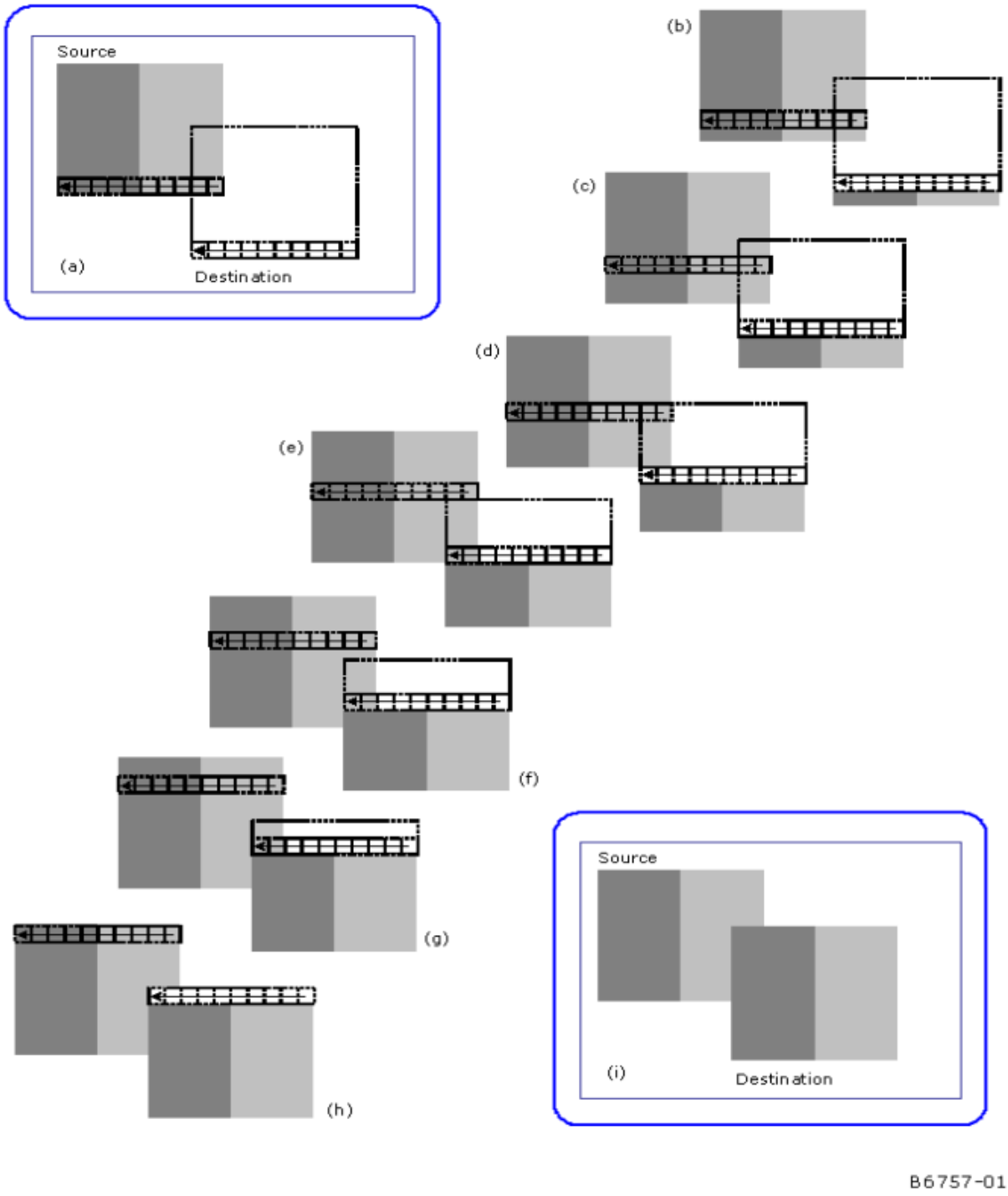


The BLT engine can alter the order in which source data is read and destination data is written when necessary to avoid source data corruption problems when the source and destination locations overlap. The command packets provide the ability to change the point at which the BLT engine begins reading

and writing data from the upper left-hand corner (the usual starting point) to one of the other three corners. The BLT engine may be set to read data from the source and write it to the destination starting at any of the four corners of the panel.

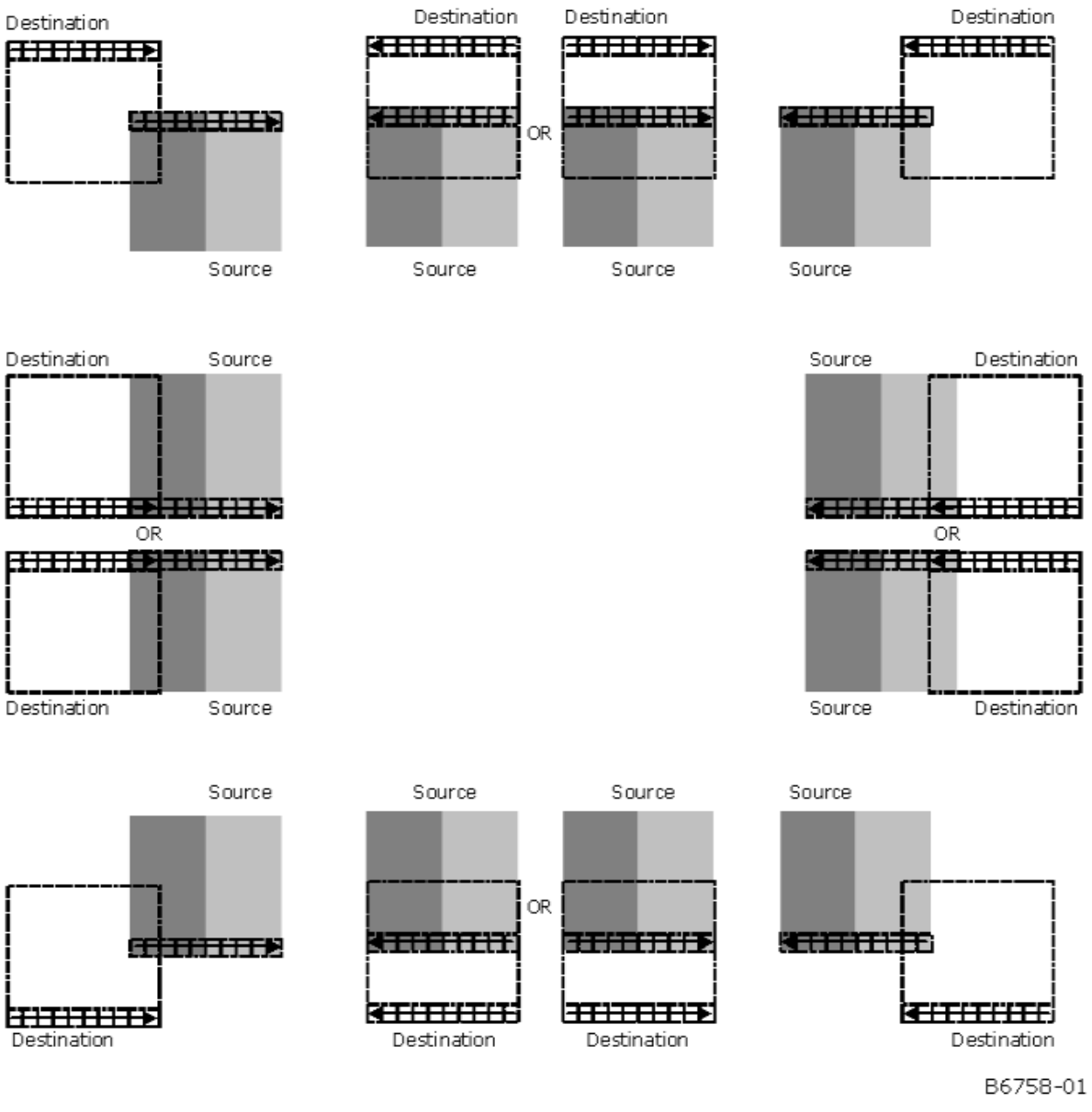
The XY command packets perform the necessary comparisons and start at the proper corner of each operand which avoids data corruption.

Correctly Performed BLT with Overlapping Source and Destination Locations



The following figure illustrates how this feature of the BLT engine can be used to perform the same BLT operation as was illustrated in the figure above, while avoiding the corruption of source data. As shown in the figure below, the BLT engine reads the source data and writes the data to the destination starting with the right-most pixel of the bottom-most line. By doing this, no pixel existing where the source and destination locations overlap will ever be written to before it is read from by the BLT engine. By the time the BLT operation has reached step (e) where two pixels existing where the source and destination locations overlap are about to be over written, the source data for those two pixels has already been read.

Suggested Starting Points for Possible Source and Destination Overlap Situations



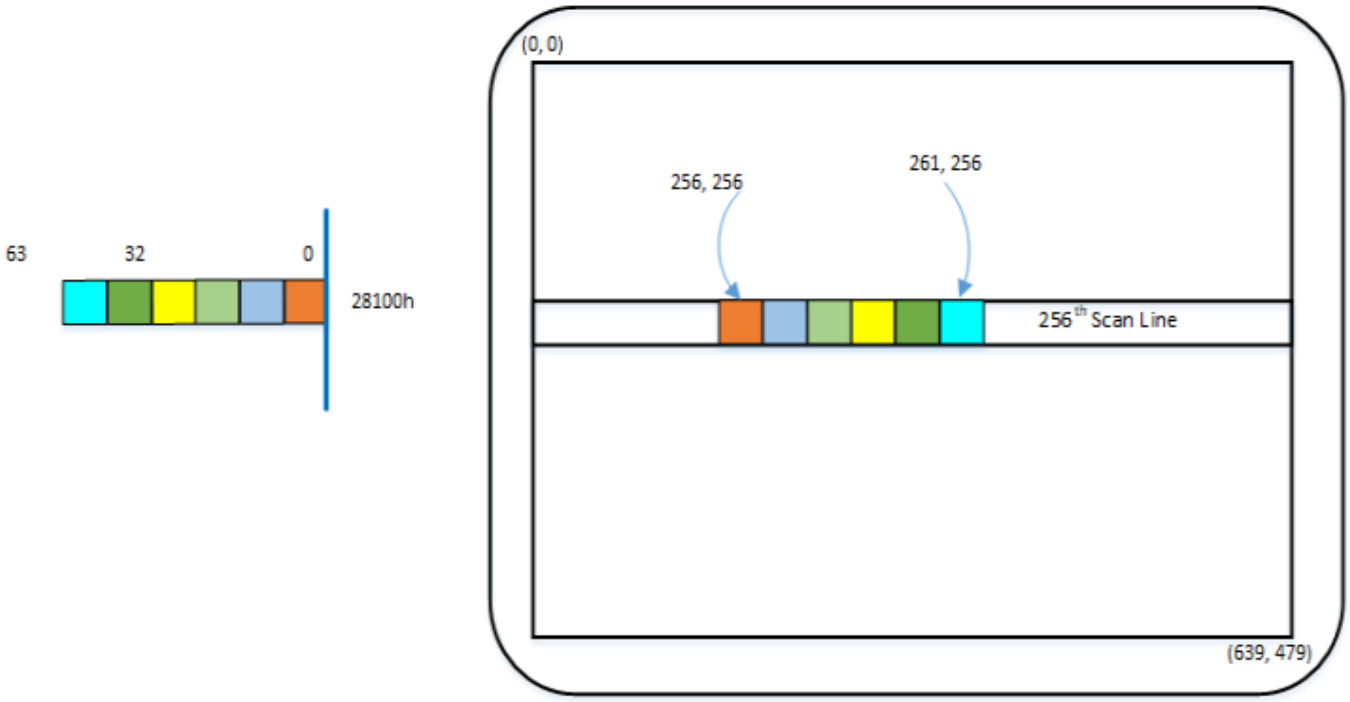
The figure above shows the recommended lines and pixels to be used as starting points in each of 8 possible ways in which the source and destination locations may overlap. In general, the starting point should be within the area in which the source and destination overlap.

Basic Graphics Data Considerations

Contiguous vs. Discontinuous Graphics Data

Graphics data stored in memory, particularly in the frame buffer of a graphics system, has organizational characteristics that often distinguish it from other varieties of data. The main distinctive feature is the tendency for graphics data to be organized in a discontinuous block of graphics data made up of multiple sub-blocks of bytes, instead of a single contiguous block of bytes.

Representation of On-Screen Single 6-Pixel Line in the Frame Buffer

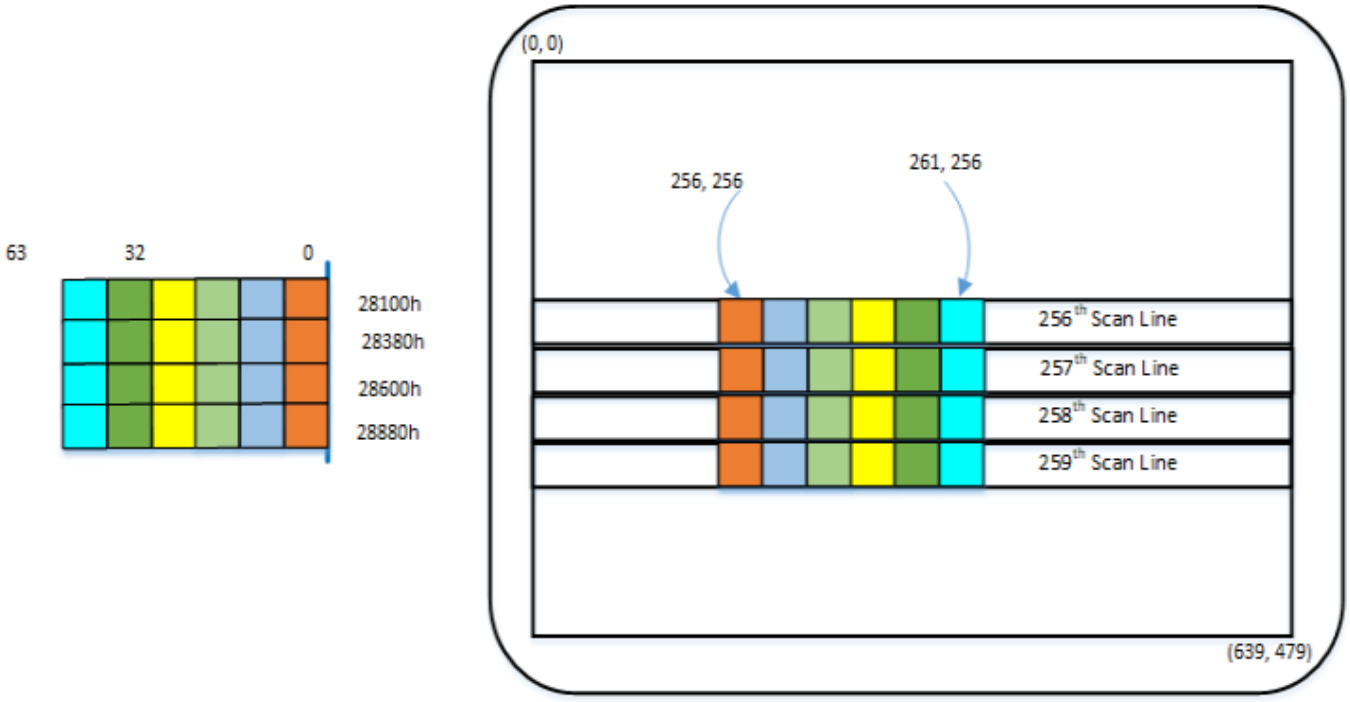


The figure above shows an example of contiguous graphics data -- a horizontal line made up of six adjacent pixels within a single scan line on a display with a resolution of 640x480. Presuming that the graphics system driving this display has been set to 8 bits per pixel and that the frame buffer's starting address of 0h corresponds to the upper left-most pixel of this display, then the six pixels that make this horizontal line starting at coordinates (256, 256) occupies the six bytes starting at frame buffer address 28100h, and ending at address 28105h.

In this case, there is only one scan line's worth of graphics data in this single horizontal line, so the block of graphics data for all six of these pixels exists as a single, contiguous block comprised of only these six bytes. The starting address and the number of bytes are the only pieces of information that a BLT engine would require to read this block of data.

The simplicity of the above example of a single horizontal line contrasts sharply to the example of discontinuous graphics data depicted in the figure below. The simple six-pixel line of the figure above is now accompanied by three more six-pixel lines placed on subsequent scan lines, resulting in the 6x4 block of pixels shown.

Representation of On-Screen 6x4 Array of Pixels in the Frame Buffer



Since there are other pixels on each of the scan lines on which this 6x4 block exists that are not part of this 6x4 block, what appears to be a single 6x4 block of pixels on the display must be represented by a discontinuous block of graphics data made up of 4 separate sub-blocks of six bytes apiece in the frame buffer at addresses 28100h, 28380h, 28600h, and 28880h. This situation makes the task of reading what appears to be a simple 6x4 block of pixels more complex. However, there are two characteristics of this 6x4 block of pixels that help simplify the task of specifying the locations of all 24 bytes of this discontinuous block of graphics data: all four of the sub-blocks are of the same length, and the four sub-blocks are separated from each other at equal intervals.

The BLT engine is designed to make use of these characteristics of graphics data to simplify the programming required to handle discontinuous blocks of graphics data. For such a situation, the BLT engine requires only four pieces of information: the starting address of the first sub-block, the length of a sub-block, the offset (in bytes), pitch, of the starting address of each subsequent sub-block, and the quantity of sub-blocks.

Source Data

The source data may exist in the frame buffer or elsewhere in the graphics aperture where the BLT engine may read it directly, or it may be provided to the BLT engine by the host CPU through the command packets. The block of source graphics data may be either contiguous or discontinuous, and may be either in color (with a color depth that matches that to which the BLT engine has been set) or monochrome.

The source select bit in the command packets specifies whether the source data exists in the frame buffer or is provided through the command packets. Monochrome source data is always specified as being supplied through an immediate command packet.

If the color source data resides within the frame buffer or elsewhere in the graphics aperture, then the Source Address Register, specified in the command packets is used to specify the address of the source.

In cases where the host CPU provides the source data, it does so by writing the source data to ring buffer directly after the BLT command that requires the data or uses an IMMEDIATE_INDIRECT_BLT command packet which has a size and pointer to the operand in Graphics aperture.

The block of bytes sent by the host CPU through the command packets must be quadword-aligned and the source data contained within the block of bytes must also be aligned.

To accommodate discontinuous source data, the source and destination pitch registers can be used to specify the offset in bytes from the beginning of one scan line's worth source data to the next. Otherwise, if the source data is contiguous, then an offset equal to the length of a scan line's worth of source data should be specified.

Monochrome Source Data

The opcode of the command packet specifies whether the source data is color or monochrome. Since monochrome graphics data only uses one bit per pixel, each byte of monochrome source data typically carries data for 8 pixels which hinders the use of byte-oriented parameters when specifying the location and size of valid source data. Some additional parameters must be specified to ensure the proper reading and use of monochrome source data by the BLT engine. The BLT engine also provides additional options for the manipulation of monochrome source data versus color source data.

The various bit-wise logical operations and per-pixel write-masking operations were designed to work with color data. In order to use monochrome data, the BLT engine converts it into color through a process called color expansion, which takes place as a BLT operation is performed. In color expansion the single bits of monochrome source data are converted into one, two, or four bytes (depending on the color depth) of color data that are set to carry value corresponding to either the foreground or background color that have been specified for use in this conversion process. If a given bit of monochrome source data carries a value of 1, then the byte(s) of color data resulting from the conversion process will be set to carry the value of the foreground color. If a given bit of monochrome source data carries a value of 0, then the resulting byte(s) will be set to the value of the background color. The foreground and background colors used in the color expansion of monochrome source data can be set in the source expansion foreground color register and the source expansion background color register.

The BLT Engine requires that the bit alignment of each scan line's worth of monochrome source data be specified. Each scan line's worth of monochrome source data is word aligned but can actually start on any bit boundary of the first byte. Monochrome text is special cased, and it is bit or byte packed, where in bit packed there are no invalid pixels (bits) between scan lines. There is a 3 bit field which indicates the starting pixel position within the first byte for each scan line, Mono Source Start.

Note that the Monosource surface start Base Address, should always be Cache Line (64byte) aligned.

The BLT engine also provides various clipping options for use with specific BLT commands (BLT_TEXT) with a monochrome source. Clipping is supported through: Clip rectangle Y addresses or coordinates and X coordinates along with scan line starting and ending addresses (with Y addresses) along with X starting and ending coordinates.

The maximum immediate source size is 128 bytes.

Pattern Data

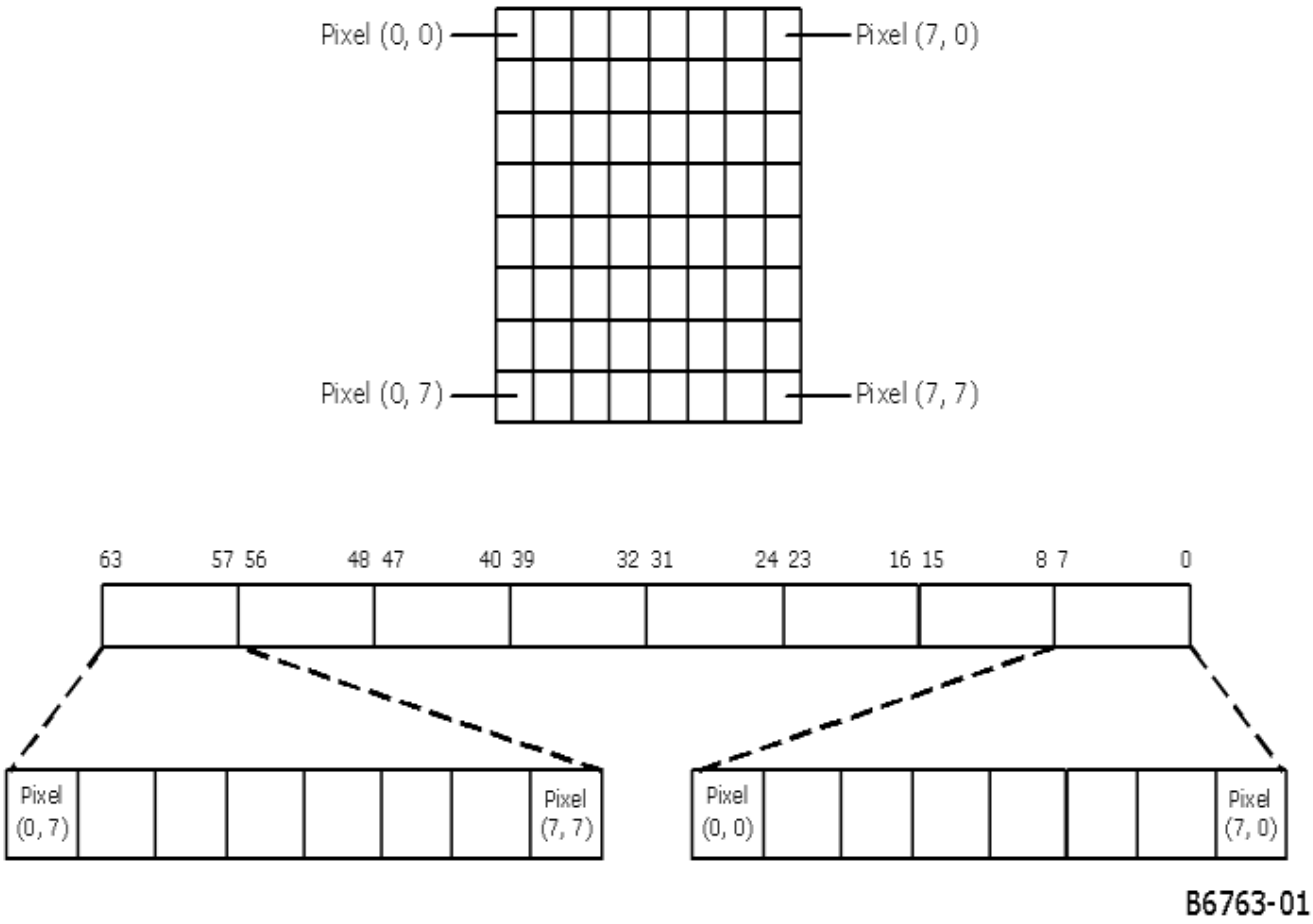
The color pattern data must exist within the frame buffer or Graphics aperture where the BLT engine may read it directly or it can be sent through the command stream. The pattern data must be located in linear memory.

Note also that the Color Pattern surface start Base Address, should always be Cache Line (64byte) aligned.

Monochrome pattern data is supplied by the command packet when it is to be used. As shown in figure below, the block of pattern graphics data always represents a block of 8x8 pixels. The bits or bytes of a block of pattern data may be organized in the frame buffer memory in only one of three ways, depending upon its color depth which may be 8, 16, or 32 bits per pixel (whichever matches the color depth to which the BLT engine has been set), or monochrome.

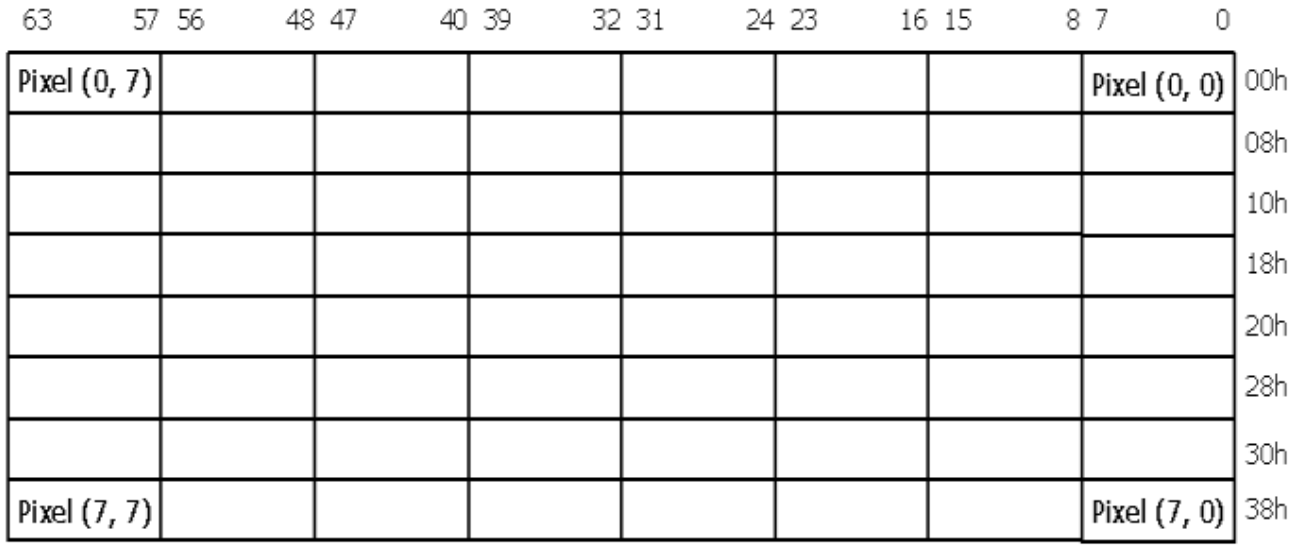
The maximum color pattern size is 256 bytes.

Pattern Data -- Always an 8x8 Array of Pixels



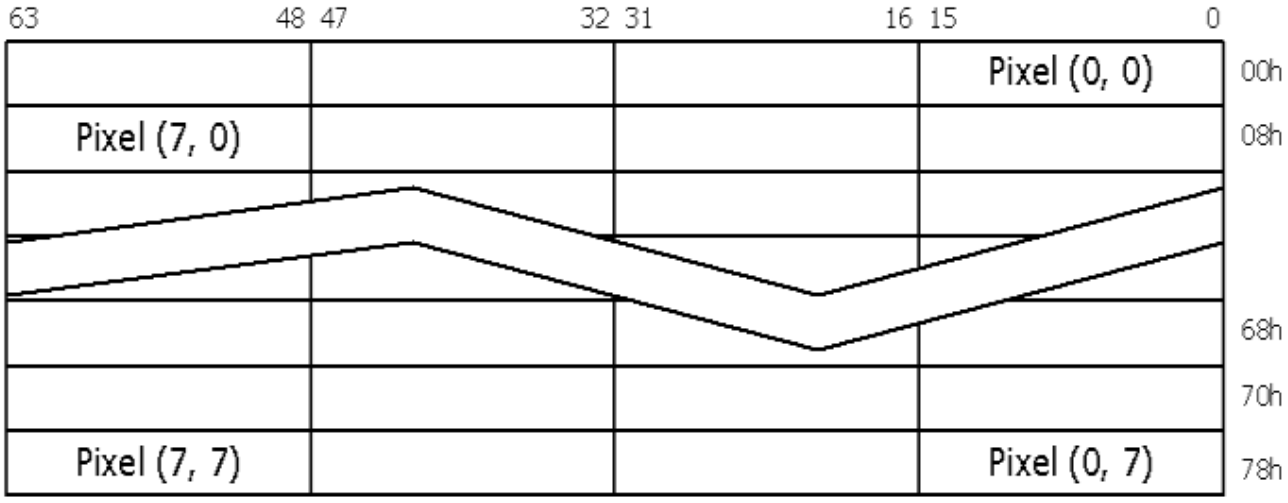
The Pattern Address Register is used to specify the address of the color pattern data at which the block of pattern data begins. The three least significant bits of the address written to this register are ignored, because the address must be in terms of quadwords. This is because the pattern must always be located on an address boundary equal to its size. Monochrome patterns take up 8 bytes, or a single quadword of space, and are loaded through the command packet that uses it. Similarly, color patterns with color depths of 8, 16, and 32 bits per pixel must start on 64-byte, 128-byte and 256-byte boundaries, respectively. The next 3 figures show how monochrome, 8bpp, 16bpp, and 32bpp pattern data , respectively, is organized in memory.

8bpp Pattern Data -- Occupies 64 Bytes (8 quadwords)



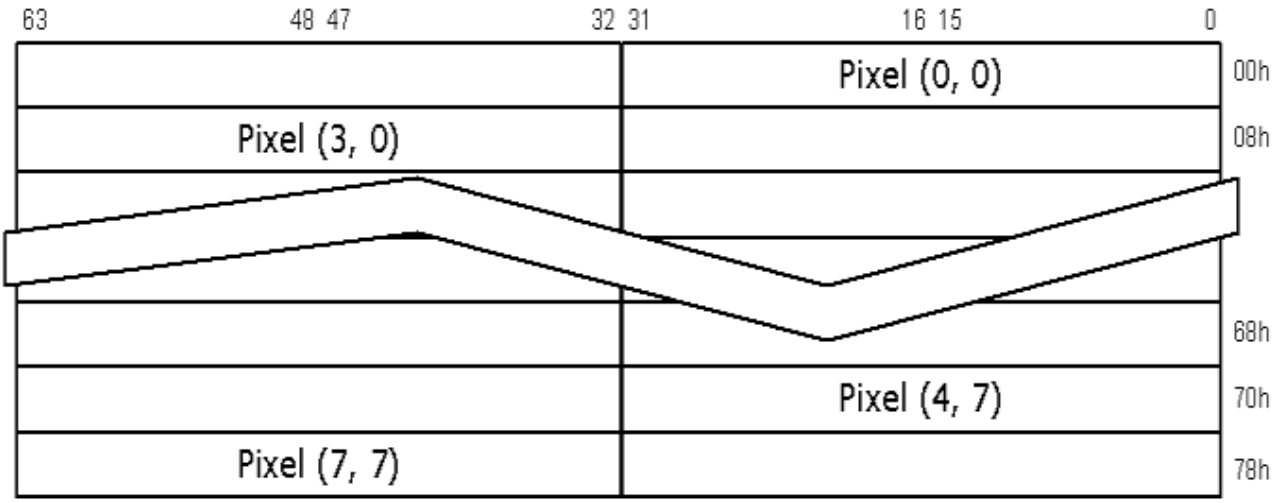
B6764-01

16bpp Pattern Data -- Occupies 128 Bytes (16 quadwords)



B6765-01

32bpp Pattern Data -- Occupies 256 Bytes (32 quadwords)



The opcode of the command packet specifies whether the pattern data is color or monochrome. The various bit-wise logical operations and per-pixel write-masking operations were designed to work with color data. In order to use monochrome pattern data, the BLT engine is designed to convert it into color through a process called "color expansion" which takes place as a BLT operation is performed. In color expansion, the single bits of monochrome pattern data are converted into one, two, or four bytes (depending on the color depth) of color data that are set to carry values corresponding to either the foreground or background color that have been specified for use in this process. The foreground color is used for pixels corresponding to a bit of monochrome pattern data that carry the value of 1, while the background color is used where the corresponding bit of monochrome pattern data carries the value of 0. The foreground and background colors used in the color expansion of monochrome pattern data can be set in the Pattern Expansion Foreground Color Register and Pattern Expansion Background Color Register.

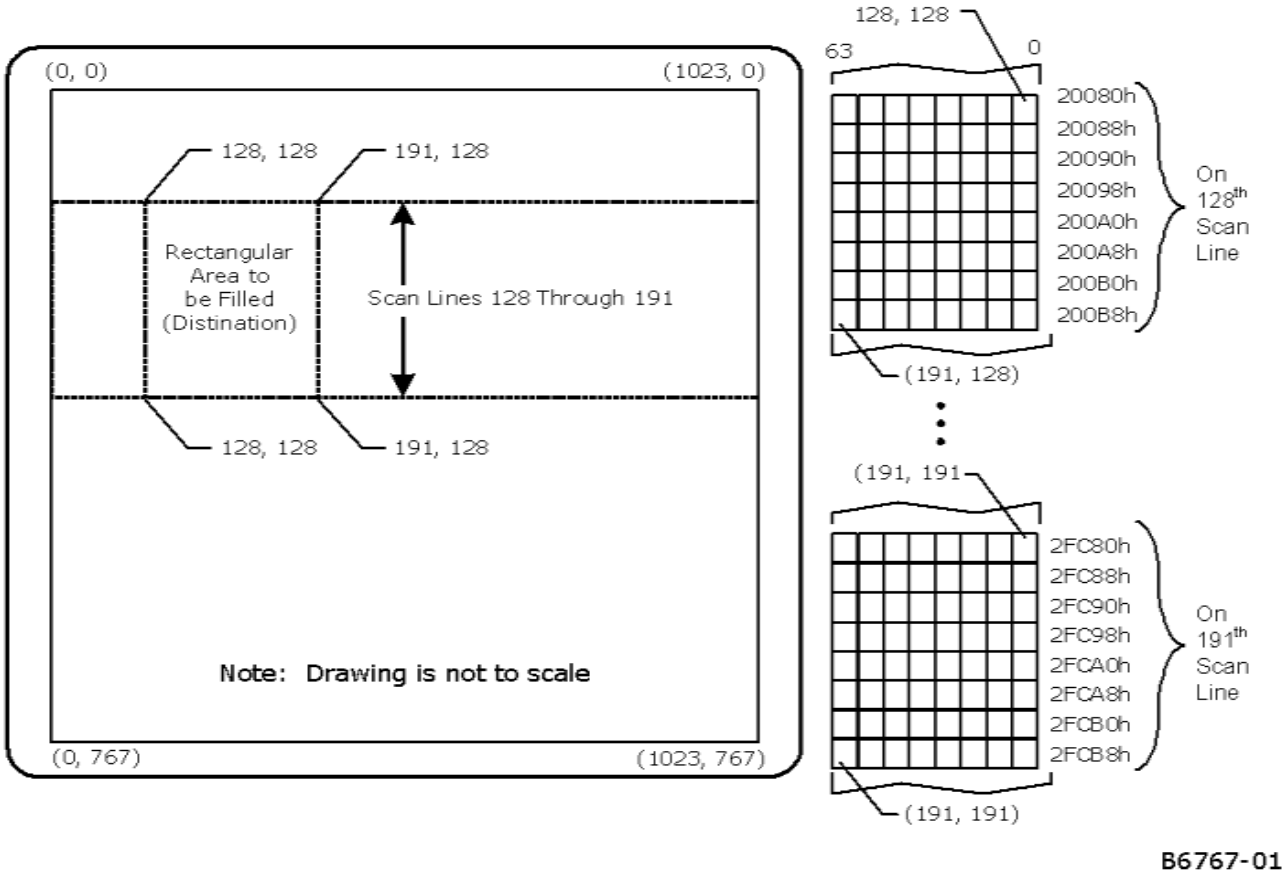
Destination Data

BLT Programming Examples

Pattern Fill -- A Very Simple BLT

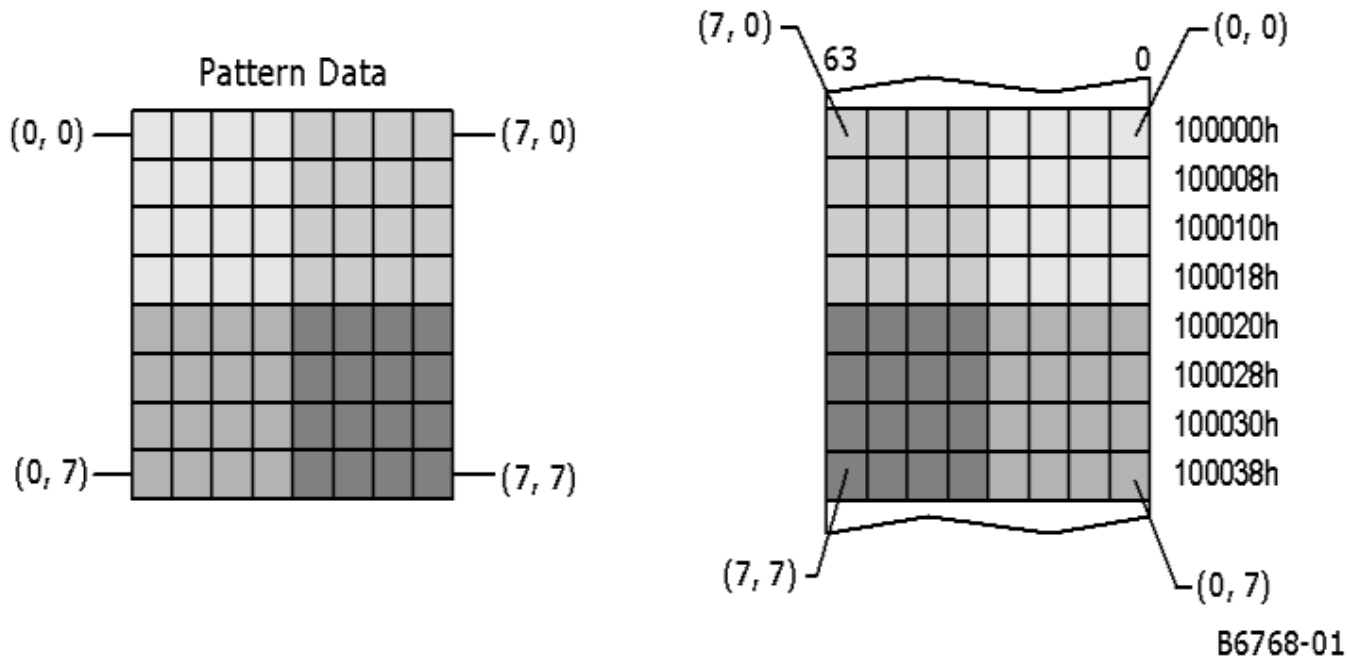
In this example, a rectangular area on the screen is to be filled with a color pattern stored as pattern data in off-screen memory. The screen has a resolution of 1024x768 and the graphics system has been set to a color depth of 8 bits per pixel.

On-Screen Destination for Example Pattern Fill BLT



As shown in the figure above, the rectangular area to be filled has its upper left-hand corner at coordinates (128, 128) and its lower right-hand corner at coordinates (191, 191). These coordinates define a rectangle covering 64 scan lines, each scan line's worth of which is 64 pixels in length -- in other words, an array of 64x64 pixels. Presuming that the pixel at coordinates (0, 0) corresponds to the byte at address 00h in the frame buffer memory, the pixel at (128, 128) corresponds to the byte at address 20080h.

Pattern Data for Example Pattern Fill BLT



As shown in figure above, the pattern data occupies 64 bytes starting at address 100000h. As always, the pattern data represents an 8x8 array of pixels.

The BLT command packet is used to select the features to be used in this BLT operation, and must be programmed carefully. The vertical alignment field should be set to 0 to select the top-most horizontal row of the pattern as the starting row used in drawing the pattern starting with the top-most scan line covered by the destination. The pattern data is in color with a color depth of 8 bits per pixel, so the dynamic color enable should be asserted with the dynamic color depth field should be set to 0. Since this BLT operation does not use per-pixel write-masking (destination transparency mode), this field should be set to 0. Finally, the raster operation field should be programmed with the 8-bit value of F0h to select the bit-wise logical operation in which a simple copy of the pattern data to the destination takes place. Selecting this bit-wise operation in which no source data is used as an input causes the BLT engine to automatically forego either reading source data from the frame buffer.

The Destination Pitch Register must be programmed with number of bytes in the interval from the start of one scan line's worth of destination data to the next. Since the color depth is 8 bits per pixel and the horizontal resolution of the display is 1024, the value to be programmed into these bits is 400h, which is equal to the decimal value of 1024.

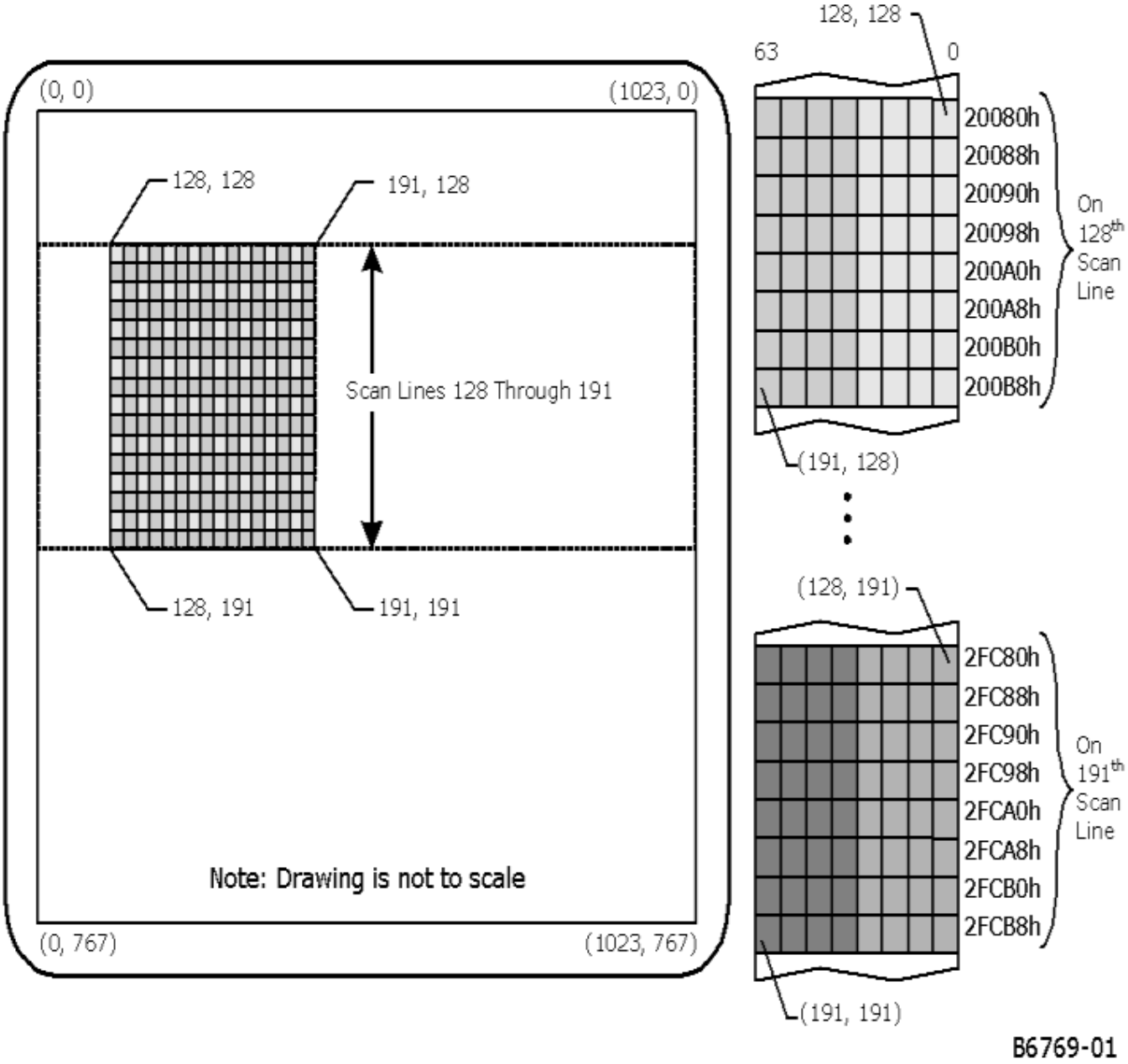
Bits [31:3] of the Pattern Address Register must be programmed with the address of the pattern data.

Similarly, bits [31:0] of the Destination Address Register must be programmed with the byte address at the destination that will be written to first. In this case, the address is 20080h, which corresponds to the byte representing the pixel at coordinates (128, 128).

This BLT operation does not use the values in the Source Address Register or the Source Expansion Background or Foreground Color Registers.

The Destination Width and Height Registers (or the Destination X and Y Coordinates) must be programmed with values that describe to the BLT engine the 64x64 pixel size of the destination location. The height should be set to carry the value of 40h, indicating that the destination location covers 64 scan lines. The width should be set to carry the value of 40h, indicating that each scan line's worth of destination data occupies 64 bytes. All of this information is written to the ring buffer using the PAT_BLT (or XY_PAT_BLT) command packet.

Results of Example Pattern Fill BLT

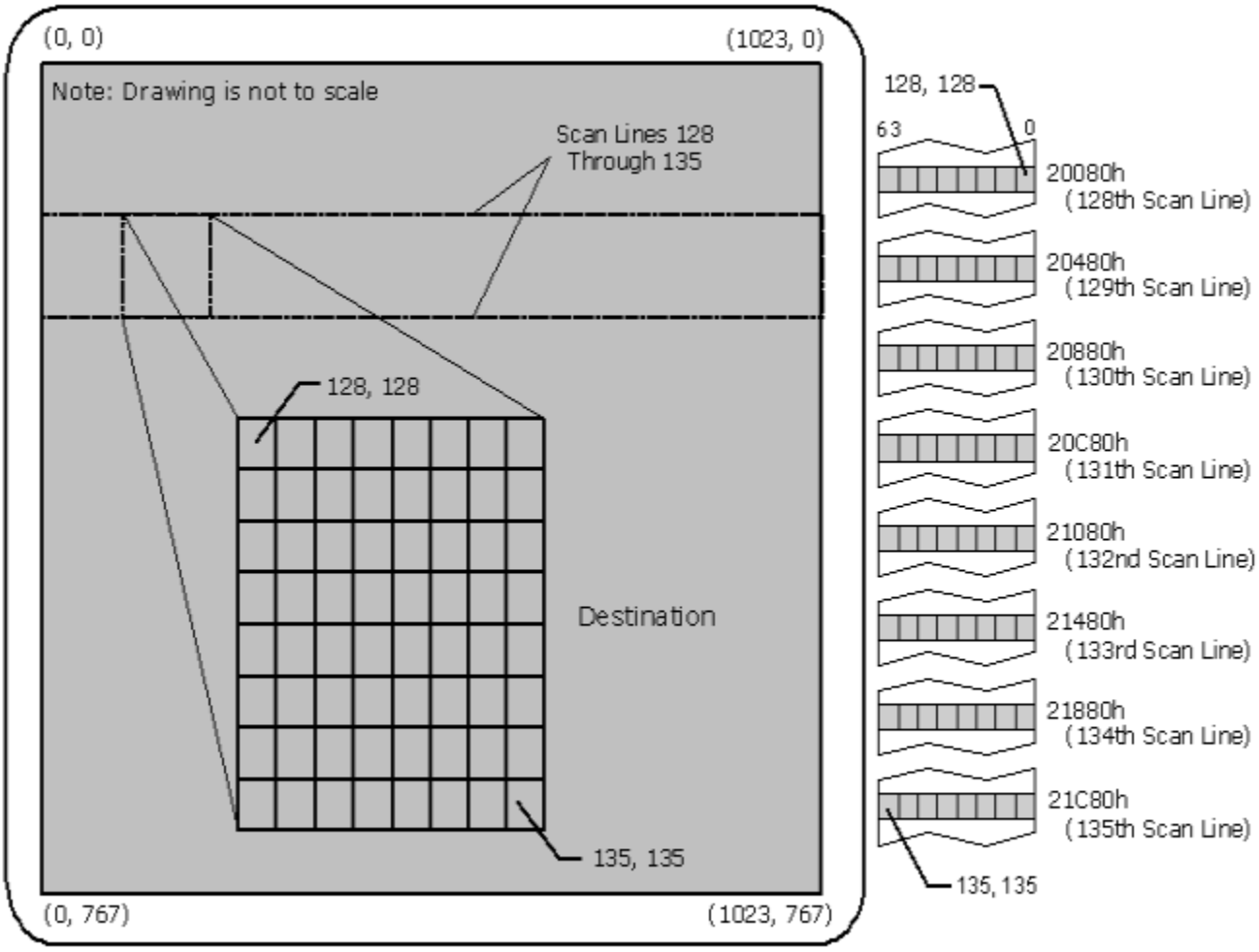


The figure above shows the end result of performing this BLT operation. The 8x8 pattern has been repeatedly copied ("tiled") into the entire 64x64 area at the destination.

Drawing Characters Using a Font Stored in System Memory

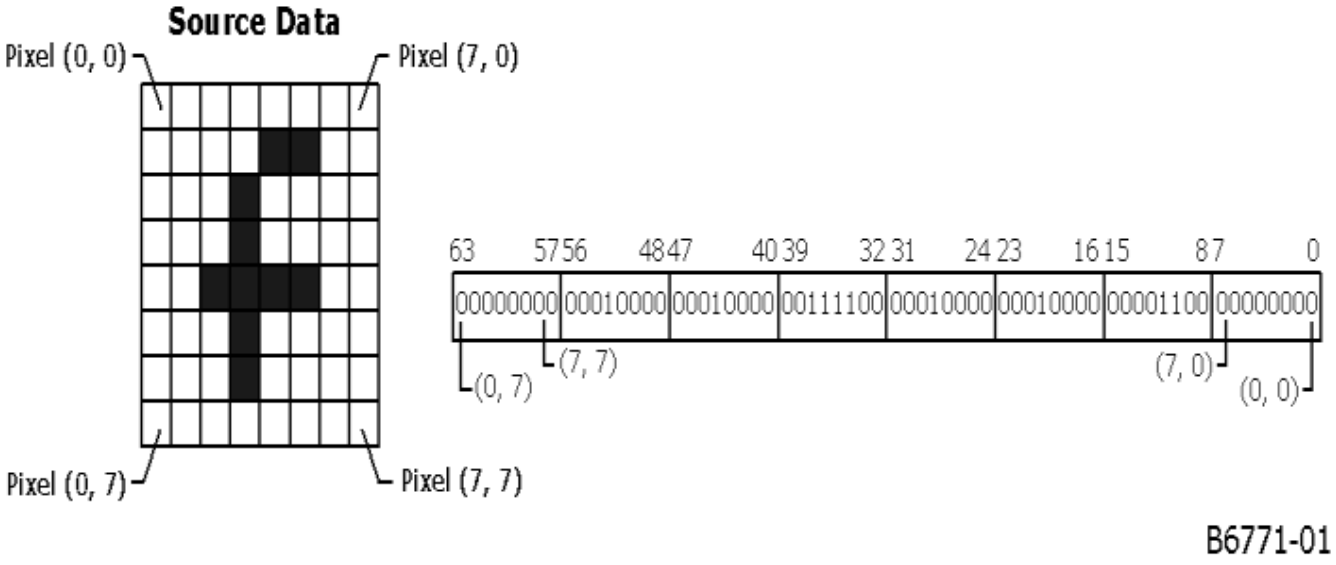
In this example BLT operation, a lowercase letter "f" is to be drawn in black on a display with a gray background. The resolution of the display is 1024x768, and the graphics system has been set to a color depth of 8 bits per pixel.

On-Screen Destination for Example Character Drawing BLT



The figure above shows the display on which this letter "f" is to be drawn. As shown in this figure, the entire display has been filled with a gray color. The letter "f" is to be drawn into an 8x8 region on the display with the upper left-hand corner at the coordinates (128, 128).

Source Data in System Memory for Example Character Drawing BLT



The figure above shows both the 8x8 pattern making up the letter "f" and how it is represented somewhere in the host's system memory -- the actual address in system memory is not important. The letter "f" is represented in system memory by a block of monochrome graphics data that occupies 8 bytes. Each byte carries the 8 bits needed to represent the 8 pixels in each scan line's worth of this graphics data. This type of pattern is often used to store character fonts in system memory.

During this BLT operation, the host CPU will read this representation of the letter "f" from system memory, and write it to the BLT engine by performing memory writes to the ring buffer as an immediate monochrome BLT operand following the BLT_TEXT command. The BLT engine will receive this data through the command stream and use it as the source data for this BLT operation. The BLT engine will be set to the same color depth as the graphics system -- 8 bits per pixel, in this case. Since the source data in this BLT operation is monochrome, color expansion must be used to convert it to an 8 bpp color depth. To ensure that the gray background behind this letter "f" is preserved, per-pixel write masking will be performed, using the monochrome source data as the pixel mask.

The BLT Setup and Text_immediate command packets are used to select the features to be used in this BLT operation. Only the fields required by these two command packets must be programmed carefully. The BLT engine ignores all other registers and fields. The source select field in the Text_immediate command must be set to 1, to indicate that the source data is provided by the host CPU through the command packet. Finally, the raster operation field should be programmed with the 8-bit value CCh to select the bit-wise logical operation that simply copies the source data to the destination. Selecting this bit-wise operation in which no pattern data is used as an input, causes the BLT engine to automatically forego reading pattern data from the frame buffer.

The Setup Pattern/Source Expansion Foreground Color Register to specify the color with which the letter "f" will be drawn. There is no Source address. All scan lines of the glyph are bit packed and the clipping is controlled by the ClipRect registers from the SETUP_BLT command and the Destination Y1, Y2, X1, and

X2 registers in the TEXT_BLT command. Only the pixels that are within (inclusive comparisons) the clip rectangle are written to the destination surface.

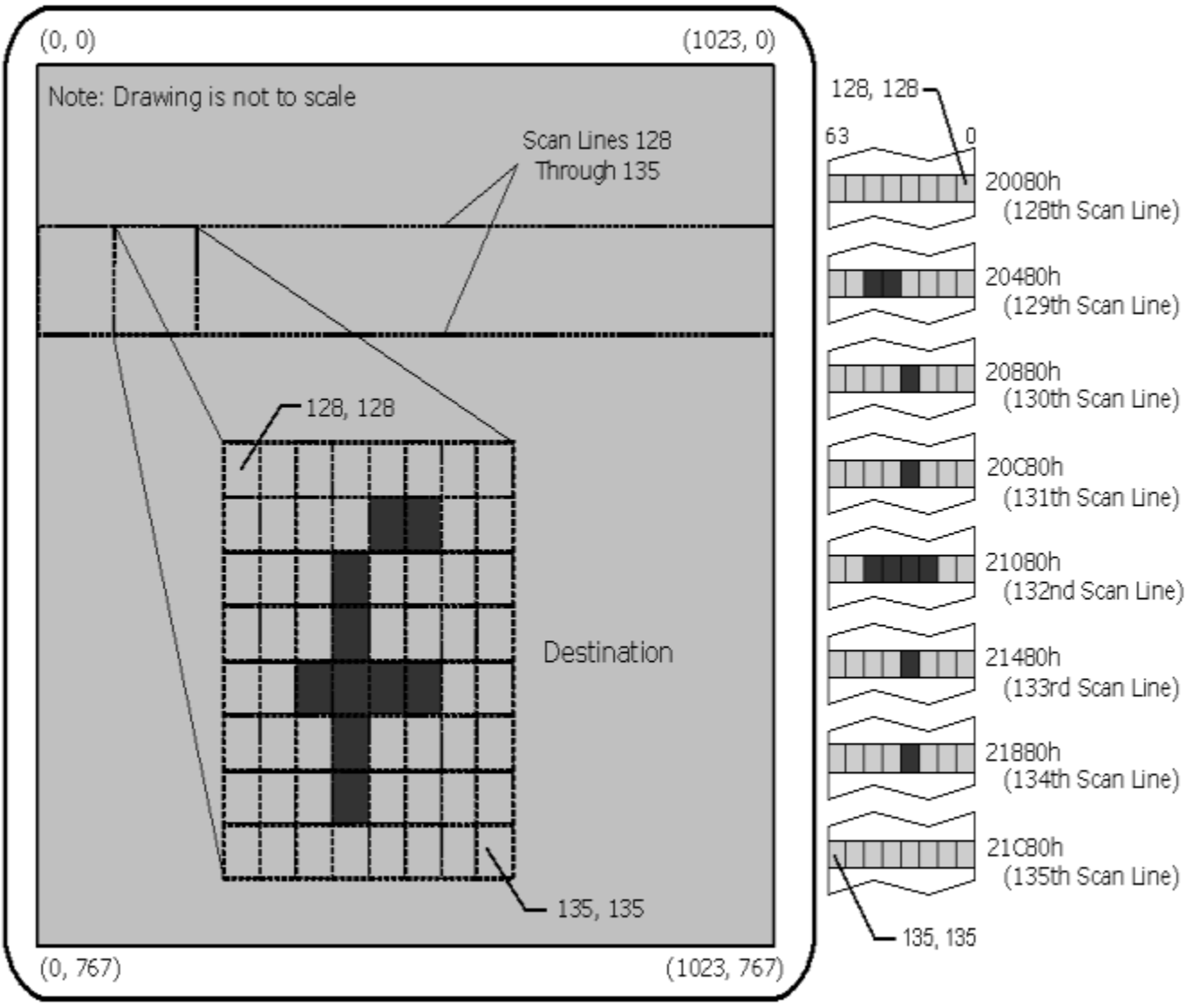
The Destination Pitch Register must be programmed with a value equal to the number of bytes in the interval between the first bytes of each adjacent scan line's worth of destination data. Since the color depth is 8 bits per pixel and the horizontal resolution of the display is 1024 pixels, the value to be programmed into these bits is 400h, which is equal to the decimal value of 1024. Since the source data used in this BLT operation is monochrome, the BLT engine will not use a byte-oriented pitch value for the source data.

Since the source data is monochrome, color expansion is required to convert it to color with a color depth of 8 bits per pixel. Since the Setup Pattern/Source Expansion Foreground Color Register is selected to specify the foreground color of black to be used in drawing the letter "f", this register must be programmed with the value for that color. With the graphics system set for a color depth of 8 bits per pixel, the actual colors are specified in the RAMDAC palette, and the 8 bits stored in the frame buffer for each pixel actually specify the index used to select a color from that palette. This example assumes that the color specified at index 00h in the palette is black, and therefore bits [7:0] of this register should be set to 00h to select black as the foreground color. The BLT engine ignores bits [31:8] of this register because the selected color depth is 8 bits per pixel. Even though the color expansion being performed on the source data normally requires that both the foreground and background colors be specified, the value used to specify the background color is not important in this example. Per-pixel write-masking is being performed with the monochrome source data as the pixel mask, which means that none of the pixels in the source data that will be converted to the background color will ever be written to the destination. Since these pixels will never be seen, the value programmed into the Pattern/Source Expansion Background Color Register to specify a background color is not important.

The Destination Width and Height Registers are not used. The Y1, Y2, X1, and X2 are used to describe to the BLT engine the 8x8 pixel size of the destination location. The Destination Y1 and Y2 address (or coordinate) registers must be programmed with the starting and ending scan line address (or Y coordinates) of the destination data. This address is specified as an offset from the start of the frame buffer of the scan line at the destination that will be written to first. The destination X1 and X2 registers must be programmed with the starting and ending pixel offsets from the beginning of the scan line.

This BLT operation does not use the values in the Pattern Address Register, the Source Expansion Background Color Register, or the Source Expansion Foreground Color Register.

Results of Example Character Drawing BLT



The preceding shows the end result of performing this BLT operation. Only the pixels that form part of the actual letter "f" have been drawn into the 8x8 destination location on the display, leaving the other pixels within the destination with their original gray color.

Fast Copy Engine

Fast Copy Engine is capable of performing sub-resource copy or fill operation saturating the full memory bandwidth. It is a resource processor that can copy from a source of arbitrary format. Fast Copy Engine is capable of understanding the concept of resource and sub-resources. It can support data transfer between overlapping surfaces. It can support state-full compression and decompression. Copy Engine supports both linear as well as tiled memory and operate on both system as well as local memory (when it is not a UMA system). In case of UMA system there is just a single memory.

Fast Copy Engine can be used for paging in or paging out resources to and from GPU. It also supports Fast Clear and regular clear operation of AUX enabled surfaces. It can be used to perform resolve operation on compressed surfaces. Copy Engines can operate on 2D surfaces defined using pixel coordinates as well as linear surfaces defined using virtual addresses.

Fast Copy Engine performance scales with number of memory slices in the system.

Copy workloads are submitted to the appropriate copy engines by the software through copy queues. BLT command streamers parses the commands and submits them to Fast Copy Engine. Copy Engines are split into two parts:

1. Copy Engine Front-end - splits an incoming command/BLIT into smaller commands/sub-blits
2. Copy Engine Back-end - it takes sub-blits and performs actual cachline read/write operations

Performance scaling of the Fast Copy Engine is achieved by increasing the number of copies of Back-end (Sub-copy Engine) attached to the single front-end of the fast copy engine. Fast copy engine can handle sub-blit level dependencies (producer consumer relationship). Software may have to insert appropriate flushes between commands to handle some dependencies not supported by copy engine hardware, as defined at a later section.

BLT Instruction Overview

There are two blitter back-ends behind the single engine viz. Classical Blitter or Legacy Blitter Engine and Fast Copy Engine. Each of which operates on a separate mutually exclusive set of instructions. Commands for both the back-ends can be submitted to the same Blitter Command Streamer and based on the command it is passed on to the appropriate back-end and executed there.

Each instruction consists of multiple dwords, where first dword is the header, which can identify the type of instruction, instruction opcode, length of instruction etc. Header may contain additional instruction fields as well. Fields may sometime need more than one dword.

Most of the instructions can operate independently without state other than some Legacy Blitter commands involving clip rectangles that programs some BRs (Blitter Registers) before executing those commands.

There are specific MMIO based config registers that configures how Legacy Blitter or Fast Copy Engine is expected to operate.

Legacy Blitter uses cache, so unless a flush is inserted after an instruction there is no guarantee of final data reaching the memory.

The flush is common for both the backends. Once a flush is sent command streamer stops sending further commands to the back-ends and back-ends ensure that when the current commands being processed by the back-ends are completed flush dones are sent to blitter command streamer.

The actual commands have been listed out in a later section.

BLT Engine State

Most of the BLT instructions are state-free, which means that all states required to execute the command is within the instruction. If clipping is not used, then there is no shared state for many of the BLT instructions. This allows the BLT Engine to be shared by many drivers with minimal synchronization between the drivers.

Instructions which share state are:

All instructions that are X,Y commands and use the Clipping Rectangle by asserting the Clip Enable field

All XY_Setup Commands (XY_SETUP_BLT and XY_SETUP_MONO_PATTERN_SL_BLT, XY_SETUP_CLIP_BLT) load the shared state for the following commands:

XY_PIXEL_BLT	(Negative Stride (=Pitch) Not Allowed)
XY_SCANLINES_BLT	
XY_TEXT_BLT	(Negative Stride (=Pitch) Not Allowed)
XY_TEXT_IMMEDIATE_BLT	(Negative Stride (=Pitch) Not Allowed)

State registers that are saved & restored in the Logical Context:

BR1+ Setup Control (Solid Pattern Select, Clipping Enable, Mono Source Transparency Mode, Mono Pattern Transparency Mode, Color Depth[1:0], Raster Operation[7:0], & Destination Pitch[15:0]) + 32bpp Channel

Mask[1:0], Mono / Color Pattern

BR05 Setup Background Color

BR06 Setup Foreground Color

BR07 Setup Pattern Base Address

BR09 Setup Destination Base Address

BR20 DW0 for a Monochrome Pattern

BR21 DW1 for a Monochrome Pattern

BR24 ClipRectY1'X1

BR25 ClipRectY2'X2

The definition of each of the above registers which are part of logical context and are saved and restored during context save and restore are found in "BLT Engine Instruction Field Definitions" section.

Device Cache Coherency: Render & Texture Caches

Software must initiate cache flushes to enforce coherency between the render and texture caches, i.e., both the render and texture caches must be flushed before a BLT destination surface can be reused as a texture source. Color sources and destinations use the render cache, while patterns and monochrome sources use the texture cache.

Verbatim Copy operation

Verbatim Copy is the special copy operation defined for compressed surface copy where both the main surface data and the associated CCS data are moved as it is from one location to another. During this operation main surface data copy and CCS data copy are performed sequentially. Driver needs to use two separate commands, XY_FAST_COPY_BLT to copy the main surface data and XY_CTRL_SURF_COPY_BLT to copy the CCS surface, together for verbatim copy. In case of regular compressed data access from Copy Engine, read data from source is always made available to copy engine as fully decompressed data and Copy Engine writes fully decompressed data back to memory and if destination compression is required this data is again compressed and written to memory. The compressed write causes both CCS and main surface write data to be modified. In case of verbatim copy "data read" operation from source the data does not decompressed before sending to Copy Engine. Driver needs to set "Enable Compressed Surface Read" bit in the XY_FAST_COPY_BLT command to enable this special operation. In case of verbatim copy "data write" operation to destination surface this "raw" compressed data is sent out and this flow can be enabled by driver by setting the "Enable Compressed Surface Write" bit in the XY_FAST_COPY_BLT command.

These two commands must be used in a pair and the sequence in which the two commands must be used during verbatim copy operation varies based on whether it is being used for swap-in (moving data from local memory to system memory) or swap-out (moving data from system memory to local memory) operation for compressed data. In case of swap-out operation the sequence required is XY_FAST_COPY_BLT followed by XY_CTRL_SURF_COPY_BLT. In case of swap-in operation the sequence required is XY_CTRL_SURF_COPY_BLT followed by XY_FAST_COPY_BLT.

During verbatim copy the copy rectangles used for XY_CTRL_SURF_COPY_BLT and XY_FAST_COPY_BLT must be same. They must also use same BPP and pitch. Using base address X1, X2, Y1, Y2 from XY_FAST_COPY_BLT command start address and range is calculated.

BLT Engine Instructions

The Instruction Target field is used as an opcode by the BLT Engine state machine to qualify the control bits that are relevant for executing the instruction. The descriptions for each DWord and bit field are contained in the *BLT Engine Instruction Field Definition* section. Each DWord field is described as a register, but none of these registers can be written or read through a memory mapped location; they are internal state only.

BLT Programming Restrictions

Overlapping Source/Destination BLTs:

For all products *negative pitch* programming is allowed only when the source and destination surfaces are of the same type: linear source to linear destination copy, or tiled source to tiled destination copy. This is a must requirement. In such cases:

- Both the pitches must be programmed to be a negative value, if the source and destination surfaces are overlapping.
- Either of the pitches can be programmed to a negative value if required to do mirroring, but only if the complete source and destination surfaces are not overlapping anywhere on the surfaces.

Description
For the XY_FAST_COPY_BLT and XY_BLOCK_COPY_BLT instruction, these restrictions apply: When two sequential fast copy blits have different source surfaces, but their destinations refer to the same destination surface and therefore destinations overlap, a Flush must be inserted between the two blits. For two sequential fast copy blits when the source of the second blit is the destination of the first blit or they overlap a Flush must be inserted between the two blits.
For Tiled surfaces, the pitch length is always Cacheline aligned (64-byte multiple), as Tile surface pitches must be a multiple of Tile widths, which are always cacheline aligned. For XY_FAST_COPY_BLT command (X1, X2) and (Y1, Y2) must be programmed different values so that start pixel/end pixel and start line/end line do not coincide.

Legacy Blits:

The following condition must be avoided when programming the BLT engine: Linear surfaces with a cache line in scan line Y for the source stream overlapping with a cache line in scan line Y-1 for the dest stream (=> non-aligned surface pitches). The cache coherency rules combined with the Blitter data consumption rules result in UNDEFINED operation.

All reserved fields must be programmed to 0s.

When using monosource or text data (bit/byte/word aligned): Do not program pixel widths greater than 32,745 pixels.

The other way to do this is driver should always program a dummy 3D.

NON-PIPELINE state following the BLT commands:

Description
For Monosource and Color Pattern surfaces, and also linear color source and destination surfaces, the start **Base Address** programmed should always be Cache Line (64 byte) aligned.

Programming Note	
Context:	Fast Copy Blitter + Frame Buffer Compression
Fast Copy, Block Copy and Fast Color workloads targeting the Front Buffer with FBC enabled is not supported.	

Programming Note	
Context:	XY_FAST_COPY_BLT command programming restriction
XY_FAST_COPY_BLT command's (X1, Y1), (X2, Y2) must be programmed such way so that neither W nor H of the surface to copy are "0".	

Programming Note	
Context:	Blitter
Support for Tile-Yf format has been removed, so Blitter should not be programmed to support Tile-Yf for source or destination tiling.	

Programming Note	
Project:	
Context:	Blitter (XY_BLOCK_COPY_BLT, XY_FAST_COLOR_BLT commands)
X offset, Y offset and (X1, Y1), (X2, Y2) values must be programmed so that X coordinate values when added to X offset remains within the surface boundary (16384 pixels). Similarly Y coordinate values when added to Y offset must remain within the surface boundary (16384 lines). Copy operation between two overlapping surfaces are supported only when the base address and pitch of both the source and destination surfaces are identical, irrespective of the surface formats.	

2D (XY) BLT Instructions

Most BLT instructions (prefixed with "XY_") use 2D X,Y coordinate specifications vs. lower-level linear addresses These instructions also support simple 2D clipping against a clip rectangle. The top and left Clipping coordinates are inclusive. The bottom and right coordinates are exclusive. The BLT Engine performs a trivial reject for all CLIP BLT instructions before performing any accesses.

Negative destination and source coordinates are supported. In the case of negative source coordinates, the destination X1 and Y1 are modified by the absolute value of the negative source coordinate before the destination clip checking and final drawing coordinates are calculated. The absolute value of the source negative coordinate is added to the corresponding destination coordinate. The BLT engine clipping also checks for (DX2 [or = DX1) or (DY2 [or = DY1) after this calculation and if true, then the BLT is totally rejected.

Source and destination pitches have the additional explanation given next. The below statements are applicable for pitch field in all of the Blit commands:

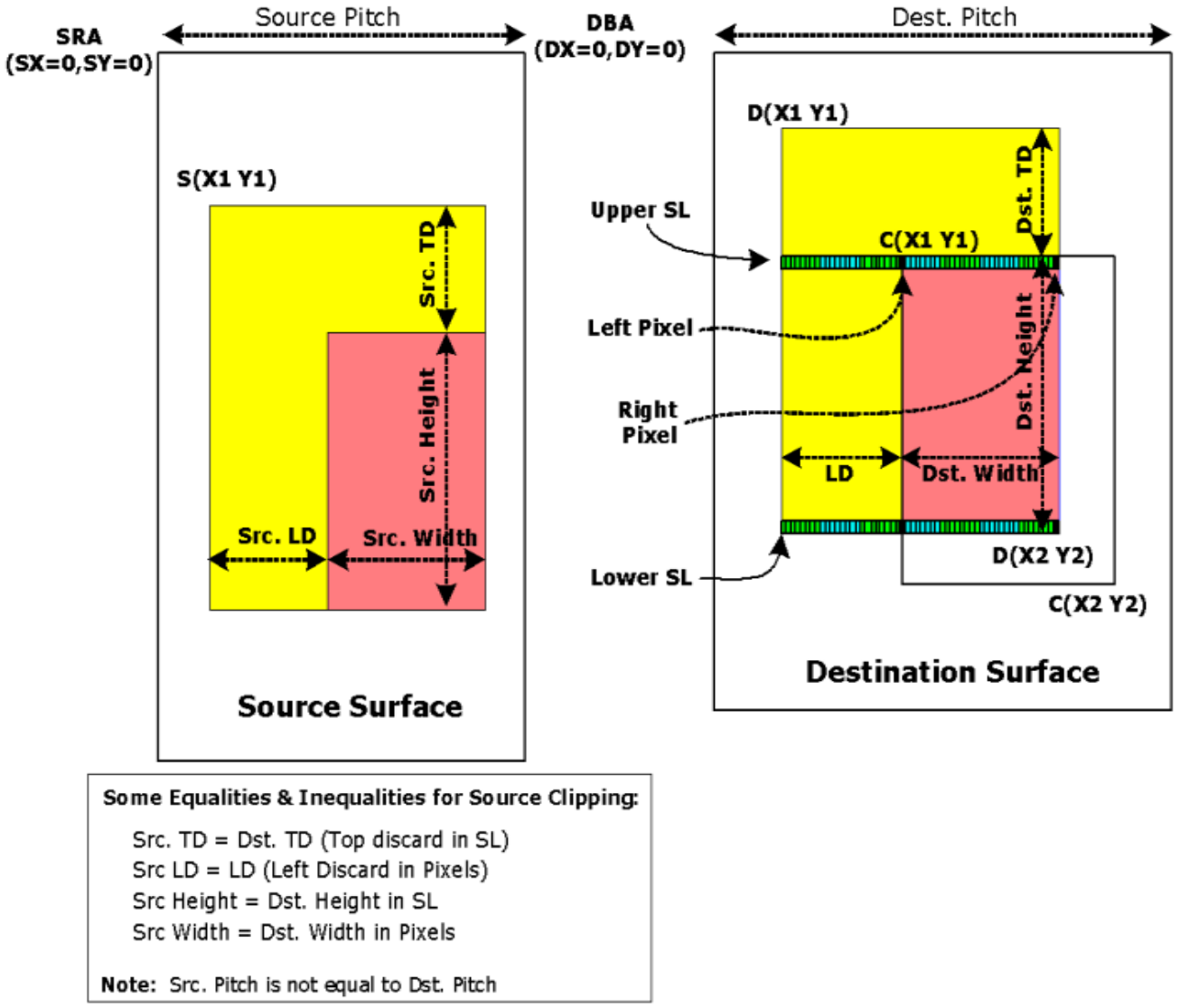
1. For Linear surfaces, the pitch is programmed in bytes. For Tiled surfaces the pitch programmed is in Dwords count.
2. For Tiled surfaces this pitch is of 512Byte granularity for Tile-X: This means the tiled-x surface pitch can be (512, 1024, 1536, 2048...)/4 (in Dwords).
3. For Tiled surfaces this pitch is of 128B granularity for Tile-Y/Yf and of 512B granularity for Tile-Ys: This means the tiled-y surface pitch can be (128, 256, 384, 512...)/4 (in Dwords).
4. Another way to indicate this is, for tiled surfaces, the pitch is programmed in Dwords and is an integral multiple of the tile width.

BLT Instructions

Instructions
XY_BLOCK_COPY_BLT
XY_FAST_COPY_BLT
XY_FAST_COLOR_BLT
XY_COLOR_BLT
XY_FULL_BLT
XY_FULL_IMMEDIATE_PATTERN_BLT
XY_FULL_MONO_PATTERN_BLT
XY_FULL_MONO_PATTERN_MONO_SRC_BLT
XY_FULL_MONO_SRC_BLT
XY_FULL_MONO_SRC_IMMEDIATE_PATTERN_BLT
XY_MONO_PAT_BLT
XY_MONO_PAT_FIXED_BLT
XY_MONO_SRC_COPY_BLT
XY_MONO_SRC_COPY_IMMEDIATE_BLT
XY_PAT_BLT
XY_PAT_BLT_IMMEDIATE

Instructions
XY_PAT_CHROMA_BLT
XY_PAT_CHROMA_BLT_IMMEDIATE
XY_PIXEL_BLT
XY_SCANLINES_BLT
XY_SETUP_BLT
XY_SETUP_CLIP_BLT
XY_SETUP_MONO_PATTERN_SL_BLT
XY_SRC_COPY_BLT
XY_SRC_COPY_CHROMA_BLT
XY_TEXT_BLT
XY_TEXT_IMMEDIATE_BLT

Some Equalities & Inequalities for Source Clipping



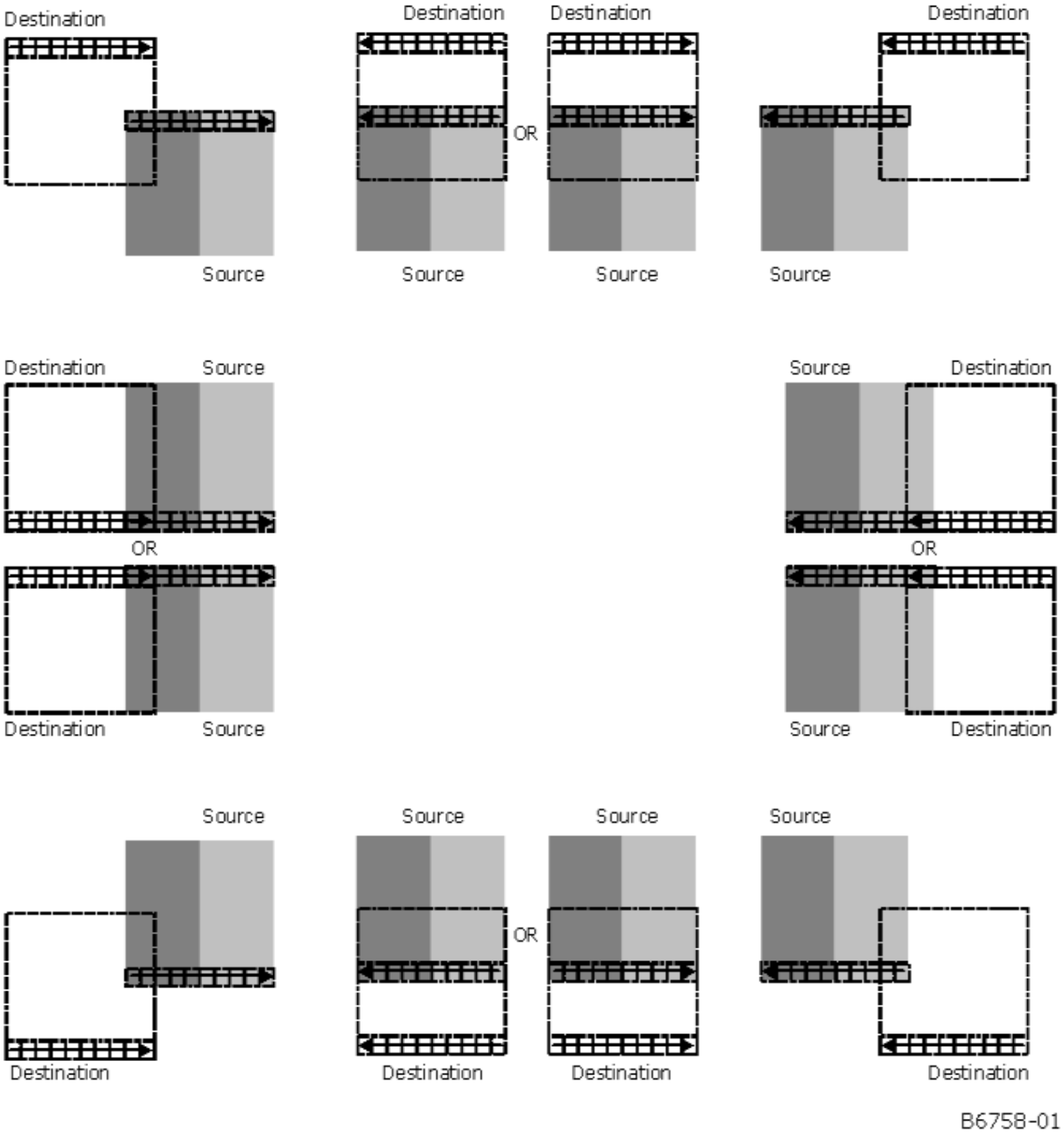
DX1, DY1, CX1, and CY1 are inclusive, while DX2, DY2, CX2, and CY2 are exclusive.

Destination pixel address = (Destination Base Address + (Destination Y coordinate * Destination pitch) + (Destination X coordinate * bytes per pixel)).

Source pixel address = (Source Base Address + (Source Y coordinate * Source pitch) + (Source X coordinate * bytes per pixel)).

Since there is 1 set of Clip Rectangle registers, the Interrupt Ring BLT commands either MUST NEVER enable clipping with these commands and never use the XY_Pixel_BLT, XY_Scanline_BLT, nor XY_Text_BLT commands or it must use context switching. The Interrupt rings can also use the non-clipped, linear address commands specified before this section.

The base addresses plus the X and Y coordinates determine if there is an overlap between the source and destination operands. If the base addresses of the source and destination are the same and the Source X1 is less than Destination X1, then the BLT Engine performs the accesses in the X-backwards access pattern. There is no need to look for an actual overlap. If the base addresses are the same and Source Y1 is less than Destination Y1, then the scan line accesses are performed backwards.



BLT Engine Instruction Field Definitions

This section describes the BLT Engine instruction fields. These descriptions are in the format of register descriptions. These registers are internal and are not readable. Some of these registers are state that is saved and restored for supporting separate software threads.

Register
BR00 - BLT Opcode and Control
BR01 - Setup BLT Raster OP, Control, and Destination Offset
BR05 - Setup Expansion Background Color
BR06 - Setup Expansion Foreground Color
BR07 - Setup Blit Color Pattern Address Lower Order Address bits
BR30 - Setup Blit Color Pattern Address Higher Order Address
BR09 - Destination Address Lower Order Address Bits
BR27 - Destination Address Higher Order Address
BR11 - BLT Source Pitch (Offset)
BR12 - Source Address Lower order Address bits
BR28 - Source Address Higher order Address
BR13 - BLT Raster OP, Control, and Destination Pitch
BR14 - Destination Width and Height
BR15 - Color Pattern Address Lower order Address bits
BR29 - Color Pattern Address Higher order Address
BR16 - Pattern Expansion Background and Solid Pattern Color
BR17 - Pattern Expansion Foreground Color
BR18 - Source Expansion Background and Destination Color
BR19 - Source Expansion Foreground Color

Intel® Iris® Xe and UHD Graphics Open Source

Programmer's Reference Manual

For the 2020-2021 11th Generation Intel Xeon®, Core™, Celeron®, Pentium® Gold Processors based on the "Tiger Lake" Platform

Volume 11: Media Engines

December 2021, Revision 1.0

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Table of Contents

Media 1

Media VDBOX 1

AVP 1

Video Command Streamer (VCS) 8

HCP 16

MFX Pipe 164

VDBOX Registers 324

Media VEBOX 326

Media VEBOX Introduction 326

SFC 332

SFC Overview 332

SFC Commands Definition 333

Media

Media VDBOX

This chapter describes the VDBOX Media Engine.

AVP

The AV1 Codec Pipeline (AVP) is a fixed function hardware video codec responsible for decoding AV1 (AOMedia Video 1) video streams.

AVP Register Definitions

The Message Channel Interface is a read-only bus used to access the AVP status registers. All registers are 32 bits where reserved bits return a value of zero and subtractive-decode is used to return 0x0000 for all register holes.

Register Attributes Description

Host Register Attributes gives the defined register tags and their description.

Host Register Attributes

Tag	Name	Description
R/W	Read/Write	Bit is read and writeable.
R/SW	Read/Special Write	Bit is readable. Write is only allowed once after a reset.
RO	Read Only	Bit is only readable, but writes have no effects.
WO	Write Only	Bit is only writeable, reads return zeros.
RV	Reserved	Bit is reserved and not visible. Reads will return 0, and writes have no effect.
NA	Not Accessible	This bit is not accessible.

AVP Decoder Register Map

This documents all AVP Decoder MMIO Registers.

AVP Decoder Register Descriptions

Reserved.

AVP Command Summary

The AV1 is configured through a set of batch commands defined in the following sections. The software driver builds a frame level workload using these commands and stores these workloads in graphics memory where they are fetched by the Video Command Streamer (VCS) and presented to the AVP for processing. The commands are processed by the Workload Parser within the AVP and the hardware is

configured by the Workload Parser prior to each frame level encode or decode. A workload is defined as a set of commands necessary to encode or decode one frame.

The software driver is required to read the AVP disable fuse to determine if the AVP is enabled. If it is disabled, then the software driver must not enable AVP batch commands to be sent to the AVP or a hang event may occur. Only when the AVP is enabled through the fuse, should the batch commands be sent to the AVP.

AVP Workload Command Model

DWord0 of each command is defined in AVP DWord0 Command Definition. The AVP is selected with the **Media Instruction Opcode "8h**" for all AVP Commands.

HCP DWord0 Command Definition

DWord	Bits	Description
0	31:29	**Command Type** = PARALLEL_VIDEO_PIPE = 3h
	28:27	**Pipeline Type** = 2h
	26:23	**Media Instruction Opcode = Codec/Engine Name** = AVP = 8h
	22:16	**Media Instruction Command** = <*see* HCP Media Instruction Commands (Opcode=7h)>
	15:12	**Reserved: MBZ**
	11:0	**Dword Length** (Excludes Dwords 0, 1) = <*command length*>

Each AVP command has assigned a media instruction command as defined in AVP Media Instruction Commands (Opcode=8h).

AVP Media Instruction Commands (Opcode=8h)

Media Instruction Command	Command DWord0 [22:16]	Mode	Scope
AVP_PIPE_MODE_SELECT	0h	Dec	Picture
AVP_SURFACE_STATE	1h	Dec	Picture
AVP_PIPE_BUF_ADDR_STATE	2h	Dec	Picture
AVP_IND_OBJ_BASE_ADDR_STATE	3h	Dec	Picture
Reserved	4h-5h		
Reserved	8h-9h		

Media Instruction Command	Command DWord0 [22:16]	Mode	Scope
VD_CONTROL_STATE	Ah	Dec	Picture
Reserved	Bh-Fh		
AVP_PIC_STATE	10h	Dec	Picture
Reserved	11h		
AVP_REF_IDX_STATE	12h	Dec	Tile
Reserved	13h-14h		
AVP_TILE_CODING	15h	Dec	Tile
Reserved	16h-1Fh		
AVP_BSD_OBJECT_STATE	20h	Dec	Tile
Reserved	21h-31h		
Reserved	33h-7Fh		

AVP Command Sequence

The AV1 is configured for encoding or decoding through a set of batch commands defined in the following sections. The software driver builds a frame level workload using these commands and stores these workloads in graphics memory where they are fetched by the Video Command Streamer (VCS) and presented to the AVP for processing. The commands are processed by the Workload Parser within the AVP and the hardware is configured by the Workload Parser prior to each frame level encode or decode. A workload is defined as a set of commands necessary to encode or decode one frame.

The software driver is required to read the AVP disable fuse to determine if the AVP is enabled. If it is disabled, then the software driver must not enable AVP batch commands to be sent to the AVP or a hang event may occur. Only when the AVP is enabled through the fuse, should the batch commands be sent to the AVP.

AVP Command Sequence Examples for Decoder

AV1 workload is based upon a single tile decode. There are no states saved between tile decodes in the AVP.

The following programming sequence will be used by single pipe decode.

«Start Workload»	
|	

VD_CONTROL (AVP_Pipe_Initialization)	AVP PIpe Reset
|	
AVP_PIPE_MODE_SELECT	AVP Pipe Setup
|	

AVP_SURFACE_STATE (Multiple)	Frame Level Commands
AVP_PIPE_BUF_ADDR_STATE	Frame Level Commands
AVP_IND_OBJ_BASE_ADDR_STATE	Frame Level Commands
AVP_PIC_STATE	Frame Level Commands
AVP_SEGMENT_STATE	Frame Level Commands
AVP_INLOOP_FILTER_STATE	Frame Level Commands
|	
AVP_TILE_CODING	Tile Level Commands
|	

AVP_BSD_OBJECT	Decode Start
|	
« Tile Done »	Decode Finish
|	
VD_CONTROL (AVP_Memory_Implicit_Fush)	HW Data Flush to Memory
|	
«End Workload»	

Each tile should be programmed independently

AVP Buffer Size Requirements

This documents all the Memory Buffer Size Requirement and Media Internal Storage Programming.

The following tables indicate AV1 rowstore size requirement.

The number below indicates is number of cacheline per SB (SB can be 64x64 or 128x128).

The rowstore data can be stored in Internal Media Storage.

To allocate the following: **Total CLs = (#CLs_per_SB * num_of_SB_per_tile_width)**

Surface	8 bits		10 bits		Comments
	SB64	SB128	SB64	SB128	
Bitstream Decoder/~~Encode~~ Line Rowstore (BTDL)	2	4	2	4	**Decoder Only**
Spatial Motion Vector Line Rowstore (SMVL)	4	8	4	8	
Intra Prediction Line Rowstore (IPDL)	2	4	4	8	
Deblocker Filter Line Y Buffer (DFLY)	9	17	11	21	
Deblocker Filter Line U Buffe (DFLU)	3	4	3	5	
Deblocker Filter Line V Buffe (DFLV)	3	4	3	5	

The following rowstore requires extra CL allocation in addition to CL per SB.

To allocate the following: **Total CLs = (#CLs_per_SB * num_of_SB_per_tile_width) + #CLs_extra_per_surface**

Surface	#CLs per SB				#CLs extra per surface			
	8 bit		10 bit		8 bit		10 bit	
	SB64	SB128	SB64	SB128	SB64	SB128	SB64	SB128
CDEF Filter Line Buffer (CDEF)	8	16	10	20	1	1	2	2

The following tables indicate AV1 tile storage. These will NOT be stored in Internal Media Storage.

To allocate the following, use the following equations based on Tile Line Rowstore or Tile Column Rowstore:

Total Tile Line CLs = (#CLs_per_SB * num_of_SB_per_FRAME_width)

Total Tile Column CLs = (#CLs_per_SB * num_of_SB_per_FRAME_height)

[Programming suggestion: The largest tile width is 4096 in pixels. It is recommended to allocate the buffer based on 4096 tile width

and it can be reused for all tiles within the frame]

Surface	8 bit		10 bit		Comments
	SB64	SB128	SB64	SB128	
Bitstream Decode/Encode Tile Line Rowstore	2	4	2	4	**Decode/Encode**
Spatial Motion Vector Tile Line Rowstore	4	8	4	8	
Intra Prediction Tile Line Rowstore Tile Row	2	4	4	8	
Deblocker Filter Tile Line Y Buffer	9	17	11	21	
Deblocker Filter Tile Column Y Buffer	8	16	10	20	
Deblocker Filter Tile Line U Buffer	3	4	3	5	
Deblocker Filter Tile Column U Buffer	2	4	3	5	
Deblocker Filter Tile Line V Buffer	3	4	3	5	
Deblocker Filter Tile Column V Buffer	2	4	3	5	
CDEF Filter Top-Left Corner Buffer	**1 * num_Tile_Horz * num_Tile_Vert**				1 CL per tiles
CDEF Filter Meta Tile Line Buffer	**1 * num_Tile_Horz**				1 CL per horz

		tile
Loop Restoration Tile Line Y Buffer	**7 * num_Tile_Horz**	7 CL per horz tile
Loop Restoration Tile Line U Buffer	**5 * num_Tile_Horz**	5 CL per horz tile
Loop Restoration Tile Line V Buffer	**5 * num_Tile_Horz**	5 CL per horz tile

The following buffer requires extra CLs at the end of frame width/height. These will NOT be stored in Internal Media Storage.

To allocate the following, use the following equations based on Tile Line Rowstore or Tile Column Rowstore:

Total Tile Line CLs = (#CLs_per_SB * num_of_SB_per_FRAME_width) + #CLs_extra_per_surface

Total Tile Column CLs = (#CLs_per_SB * num_of_SB_per_FRAME_height) + #CLs_extra_per_surface

[Programming suggestion: The largest tile width is 4096 in pixels. It is recommended to allocate the buffer based on 4096 tile width

and it can be reused for all tiles within the frame]

Surface	**#CLs per SB**				**#CLs extra per surface**			
	8 bit		**10 bit**		**8 bit**		**10 bit**	
	SB64	**SB128**	**SB64**	**SB128**	**SB64**	**SB128**	**SB64**	**SB128**
CDEF Filter Tile Line Buffer	8	16	10	20	1	1	2	2
CDEF Filter Tile Column Buffer	8	16	10	20	1	1	2	2
CDEF Filter Meta Tile Column Buffer	1	1	1	1	0	0	0	0
Super-Res Tile Column Y Buffer	22	44	29	58	22	44	29	58
Super-Res Tile Column U Buffer	8	16	10	20	8	16	10	20
Super-Res Tile Column V Buffer	8	16	10	20	8	16	10	20
Loop Restoration Filter Tile Column Y Buffer	9	17	11	22	2	2	2	2
Loop Restoration Filter Tile Column U Buffer	5	9	6	12	1	1	1	1
Loop Restoration Filter Tile Column V Buffer	5	9	6	12	1	1	1	1
Loop Restoration Meta Tile Column Buffer	1	1	1	1	1	1	1	1

The following table indicates AV1 frame buffer (except pixel buffer).

Surface	**SB64**	**SB128**	**Comments**
CDF Tables Initialization Buffer	**242** CLs		Intra frame uses 183 CLs; inter frame uses 242 CLs
CDF Tables Backward Adaptation Buffer	**242** CLs		
AV1 Segment ID Read	**2** *	**8** *	

Buffer	Total_Num_SB64_in_Frame	Total_Num_SB128_in_Frame	
AV1 Segment ID Write Buffer	**2** * Total_Num_SB64_in_Frame	**8** * Total_Num_SB128_in_Frame	
Collocated Motion Vector Temporal Buffer	**4** * Total_Num_SB64_in_Frame	**16** * Total_Num_SB128_in_Frame	
Current Frame Motion Vector Write Buffer	**4** * Total_Num_SB64_in_Frame	**16** * Total_Num_SB128_in_Frame	

This section documents the Internal Media Storage Programming for AV1 decoder.

The following table is created for a maximum of 4k tile width.

Since AV1 has a restriction of maximum 4k tile width and each tile is programmed

per tile independently, only 4k tile programming is needed (unlike other codec)

			Address Programming (N/A means disable)							
Format	Bitdepth	TileSize	BTDL	SMVL	IPDL	DFLY	DFLU	DFLV	CDEF	
420	8/10 bit	<= 4k	0	128	384	640	1344	1536	1728	N/A

AVP Common Commands

This documents commands only for AVP codec decoder

Commands
AVP_REF_IDX_STATE
AVP_SEGMENT_STATE
AVP_BSD_OBJECT

AVP Pipe Common Commands

The AVP Pipe Common Commands specify the AVP Decoder pipeline level configuration.

Shared Commands
VD_CONTROL_STATE

AVP Commands
AVP_PIPE_MODE_SELECT
AVP_SURFACE_STATE
AVP_PIC_STATE
AVP_PIPE_BUF_ADDR_STATE
AVP_TILE_CODING
AVP_IND_OBJ_BASE_ADDR_STATE

Video Command Streamer (VCS)

The VCS (Video Command Streamer) unit primarily serves as the software programming interface between the O/S driver and the MFD Engine. It is responsible for fetching, decoding, and dispatching of data packets (Media Commands with the header DWord removed) to the front end interface module of MFX Engine.

Its logic functions include:

- MMIO register programming interface
- DMA action for fetching of execlists and ring data from memory
- Management of the Head pointer for the Ring Buffer
- Decode of ring data and sending it to the appropriate destination: AVC, VC1, or MPEG2 engine
- Handling of user interrupts
- Handling of ring context switch interrupt
- Flushing the MFX Engine
- Handle NOP

The register programming (RM) bus is a DWord interface bus that is driven by the Gx Command Streamer. The VCS unit only claims memory mapped I/O cycles that are targeted to its range of 0x4000 to 0x4FFFF. The Gx and MFX Engines use semaphore to synchronize their operations.

VCS operates completely independent of the Gx CS.

The simple sequence of events is as follows: a ring (say PRB0) is programmed by a memory-mapped register write cycle. The DMA inside VCS is kicked off. The DMA fetches commands from memory based on the starting address and head pointer. The DMA requests cache lines from memory (one cacheline CL at a time). There is guaranteed space in the DMA FIFO (16 CL deep) for data coming back from memory. The DMA control logic has copies of the head pointer and the tail pointer. The DMA increments the head pointer after making requests for ring commands. Once the DMA copy of the head pointer becomes equal to the tail pointer, the DMA stops requesting.

The parser starts executing once the DMA FIFO has valid commands. All the commands have a header DWord packet. Based on the encoding in the header packet, the command may be targeted towards AVC/VC1/MPEG2 engine or the command parser. After execution of every command, the actual head pointer is updated. The ring is considered empty when the head pointer becomes equal to the tail pointer.

Context Management

Video Engine Power Context

This section lists the power context image of Video Engine across generations.

Video Engine Power Context

Table below captures the data from VCS power context save/restored by PM. Address offset in the below table is relative to the starting location of VCS in the overall power context image managed by PM.

Address offsets in this table are relative to the starting location of VCS in the power context image managed by each engine. MMIO offset mentioned for the registers in the below table are offset from the units "MMIO Base Offset" mentiond in the table " Base Offset for all engines in the section Register Access and User Mode Privileges. For Example: VCS has MMIO Base Offset as "0x1C_0000". In the below table GFX_MODE register has 0x0029C as offset against it, actual MMIO Offset of GFX_MODE register for VCS is 0xx1C_029C and for VECS it would be 0x1C_829C.

VCS Power Context Image

Description	MMIO Offset	Unit	# of DW	Address Offset (PWR)	CSFE/CSBE
CSFE Power context without Display		VCS	192	**0**	**CSFE**
NOOP		VCS	1	**00C0**	**CSBE**
Load_Register_Immediate header	0x1100_1005	VCS	1	**00C1**	**CSBE**
GAC MODE REGISTER	0x000a0	VCS	2	**00C6**	**CSBE**
VCS_WAKERATE_HCP	0x006E0	VCS	2	**00CA**	**CSBE**
VCS_WAKERATE_MFX	0x006E4	VCS	2	**00CC**	**CSBE**
VCS_SUBWELL	0x006E8	VCS	2	**00CE**	**CSBE**
VCS_BUSYNESS_VCS	0x006EC	VCS	2	**00D0**	**CSBE**
VCS_BUSYNESS_HCP	0x006F0	VCS	2	**00D2**	**CSBE**
VCS_BUSYNESS_MFX	0x006F4	VCS	2	**00D4**	**CSBE**
NOOP		VCS	9	**00D9**	**CSBE**
MI_BATCH_BUFFER_END		VCS	1	**00DF**	**CSBE**

VDBOX - Engine Register State and Context

This section discusses the following topics for the BSD Logical Render Context Address (LRCA):

- Ring Context
- Register State Context

Register State Context

EXECLIST CONTEXT
EXECLIST CONTEXT(PPGTT Base)
ENGINE CONTEXT

Description	Description When Suspended Context	Unit	Dword Count	Address Offset (Dword)
CSFE Execlist Context		VCSFE	192	**0**
MI_BATCH_BUFFER_END		CSEND	1	**00C0**
NOOP		CSEND	127	**00C1**
			DW	320
			K Bytes	1.25

Video Command Formats

MFX Commands

The MFX (MFD for decode and MFC for encode) commands are used to program the multi-format codec engine attached to the Video Codec Command Parser. See the *MFD* and *MFC* chapters for a description of these commands.

MFX state commands support direct state model and indirect state model. Recommended usage of indirect state model is provided here (as a software usage guideline).

Pipeline Type (28:27)	Opcode (26:24)	SubopA (23:21)	SubopB (20:16)	Command	Chapter	Recommended Indirect State Pointer Map	Interruptable?
MFX Common (State)							
2h	0h	0h	0h	MFX_PIPE_MODE_SELECT	MFX	IMAGE	N/A
2h	0h	0h	1h	MFX_SURFACE_STATE	MFX	IMAGE	N/A
2h	0h	0h	2h	MFX_PIPE_BUF_ADDR_STATE	MFX	IMAGE	N/A
2h	0h	0h	3h	MFX_IND_OBJ_BASE_ADDR_STATE	MFX	IMAGE	N/A
2h	0h	0h	4h	MFX_BSP_BUF_BASE_ADDR_STATE	MFX	IMAGE	N/A
2h	0h	0h	6h	MFX_ STATE_POINTER	MFX	IMAGE	N/A
2h	0h	0h	7-8h	Reserved	N/A	N/A	N/A
MFX Common (Object)							
2h	0h	1h	9h	MFD_ IT_OBJECT	MFX	N/A	Yes

Pipeline Type (28:27)	Opcode (26:24)	SubopA (23:21)	SubopB (20:16)	Command	Chapter	Recommended Indirect State Pointer Map	Interruptable?
2h	0h	0h	4-1Fh	Reserved	N/A	N/A	N/A
AVC Common (State)							
2h	1h	0h	0h	MFX_AVC_IMG_STATE	MFX	IMAGE	N/A
2h	1h	0h	1h	MFX_AVC_QM_STATE	MFX	IMAGE	N/A
2h	1h	0h	2h	MFX_AVC_DIRECTMODE_STATE	MFX	SLICE	N/A
2h	1h	0h	3h	MFX_AVC_SLICE_STATE	MFX	SLICE	N/A
2h	1h	0h	4h	MFX_AVC_REF_IDX_STATE	MFX	SLICE	N/A
2h	1h	0h	5h	MFX_AVC_WEIGHTOFFSET_STATE	MFX	SLICE	N/A
2h	1h	0h	6-1Fh	Reserved	N/A	N/A	N/A
AVC Dec							
2h	1h	1h	0-7h	Reserved	N/A	N/A	N/A
2h	1h	1h	8h	MFD_AVC_BSD_OBJECT	MFX	N/A	No
2h	1h	1h	9-1Fh	Reserved	N/A	N/A	N/A
AVC Enc							
2h	1h	2h	0-1h	Reserved	N/A	N/A	N/A
2h	1h	2h	2h	MFC_AVC_FQM_STATE	MFX	IMAGE	N/A
2h	1h	2h	3-7h	Reserved	N/A	N/A	N/A
2h	1h	2h	8h	MFC_AVC_PAK_INSERT_OBJECT	MFX	N/A	N/A
2h	1h	2h	9h	MFC_AVC_PAK_OBJECT	MFX	N/A	Yes
2h	1h	2h	A-1Fh	Reserved	N/A	N/A	N/A
2h	1h	2h	0-1Fh	Reserved	N/A	N/A	N/A
VC1 Common							
2h	2h	0h	0h	MFX_VC1_PIC_STATE	MFX	IMAGE	N/A
2h	2h	0h	1h	MFX_VC1_PRED_PIPE_STATE	MFX	IMAGE	N/A
2h	2h	0h	2h	MFX_VC1_DIRECTMODE_STATE	MFX	SLICE	N/A
2h	2h	0h	2-1Fh	Reserved	N/A	N/A	N/A
VC1 Dec							
2h	2h	1h	0-7h	Reserved	N/A	N/A	N/A
2h	2h	1h	8h	MFD_VC1_BSD_OBJECT	MFX	N/A	Yes
2h	2h	1h	9-1Fh	Reserved	N/A	N/A	N/A
VC1 Enc							
2h	2h	2h	0-1Fh	Reserved	N/A	N/A	N/A
MPEG2 Common							
2h	3h	0h	0h	MFX_MPEG2_PIC_STATE	MFX	IMAGE	N/A

Pipeline Type (28:27)	Opcode (26:24)	SubopA (23:21)	SubopB (20:16)	Command	Chapter	Recommended Indirect State Pointer Map	Interruptable?
2h	3h	0h	1h	MFX_MPEG2_QM_STATE	MFX	IMAGE	N/A
2h	3h	0h	2-1Fh	Reserved	N/A	N/A	N/A
MPEG2 Dec							
2h	3h	1h	1-7h	Reserved	N/A	N/A	N/A
2h	3h	1h	8h	MFD_MPEG2_BSD_OBJECT	MFX	N/A	Yes
2h	3h	1h	9-1Fh	Reserved	N/A	N/A	N/A
MPEG2 Enc							
2h	3h	2h	0-1Fh	Reserved	N/A	N/A	N/A
The Rest							
2h	4-5h, 7h	x	x	Reserved	N/A	N/A	N/A

Video Command Header Format

Type	Bits				
	31:29	28:24	23	22	21:0
Memory Interface (MI)	000	Opcode 00h - NOP 0Xh - Single DWord Commands 1Xh - Reserved 2Xh - Store Data Commands 3Xh - Ring/Batch Buffer Cmds		Identification No./DWord Count Command Dependent Data 5:0 - DWord Count 5:0 - DWord Count 5:0 - DWord Count	

Type	Bits				
	31:29	28:27	26:24	23:16	15:0
Reserved	011	00	XXX	XX	
MFX Single DW	011	01	000	Opcode: 0h	0
Reserved	011	01	1XX		
Reserved	011	10	0XX		
AVC State	011	10	100	Opcode: 0h - 4h	DWord Count
AVC Object	011	10	100	Opcode: 8h	DWord Count
VC1 State	011	10	101	Opcode: 0h - 4h	DWord Count
VC1 Object	011	10	101	Opcode: 8h	DWord Count
Reserved	011	10	11X		
Reserved	011	11	XXX		

Type	Bits					
	31:29	28:27	26:24	23:21	20:16	15:0
MFX Common	011	10	000	000	subopcode	DWord Count
Reserved	011	10	000	001-111	subopcode	DWord Count
AVC Common	011	10	001	000	subopcode	DWord Count
AVC Dec	011	10	001	001	subopcode	DWord Count
AVC Enc	011	10	001	010	subopcode	DWord Count
Reserved	011	10	001	011-111	subopcode	DWord Count
Reserved (for VC1 Common)	011	10	010	000	subopcode	DWord Count
VC1 Dec	011	10	010	001	subopcode	DWord Count
Reserved (for VC1 Enc)	011	10	010	010	subopcode	DWord Count
Reserved	011	10	010	011-111	subopcode	DWord Count
Reserved (MPEG2 Common)	011	10	011	000	subopcode	DWord Count
MPEG2Dec	011	10	011	001	subopcode	DWord Count
Reserved (for MPEG2 Enc)	011	10	011	010	subopcode	DWord Count
Reserved	011	10	011	011-111	subopcode	DWord Count
Reserved	011	10	100-111	XXX		

Watchdog Timer Registers

The following registers are defined as Watchdog Timer registers:

Register
PR_CTR_CTL - Watchdog Counter Control
PR_CTR_THRSH - Watchdog Counter Threshold

Logical Context Support

This section contains the registers for Logical Context Support.

Register
BB_STATE - Batch Buffer State Register
CXT_EL_OFFSET - Exec-List Context Offset
BB_START_ADDR - Batch Buffer Start Head Pointer Register
BB_START_ADDR_UDW - Batch Buffer Start Upper Head Pointer Register
BB_ADDR_DIFF - Batch Address Difference Register
BB_ADDR - Batch Buffer Head Pointer Register
SBB_ADDR - Second Level Batch Buffer Head Pointer Register
SBB_ADDR_UDW - Second Level Batch Buffer Upper Head Pointer Register
WAIT_FOR_RC6_EXIT - Control Register for Power Management
SBB_STATE - Second Level Batch Buffer State Register

Register
BB_OFFSET - Batch Offset Register
RING_BUFFER_HEAD_PREEMPT_REG - RING_BUFFER_HEAD_PREEMPT_REG
BB_PREEMPT_ADDR - Batch Buffer Head Pointer Preemption Register
BB_PREEMPT_ADDR_UDW - Batch Buffer Upper Head Pointer Preemption Register
SBB_PREEMPT_ADDR - Second Level Batch Buffer Head Pointer Preemption Register
SBB_PREEMPT_ADDR_UDW - Second Level Batch Buffer Upper Head Pointer Preemption Register
MI_PREDICATE_RESULT_1 - Predicate Rendering Data Result 1
MI_PREDICATE_RESULT_2 - Predicate Rendering Data Result 2
FORCE_TO_NONPRIV - FORCE_TO_NONPRIV
INDIRECT_CTX - Indirect Context Pointer
INDIRECT_CTX_OFFSET - Indirect Context Offset Pointer
BB_PER_CTX_PTR - Batch Buffer Per Context Pointer
SYNC_FLIP_STATUS - Wait For Event and Display Flip Flags Register
SYNC_FLIP_STATUS_1 - Wait For Event and Display Flip Flags Register 1
SYNC_FLIP_STATUS_2 - Wait For Event and Display Flip Flags Register 2

Mode Registers

The following are Mode Registers:

Mode Register
MI_MODE - Mode Register for Software Interface
INSTPM - Instruction Parser Mode Register
NOPID - NOP Identification Register
IDLEDLY - Idle Switch Delay
RESET_CTRL - Reset Control Register
PREEMPTION_HINT - Preemption Hint
PREEMPTION_HINT_UDW - Preemption Hint Upper DWord
SEMA_WAIT_POLL - Semaphore Polling Interval on Wait

Misc Register
HWS_PGA - Hardware Status Page Address Register

Registers in Media Engine

This topic describes the memory-mapped registers associated with the Memory Interface, including brief descriptions of their use. The functions performed by some of these registers are discussed in more detail in the Memory Interface Functions, Memory Interface Instructions, and Programming Environment chapters.

The registers detailed in this chapter are used across multiple projects and are extensions to previous projects. However, slight changes may be present in some registers (i.e., for features added or removed), or some registers may be removed entirely. These changes are clearly marked within this chapter.

Register
TIMESTAMP - Reported Timestamp Count
CTX_TIMESTAMP - Context Timestamp Count

Memory Interface Commands for Video Codec Engine

This chapter describes the formats of the "Memory Interface" commands, including brief descriptions of their use. The functions performed by these commands are discussed fully in the *Memory Interface Functions* Device Programming Environment chapter.

This chapter describes MI Commands for the Video Codec Engine.

The commands detailed in this chapter are used across product families. However, slight changes may be present in some commands (i.e., for features added or removed), or some commands may be removed entirely. Refer to the *Preface* chapter for details.

MI Commands
MI_BATCH_BUFFER_END
MI_CONDITIONAL_BATCH_BUFFER_END
MI_BATCH_BUFFER_START
MI_FLUSH_DW
MI_COPY_MEM_MEM
MI_LOAD_REGISTER_REG
MI_MATH
MI_NOOP
MI_REPORT_HEAD
MI_SEMAPHORE_SIGNAL
MI_SEMAPHORE_WAIT
MI_STORE_REGISTER_MEM
MI_STORE_DATA_IMM
MI_SUSPEND_FLUSH
MI_USER_INTERRUPT
MI_LOAD_REGISTER_MEM
MI_ATOMIC
MI_FORCE_WAKEUP

HCP

HCP HW Codec Pipeline Introduction

The HEVC/VP9 Codec Pipeline (HCP) is a fixed function hardware video codec responsible for decoding and encoding HEVC/VP9 (High Efficiency Video Coding) video streams.

Scope

The primary scope of the HCP BSpec document is to provide a description of the HCP commands processed by the Video Command Streamer (VCS). The secondary scope is to provide a description of the status registers on the Message Channel Interface to support encoding and decoding of the HEVC and VP9 video formats.

The BSpec sections include:

- Summary of Features
- Architecture Overview
- Commands
- Register Definitions

Acronyms and Applicable Standards

Summary of Features

The following sections define the general features of the HCP HW Decoder and Encoder pipeline, and the features specific to HEVC and VP9 decoding and encoding, respectively.

VP9 Decoder Features

• Support full-featured VP9 Profile 1 and part of Profile 2 (444 only), up to 8K.

• All headers (uncompressed and compressed header) are parsed and decoded outside the HCP HW pipeline. They are then fed to the HW through a set of HCP (with VP9 specific) state commands.

• Supports inner-loop decode part of the VP9 encoder implementation.

VP9 Encoder Features

- Supports ENC-PAK architecture
- Supports multiple pass BRC rate control operation flow
- Supports the VP9 Main Profile standard
- VP9 PAK Only mode is not supported.

HCP Hardware Pipeline Features

- Supports both decoder and encoder functions, setup on a per picture basis:
  - Hardware acceleration provides Ctb/CU level decode and encode.
  - No context switch is supported within a frame process.

- Supports Video Command Streamer (VCS):
  - Shared with MFX HW pipeline, and at any one time, only one pipeline (MFX or HCP) and one operation (decoding or encoding) can be active.
- Supports Message Channel Interface:

Feature
Supports Tile-YS and Tile-YF.
Supports Tile-Y Legacy.

- Supports NV12 video buffer plane:
  - Supports 4:2:0, 8-bit per pixel component (Y, Cb and Cr) video.
- Supports 8Kx8K frame size.

HEVC Decoder Features

- Supports full-featured HEVC Main Profile standard, up to Level 6.2.
- Supports the long format HW decoding interface:
  - All headers (SPS, PPS, Slice Header) are parsed and decoded outside the HCP HW pipeline. They are then fed to the HW through a set of HCP state commands.
- Supports inner-loop decode with hardware entry points for Encoder.
- Error detection/resiliency down to the Ctb/CU level.

All 41 HEVC profiles:

- Yellow colored profiles have explicit GUID assigned
- Pink colored profiles are subset of Yellow colored profiles, and do not have their own GUID.
- Grey colored profiles are not supported by Intel HW at all.

No.	Version	HEVC Profiles	Spec. Description	Intel HW Decoder
1	Base	Main	8b 4:2:0 only	
2	Base	Main 10	8/9/10b 4:2:0 only	
3	Base	Main Still Picture	8b 4:2:0, 1 frame only	
7	RExt	Main 12	8 to 12b 400/420	-Mono
8	RExt	Main 4:2:2 10	8/9/10b 400/420/422	-Mono
9	RExt	Main 4:2:2 12	8-12b 400/420/422	-Mono
10	RExt	Main 4:4:4	8b 400/420/422/444	-Mono
11	RExt	Main 4:4:4 10	8/9/10b 400/420/422/444	-Mono

No.	Version	HEVC Profiles	Spec. Description	Intel HW Decoder
12	RExt	Main 4:4:4 12	8-12b 400/420/422/444	-Mono
15	RExt	Main 12 Intra		-Mono
16	RExt	Main 4:2:2 10 Intra		-Mono
17	RExt	Main 4:2:2 12 Intra		-Mono
18	RExt	Main 4:4:4 Intra		-Mono
19	RExt	Main 4:4:4 10 Intra		-Mono
20	RExt	Main 4:4:4 12 Intra		-Mono
22	RExt	Main 4:4:4 Still Pic	8b 400/420/422/444	-Mono
28	V3	Screen-Extended Main	8b 400/420	-Mono
29	V3	Screen-Extended Main 10	8/9/10b 400/420	-Mono
30	V3	Screen-Extended Main 4:4:4	8b 400/420/444 + Main 4:4:4	-Mono
31	V3	Screen-Extended Main 4:4:4 10	8/9/10b 400/420/444 + Main 4:4:4 10	-Mono

HEVC Encoder Features

- Supports ENC-PAK architecture
- Supports multiple pass BRC rate control operation flow
- Supports the HEVC Main Profile standard, with certain restrictions on the feature set and coding parameters, listed in the following table:

HEVC Encoder Features and Restrictions

Note that there is a difference between what PAK supported and what ENC supported. A feature/function that is supported in the PAK, does not necessary being supported by ENC and MediaSDK and the like.

Coding Tool	Support	Restriction	Comments
LCU Size	Yes (spec)		Support all 3 sizes: 16x16, 32x32, 64x64.
CU Size	Yes (spec)		Support 8x8, 16x16, 32x32, 64x64. Max 64 CU per LCU; min. CU size intel supported is 8x8 for all LCU size.
PU Partition	Yes (spec)		Support all inter symmetric (square) and asymmetric (non-square) PU partitioning, according to HEVC spec. PU Size for inter : Smallest allowed is 4x8 and 8x4, and they cannot be bidirectional. Inter 4x4 PU is not allowed in Main Profile.
TU QuadTree		Partial (intel) max depth is set to 3	Max depth is 3 (64x64 CU with 4x4 TU). Decoder supports this depth, but probably no need to search this for encode. Better to just split CU. HM common conditions set the max to 2 for both inter and intra. Intel Encoder only supports 2 levels of quad-tree. That is, max_transform_hierarchy_depth_inter/intra <= 2. Max num of TUs per CU is 16.
AMP	Yes (spec)		Asymmetric Motion Partition (rectangular PU partitioning - 2NxnU, 2NxnD, nLx2N or nRx2N). Available only for 64x64 to 16x16 CU.
AMVP	Yes (spec)		Adaptive/Advanced Motion Vector Prediction : spatial and PU-based temporal co-located MV candidates with scaling. Logic available from decoder. HW PAK is supporting temporal MV candidates.
Merge	Yes (spec)		Merge Skip and Regular Merge. Max. 5 MV Merge candidates (4 spatial + 1 temporal co-located) with scaling. Logic available from decoder. [merge_flag, merge_index, skip_flag]
Parallel Motion Merge		No (intel)	Tool for parallel decode of MVs. Since this isn't constrained by Main Profile, the decoder has to meet performance targets in the worst case anyway.
MC Interpolation Filter	Yes (spec)		1/4-pel Luma MV precision, 1/8-pel Chroma MV precision. 8-tap Luma filtering for both 1/2-pel and 1/4-pel locations (1-pass). 4-tap Chroma filtering. Use separable (first horizontal then vertical 1-D filtering) filter coefficients. Not all filter kernels are symmetrical and can map into simple arithmetic. It is a DCT-IF based filter. All operations are within 16-bit data.
Weighted Prediction	Yes (spec)		Free for PAK since decoder already has it.
Combined Reference Frame List		No (intel)	Combine List0 and List1 into a single list to remove uni-prediction signaling overhead.
Intra modes	Yes (spec)		33 directions and DC/Planar modes, with adaptive pre-filtering on reference pixels and boundary smoothing.

Coding Tool	Support	Restriction	Comments
IPCM (intra)		No (intel)	Can be disabled completely by SPS, or only allowed at certain CU sizes. No bit maximum on CUs in any profile/level (yet), so as of today there's no mandate to support this for encode.
Constrained Intra		No (intel)	Allow only intra neighboring blocks for current block intra-prediction. Enabling this is a coding loss and does not result in a performance improvement in HW designs.
2D DCT Transform	Yes (spec)		Square shape only; 32x32, 16x16, 8x8 and 4x4.
Transform Skip Evaluation		No (intel) ENC will estimate the use of transform skip	Significant coding gains for screen content (PowerPoint etc.). This tool is disabled in the common conditions but isn't explicitly disallowed by Main Profile. FQ is not bypass.
Sign bit hiding		No (intel)	Coding gain by removing one bypass bin per TU. Requires some smarts in the PAK.
Trellis		No (intel)	Trellis Quantization
SAO		No (intel)	Difficult to implement in single pass, performance impact in 2-pass or with previous frame search. Needs investigation. Decoder will support it.
Loop Filter across tiles/slices boundary		No (intel)	Can be disabled for tiles and or slices in SPS, so that filter across all tiles and slices boundaries. Main profile doesn't constrain.
Scaling List	Yes (spec)		This uses the default (or custom) qp adjustment on a per-frequency basis within a TU. Good coding improvement over flat scaling.
dQP		Partial (intel) Yes for LCU; No for CU	Being able to change QP per LCU or even up to once per 8x8 CU can lead to significant coding gains.
Chroma QP offset		No (intel)	No ROI.
Dependent slices		No (intel)	It is now part of Main Profile, but Intel will not support it. SW can perform the slice repackaging without re-encoding.
Tiles		No (intel)	Although in Main Profile, it results in coding loss and doesn't improve performance on HW. SW parallel processing (multithreaded) tool
Wavefront (aka WPP)		No (intel)	Latest Main Profile spec has included Wavefront. We got a feedback that this feature is highly desirable to support high performance multithreading HEVC decoder.
Lossless coding		No (intel)	Note: this is not the same as IPCM. Also details of this are in flux.
Interlaced Video		No (intel)	Only progressive video encoding is supported.
LM mode	No (spec)		Chroma-from-luma intra prediction (Linear Mode) is not allowed in Main Profile (yet).

Coding Tool	Support	Restriction	Comments
NSQT	No (spec)		Non square transform is not allowed in the Main Profile
ALF	No (spec)		Expensive and not in Main Profile currently. If decoder is going to support it, may consider for encoder as well.
Entropy slices	No (spec)		Not allowed in Main Profile, it is a SW parallel processing (multithreaded) tool.
Slice granularity != 0	No (spec)		Not allowed by Main Profile, and highly likely to be removed from the standard completely.

Architecture Overview

HCP HW pipeline is designed to support two codec standards: HEVC and VP9. It implements the complete decoder process, but does not handle header (sequence header, frame/picture header, slice header and tile header) parsing which is to be done by application/driver at software level. It also implements the bitstream coding, residual generation and frame reconstruction part of the encoding process (namely PAK), whereas the bit rate control, motion estimation and the block coding decision are done either in software and/or in a separate HW modules.

For decoder, both HEVC and VP9 are fully compliant to the standards, while for PAK, only a subset of coding tools are implemented.

The HCP can be programmed to function as either VP9 or HEVC at frame level at a time. The command sequence for each codec is frame based.

HEVC/VP9 Encoder

The HEVC/VP9 encoder architecture consists of 2 major HW components: VDENC and PAK. In addition, the HEVC architecture also supports a 3 HW components mode: Media ENC (EUs/Kernels+VME), and PAK. Media EUs/Kernels implement the ENC portion of the encoding process. It communicates with the VME to determine the best inter and intra coding modes for each block based on a set of cost functions and algorithms. It also responsible for setting up multiple encoding passes to meet the target coding efficiency. For both modes, the PAK is used to generate the final compressed bitstream on a per LCU basis with coding parameters received from the ENC. It also provides feedback information for BRC rate control purpose. As part of the PAK operation, it invokes the decoder in the reconstruction process.

HCP Command Summary

The HCP is configured for encoding or decoding through a set of batch commands defined in the following sections. The software driver builds a frame level workload using these commands and stores these workloads in graphics memory where they are fetched by the Video Command Streamer (VCS) and presented to the HCP for processing. The commands are processed by the Workload Parser within the HCP and the hardware is configured by the Workload Parser prior to each frame level encode or decode. A workload is defined as a set of commands necessary to encode or decode one frame.

The software driver is required to read the HCP disable fuse to determine if the HCP is enabled. If it is disabled, then the software driver must not enable HCP batch commands to be sent to the HCP or a hang event may occur. Only when the HCP is enabled through the fuse, should the batch commands be sent to the HCP.

Workload Command Model

DWord0 of each command is defined in HCP DWord0 Command Definition. The HCP is selected with the **Media Instruction Opcode "7h**" for all HCP Commands.

HCP DWord0 Command Definition

DWord	Bits	Description
0	31:29	**Command Type** = PARALLEL_VIDEO_PIPE = 3h
	28:27	**Pipeline Type** = 2h
	26:23	**Media Instruction Opcode = Codec/Engine Name** = HCP = 7h
	22:16	**Media Instruction Command** = <*see* HCP Media Instruction Commands (Opcode=7h)>
	15:12	**Reserved: MBZ**
	11:0	**Dword Length** (Excludes Dwords 0, 1) = <*command length*>

Each HCP command has assigned a media instruction command as defined in HCP Media Instruction Commands (Opcode=7h).

HCP Media Instruction Commands (Opcode=7h)

Media Instruction Command	Command DWord0 [22:16]	Mode	Scope
HCP_PIPE_MODE_SELECT	0h	Enc/Dec	Picture
HCP_SURFACE_STATE	1h	Enc/Dec	Picture
HCP_PIPE_BUF_ADDR_STATE	2h	Enc/Dec	Picture
HCP_IND_OBJ_BASE_ADDR_STATE	3h	Enc/Dec	Picture
HCP_QM_STATE	4h	Enc/Dec	Picture
HCP_FQM_STATE (encoder only)	5h	Enc	Picture
Reserved	8h-Fh		

Media Instruction Command	Command DWord0 [22:16]	Mode	Scope
HCP_PIC_STATE	10h	Enc/Dec	Picture
HCP_TILE_STATE	11h	Dec	Picture
HCP_REF_IDX_STATE	12h	Enc/Dec	Slice
HCP_WEIGHTOFFSET_STATE	13h	Enc/Dec	Slice
HCP_SLICE_STATE	14h	Enc/Dec	Slice
HCP_TILE_CODING	15h	Enc/Dec	Tile
Reserved	16h-1Fh		
HCP_BSD_OBJECT_STATE (decoder only)	20h	Dec	Slice
HCP_PAK_OBJECT (encoder only)	21h	Enc	LCU
HCP_INSERT_PAK_OBJECT (encoder only)	22h	Enc	Bitstream
Reserved	23h-2Fh		
HCP_VP9_PIC_STATE	30h	Dec	Picture
HCP_VP9_SEGMENT_STATE	32h	Dec	Picture
HCP_VP9_PAK_STATE	35h	Enc	LCU
HCP_VP9_RDOQ_STATE	3Ch	Enc	LCU
Reserved	3Dh-7Fh		

HCP Command Sequence Examples

VP9 Encoder Command Sequence

For a single frame encoding process the command sequence is listed below. There are no states saved between frame encoded in the HCP. There should be no other commands or context switch within a group of PAK OBJECT Commands. HCP and MFX share the same VCS, but there is no common encoding and decoding command that can be executed in both pipes, mi_flush and MMIO commands.

------ Per Frame Level Commands

HCP_PIPE_MODE_SELECT

HCP_SURFACE_STATE

HCP_PIPE_BUF_ADDR_STATE

HCP_IND_OBJ_BASE_ADDR_STATE

HCP_VP9_PIC_STATE

HCP_VP9_SEGMENT_STATE

HCP_VP9_QUANT_LOOKUP_TABLES

HCP_PAK_INSERT_OBJECT - if header present at the beginning of frame

------- A group of LCUs

HCP_PAK_OBJECT

...

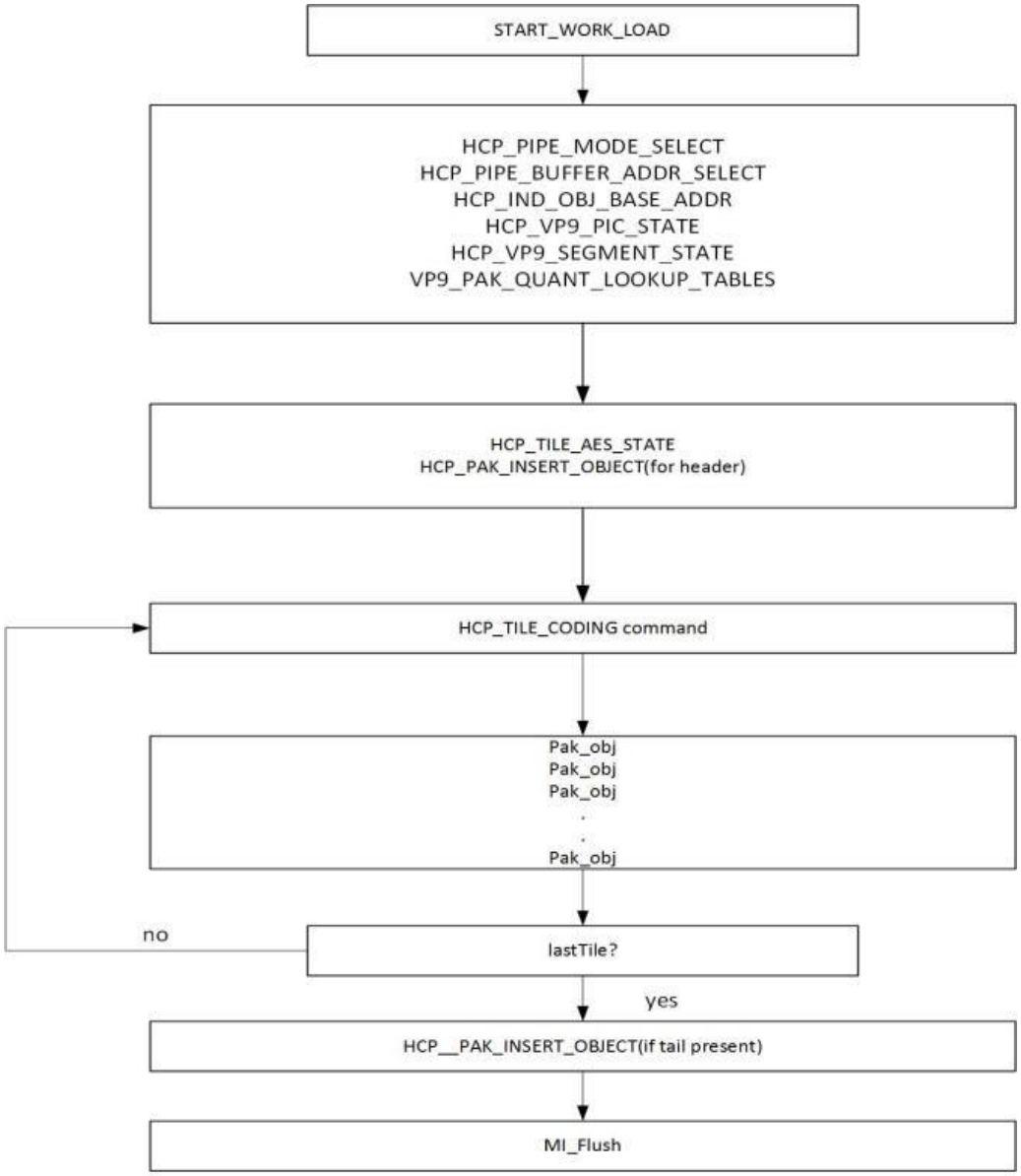
HCP_PAK_INSERT_OBJECT - if tail present at frame end

MI_FLUSH - when the frame is done

Command Sequence in Single Pipe Mode

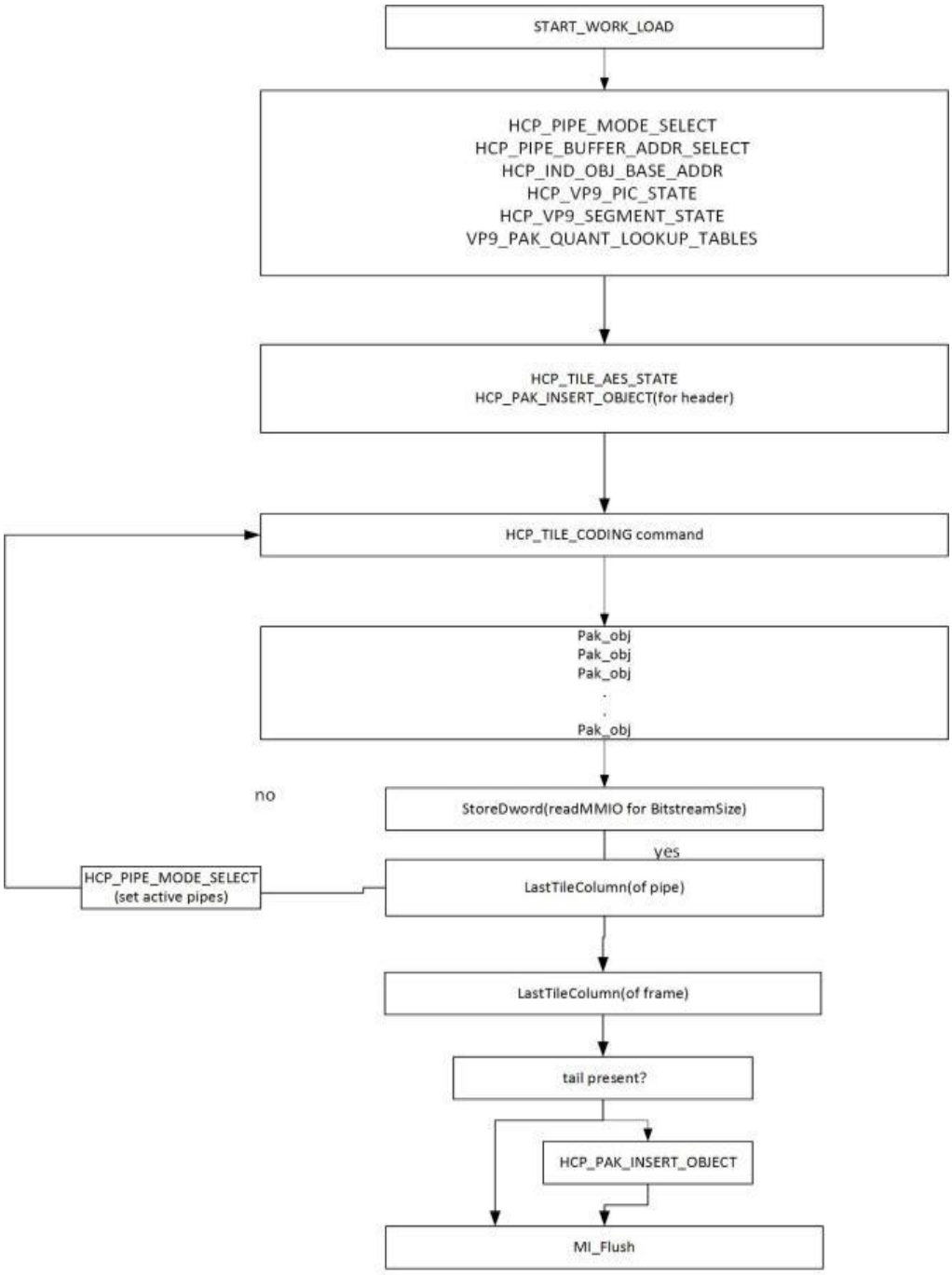
Command Sequences with Tile Support

Single Pipe Mode- Following flow chart shows the command sequence when encoding frame using a single pipe(VDbox).



VP9 encode command sequence for Single Pipe mode

Multiple Pipe Mode- When encoding a frame using multiple pipes, each pipe gets a single Tile Column or multiple Tile columns depending upon Number of tile columns to encode. Following flow chart shows commands sequence in a pipe (VDbox) when multiple pipes are used for encoding.



VP9 encode command sequence in Multiple Pipes mode

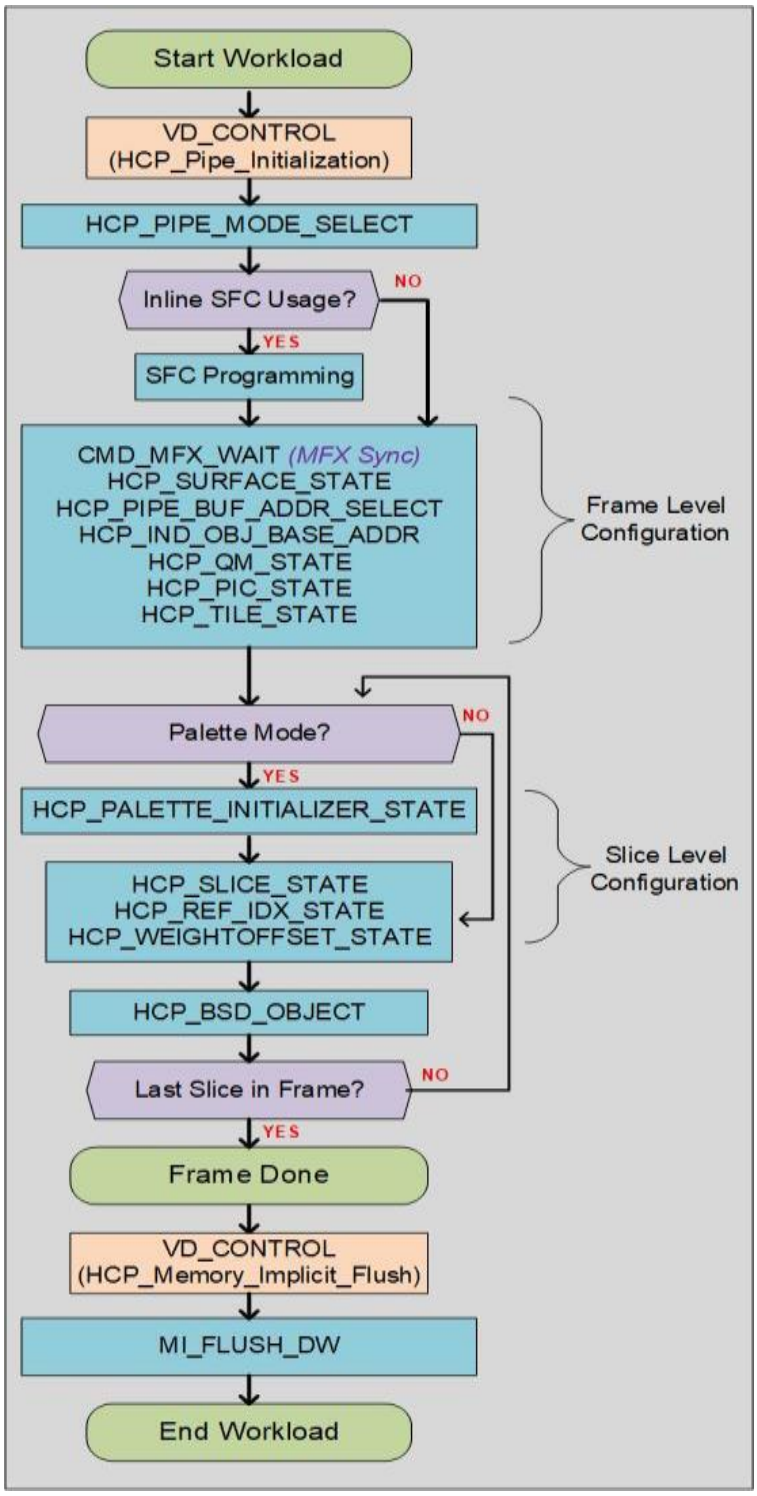
HCP Decoder Command Sequence

The long format workload for the HCP is based upon a single frame decode. There are no states saved between frame decodes in the HCP. Once the bit stream DMA is configured with the HCP_BSD_OBJECT command, and the bit stream is presented to the HCP, the frame decode will begin.

HCP Long Format Decode Workload Chart

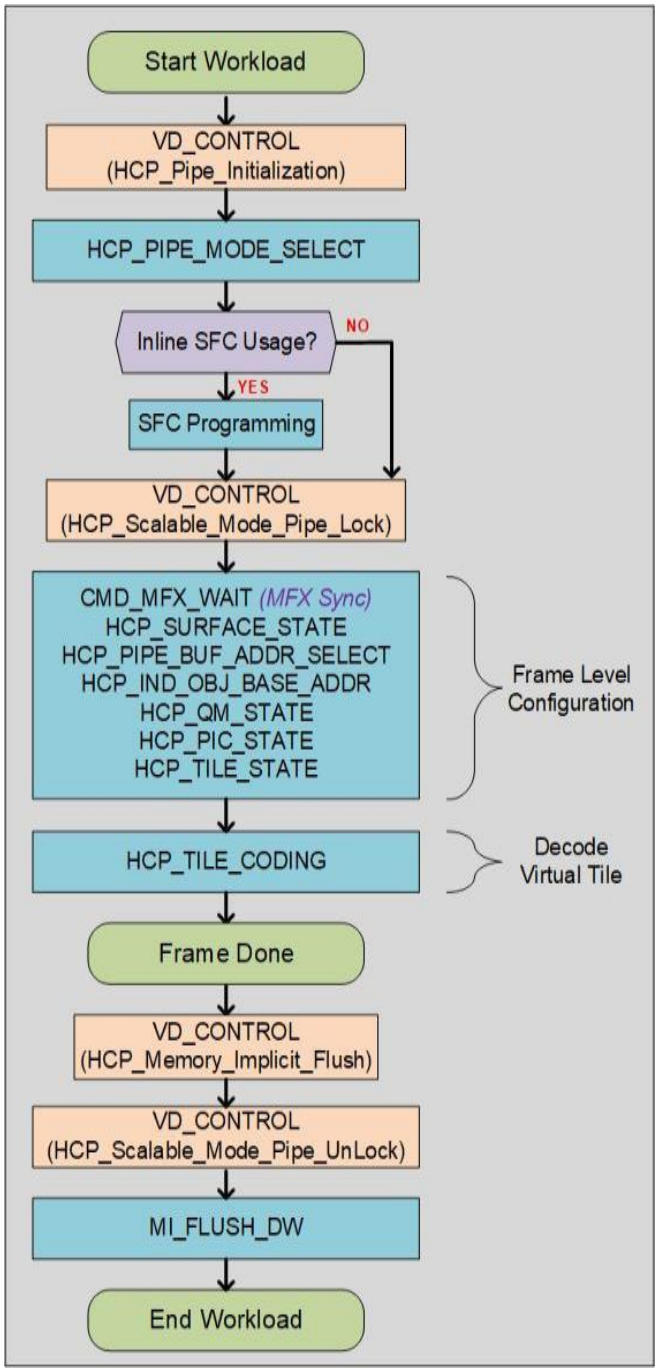
The following programming sequence will be used by single pipe decode (CABAC+BE reconstruction).

This is also used in scalable CABAC only decode mode.



HCP Scalabe Backend Only Workload Chart

The scalable decoder workload for the HCP backend pipe is based upon a single frame decode using multiple backend pipes. The frame is split into multple "virtual" vertical (column) tiles and they are processed by multple linked backend pipes. [NOTE: the above command sequence is still used for HCP CABAC decode and the decoded syntax elements are streamed to memory.]

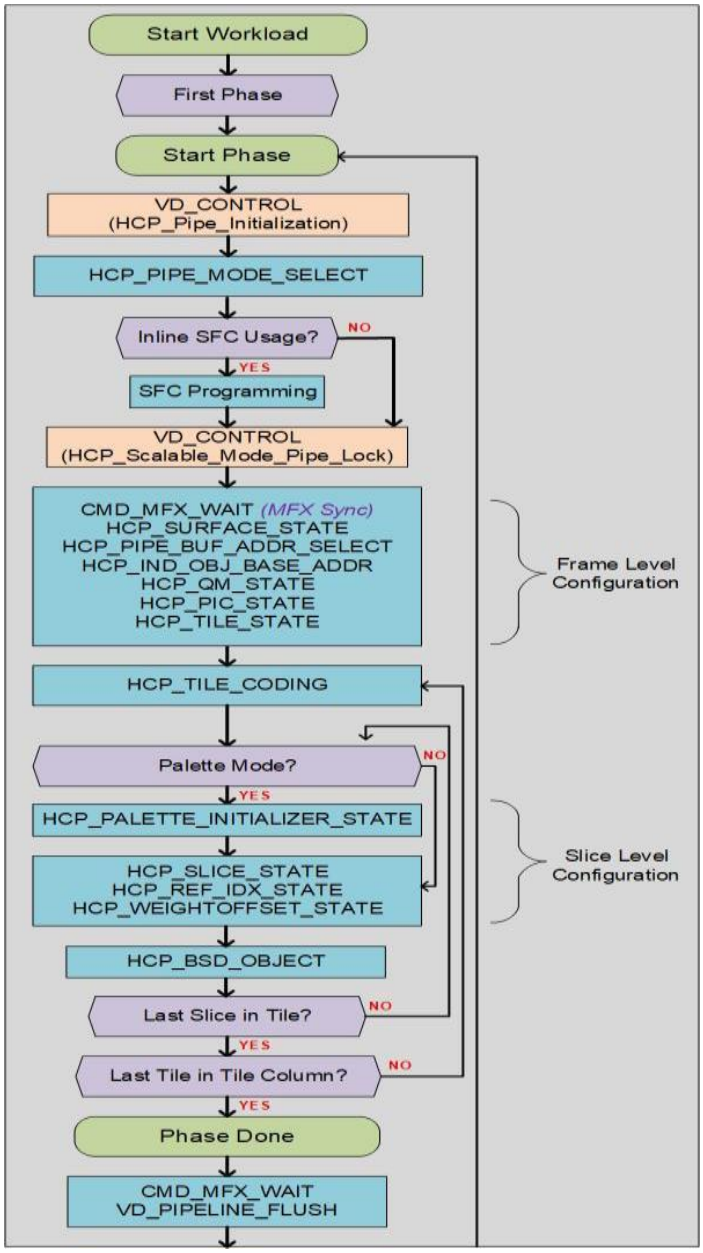


The scalable decoder with CABAC in real tiles allows frame with tiles to be decodes by multiple pipes. Each pipes will decode separate tile columns.

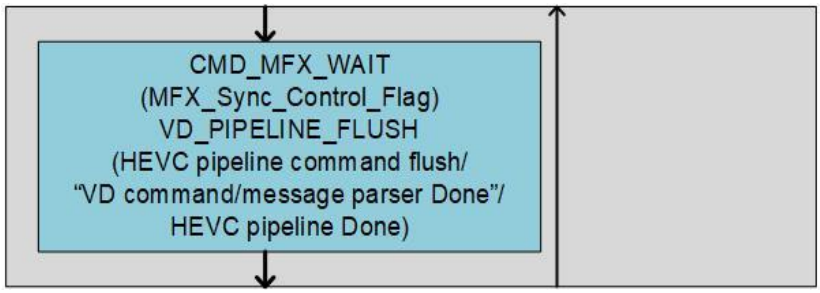
In this mode, the CABAC and BE will link and decode together. The following is the programming sequence.

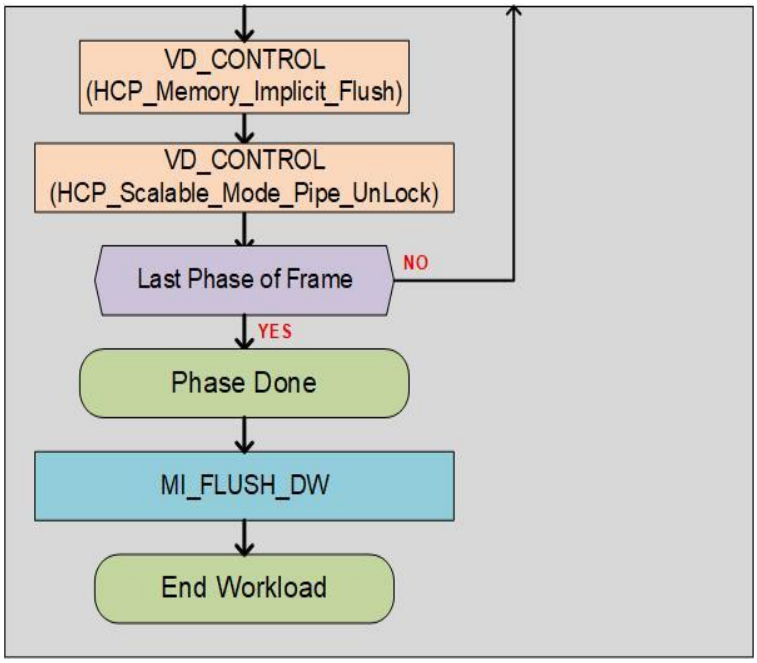
Also, the number of tiles may be greater than the number of pipes used to decode the frame. In this case, multiple phases will be introduced in the programming. For example, if N number of pipes are used to decode the frame. The first phases will decode 0 to (N-1) tile column. The next phase will decode N to (2N -1). This will continue till all the tile columns are processed.

[NOTE: The last phases may have fewer than N tile columns. In this case, only the needed pipes will be programmed and used.



This needs to be added between tiles.





HCP Encoder Command Sequence

For a single frame encoding process (w/o multiple slices per frame), the command sequence is listed below. There are no states saved between frame encoded in the HCP. There should be no other commands or context switch within a group of PAK OBJECT Commands, representing a complete slice. HCP and MFX share the same VCS, but there is no common encoding and decoding command that can be executed in both pipes, mi_flush and MMIO commands.

------ Per Frame Level Commands

HCP_PIPE_MODE_SELECT

HCP_SURFACE_STATE

HCP_PIPE_BUF_ADDR_STATE

HCP_IND_OBJ_BASE_ADDR_STATE

HCP_FQM_STATE - issue n number of times

HCP_QM_STATE - issue n number of times

HCP_PIC_STATE

------- Per Slice Level Commands (2 cases)

------- A Frame with only 1 Slice:

HCP_REF_IDX_STATE - set to provide L0 list for a P or B-Slice

HCP_REF_IDX_STATE - set to provide L1 list for a B-Slice

HCP_WEIGHTOFFSET_STATE Command - set to provide for L0 of a P or B-Slice

HCP_WEIGHTOFFSET_STATE Command - set to provide for L1 of a B-Slice

HCP_SLICE_STATE

HCP_PAK_INSERT_OBJECT - if header present at 1st slice start

    ------- A group of LCUs Per Slice

    HCP_PAK_OBJECT

    ...

HCP_PAK_INSERT_OBJECT - if tail present at frame end

MI_FLUSH - when the frame is done

------ A Frame with Multiple Slices:

HCP_REF_IDX_STATE - set to provide L0 list for a P or B-Slice

HCP_REF_IDX_STATE - set to provide L1 list for a B-Slice

HCP_WEIGHTOFFSET_STATE Command - set to provide for L0 of a P or B-Slice

HCP_WEIGHTOFFSET_STATE Command - set to provide for L1 of a B-Slice

HCP_SLICE_STATE

HCP_PAK_INSERT_OBJECT - if header present at 1st slice start of a frame

    HCP_PAK_OBJECT - a group of LCUs for a slice or a frame

    ...

HCP_PAK_INSERT_OBJECT - if tail present at slice or frame end

HCP_REF_IDX_STATE - set to provide L0 list for a P or B-Slice

HCP_REF_IDX_STATE - set to provide L1 list for a B-Slice

HCP_WEIGHTOFFSET_STATE Command - set to provide for L0 of a P or B-Slice

HCP_WEIGHTOFFSET_STATE Command - set to provide for L1 of a B-Slice

HCP_SLICE_STATE

HCP_PAK_INSERT_OBJECT - if header present at slice start

    HCP_PAK_OBJECT - a group of LCUs for a slice or a frame

...

HCP_PAK_INSERT_OBJECT - if tail present at last slice end (frame end)

MI_FLUSH - when the frame is done

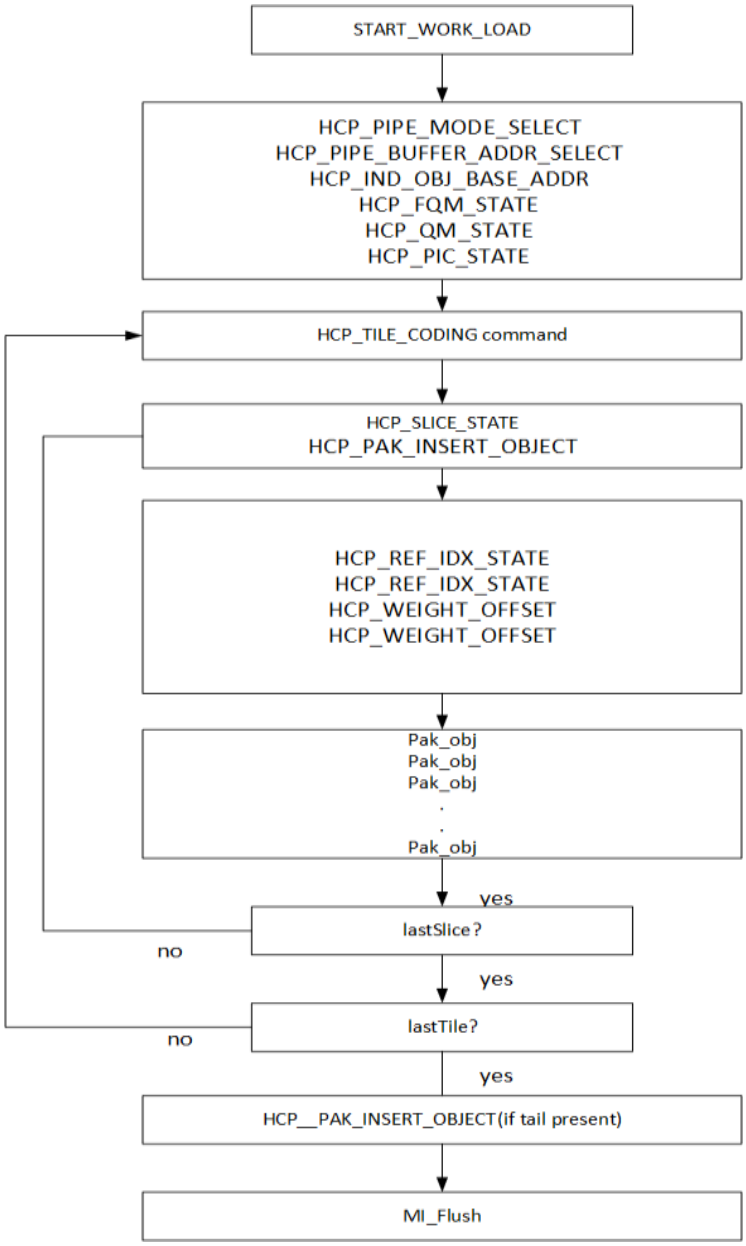
MFX_STITCH_OBJECT - a generic bitstream stitching command from MFX pipe

MI_FLUSH

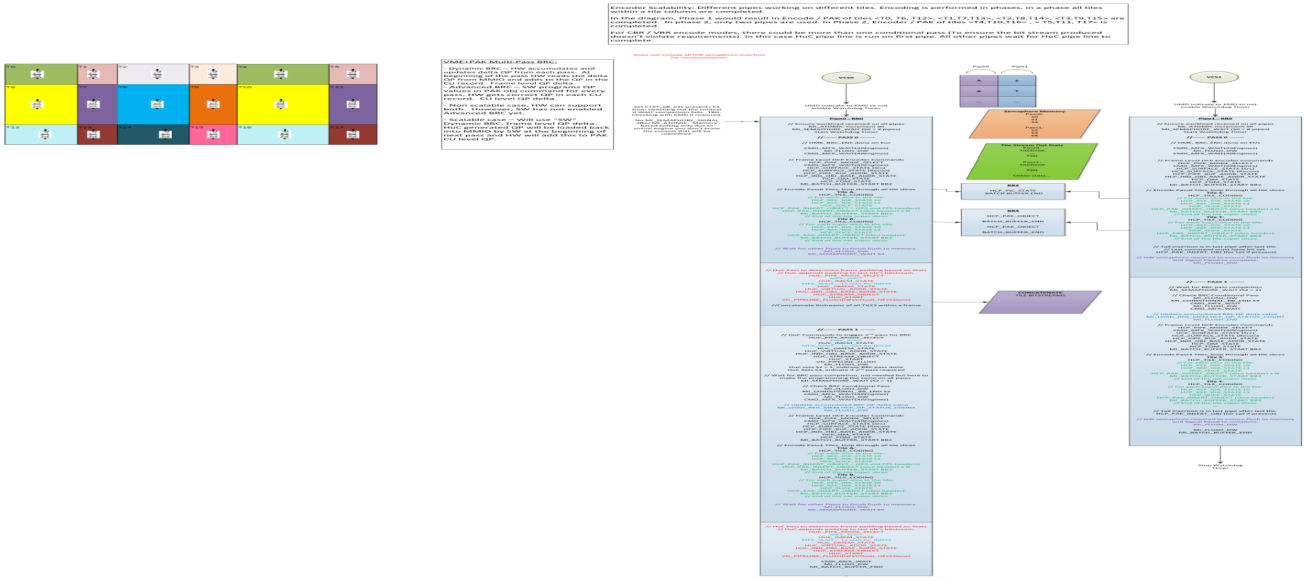
MI_FLUSH is not allowed between Slices. HEVC CABAC has simplified its operation from AVC. There is no longer a BSP_BUF_BASE_ADDR_STATE Command, as only a small local internal buffer is needed for BSP/BSE row store. THE HCP PAK_INSERT_OBJECT has been designed to support both inline and indirectly payload. Nevertheless, the MFX_STITCH_OBJECT command can still be used to stitch HEVC bitstreams together, and is run in the MFX pipe. No HEVC specific STITCH command is implemented. The SURFACE_STATE command for HEVC is redesigned and much simplified from that of MFX pipe.

Command Sequences with Tile Support

Single Pipe Mode- Following flow chart shows the command sequence when encoding frame using a single pipe(VDbox).



Multiple Pipe Mode- When encoding a frame using multiple pipes, each pipe gets a single Tile Column or multiple Tile columns depending upon Number of tile columns to encode. Following flow chart shows commands sequence in a pipe (VDbox) when multiple pipes are used for encoding.



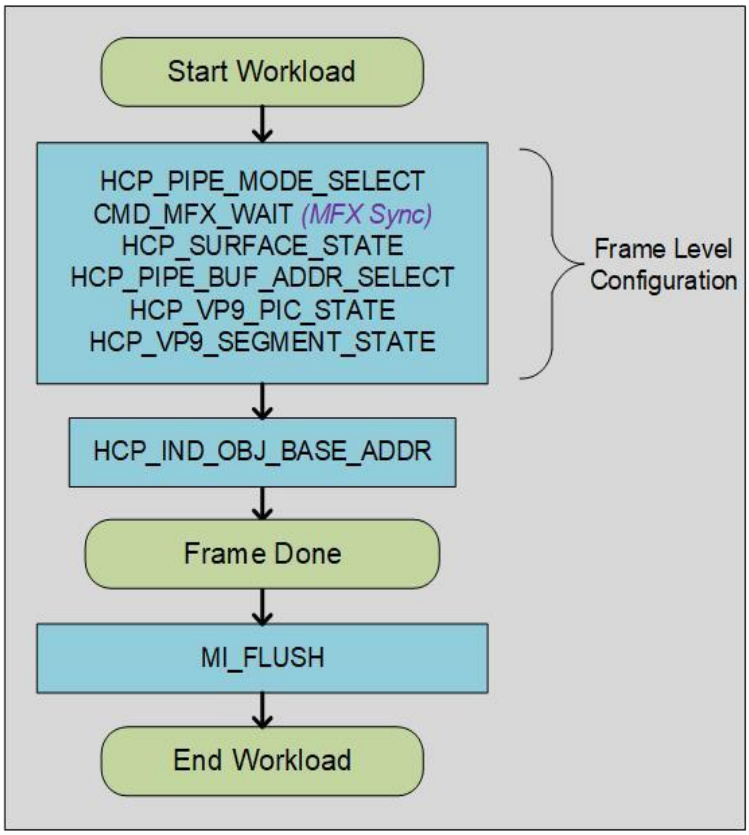
VP9 Decoder Command Sequence

VP9 decode programming is based upon a single frame decode. There are no states saved between frame decodes in the HCP. Once the bit stream DMA is configured with the HCP_IND_OBJ_BSD_OBJECT command, and the bit stream is presented to the HCP, the frame decode will begin.

VP9 Long Format Workflow Chart

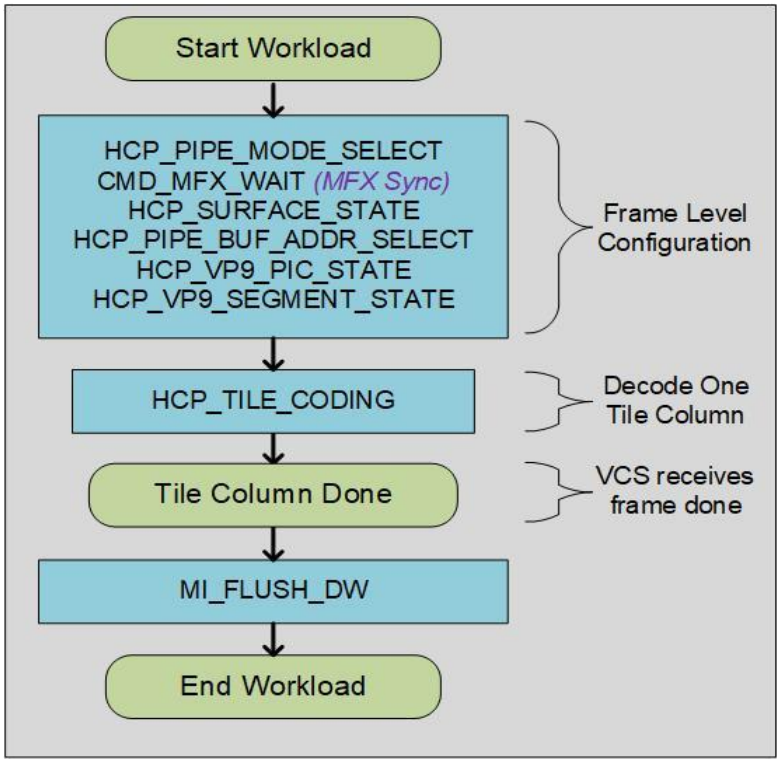
The following is the programming for single pipe decode.

The following programming also for scalable mode CABAC FE decode pass. There will be only one bitstream programming even if there are multiple tiles in the frame.



VP9 Scalable BE Only Workflow Chart

The following is the programming for scalable mode BE reconstruction pass (with multiple pipes). The following will be programmed on all the pipes.



Memory Address Attributes

This section defines the memory address attributes for the third DWord of the HCP command buffer address.

NOTE: The first DWord defines the lower address range and the second Dword defines the upper address range in the HCP command buffer address.

MemoryAddressAttributes

HCP Pipe Common Commands

The HCP Pipe Common Commands specify the HEVC Decoder pipeline level configuration.

Commands
HCP_PIPE_MODE_SELECT_VideoCS
HCP_SURFACE_STATE_VideoCS
HCP_PIPE_BUF_ADDR_STATE_VideoCS
HCP_IND_OBJ_BASE_ADDR_STATE_VideoCS
HCP_QM_STATE_VideoCS
HCP_FQM_STATE_VideoCS
HCP_TILE_CODING
VD_CONTROL_STATE

Buffer Size Requirements

HEVC Buffer Requirement

The following table indicates the buffer size in CLs per LCU. For memory allocation, the size will be the CLs per LCU * the number of LCU horizontally (if it is line) or vertically (if is column)

Mode	HEVC					
Lcu_y_size in pixels (64 for 64x64)						
Lcu_size is in number of 4x4 in the LCU per column (16 for 64x64)	8 bit			>8 bit		
	4:2:0	4:2:2	4:4:4	4:2:0	4:2:2	4:4:4
Surface						
Deblock Line (per LCU)	[(2*lcu_size*128) + 511]/512	[(2*lcu_size*128) + 511]/512	[(2*lcu_size*1.5*128) + 511]/512	[(2*lcu_size*256) + 511]/512	[(2*lcu_size*256) + 511]/512	[(2*lcu_size*1.5*256) + 511]/512
Deblock Tile Line (per LCU)	**(** [(2*lcu_size*128) + 511]/512**) * 2**	**(** [(2*lcu_size*128) + 511]/512**) * 2**	**(** [(2*lcu_size*1.5*128) + 511]/512**) * 2**	**(** [(2*lcu_size*256) + 511]/512**) * 2**	**(** [(2*lcu_size*256) + 511]/512**) * 2**	**(** [(2*lcu_size*1.5*256) + 511]/512**) * 2**
Deblock Tile Column (per LCU)	**(** [(2*lcu_size*128 + 3*128) + 511]/512**) * 2**	**(** [(2*lcu_size*1.5*128 + 3*128) + 511]/512**) * 2**	**(** [(2*lcu_size*1.5*128 + 3*128) + 511]/512**) * 2**	**(** [(2*lcu_size*256 + 3*256) + 511]/512**) * 2**	**(** [(2*lcu_size*1.5*256 + 3*256) + 511]/512**) * 2**	**(** [(2*lcu_size*1.5*256 + 3*256) + 511]/512**) * 2**
Top Right Motion Vector Tile Column (per LCU)	1	1	1	1	1	1
Motion Vector Line (per LCU)	LCU16/32 : 1 LCU64:2	LCU16/32 : 1 LCU64:2	LCU16/32 : 1 LCU64:2	LCU16/32 : 1 LCU64:2	LCU16/32 : 1 LCU64:2	LCU16/32 : 1 LCU64:2
Motion Vector Tile Line (per LCU)	LCU16/32 : 2 LCU64 : 4	LCU16/32 : 2 LCU64 : 4	LCU16/32 : 2 LCU64 : 4	LCU16/32 : 2 LCU64 : 4	LCU16/32 : 2 LCU64 : 4	LCU16/32 : 2 LCU64 : 4
Right Motion Vector Tile Column(per LCU)	LCU16/32 : 2 LCU64 : 4	LCU16/32 : 2 LCU64 : 4	LCU16/32 : 2 LCU64 : 4	LCU16/32 : 2 LCU64 : 4	LCU16/32 : 2 LCU64 : 4	LCU16/32 : 2 LCU64 : 4
Top Right Neighbor(pe	1 CL	1 CL	LCU16 : 1 LCU32/64 : 2	LCU16 : 1 LCU32/64 : 2	LCU16 : 1 LCU32/64 : 2	LCU16 : 2 LCU32/64 : 3

r LCU)						
HPR Left Recon Column(per LCU)	LCU16/32 : 1 LCU64 : 2	LCU16:1 LCU32:2 LCU64:3	LCU16:1 LCU32:2 LCU64:3	LCU16:1 LCU32:2 LCU64:4	LCU16:2 LCU32:3 LCU64:6	LCU16:2 LCU32:3 LCU64:6
HSF						
	8/10 bit			12 bit		
	4:2:0	4:2:2	4:4:4	4:2:0	4:2:2	4:4:4
Surface						
SAO Line (per LCU)	LCU16 : 2 LCU32 : 3 LCU64: 5	LCU16 : 2 LCU32 : 3 LCU64: 5	LCU16 : 3 LCU32 : 4 LCU64: 7	LCU16 : 2 LCU32 : 4 LCU64: 6	LCU16 : 2 LCU32 : 4 LCU64: 6	[LCU16 : 3 LCU32 : 5 LCU64: 8
SAO Tile Line (per LCU)	LCU16 : **4** LCU32 : **6** LCU64 : **10**	LCU16 : **4** LCU32 : **6** LCU64 : **10**	LCU16 : **6** LCU32 : **8** LCU64 : **14**	LCU16 : **4** LCU32 : **8** LCU64 : **12**	LCU16 : **4** LCU32 : **8** LCU64 : **12**	[LCU16 : **6** LCU32 : **10** LCU64 : **16**
SAO Tile Column (per LCU)	LCU16 : **8** LCU32 : **10** LCU64 : **18**	LCU16 : **10** LCU32 : **14** LCU64 : **24**	LCU16 : **10** LCU32 : **14** LCU64 : **24**	[LCU16 : **8** LCU32 : **10** LCU64 : **18**	LCU16 : **10** LCU32 : **14** LCU64 : **24**	LCU16 : **10** LCU32 : **14** LCU64 : **24**

The following table indicates the buffer size in CLs for each row of frame or tile column.

Mode	HEVC					
	8 bit			>8 bit		
	4:2:0	4:2:2	4:4:4	4:2:0	4:2:2	4:4:4
Surface						
HSSE(per row of frame or tile column)	16*(frame/tile_column width_in_lcu + 3)	16*(frame/tile_column width_in_lcu + 3)	16*(frame/tile_column width_in_lcu + 3)	16*(frame/tile_column width_in_lcu + 3)	16*(frame/tile_column width_in_lcu + 3)	16*(frame/tile_column width_in_lcu + 3)
HSAO(per row of frame or tile column)	(frame/tile_column width_in_lcu + 3)/4	(frame/tile column width_in_lcu + 3)/4	(frame(tile_column)_width_in_lcu + 3)/4	(frame/tile_column width_in_lcu + 3)/4	(frame/tile_column width_in_lcu + 3)/4	(frame/tile_column width_in_lcu + 3)/4

VP9 Buffer Size Requirements

The following table indicates the buffer size in CLs per LCU. For memory allocation, the size will be the CLs per LCU * the number of LCU horizontally (if it is line) or vertically (if is column)

Mode	VP9					
	8 bits			> 8 bits		
	4:2:0	4:2:2	4:4:4	4:2:0	4:2:2	4:4:4
Deblock Line (per SB64)	18	18	27	36	36	54
Deblock Tile Line (per SB64)	18	18	27	36	36	54
Deblock Tile Column (per SB64)	17	25	25	34	50	50
Top Right Motion Vector Tile Column (per LCU)	NA	NA	NA	NA	NA	NA
Right Motion Vector Line	5	5	5	5	5	5
Right Motion Vector Tile Line	5	5	5	5	5	5
Right Motion Vector Tile Column	NA	NA	NA	NA	NA	NA
HPR Left Recon Column(per LCU)	2	3	3	4	6	6
Top Right Neighbor	1	1	1	1	1	1
HSAO	NA	NA	NA	NA	NA	NA
VP9 HVD Line Rowstore (per SB64)	1	1	1	1	1	1
VP9 HVD Tile Rowstore (per SB64)	1	1	1	1	1	1
VP9 Probability buffer (per frame)	32	32	32	32	32	32

The following table indicates the buffer size in CLs for the each row of frame or tile column.

Mode	VP9					
	8 bit			> 8 bits		
	4:2:0	4:2:2	4:4:4	4:2:0	4:2:2	4:4:4
HSSE	32*(frame_width_in_sb64 + 3)	32*(frame_width_in_sb64 + 3)	32*(frame_width_in_sb64 + 3)	32*(frame_width_in_sb64 + 3)	32*(frame_width_in_sb64 + 3)	32*(frame_width_in_sb64 + 3)

The following table indicates the buffer size of each buffer for the whole frame. These data will be used across frames.

Mode	VP9					
	8 bit			> 8 bits		
	4:2:0	4:2:2	4:4:4	4:2:0	4:2:2	4:4:4
Current Motion Vector Temporal Buffer	(num_width_in_SB * num_height_in_SB) * 9	(num_width_in_SB * num_height_in_SB) * 9	(num_width_in_SB * num_height_in_SB) * 9	(num_width_in_SB * num_height_in_SB) * 9	(num_width_in_SB * num_height_in_SB) * 9	(num_width_in_SB * num_height_in_SB) * 9
Collocated	(num_width_in_SB *	(num_width_in_SB *	(num_width_in_SB *	(num_width_in_SB *	(num_width_in_SB *	(num_width_in_SB *

Mode	VP9					
	8 bit			> 8 bits		
	4:2:0	4:2:2	4:4:4	4:2:0	4:2:2	4:4:4
Motion Vector Temporal Buffer	num_height_in_SB) * 9	num_height_in_SB) * 9	num_height_in_SB) * 9	num_height_in_SB) * 9	num_height_in_SB) * 9	num_height_in_SB) * 9
VP9 Probability Buffer	32	32	32	32	32	32
VP9 Segment ID buffer (per frame)	(frame_width_in_sb64 * frame_height_in_sb64	(frame_width_in_sb64 * frame_height_in_sb64	(frame_width_in_sb64 * frame_height_in_sb64	(frame_width_in_sb64 * frame_height_in_sb64	(frame_width_in_sb64 * frame_height_in_sb64	(frame_width_in_sb64 * frame_height_in_sb64

Internal Media Rowstore table - If the internal Media Rowstore exists, driver should use the storage as the following table indicates.

This is for HEVC VMM setting. FrameWidth means frame width in picture width in decode/encode mode.

HEVC				Enable Setting					Addr Setting				
ArrayType	Bitdepth	LCU Size	FrameWidth	Meta/MV	Deblock	SAO	HSAO	VDEnc	DAT	DF	SAO	HSAO	VDEnc
420/422	8/10/12	16	<= 4096	Y	Y	Y	Y	N	0	256	1280	2048	N/A
		32/64	<= 4096	Y	Y	Y	Y	Y	0	256	1280	1792	1824
	8/10/12	16	4097 - 8192	Y	Y	N	N	N	0	512	N/A	N/A	N/A
		32/64	4097 - 8192	Y	Y	N	N	Y	0	256	N/A	N/A	2304
444	8	16	<= 4096	Y	Y	Y	Y	Y	0	256	1024	1792	N/A
			4097 - 8192	Y	Y	N	Y	N	0	512	N/A	2048	N/A
	10		<= 4096	Y	Y	Y	N	N	0	256	1792	N/A	N/A
			4097 - 8192	Y	N	Y	Y	N	0	N/A	512	2048	N/A
	12		<= 4096	Y	Y	Y	N	N	0	256	1792	N/A	N/A
			4097 - 8192	Y	N	Y	Y	N	0	N/A	256	1792	N/A

	8	32/64	<= 4096	Y	Y	Y	Y	Y	0	256	1024	1536	1568
			4097 - 8192	Y	Y	N	Y	Y	0	512	N/A	2048	2112
	10		<= 4096	Y	Y	Y	Y	Y	0	256	1792	2304	2336
			4097 - 8192	Y	N	Y	Y	Y	0	N/A	512	1536	1600
	12		<= 4096	Y	Y	Y	Y	Y	0	128	1664	2304	2336
			4097 - 8192	Y	N	Y	Y	Y	0	N/A	256	1536	1600

The following table is for VP9 VMM setting. FrameWidth means frame width in picture for decode/encode mode

VP9				Enable Setting				Addr Setting			
ArrayType	**Bitdepth**	**LCU Size**	**FrameWidth**	**HVD**	**Meta/MV**	**Deblock**	**VDENC**	**HVD**	**Meta/MV**	**Deblock**	**VDEnc**
420	8	64	<= 4096	Y	Y	Y	Y	0	64	384	1536
		64	4097 - 8192	N	N	Y	Y	N/A	N/A	0	2304
	10/12	64	<= 4096	Y	N	Y	Y	0	N/A	64	2368
		64	4097 - 8192	Y	Y	N	Y	0	128	N/A	768
422	8	64	<= 4096	Y	Y	Y	Y	0	64	384	1536
		64	4097 - 8192	N	N	Y	Y	N/A	N/A	0	2304
	10/12	64	<= 4096	N	N	Y	N	N/A	N/A	0	N/A
		64	4097 - 8192	Y	Y	N	Y	0	128	N/A	768
444	8	64	<= 4096	Y	Y	Y	Y	0	64	384	2112
		64	4097 - 8192	Y	Y	N	Y	0	128	N/A	768
	10/12	64	<= 2048	Y	Y	Y	Y	0	32	192	1920
		64	2049 - 4096	Y	Y	N	Y	0	128	N/A	768
		64	4097 - 8192	Y	Y	N	Y	0	128	N/A	768

VP9 Common Commands

Commands
HCP_PIPE_MODE_SELECT
HCP_SURFACE_STATE
HCP_PIPE_BUF_ADDR_STATE
HCP_IND_OBJ_BASE_ADDR_STATE
HCP_VP9_SEGMENT_STATE
HCP_VP9_PIC_STATE

HCP Common Commands

HCP Common Commands
HCP_PIC_STATE
HCP_TILE_STATE
HCP_REF_IDX_STATE
HCP_WEIGHTOFFSET_STATE
HCP_SLICE_STATE
HEVC_VP9_RDOQ_STATE
HCP_BSD_OBJECT
HCP_PAK_OBJECT
HCP_PAK_INSERT_OBJECT
HCP_PALETTE_INITIALIZER_STATE

HCP and VP9 Commands

HCP_BSD_OBJECT (triggers HW start)

HCP_VP9_PAK_OBJECT

Tile Size and CU Stream-out Records

HEVC: Streamout0 cacheline is composed of 4 quarter cachelines, each containing information on CU skip flag, coding block flag for the TUs in a PU, residual/coefficient bit count for a PU, total bit count for CU, SB exceed limit flag. A typical streamout0 cacheline, therefore, has information on statistics for 4 PUs and Super Block exceed limit flag.

Pak pipeline streamout enable bit, set by HCP_PIPE_MODE_SELECT command, enables or disables the streamout.

Programming Note				
Context:		CU level statistics		
Level	**Field**	**Width**	**Cacheline**	**Comment**
PU	PU Skip Flag	1	qcacheline[0]	Packed in Quarter Cacheline in PU format
SB	SB exceed limit	1	qcacheline[1]	Packed in Quarter Cacheline in PU format (valid on last PU of SB)
	Reserved	14	qcacheline[15:2	Reserved
PU	TU CBF Y/U/V	48	qcacheline[63:16]	Packed in Quarter Cacheline in PU format
PU	PU Coefficient Bit Count (Only residual)	18	qcacheline[81:64]	Packed in Quarter Cacheline in PU format
PU	PU Bit Count (all PU Syntax)	18	qcacheline[113:96]	Packed in Quarter Cacheline in PU format
	Reserved	14	qcacheline[127:114]	Reserved

Programming Note

HEVC Streamout 1: Per Tile Quarter Cacheline

Level	Field	Width	Cacheline	Comment
Tile	Tile Bit Count (header + data + tail)	32	cacheline[31:0]	
	Reserved(MBZ)	32	cacheline[63:32]	
	TilePositionX[15:0]	16	cacheline[79:64]	
	TilePositionY[15:0]	16	cacheline[95:80]	
	Reserved(MBZ)	32	cacheline[127:96]	

VP9: CU statistics record (individual PUs per record down to 8x8 only)

Fields	Bits	
Skip	3:0	Indicates Skip flag Group 4 4x4s -> 4 bits
InterMode	11:4	InterMode: 0 NEARESTMV, 1 NEARMV, 2 ZEROMV, 3NEWMV Group 4 4x4s total 8 bits
Reserved	15:12	
NZ coeff count	28:16	Number of non-zero coeffs; sum of YUV, 13bits
Reserved	31:29	
NumBitsforCoeffs	47:32	Number of Bits for coefficients per block, 16bits
NumBitsforBlock	63:48	Number of Bits in block

Stream-in Probability Table

In Encoder mode, two sets of this table will be streamed out: one for the current frame probability update and one for future frame.

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT		State counter EBB Address	Coefficient counter EBB Address							
												4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
CL aligned	0	1	tx_probs_8x8 [0] [0..0]	100	100	0	0	0	0	MODE COUNTERS (counts tx)	0-17								
	1	1	tx_probs_8x8 [1] [0..0]	66	66	0	1	1											
	2	2	tx_probs_16x16 [0] [0..1]	20, 152	20, 152	0	2	2											
	4	2	tx_probs_16x16 [1] [0..1]	15, 101	15, 101	0	4	4											
	6	3	tx_probs_32x32 [0] [0..2]	3, 136, 37	3, 136, 37	0	6	6											
	9	3	tx_probs_32x32 [1] [0..2]	5, 52, 13	5, 52, 13	0	9	9											
	12	52	DUMMY	0, 0, 0, 0	0, 0, 0, 0	52	12												
CL aligned	64	3	coef_probs_4x4 [0] [0] [0] [0] [0..2]	195, 29, 183	195, 29, 183	0	12		8	COEFF COUNTERS (coeff_count_model_coeff)		0-287							
	67	3	coef_probs_4x4 [0] [0] [0] [1] [0..2]	84, 49, 136	84, 49, 136	0	15												

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	70	3	coef_probs_4x4 [0] [0] [0] [2] [0..2]	8, 42, 71	8, 42, 71	0	18											
	73	3	coef_probs_4x4 [0] [0] [1] [0] [0..2]	31, 107, 169	31, 107, 169	0	21											
	76	3	coef_probs_4x4 [0] [0] [1] [1] [0..2]	35, 99, 159	35, 99, 159	0	24											
	79	3	coef_probs_4x4 [0] [0] [1] [2] [0..2]	17, 82, 140	17, 82, 140	0	27											
	82	3	coef_probs_4x4 [0] [0] [1] [3] [0..2]	8, 66, 114	8, 66, 114	0	30											
	85	3	coef_probs_4x4 [0] [0] [1] [4] [0..2]	2, 44, 76	2, 44, 76	0	33											
	88	3	coef_probs_4x4 [0] [0] [1] [5] [0..2]	1, 19, 32	1, 19, 32	0	36											
	91	3	coef_probs_4x4 [0] [0] [2] [0] [0..2]	40, 132, 201	40, 132, 201	0	39											
	94	3	coef_probs_4x4 [0] [0] [2] [1]	29, 114, 187	29, 11	0	42											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT		State counter EBB Address	Coefficient counter EBB Address							
												4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			[0..2]		4, 187														
	97	3	coef_probs_4x4 [0] [0] [2] [2] [0..2]	13, 91, 157	13, 91, 157	0	45												
	100	3	coef_probs_4x4 [0] [0] [2] [3] [0..2]	7, 75, 127	7, 75, 127	0	48												
	103	3	coef_probs_4x4 [0] [0] [2] [4] [0..2]	3, 58, 95	3, 58, 95	0	51												
	106	3	coef_probs_4x4 [0] [0] [2] [5] [0..2]	1, 28, 47	1, 28, 47	0	54												
	109	3	coef_probs_4x4 [0] [0] [3] [0] [0..2]	69, 142, 221	69, 142, 221	0	57												
	112	3	coef_probs_4x4 [0] [0] [3] [1] [0..2]	42, 122, 201	42, 122, 201	0	60												
	115	3	coef_probs_4x4 [0] [0] [3] [2] [0..2]	15, 91, 159	15, 91, 159	0	63												
	118	3	coef_probs_4x4 [0] [0] [3] [3] [0..2]	6, 67, 121	6, 67, 12	0	66												

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					1													
	121	3	coef_probs_4x4 [0] [0] [3] [4] [0..2]	1, 42, 77	1, 42, 77	0	69											
	124	3	coef_probs_4x4 [0] [0] [3] [5] [0..2]	1, 17, 31	1, 17, 31	0	72											
	127	3	coef_probs_4x4 [0] [0] [4] [0] [0..2]	102, 148, 228	102, 148, 228	0	75											
	130	3	coef_probs_4x4 [0] [0] [4] [1] [0..2]	67, 117, 204	67, 117, 204	0	78											
	133	3	coef_probs_4x4 [0] [0] [4] [2] [0..2]	17, 82, 154	17, 82, 154	0	81											
	136	3	coef_probs_4x4 [0] [0] [4] [3] [0..2]	6, 59, 114	6, 59, 114	0	84											
	139	3	coef_probs_4x4 [0] [0] [4] [4] [0..2]	2, 39, 75	2, 39, 75	0	87											
	142	3	coef_probs_4x4 [0] [0] [4] [5] [0..2]	1, 15, 29	1, 15, 29	0	90											
	145	3	coef_probs_4x4 [0] [0] [5] [0]	156, 57, 233	156, 57	0	93											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			[0..2]		, 233													
	148	3	coef_probs_4x4 [0] [0] [5] [1] [0..2]	119, 57, 212	119, 57, 212	0	96											
	151	3	coef_probs_4x4 [0] [0] [5] [2] [0..2]	58, 48, 163	58, 48, 163	0	99											
	154	3	coef_probs_4x4 [0] [0] [5] [3] [0..2]	29, 40, 124	29, 40, 124	0	102											
	157	3	coef_probs_4x4 [0] [0] [5] [4] [0..2]	12, 30, 81	12, 30, 81	0	105											
	160	3	coef_probs_4x4 [0] [0] [5] [5] [0..2]	3, 12, 31	3, 12, 31	0	108											
	163	3	coef_probs_4x4 [0] [1] [0] [0] [0..2]	191, 107, 226	191, 107, 226	0	111											
	166	3	coef_probs_4x4 [0] [1] [0] [1] [0..2]	124, 117, 204	124, 117, 204	0	114											
	169	3	coef_probs_4x4	25, 99,	25,	0	117											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			[0] [1] [0] [2] [0..2]	155	99, 155													
	172	3	coef_probs_4x4 [0] [1] [1] [0] [0..2]	29, 148, 210	29, 148, 210	0	120											
	175	3	coef_probs_4x4 [0] [1] [1] [1] [0..2]	37, 126, 194	37, 126, 194	0	123											
	178	3	coef_probs_4x4 [0] [1] [1] [2] [0..2]	8, 93, 157	8, 93, 157	0	126											
	181	3	coef_probs_4x4 [0] [1] [1] [3] [0..2]	2, 68, 118	2, 68, 118	0	129											
	184	3	coef_probs_4x4 [0] [1] [1] [4] [0..2]	1, 39, 69	1, 39, 69	0	132											
	187	3	coef_probs_4x4 [0] [1] [1] [5] [0..2]	1, 17, 33	1, 17, 33	0	135											
	190	3	coef_probs_4x4 [0] [1] [2] [0] [0..2]	41, 151, 213	41, 151, 213	0	138											
	193	3	coef_probs_4x4 [0] [1] [2] [1] [0..2]	27, 123, 193	27, 123,	0	141											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					193													
	196	3	coef_probs_4x4 [0] [1] [2] [2] [0..2]	3, 82, 144	3, 82, 144	0	144											
	199	3	coef_probs_4x4 [0] [1] [2] [3] [0..2]	1, 58, 105	1, 58, 105	0	147											
	202	3	coef_probs_4x4 [0] [1] [2] [4] [0..2]	1, 32, 60	1, 32, 60	0	150											
	205	3	coef_probs_4x4 [0] [1] [2] [5] [0..2]	1, 13, 26	1, 13, 26	0	153											
	208	3	coef_probs_4x4 [0] [1] [3] [0] [0..2]	59, 159, 220	59, 159, 220	0	156											
	211	3	coef_probs_4x4 [0] [1] [3] [1] [0..2]	23, 126, 198	23, 126, 198	0	159											
	214	3	coef_probs_4x4 [0] [1] [3] [2] [0..2]	4, 88, 151	4, 88, 151	0	162											
	217	3	coef_probs_4x4 [0] [1] [3] [3] [0..2]	1, 66, 114	1, 66, 114	0	165											
	220	3	coef_probs_4x	1, 38,	1,	0	16											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			4 [0] [1] [3] [4] [0..2]	71	38, 71		8											
	223	3	coef_probs_4x4 [0] [1] [3] [5] [0..2]	1, 18, 34	1, 18, 34	0	171											
	226	3	coef_probs_4x4 [0] [1] [4] [0] [0..2]	114, 136, 232	114, 136, 232	0	174											
	229	3	coef_probs_4x4 [0] [1] [4] [1] [0..2]	51, 114, 207	51, 114, 207	0	177											
	232	3	coef_probs_4x4 [0] [1] [4] [2] [0..2]	11, 83, 155	11, 83, 155	0	180											
	235	3	coef_probs_4x4 [0] [1] [4] [3] [0..2]	3, 56, 105	3, 56, 105	0	183											
	238	3	coef_probs_4x4 [0] [1] [4] [4] [0..2]	1, 33, 65	1, 33, 65	0	186											
	241	3	coef_probs_4x4 [0] [1] [4] [5] [0..2]	1, 17, 34	1, 17, 34	0	189											
	244	3	coef_probs_4x4 [0] [1] [5] [0] [0..2]	149, 65, 234	149, 65, 234	0	192											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	247	3	coef_probs_4x4 [0] [1] [5] [1] [0..2]	121, 57, 215	121, 57, 215	0	195											
	250	3	coef_probs_4x4 [0] [1] [5] [2] [0..2]	61, 49, 166	61, 49, 166	0	198											
	253	3	coef_probs_4x4 [0] [1] [5] [3] [0..2]	28, 36, 114	28, 36, 114	0	201											
	256	3	coef_probs_4x4 [0] [1] [5] [4] [0..2]	12, 25, 76	12, 25, 76	0	204											
	259	3	coef_probs_4x4 [0] [1] [5] [5] [0..2]	3, 16, 42	3, 16, 42	0	207											
	262	3	coef_probs_4x4 [1] [0] [0] [0] [0..2]	214, 49, 220	214, 49, 220	0	210					0-287						
	265	3	coef_probs_4x4 [1] [0] [0] [1] [0..2]	132, 63, 188	132, 63, 188	0	213											
	268	3	coef_probs_4x4 [1] [0] [0] [2] [0..2]	42, 65, 137	42, 65, 13	0	216											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					7													
	271	3	coef_probs_4x4 [1] [0] [1] [0] [0..2]	85, 137, 221	85, 137, 221	0	219											
	274	3	coef_probs_4x4 [1] [0] [1] [1] [0..2]	104, 131, 216	104, 131, 216	0	222											
	277	3	coef_probs_4x4 [1] [0] [1] [2] [0..2]	49, 111, 192	49, 111, 192	0	225											
	280	3	coef_probs_4x4 [1] [0] [1] [3] [0..2]	21, 87, 155	21, 87, 155	0	228											
	283	3	coef_probs_4x4 [1] [0] [1] [4] [0..2]	2, 49, 87	2, 49, 87	0	231											
	286	3	coef_probs_4x4 [1] [0] [1] [5] [0..2]	1, 16, 28	1, 16, 28	0	234											
	289	3	coef_probs_4x4 [1] [0] [2] [0] [0..2]	89, 163, 230	89, 163, 230	0	237											
	292	3	coef_probs_4x4 [1] [0] [2] [1] [0..2]	90, 137, 220	90, 137, 22	0	240											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults					Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
												4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					0														
	295	3	coef_probs_4x4 [1] [0] [2] [2] [0..2]	29, 100, 183	29, 100, 183	0	243												
	298	3	coef_probs_4x4 [1] [0] [2] [3] [0..2]	10, 70, 135	10, 70, 135	0	246												
	301	3	coef_probs_4x4 [1] [0] [2] [4] [0..2]	2, 42, 81	2, 42, 81	0	249												
	304	3	coef_probs_4x4 [1] [0] [2] [5] [0..2]	1, 17, 33	1, 17, 33	0	252												
	307	3	coef_probs_4x4 [1] [0] [3] [0] [0..2]	108, 167, 237	108, 167, 237	0	255												
	310	3	coef_probs_4x4 [1] [0] [3] [1] [0..2]	55, 133, 222	55, 133, 222	0	258												
	313	3	coef_probs_4x4 [1] [0] [3] [2] [0..2]	15, 97, 179	15, 97, 179	0	261												
	316	3	coef_probs_4x4 [1] [0] [3] [3] [0..2]	4, 72, 135	4, 72, 135	0	264												

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	319	3	coef_probs_4x4 [1] [0] [3] [4] [0..2]	1, 45, 85	1, 45, 85	0	267											
	322	3	coef_probs_4x4 [1] [0] [3] [5] [0..2]	1, 19, 38	1, 19, 38	0	270											
	325	3	coef_probs_4x4 [1] [0] [4] [0] [0..2]	124, 146, 240	124, 146, 240	0	273											
	328	3	coef_probs_4x4 [1] [0] [4] [1] [0..2]	66, 124, 224	66, 124, 224	0	276											
	331	3	coef_probs_4x4 [1] [0] [4] [2] [0..2]	17, 88, 175	17, 88, 175	0	279											
	334	3	coef_probs_4x4 [1] [0] [4] [3] [0..2]	4, 58, 122	4, 58, 122	0	282											
	337	3	coef_probs_4x4 [1] [0] [4] [4] [0..2]	1, 36, 75	1, 36, 75	0	285											
	340	3	coef_probs_4x4 [1] [0] [4] [5] [0..2]	1, 18, 37	1, 18, 37	0	288											
	343	3	coef_probs_4x4 [1] [0] [5] [0] [0..2]	141, 79, 241	141, 79, 24	0	291											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					1													
	346	3	coef_probs_4x4 [1] [0] [5] [1] [0..2]	126, 70, 227	126, 70, 227	0	294											
	349	3	coef_probs_4x4 [1] [0] [5] [2] [0..2]	66, 58, 182	66, 58, 182	0	297											
	352	3	coef_probs_4x4 [1] [0] [5] [3] [0..2]	30, 44, 136	30, 44, 136	0	300											
	355	3	coef_probs_4x4 [1] [0] [5] [4] [0..2]	12, 34, 96	12, 34, 96	0	303											
	358	3	coef_probs_4x4 [1] [0] [5] [5] [0..2]	2, 20, 47	2, 20, 47	0	306											
	361	3	coef_probs_4x4 [1] [1] [0] [0] [0..2]	229, 99, 249	229, 99, 249	0	309											
	364	3	coef_probs_4x4 [1] [1] [0] [1] [0..2]	143, 111, 235	143, 111, 235	0	312											
	367	3	coef_probs_4x4 [1] [1] [0] [2] [0..2]	46, 109, 192	46, 109,	0	315											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					192													
	370	3	coef_probs_4x4 [1] [1] [1] [0] [0..2]	82, 158, 236	82, 158, 236	0	318											
	373	3	coef_probs_4x4 [1] [1] [1] [1] [0..2]	94, 146, 224	94, 146, 224	0	321											
	376	3	coef_probs_4x4 [1] [1] [1] [2] [0..2]	25, 117, 191	25, 117, 191	0	324											
	379	3	coef_probs_4x4 [1] [1] [1] [3] [0..2]	9, 87, 149	9, 87, 149	0	327											
	382	3	coef_probs_4x4 [1] [1] [1] [4] [0..2]	3, 56, 99	3, 56, 99	0	330											
	385	3	coef_probs_4x4 [1] [1] [1] [5] [0..2]	1, 33, 57	1, 33, 57	0	333											
	388	3	coef_probs_4x4 [1] [1] [2] [0] [0..2]	83, 167, 237	83, 167, 237	0	336											
	391	3	coef_probs_4x4 [1] [1] [2] [1] [0..2]	68, 145, 222	68, 145, 22	0	339											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					2													
	394	3	coef_probs_4x4 [1] [1] [2] [2] [0..2]	10, 103, 177	10, 103, 177	0	342											
	397	3	coef_probs_4x4 [1] [1] [2] [3] [0..2]	2, 72, 131	2, 72, 131	0	345											
	400	3	coef_probs_4x4 [1] [1] [2] [4] [0..2]	1, 41, 79	1, 41, 79	0	348											
	403	3	coef_probs_4x4 [1] [1] [2] [5] [0..2]	1, 20, 39	1, 20, 39	0	351											
	406	3	coef_probs_4x4 [1] [1] [3] [0] [0..2]	99, 167, 239	99, 167, 239	0	354											
	409	3	coef_probs_4x4 [1] [1] [3] [1] [0..2]	47, 141, 224	47, 141, 224	0	357											
	412	3	coef_probs_4x4 [1] [1] [3] [2] [0..2]	10, 104, 178	10, 104, 178	0	360											
	415	3	coef_probs_4x4 [1] [1] [3] [3] [0..2]	2, 73, 133	2, 73, 133	0	363											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	418	3	coef_probs_4x4 [1] [1] [3] [4] [0..2]	1, 44, 85	1, 44, 85	0	366											
	421	3	coef_probs_4x4 [1] [1] [3] [5] [0..2]	1, 22, 47	1, 22, 47	0	369											
	424	3	coef_probs_4x4 [1] [1] [4] [0] [0..2]	127, 145, 243	127, 145, 243	0	372											
	427	3	coef_probs_4x4 [1] [1] [4] [1] [0..2]	71, 129, 228	71, 129, 228	0	375											
	430	3	coef_probs_4x4 [1] [1] [4] [2] [0..2]	17, 93, 177	17, 93, 177	0	378											
	433	3	coef_probs_4x4 [1] [1] [4] [3] [0..2]	3, 61, 124	3, 61, 124	0	381											
	436	3	coef_probs_4x4 [1] [1] [4] [4] [0..2]	1, 41, 84	1, 41, 84	0	384											
	439	3	coef_probs_4x4 [1] [1] [4] [5] [0..2]	1, 21, 52	1, 21, 52	0	387											
	442	3	coef_probs_4x4 [1] [1] [5] [0] [0..2]	157, 78, 244	157, 78, 24	0	390											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					4													
	445	3	coef_probs_4x4 [1] [1] [5] [1] [0..2]	140, 72, 231	140, 72, 231	0	393											
	448	3	coef_probs_4x4 [1] [1] [5] [2] [0..2]	69, 58, 184	69, 58, 184	0	396											
	451	3	coef_probs_4x4 [1] [1] [5] [3] [0..2]	31, 44, 137	31, 44, 137	0	399											
	454	3	coef_probs_4x4 [1] [1] [5] [4] [0..2]	14, 38, 105	14, 38, 105	0	402											
	457	3	coef_probs_4x4 [1] [1] [5] [5] [0..2]	8, 23, 61	8, 23, 61	0	405											
	460	3	coef_probs_8x8 [0] [0] [0] [0] [0..2]	125, 34, 187	125, 34, 187	0	408		57.5				0-287					
	463	3	coef_probs_8x8 [0] [0] [0] [1] [0..2]	52, 41, 133	52, 41, 133	0	411											
	466	3	coef_probs_8x8 [0] [0] [0] [2]	6, 31, 56	6, 31,	0	414											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			[0..2]		56													
	469	3	coef_probs_8x8 [0] [0] [1] [0] [0..2]	37, 109, 153	37, 109, 153	0	417											
	472	3	coef_probs_8x8 [0] [0] [1] [1] [0..2]	51, 102, 147	51, 102, 147	0	420											
	475	3	coef_probs_8x8 [0] [0] [1] [2] [0..2]	23, 87, 128	23, 87, 128	0	423											
	478	3	coef_probs_8x8 [0] [0] [1] [3] [0..2]	8, 67, 101	8, 67, 101	0	426											
	481	3	coef_probs_8x8 [0] [0] [1] [4] [0..2]	1, 41, 63	1, 41, 63	0	429											
	484	3	coef_probs_8x8 [0] [0] [1] [5] [0..2]	1, 19, 29	1, 19, 29	0	432											
	487	3	coef_probs_8x8 [0] [0] [2] [0] [0..2]	31, 154, 185	31, 154, 185	0	435											
	490	3	coef_probs_8x8 [0] [0] [2] [1] [0..2]	17, 127, 175	17, 127, 175	0	438											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	493	3	coef_probs_8x8 [0] [0] [2] [2] [0..2]	6, 96, 145	6, 96, 145	0	441											
	496	3	coef_probs_8x8 [0] [0] [2] [3] [0..2]	2, 73, 114	2, 73, 114	0	444											
	499	3	coef_probs_8x8 [0] [0] [2] [4] [0..2]	1, 51, 82	1, 51, 82	0	447											
	502	3	coef_probs_8x8 [0] [0] [2] [5] [0..2]	1, 28, 45	1, 28, 45	0	450											
	505	3	coef_probs_8x8 [0] [0] [3] [0] [0..2]	23, 163, 200	23, 163, 200	0	453											
	508	3	coef_probs_8x8 [0] [0] [3] [1] [0..2]	10, 131, 185	10, 131, 185	0	456											
	511	3	coef_probs_8x8 [0] [0] [3] [2] [0..2]	2, 93, 148	2, 93, 148	0	459											
	514	3	coef_probs_8x8 [0] [0] [3] [3] [0..2]	1, 67, 111	1, 67, 111	0	462											
	517	3	coef_probs_8x8 [0] [0] [3] [4] [0..2]	1, 41, 69	1, 41, 69	0	465											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	520	3	coef_probs_8x8 [0] [0] [3] [5] [0..2]	1, 14, 24	1, 14, 24	0	468											
	523	3	coef_probs_8x8 [0] [0] [4] [0] [0..2]	29, 176, 217	29, 176, 217	0	471											
	526	3	coef_probs_8x8 [0] [0] [4] [1] [0..2]	12, 145, 201	12, 145, 201	0	474											
	529	3	coef_probs_8x8 [0] [0] [4] [2] [0..2]	3, 101, 156	3, 101, 156	0	477											
	532	3	coef_probs_8x8 [0] [0] [4] [3] [0..2]	1, 69, 111	1, 69, 111	0	480											
	535	3	coef_probs_8x8 [0] [0] [4] [4] [0..2]	1, 39, 63	1, 39, 63	0	483											
	538	3	coef_probs_8x8 [0] [0] [4] [5] [0..2]	1, 14, 23	1, 14, 23	0	486											
	541	3	coef_probs_8x8 [0] [0] [5] [0] [0..2]	57, 192, 233	57, 192, 233	0	489											
	544	3	coef_probs_8x8 [0] [0] [5] [1] [0..2]	25, 154, 215	25, 154,	0	492											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					215													
	547	3	coef_probs_8x8 [0] [0] [5] [2] [0..2]	6, 109, 167	6, 109, 167	0	495											
	550	3	coef_probs_8x8 [0] [0] [5] [3] [0..2]	3, 78, 118	3, 78, 118	0	498											
	553	3	coef_probs_8x8 [0] [0] [5] [4] [0..2]	1, 48, 69	1, 48, 69	0	501											
	556	3	coef_probs_8x8 [0] [0] [5] [5] [0..2]	1, 21, 29	1, 21, 29	0	504											
	559	3	coef_probs_8x8 [0] [1] [0] [0] [0..2]	202, 105, 245	202, 105, 245	0	507											
	562	3	coef_probs_8x8 [0] [1] [0] [1] [0..2]	108, 106, 216	108, 106, 216	0	510											
	565	3	coef_probs_8x8 [0] [1] [0] [2] [0..2]	18, 90, 144	18, 90, 144	0	513											
	568	3	coef_probs_8x8 [0] [1] [1] [0] [0..2]	33, 172, 219	33, 172, 219	0	516											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	571	3	coef_probs_8x8 [0] [1] [1] [1] [0..2]	64, 149, 206	64, 149, 206	0	519											
	574	3	coef_probs_8x8 [0] [1] [1] [2] [0..2]	14, 117, 177	14, 117, 177	0	522											
	577	3	coef_probs_8x8 [0] [1] [1] [3] [0..2]	5, 90, 141	5, 90, 141	0	525											
	580	3	coef_probs_8x8 [0] [1] [1] [4] [0..2]	2, 61, 95	2, 61, 95	0	528											
	583	3	coef_probs_8x8 [0] [1] [1] [5] [0..2]	1, 37, 57	1, 37, 57	0	531											
	586	3	coef_probs_8x8 [0] [1] [2] [0] [0..2]	33, 179, 220	33, 179, 220	0	534											
	589	3	coef_probs_8x8 [0] [1] [2] [1] [0..2]	11, 140, 198	11, 140, 198	0	537											
	592	3	coef_probs_8x8 [0] [1] [2] [2] [0..2]	1, 89, 148	1, 89, 148	0	540											
	595	3	coef_probs_8x8	1, 60, 104	1, 60	0	543											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT		State counter EBB Address	Coefficient counter EBB Address							
												4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			[0] [1] [2] [3] [0..2]		, 104														
	598	3	coef_probs_8x8 [0] [1] [2] [4] [0..2]	1, 33, 57	1, 33, 57	0	546												
	601	3	coef_probs_8x8 [0] [1] [2] [5] [0..2]	1, 12, 21	1, 12, 21	0	549												
	604	3	coef_probs_8x8 [0] [1] [3] [0] [0..2]	30, 181, 221	30, 181, 221	0	552												
	607	3	coef_probs_8x8 [0] [1] [3] [1] [0..2]	8, 141, 198	8, 141, 198	0	555												
	610	3	coef_probs_8x8 [0] [1] [3] [2] [0..2]	1, 87, 145	1, 87, 145	0	558												
	613	3	coef_probs_8x8 [0] [1] [3] [3] [0..2]	1, 58, 100	1, 58, 100	0	561												
	616	3	coef_probs_8x8 [0] [1] [3] [4] [0..2]	1, 31, 55	1, 31, 55	0	564												
	619	3	coef_probs_8x8 [0] [1] [3] [5] [0..2]	1, 12, 20	1, 12, 20	0	567												
	622	3	coef_probs_8x8 [0] [1] [4] [0]	32, 186, 224	32, 18	0	570												

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			[0..2]		6, 224													
	625	3	coef_probs_8x8 [0] [1] [4] [1] [0..2]	7, 142, 198	7, 142, 198	0	573											
	628	3	coef_probs_8x8 [0] [1] [4] [2] [0..2]	1, 86, 143	1, 86, 143	0	576											
	631	3	coef_probs_8x8 [0] [1] [4] [3] [0..2]	1, 58, 100	1, 58, 100	0	579											
	634	3	coef_probs_8x8 [0] [1] [4] [4] [0..2]	1, 31, 55	1, 31, 55	0	582											
	637	3	coef_probs_8x8 [0] [1] [4] [5] [0..2]	1, 12, 22	1, 12, 22	0	585											
	640	3	coef_probs_8x8 [0] [1] [5] [0] [0..2]	57, 192, 227	57, 192, 227	0	588											
	643	3	coef_probs_8x8 [0] [1] [5] [1] [0..2]	20, 143, 204	20, 143, 204	0	591											
	646	3	coef_probs_8x8 [0] [1] [5] [2] [0..2]	3, 96, 154	3, 96, 154	0	594											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	649	3	coef_probs_8x8 [0] [1] [5] [3] [0..2]	1, 68, 112	1, 68, 112	0	597											
	652	3	coef_probs_8x8 [0] [1] [5] [4] [0..2]	1, 42, 69	1, 42, 69	0	600											
	655	3	coef_probs_8x8 [0] [1] [5] [5] [0..2]	1, 19, 32	1, 19, 32	0	603											
	658	3	coef_probs_8x8 [1] [0] [0] [0] [0..2]	212, 35, 215	212, 35, 215	0	606								0-287			
	661	3	coef_probs_8x8 [1] [0] [0] [1] [0..2]	113, 47, 169	113, 47, 169	0	609											
	664	3	coef_probs_8x8 [1] [0] [0] [2] [0..2]	29, 48, 105	29, 48, 105	0	612											
	667	3	coef_probs_8x8 [1] [0] [1] [0] [0..2]	74, 129, 203	74, 129, 203	0	615											
	670	3	coef_probs_8x8 [1] [0] [1] [1] [0..2]	106, 120, 203	106, 120, 203	0	618											
	673	3	coef_probs_8x	49,	49	0	62											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			8 [1] [0] [1] [2] [0..2]	107, 178	, 107, 178		1											
	676	3	coef_probs_8x8 [1] [0] [1] [3] [0..2]	19, 84, 144	19, 84, 144	0	624											
	679	3	coef_probs_8x8 [1] [0] [1] [4] [0..2]	4, 50, 84	4, 50, 84	0	627											
	682	3	coef_probs_8x8 [1] [0] [1] [5] [0..2]	1, 15, 25	1, 15, 25	0	630											
	685	3	coef_probs_8x8 [1] [0] [2] [0] [0..2]	71, 172, 217	71, 172, 217	0	633											
	688	3	coef_probs_8x8 [1] [0] [2] [1] [0..2]	44, 141, 209	44, 141, 209	0	636											
	691	3	coef_probs_8x8 [1] [0] [2] [2] [0..2]	15, 102, 173	15, 102, 173	0	639											
	694	3	coef_probs_8x8 [1] [0] [2] [3] [0..2]	6, 76, 133	6, 76, 133	0	642											
	697	3	coef_probs_8x8	2, 51, 89	2, 51	0	645											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT		State counter EBB Address	Coefficient counter EBB Address							
												4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			[1] [0] [2] [4] [0..2]		, 89														
	700	3	coef_probs_8x8 [1] [0] [2] [5] [0..2]	1, 24, 42	1, 24, 42	0	648												
	703	3	coef_probs_8x8 [1] [0] [3] [0] [0..2]	64, 185, 231	64, 185, 231	0	651												
	706	3	coef_probs_8x8 [1] [0] [3] [1] [0..2]	31, 148, 216	31, 148, 216	0	654												
	709	3	coef_probs_8x8 [1] [0] [3] [2] [0..2]	8, 103, 175	8, 103, 175	0	657												
	712	3	coef_probs_8x8 [1] [0] [3] [3] [0..2]	3, 74, 131	3, 74, 131	0	660												
	715	3	coef_probs_8x8 [1] [0] [3] [4] [0..2]	1, 46, 81	1, 46, 81	0	663												
	718	3	coef_probs_8x8 [1] [0] [3] [5] [0..2]	1, 18, 30	1, 18, 30	0	666												
	721	3	coef_probs_8x8 [1] [0] [4] [0] [0..2]	65, 196, 235	65, 196, 235	0	669												
	724	3	coef_probs_8x	25,	25	0	67												

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults			Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
										4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			8 [1] [0] [4] [1] [0..2]	157, 221	, 157, 221		2											
	727	3	coef_probs_8x8 [1] [0] [4] [2] [0..2]	5, 105, 174	5, 105, 174	0	675											
	730	3	coef_probs_8x8 [1] [0] [4] [3] [0..2]	1, 67, 120	1, 67, 120	0	678											
	733	3	coef_probs_8x8 [1] [0] [4] [4] [0..2]	1, 38, 69	1, 38, 69	0	681											
	736	3	coef_probs_8x8 [1] [0] [4] [5] [0..2]	1, 15, 30	1, 15, 30	0	684											
	739	3	coef_probs_8x8 [1] [0] [5] [0] [0..2]	65, 204, 238	65, 204, 238	0	687											
	742	3	coef_probs_8x8 [1] [0] [5] [1] [0..2]	30, 156, 224	30, 156, 224	0	690											
	745	3	coef_probs_8x8 [1] [0] [5] [2] [0..2]	7, 107, 177	7, 107, 177	0	693											
	748	3	coef_probs_8x8 [1] [0] [5] [3] [0..2]	2, 70, 124	2, 70, 12	0	696											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					4													
	751	3	coef_probs_8x8 [1] [0] [5] [4] [0..2]	1, 42, 73	1, 42, 73	0	699											
	754	3	coef_probs_8x8 [1] [0] [5] [5] [0..2]	1, 18, 34	1, 18, 34	0	702											
	757	3	coef_probs_8x8 [1] [1] [0] [0] [0..2]	225, 86, 251	225, 86, 251	0	705											
	760	3	coef_probs_8x8 [1] [1] [0] [1] [0..2]	144, 104, 235	144, 104, 235	0	708											
	763	3	coef_probs_8x8 [1] [1] [0] [2] [0..2]	42, 99, 181	42, 99, 181	0	711											
	766	3	coef_probs_8x8 [1] [1] [1] [0] [0..2]	85, 175, 239	85, 175, 239	0	714											
	769	3	coef_probs_8x8 [1] [1] [1] [1] [0..2]	112, 165, 229	112, 165, 229	0	717											
	772	3	coef_probs_8x8 [1] [1] [1] [2] [0..2]	29, 136, 200	29, 136, 20	0	720											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					0													
	775	3	coef_probs_8x8 [1] [1] [1] [3] [0..2]	12, 103, 162	12, 103, 162	0	723											
	778	3	coef_probs_8x8 [1] [1] [1] [4] [0..2]	6, 77, 123	6, 77, 123	0	726											
	781	3	coef_probs_8x8 [1] [1] [1] [5] [0..2]	2, 53, 84	2, 53, 84	0	729											
	784	3	coef_probs_8x8 [1] [1] [2] [0] [0..2]	75, 183, 239	75, 183, 239	0	732											
	787	3	coef_probs_8x8 [1] [1] [2] [1] [0..2]	30, 155, 221	30, 155, 221	0	735											
	790	3	coef_probs_8x8 [1] [1] [2] [2] [0..2]	3, 106, 171	3, 106, 171	0	738											
	793	3	coef_probs_8x8 [1] [1] [2] [3] [0..2]	1, 74, 128	1, 74, 128	0	741											
	796	3	coef_probs_8x8 [1] [1] [2] [4] [0..2]	1, 44, 76	1, 44, 76	0	744											
	799	3	coef_probs_8x	1, 17,	1,	0	74											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			8 [1] [1] [2] [5] [0..2]	28	17, 28		7											
	802	3	coef_probs_8x8 [1] [1] [3] [0] [0..2]	73, 185, 240	73, 185, 240	0	750											
	805	3	coef_probs_8x8 [1] [1] [3] [1] [0..2]	27, 159, 222	27, 159, 222	0	753											
	808	3	coef_probs_8x8 [1] [1] [3] [2] [0..2]	2, 107, 172	2, 107, 172	0	756											
	811	3	coef_probs_8x8 [1] [1] [3] [3] [0..2]	1, 75, 127	1, 75, 127	0	759											
	814	3	coef_probs_8x8 [1] [1] [3] [4] [0..2]	1, 42, 73	1, 42, 73	0	762											
	817	3	coef_probs_8x8 [1] [1] [3] [5] [0..2]	1, 17, 29	1, 17, 29	0	765											
	820	3	coef_probs_8x8 [1] [1] [4] [0] [0..2]	62, 190, 238	62, 190, 238	0	768											
	823	3	coef_probs_8x8 [1] [1] [4] [1] [0..2]	21, 159, 222	21, 159, 22	0	771											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					2													
	826	3	coef_probs_8x8 [1] [1] [4] [2] [0..2]	2, 107, 172	2, 107, 172	0	774											
	829	3	coef_probs_8x8 [1] [1] [4] [3] [0..2]	1, 72, 122	1, 72, 122	0	777											
	832	3	coef_probs_8x8 [1] [1] [4] [4] [0..2]	1, 40, 71	1, 40, 71	0	780											
	835	3	coef_probs_8x8 [1] [1] [4] [5] [0..2]	1, 18, 32	1, 18, 32	0	783											
	838	3	coef_probs_8x8 [1] [1] [5] [0] [0..2]	61, 199, 240	61, 199, 240	0	786											
	841	3	coef_probs_8x8 [1] [1] [5] [1] [0..2]	27, 161, 226	27, 161, 226	0	789											
	844	3	coef_probs_8x8 [1] [1] [5] [2] [0..2]	4, 113, 180	4, 113, 180	0	792											
	847	3	coef_probs_8x8 [1] [1] [5] [3] [0..2]	1, 76, 129	1, 76, 129	0	795											
	850	3	coef_probs_8x8	1, 46, 80	1, 46	0	798											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT		State counter EBB Address	Coefficient counter EBB Address							
												4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			[1] [1] [5] [4] [0..2]		, 80														
	853	3	coef_probs_8x8 [1] [1] [5] [5] [0..2]	1, 23, 41	1, 23, 41	0	801												
	856	3	coef_probs_16x16 [0] [0] [0] [0] [0..2]	7, 27, 153	7, 27, 153	0	804		107							0-287			
	859	3	coef_probs_16x16 [0] [0] [0] [1] [0..2]	5, 30, 95	5, 30, 95	0	807												
	862	3	coef_probs_16x16 [0] [0] [0] [2] [0..2]	1, 16, 30	1, 16, 30	0	810												
	865	3	coef_probs_16x16 [0] [0] [1] [0] [0..2]	50, 75, 127	50, 75, 127	0	813												
	868	3	coef_probs_16x16 [0] [0] [1] [1] [0..2]	57, 75, 124	57, 75, 124	0	816												
	871	3	coef_probs_16x16 [0] [0] [1] [2] [0..2]	27, 67, 108	27, 67, 108	0	819												
	874	3	coef_probs_16x16 [0] [0] [1] [3] [0..2]	10, 54, 86	10, 54, 86	0	822												
	877	3	coef_probs_1	1, 33,	1,	0	82												

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			6x16 [0] [0] [1] [4] [0..2]	52	33, 52		5											
	880	3	coef_probs_16x16 [0] [0] [1] [5] [0..2]	1, 12, 18	1, 12, 18	0	828											
	883	3	coef_probs_16x16 [0] [0] [2] [0] [0..2]	43, 125, 151	43, 125, 151	0	831											
	886	3	coef_probs_16x16 [0] [0] [2] [1] [0..2]	26, 108, 148	26, 108, 148	0	834											
	889	3	coef_probs_16x16 [0] [0] [2] [2] [0..2]	7, 83, 122	7, 83, 122	0	837											
	892	3	coef_probs_16x16 [0] [0] [2] [3] [0..2]	2, 59, 89	2, 59, 89	0	840											
	895	3	coef_probs_16x16 [0] [0] [2] [4] [0..2]	1, 38, 60	1, 38, 60	0	843											
	898	3	coef_probs_16x16 [0] [0] [2] [5] [0..2]	1, 17, 27	1, 17, 27	0	846											
	901	3	coef_probs_16x16 [0] [0] [3] [0] [0..2]	23, 144, 163	23, 144, 163	0	849											
	904	3	coef_probs_1	13,	13	0	85											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults			Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
										4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			6x16 [0] [0] [3] [1] [0..2]	112, 154	, 112, 154		2											
	907	3	coef_probs_16x16 [0] [0] [3] [2] [0..2]	2, 75, 117	2, 75, 117	0	855											
	910	3	coef_probs_16x16 [0] [0] [3] [3] [0..2]	1, 50, 81	1, 50, 81	0	858											
	913	3	coef_probs_16x16 [0] [0] [3] [4] [0..2]	1, 31, 51	1, 31, 51	0	861											
	916	3	coef_probs_16x16 [0] [0] [3] [5] [0..2]	1, 14, 23	1, 14, 23	0	864											
	919	3	coef_probs_16x16 [0] [0] [4] [0] [0..2]	18, 162, 185	18, 162, 185	0	867											
	922	3	coef_probs_16x16 [0] [0] [4] [1] [0..2]	6, 123, 171	6, 123, 171	0	870											
	925	3	coef_probs_16x16 [0] [0] [4] [2] [0..2]	1, 78, 125	1, 78, 125	0	873											
	928	3	coef_probs_16x16 [0] [0] [4] [3] [0..2]	1, 51, 86	1, 51, 86	0	876											
	931	3	coef_probs_1	1, 31,	1,	0	87											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			6x16 [0] [0] [4] [4] [0..2]	54	31, 54		9											
	934	3	coef_probs_16x16 [0] [0] [4] [5] [0..2]	1, 14, 23	1, 14, 23	0	882											
	937	3	coef_probs_16x16 [0] [0] [5] [0] [0..2]	15, 199, 227	15, 199, 227	0	885											
	940	3	coef_probs_16x16 [0] [0] [5] [1] [0..2]	3, 150, 204	3, 150, 204	0	888											
	943	3	coef_probs_16x16 [0] [0] [5] [2] [0..2]	1, 91, 146	1, 91, 146	0	891											
	946	3	coef_probs_16x16 [0] [0] [5] [3] [0..2]	1, 55, 95	1, 55, 95	0	894											
	949	3	coef_probs_16x16 [0] [0] [5] [4] [0..2]	1, 30, 53	1, 30, 53	0	897											
	952	3	coef_probs_16x16 [0] [0] [5] [5] [0..2]	1, 11, 20	1, 11, 20	0	900											
	955	3	coef_probs_16x16 [0] [1] [0] [0] [0..2]	19, 55, 240	19, 55, 240	0	903											
	958	3	coef_probs_16x16	19, 59,	19,	0	906											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			[0] [1] [0] [1] [0..2]	196	59, 196													
	961	3	coef_probs_16x16 [0] [1] [0] [2] [0..2]	3, 52, 105	3, 52, 105	0	909											
	964	3	coef_probs_16x16 [0] [1] [1] [0] [0..2]	41, 166, 207	41, 166, 207	0	912											
	967	3	coef_probs_16x16 [0] [1] [1] [1] [0..2]	104, 153, 199	104, 153, 199	0	915											
	970	3	coef_probs_16x16 [0] [1] [1] [2] [0..2]	31, 123, 181	31, 123, 181	0	918											
	973	3	coef_probs_16x16 [0] [1] [1] [3] [0..2]	14, 101, 152	14, 101, 152	0	921											
	976	3	coef_probs_16x16 [0] [1] [1] [4] [0..2]	5, 72, 106	5, 72, 106	0	924											
	979	3	coef_probs_16x16 [0] [1] [1] [5] [0..2]	1, 36, 52	1, 36, 52	0	927											
	982	3	coef_probs_16x16	35, 176,	35,	0	930											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			[0] [1] [2] [0] [0..2]	211	176, 211													
	985	3	coef_probs_16x16 [0] [1] [2] [1] [0..2]	12, 131, 190	12, 131, 190	0	933											
	988	3	coef_probs_16x16 [0] [1] [2] [2] [0..2]	2, 88, 144	2, 88, 144	0	936											
	991	3	coef_probs_16x16 [0] [1] [2] [3] [0..2]	1, 60, 101	1, 60, 101	0	939											
	994	3	coef_probs_16x16 [0] [1] [2] [4] [0..2]	1, 36, 60	1, 36, 60	0	942											
	997	3	coef_probs_16x16 [0] [1] [2] [5] [0..2]	1, 16, 28	1, 16, 28	0	945											
	1000	3	coef_probs_16x16 [0] [1] [3] [0] [0..2]	28, 183, 213	28, 183, 213	0	948											
	1003	3	coef_probs_16x16 [0] [1] [3] [1] [0..2]	8, 134, 191	8, 134, 191	0	951											
	1006	3	coef_probs_16x16 [0] [1] [3] [2] [0..2]	1, 86, 142	1, 86, 142	0	954											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	1009	3	coef_probs_16x16 [0] [1] [3] [3] [0..2]	1, 56, 96	1, 56, 96	0	957											
	1012	3	coef_probs_16x16 [0] [1] [3] [4] [0..2]	1, 30, 53	1, 30, 53	0	960											
	1015	3	coef_probs_16x16 [0] [1] [3] [5] [0..2]	1, 12, 20	1, 12, 20	0	963											
	1018	3	coef_probs_16x16 [0] [1] [4] [0] [0..2]	20, 190, 215	20, 190, 215	0	966											
	1021	3	coef_probs_16x16 [0] [1] [4] [1] [0..2]	4, 135, 192	4, 135, 192	0	969											
	1024	3	coef_probs_16x16 [0] [1] [4] [2] [0..2]	1, 84, 139	1, 84, 139	0	972											
	1027	3	coef_probs_16x16 [0] [1] [4] [3] [0..2]	1, 53, 91	1, 53, 91	0	975											
	1030	3	coef_probs_16x16 [0] [1] [4] [4] [0..2]	1, 28, 49	1, 28, 49	0	978											
	1033	3	coef_probs_16x16 [0] [1] [4] [5] [0..2]	1, 11, 20	1, 11, 20	0	981											
	1036	3	coef_probs_16x16 [0] [1] [5] [0]	13, 196, 216	13, 19	0	984											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			[0..2]		6, 216													
	1039	3	coef_probs_16x16 [0] [1] [5] [1] [0..2]	2, 137, 192	2, 137, 192	0	987											
	1042	3	coef_probs_16x16 [0] [1] [5] [2] [0..2]	1, 86, 143	1, 86, 143	0	990											
	1045	3	coef_probs_16x16 [0] [1] [5] [3] [0..2]	1, 57, 99	1, 57, 99	0	993											
	1048	3	coef_probs_16x16 [0] [1] [5] [4] [0..2]	1, 32, 56	1, 32, 56	0	996											
	1051	3	coef_probs_16x16 [0] [1] [5] [5] [0..2]	1, 13, 24	1, 13, 24	0	999											
	1054	3	coef_probs_16x16 [1] [0] [0] [0] [0..2]	211, 29, 217	211, 29, 217	0	1002									0-287		
	1057	3	coef_probs_16x16 [1] [0] [0] [1] [0..2]	96, 47, 156	96, 47, 156	0	1005											
	1060	3	coef_probs_16x16 [1] [0] [0] [2] [0..2]	22, 43, 87	22, 43, 87	0	1008											
	106	3	coef_probs_1	78,	78	0	10											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	3		6x16 [1] [0] [1] [0] [0..2]	120, 193	, 120, 193		11											
	1066	3	coef_probs_16x16 [1] [0] [1] [1] [0..2]	111, 116, 186	111, 116, 186	0	1014											
	1069	3	coef_probs_16x16 [1] [0] [1] [2] [0..2]	46, 102, 164	46, 102, 164	0	1017											
	1072	3	coef_probs_16x16 [1] [0] [1] [3] [0..2]	15, 80, 128	15, 80, 128	0	1020											
	1075	3	coef_probs_16x16 [1] [0] [1] [4] [0..2]	2, 49, 76	2, 49, 76	0	1023											
	1078	3	coef_probs_16x16 [1] [0] [1] [5] [0..2]	1, 18, 28	1, 18, 28	0	1026											
	1081	3	coef_probs_16x16 [1] [0] [2] [0] [0..2]	71, 161, 203	71, 161, 203	0	1029											
	1084	3	coef_probs_16x16 [1] [0] [2] [1] [0..2]	42, 132, 192	42, 132, 192	0	1032											
	108	3	coef_probs_1	10,	10	0	10											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	7		6x16 [1] [0] [2] [2] [0..2]	98, 150	, 98, 150		35											
	1090	3	coef_probs_16x16 [1] [0] [2] [3] [0..2]	3, 69, 109	3, 69, 109	0	1038											
	1093	3	coef_probs_16x16 [1] [0] [2] [4] [0..2]	1, 44, 70	1, 44, 70	0	1041											
	1096	3	coef_probs_16x16 [1] [0] [2] [5] [0..2]	1, 18, 29	1, 18, 29	0	1044											
	1099	3	coef_probs_16x16 [1] [0] [3] [0] [0..2]	57, 186, 211	57, 186, 211	0	1047											
	1102	3	coef_probs_16x16 [1] [0] [3] [1] [0..2]	30, 140, 196	30, 140, 196	0	1050											
	1105	3	coef_probs_16x16 [1] [0] [3] [2] [0..2]	4, 93, 146	4, 93, 146	0	1053											
	1108	3	coef_probs_16x16 [1] [0] [3] [3] [0..2]	1, 62, 102	1, 62, 102	0	1056											
	1111	3	coef_probs_16x16 [1] [0] [3] [4] [0..2]	1, 38, 65	1, 38, 65	0	1059											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	1114	3	coef_probs_16x16 [1] [0] [3] [5] [0..2]	1, 16, 27	1, 16, 27	0	1062											
	1117	3	coef_probs_16x16 [1] [0] [4] [0] [0..2]	47, 199, 217	47, 199, 217	0	1065											
	1120	3	coef_probs_16x16 [1] [0] [4] [1] [0..2]	14, 145, 196	14, 145, 196	0	1068											
	1123	3	coef_probs_16x16 [1] [0] [4] [2] [0..2]	1, 88, 142	1, 88, 142	0	1071											
	1126	3	coef_probs_16x16 [1] [0] [4] [3] [0..2]	1, 57, 98	1, 57, 98	0	1074											
	1129	3	coef_probs_16x16 [1] [0] [4] [4] [0..2]	1, 36, 62	1, 36, 62	0	1077											
	1132	3	coef_probs_16x16 [1] [0] [4] [5] [0..2]	1, 15, 26	1, 15, 26	0	1080											
	1135	3	coef_probs_16x16 [1] [0] [5] [0] [0..2]	26, 219, 229	26, 219, 229	0	1083											
	1138	3	coef_probs_16x16 [1] [0] [5] [1] [0..2]	5, 155, 207	5, 155, 207	0	1086											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	1141	3	coef_probs_16x16 [1] [0] [5] [2] [0..2]	1, 94, 151	1, 94, 151	0	1089											
	1144	3	coef_probs_16x16 [1] [0] [5] [3] [0..2]	1, 60, 104	1, 60, 104	0	1092											
	1147	3	coef_probs_16x16 [1] [0] [5] [4] [0..2]	1, 36, 62	1, 36, 62	0	1095											
	1150	3	coef_probs_16x16 [1] [0] [5] [5] [0..2]	1, 16, 28	1, 16, 28	0	1098											
	1153	3	coef_probs_16x16 [1] [1] [0] [0] [0..2]	233, 29, 248	233, 29, 248	0	1101											
	1156	3	coef_probs_16x16 [1] [1] [0] [1] [0..2]	146, 47, 220	146, 47, 220	0	1104											
	1159	3	coef_probs_16x16 [1] [1] [0] [2] [0..2]	43, 52, 140	43, 52, 140	0	1107											
	1162	3	coef_probs_16x16 [1] [1] [1] [0] [0..2]	100, 163, 232	100, 163, 232	0	1110											
	1165	3	coef_probs_16x16	179, 161,	179,	0	1113											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			[1] [1] [1] [1] [0..2]	222	161, 222													
	1168	3	coef_probs_16x16 [1] [1] [1] [2] [0..2]	63, 142, 204	63, 142, 204	0	1116											
	1171	3	coef_probs_16x16 [1] [1] [1] [3] [0..2]	37, 113, 174	37, 113, 174	0	1119											
	1174	3	coef_probs_16x16 [1] [1] [1] [4] [0..2]	26, 89, 137	26, 89, 137	0	1122											
	1177	3	coef_probs_16x16 [1] [1] [1] [5] [0..2]	18, 68, 97	18, 68, 97	0	1125											
	1180	3	coef_probs_16x16 [1] [1] [2] [0] [0..2]	85, 181, 230	85, 181, 230	0	1128											
	1183	3	coef_probs_16x16 [1] [1] [2] [1] [0..2]	32, 146, 209	32, 146, 209	0	1131											
	1186	3	coef_probs_16x16 [1] [1] [2] [2] [0..2]	7, 100, 164	7, 100, 164	0	1134											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	1189	3	coef_probs_16x16 [1] [1] [2] [3] [0..2]	3, 71, 121	3, 71, 121	0	1137											
	1192	3	coef_probs_16x16 [1] [1] [2] [4] [0..2]	1, 45, 77	1, 45, 77	0	1140											
	1195	3	coef_probs_16x16 [1] [1] [2] [5] [0..2]	1, 18, 30	1, 18, 30	0	1143											
	1198	3	coef_probs_16x16 [1] [1] [3] [0] [0..2]	65, 187, 230	65, 187, 230	0	1146											
	1201	3	coef_probs_16x16 [1] [1] [3] [1] [0..2]	20, 148, 207	20, 148, 207	0	1149											
	1204	3	coef_probs_16x16 [1] [1] [3] [2] [0..2]	2, 97, 159	2, 97, 159	0	1152											
	1207	3	coef_probs_16x16 [1] [1] [3] [3] [0..2]	1, 68, 116	1, 68, 116	0	1155											
	1210	3	coef_probs_16x16 [1] [1] [3] [4] [0..2]	1, 40, 70	1, 40, 70	0	1158											
	1213	3	coef_probs_16x16 [1] [1] [3] [5] [0..2]	1, 14, 29	1, 14, 29	0	1161											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	1216	3	coef_probs_16x16 [1] [1] [4] [0] [0..2]	40, 194, 227	40, 194, 227	0	1164											
	1219	3	coef_probs_16x16 [1] [1] [4] [1] [0..2]	8, 147, 204	8, 147, 204	0	1167											
	1222	3	coef_probs_16x16 [1] [1] [4] [2] [0..2]	1, 94, 155	1, 94, 155	0	1170											
	1225	3	coef_probs_16x16 [1] [1] [4] [3] [0..2]	1, 65, 112	1, 65, 112	0	1173											
	1228	3	coef_probs_16x16 [1] [1] [4] [4] [0..2]	1, 39, 66	1, 39, 66	0	1176											
	1231	3	coef_probs_16x16 [1] [1] [4] [5] [0..2]	1, 14, 26	1, 14, 26	0	1179											
	1234	3	coef_probs_16x16 [1] [1] [5] [0] [0..2]	16, 208, 228	16, 208, 228	0	1182											
	1237	3	coef_probs_16x16 [1] [1] [5] [1] [0..2]	3, 151, 207	3, 151, 207	0	1185											
	1240	3	coef_probs_16x16 [1] [1] [5] [2] [0..2]	1, 98, 16	1, 98, 16	0	1188											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT		State counter EBB Address	Coefficient counter EBB Address							
												4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					0														
	1243	3	coef_probs_16x16 [1] [1] [5] [3] [0..2]	1, 67, 117	1, 67, 117	0	1191												
	1246	3	coef_probs_16x16 [1] [1] [5] [4] [0..2]	1, 41, 74	1, 41, 74	0	1194												
	1249	3	coef_probs_16x16 [1] [1] [5] [5] [0..2]	1, 17, 31	1, 17, 31	0	1197												
	1252	3	coef_probs_32x32 [0] [0] [0] [0] [0..2]	17, 38, 140	17, 38, 140	0	1200		156.5									0-287	
	1255	3	coef_probs_32x32 [0] [0] [0] [1] [0..2]	7, 34, 80	7, 34, 80	0	1203												
	1258	3	coef_probs_32x32 [0] [0] [0] [2] [0..2]	1, 17, 29	1, 17, 29	0	1206												
	1261	3	coef_probs_32x32 [0] [0] [1] [0] [0..2]	37, 75, 128	37, 75, 128	0	1209												
	1264	3	coef_probs_32x32 [0] [0] [1] [1] [0..2]	41, 76, 128	41, 76, 128	0	1212												
	1267	3	coef_probs_32x32 [0] [0] [1] [2]	26, 66, 116	26, 66	0	1215												

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			[0..2]		, 116													
	1270	3	coef_probs_32x32 [0] [0] [1] [3] [0..2]	12, 52, 94	12, 52, 94	0	12 18											
	1273	3	coef_probs_32x32 [0] [0] [1] [4] [0..2]	2, 32, 55	2, 32, 55	0	12 21											
	1276	3	coef_probs_32x32 [0] [0] [1] [5] [0..2]	1, 10, 16	1, 10, 16	0	12 24											
	1279	3	coef_probs_32x32 [0] [0] [2] [0] [0..2]	50, 127, 154	50, 127, 154	0	12 27											
	1282	3	coef_probs_32x32 [0] [0] [2] [1] [0..2]	37, 109, 152	37, 109, 152	0	12 30											
	1285	3	coef_probs_32x32 [0] [0] [2] [2] [0..2]	16, 82, 121	16, 82, 121	0	12 33											
	1288	3	coef_probs_32x32 [0] [0] [2] [3] [0..2]	5, 59, 85	5, 59, 85	0	12 36											
	1291	3	coef_probs_32x32 [0] [0] [2] [4] [0..2]	1, 35, 54	1, 35, 54	0	12 39											
	129	3	coef_probs_3	1, 13,	1,	0	12											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT		State counter EBB Address	Coefficient counter EBB Address							
												4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	4		2x32 [0] [0] [2] [5] [0..2]	20	13, 20		42												
	1297	3	coef_probs_32x32 [0] [0] [3] [0] [0..2]	40, 142, 167	40, 142, 167	0	1245												
	1300	3	coef_probs_32x32 [0] [0] [3] [1] [0..2]	17, 110, 157	17, 110, 157	0	1248												
	1303	3	coef_probs_32x32 [0] [0] [3] [2] [0..2]	2, 71, 112	2, 71, 112	0	1251												
	1306	3	coef_probs_32x32 [0] [0] [3] [3] [0..2]	1, 44, 72	1, 44, 72	0	1254												
	1309	3	coef_probs_32x32 [0] [0] [3] [4] [0..2]	1, 27, 45	1, 27, 45	0	1257												
	1312	3	coef_probs_32x32 [0] [0] [3] [5] [0..2]	1, 11, 17	1, 11, 17	0	1260												
	1315	3	coef_probs_32x32 [0] [0] [4] [0] [0..2]	30, 175, 188	30, 175, 188	0	1263												
	1318	3	coef_probs_32x32 [0] [0] [4] [1] [0..2]	9, 124, 169	9, 124, 169	0	1266												

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT		State counter EBB Address	Coefficient counter EBB Address							
												4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	1321	3	coef_probs_32x32 [0] [0] [4] [2] [0..2]	1, 74, 116	1, 74, 116	0	1269												
	1324	3	coef_probs_32x32 [0] [0] [4] [3] [0..2]	1, 48, 78	1, 48, 78	0	1272												
	1327	3	coef_probs_32x32 [0] [0] [4] [4] [0..2]	1, 30, 49	1, 30, 49	0	1275												
	1330	3	coef_probs_32x32 [0] [0] [4] [5] [0..2]	1, 11, 18	1, 11, 18	0	1278												
	1333	3	coef_probs_32x32 [0] [0] [5] [0] [0..2]	10, 222, 223	10, 222, 223	0	1281												
	1336	3	coef_probs_32x32 [0] [0] [5] [1] [0..2]	2, 150, 194	2, 150, 194	0	1284												
	1339	3	coef_probs_32x32 [0] [0] [5] [2] [0..2]	1, 83, 128	1, 83, 128	0	1287												
	1342	3	coef_probs_32x32 [0] [0] [5] [3] [0..2]	1, 48, 79	1, 48, 79	0	1290												
	1345	3	coef_probs_32x32 [0] [0] [5] [4] [0..2]	1, 27, 45	1, 27, 45	0	1293												
	1348	3	coef_probs_32x32	1, 11, 17	1, 11	0	1296												

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			[0] [0] [5] [5] [0..2]		, 17													
	1351	3	coef_probs_32x32 [0] [1] [0] [0] [0..2]	36, 41, 235	36, 41, 235	0	1299											
	1354	3	coef_probs_32x32 [0] [1] [0] [1] [0..2]	29, 36, 193	29, 36, 193	0	1302											
	1357	3	coef_probs_32x32 [0] [1] [0] [2] [0..2]	10, 27, 111	10, 27, 111	0	1305											
	1360	3	coef_probs_32x32 [0] [1] [1] [0] [0..2]	85, 165, 222	85, 165, 222	0	1308											
	1363	3	coef_probs_32x32 [0] [1] [1] [1] [0..2]	177, 162, 215	177, 162, 215	0	1311											
	1366	3	coef_probs_32x32 [0] [1] [1] [2] [0..2]	110, 135, 195	110, 135, 195	0	1314											
	1369	3	coef_probs_32x32 [0] [1] [1] [3] [0..2]	57, 113, 168	57, 113, 168	0	1317											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	1372	3	coef_probs_32x32 [0] [1] [1] [4] [0..2]	23, 83, 120	23, 83, 120	0	1320											
	1375	3	coef_probs_32x32 [0] [1] [1] [5] [0..2]	10, 49, 61	10, 49, 61	0	1323											
	1378	3	coef_probs_32x32 [0] [1] [2] [0] [0..2]	85, 190, 223	85, 190, 223	0	1326											
	1381	3	coef_probs_32x32 [0] [1] [2] [1] [0..2]	36, 139, 200	36, 139, 200	0	1329											
	1384	3	coef_probs_32x32 [0] [1] [2] [2] [0..2]	5, 90, 146	5, 90, 146	0	1332											
	1387	3	coef_probs_32x32 [0] [1] [2] [3] [0..2]	1, 60, 103	1, 60, 103	0	1335											
	1390	3	coef_probs_32x32 [0] [1] [2] [4] [0..2]	1, 38, 65	1, 38, 65	0	1338											
	1393	3	coef_probs_32x32 [0] [1] [2] [5] [0..2]	1, 18, 30	1, 18, 30	0	1341											
	1396	3	coef_probs_32x32 [0] [1] [3] [0]	72, 202, 223	72, 20	0	1344											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			[0..2]		2, 223													
	1399	3	coef_probs_32x32 [0] [1] [3] [1] [0..2]	23, 141, 199	23, 141, 199	0	1347											
	1402	3	coef_probs_32x32 [0] [1] [3] [2] [0..2]	2, 86, 140	2, 86, 140	0	1350											
	1405	3	coef_probs_32x32 [0] [1] [3] [3] [0..2]	1, 56, 97	1, 56, 97	0	1353											
	1408	3	coef_probs_32x32 [0] [1] [3] [4] [0..2]	1, 36, 61	1, 36, 61	0	1356											
	1411	3	coef_probs_32x32 [0] [1] [3] [5] [0..2]	1, 16, 27	1, 16, 27	0	1359											
	1414	3	coef_probs_32x32 [0] [1] [4] [0] [0..2]	55, 218, 225	55, 218, 225	0	1362											
	1417	3	coef_probs_32x32 [0] [1] [4] [1] [0..2]	13, 145, 200	13, 145, 200	0	1365											
	1420	3	coef_probs_32x32 [0] [1] [4] [2] [0..2]	1, 86, 141	1, 86, 141	0	1368											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	1423	3	coef_probs_32x32 [0] [1] [4] [3] [0..2]	1, 57, 99	1, 57, 99	0	1371											
	1426	3	coef_probs_32x32 [0] [1] [4] [4] [0..2]	1, 35, 61	1, 35, 61	0	1374											
	1429	3	coef_probs_32x32 [0] [1] [4] [5] [0..2]	1, 13, 22	1, 13, 22	0	1377											
	1432	3	coef_probs_32x32 [0] [1] [5] [0] [0..2]	15, 235, 212	15, 235, 212	0	1380											
	1435	3	coef_probs_32x32 [0] [1] [5] [1] [0..2]	1, 132, 184	1, 132, 184	0	1383											
	1438	3	coef_probs_32x32 [0] [1] [5] [2] [0..2]	1, 84, 139	1, 84, 139	0	1386											
	1441	3	coef_probs_32x32 [0] [1] [5] [3] [0..2]	1, 57, 97	1, 57, 97	0	1389											
	1444	3	coef_probs_32x32 [0] [1] [5] [4] [0..2]	1, 34, 56	1, 34, 56	0	1392											
	1447	3	coef_probs_32x32 [0] [1] [5] [5] [0..2]	1, 14, 23	1, 14, 23	0	1395											
	1450	3	coef_probs_32x32 [1] [0] [0] [0]	181, 21, 201	181, 21	0	1398											0-287

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			[0..2]		, 201													
	1453	3	coef_probs_32x32 [1] [0] [0] [1] [0..2]	61, 37, 123	61, 37, 123	0	1401											
	1456	3	coef_probs_32x32 [1] [0] [0] [2] [0..2]	10, 38, 71	10, 38, 71	0	1404											
	1459	3	coef_probs_32x32 [1] [0] [1] [0] [0..2]	47, 106, 172	47, 106, 172	0	1407											
	1462	3	coef_probs_32x32 [1] [0] [1] [1] [0..2]	95, 104, 173	95, 104, 173	0	1410											
	1465	3	coef_probs_32x32 [1] [0] [1] [2] [0..2]	42, 93, 159	42, 93, 159	0	1413											
	1468	3	coef_probs_32x32 [1] [0] [1] [3] [0..2]	18, 77, 131	18, 77, 131	0	1416											
	1471	3	coef_probs_32x32 [1] [0] [1] [4] [0..2]	4, 50, 81	4, 50, 81	0	1419											
	1474	3	coef_probs_32x32	1, 17, 23	1, 17	0	1422											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			[1] [0] [1] [5] [0..2]		, 23													
	1477	3	coef_probs_32x32 [1] [0] [2] [0] [0..2]	62, 147, 199	62, 147, 199	0	1425											
	1480	3	coef_probs_32x32 [1] [0] [2] [1] [0..2]	44, 130, 189	44, 130, 189	0	1428											
	1483	3	coef_probs_32x32 [1] [0] [2] [2] [0..2]	28, 102, 154	28, 102, 154	0	1431											
	1486	3	coef_probs_32x32 [1] [0] [2] [3] [0..2]	18, 75, 115	18, 75, 115	0	1434											
	1489	3	coef_probs_32x32 [1] [0] [2] [4] [0..2]	2, 44, 65	2, 44, 65	0	1437											
	1492	3	coef_probs_32x32 [1] [0] [2] [5] [0..2]	1, 12, 19	1, 12, 19	0	1440											
	1495	3	coef_probs_32x32 [1] [0] [3] [0] [0..2]	55, 153, 210	55, 153, 210	0	1443											
	1498	3	coef_probs_32x32 [1] [0] [3] [1] [0..2]	24, 130, 194	24, 130,	0	1446											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT		State counter EBB Address	Coefficient counter EBB Address							
												4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					194														
	1501	3	coef_probs_32x32 [1] [0] [3] [2] [0..2]	3, 93, 146	3, 93, 146	0	1449												
	1504	3	coef_probs_32x32 [1] [0] [3] [3] [0..2]	1, 61, 97	1, 61, 97	0	1452												
	1507	3	coef_probs_32x32 [1] [0] [3] [4] [0..2]	1, 31, 50	1, 31, 50	0	1455												
	1510	3	coef_probs_32x32 [1] [0] [3] [5] [0..2]	1, 10, 16	1, 10, 16	0	1458												
	1513	3	coef_probs_32x32 [1] [0] [4] [0] [0..2]	49, 186, 223	49, 186, 223	0	1461												
	1516	3	coef_probs_32x32 [1] [0] [4] [1] [0..2]	17, 148, 204	17, 148, 204	0	1464												
	1519	3	coef_probs_32x32 [1] [0] [4] [2] [0..2]	1, 96, 142	1, 96, 142	0	1467												
	1522	3	coef_probs_32x32 [1] [0] [4] [3] [0..2]	1, 53, 83	1, 53, 83	0	1470												
	1525	3	coef_probs_32x32 [1] [0] [4] [4]	1, 26, 44	1, 26,	0	1473												

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			[0..2]		44													
	1528	3	coef_probs_32x32 [1] [0] [4] [5] [0..2]	1, 11, 17	1, 11, 17	0	1476											
	1531	3	coef_probs_32x32 [1] [0] [5] [0] [0..2]	13, 217, 212	13, 217, 212	0	1479											
	1534	3	coef_probs_32x32 [1] [0] [5] [1] [0..2]	2, 136, 180	2, 136, 180	0	1482											
	1537	3	coef_probs_32x32 [1] [0] [5] [2] [0..2]	1, 78, 124	1, 78, 124	0	1485											
	1540	3	coef_probs_32x32 [1] [0] [5] [3] [0..2]	1, 50, 83	1, 50, 83	0	1488											
	1543	3	coef_probs_32x32 [1] [0] [5] [4] [0..2]	1, 29, 49	1, 29, 49	0	1491											
	1546	3	coef_probs_32x32 [1] [0] [5] [5] [0..2]	1, 14, 23	1, 14, 23	0	1494											
	1549	3	coef_probs_32x32 [1] [1] [0] [0] [0..2]	197, 13, 247	197, 13, 247	0	1497											
	1552	3	coef_probs_32x32 [1] [1] [0] [1] [0..2]	82, 17, 222	82, 17,	0	1500											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					222													
	1555	3	coef_probs_32x32 [1] [1] [0] [2] [0..2]	25, 17, 162	25, 17, 162	0	1503											
	1558	3	coef_probs_32x32 [1] [1] [1] [0] [0..2]	126, 186, 247	126, 186, 247	0	1506											
	1561	3	coef_probs_32x32 [1] [1] [1] [1] [0..2]	234, 191, 243	234, 191, 243	0	1509											
	1564	3	coef_probs_32x32 [1] [1] [1] [2] [0..2]	176, 177, 234	176, 177, 234	0	1512											
	1567	3	coef_probs_32x32 [1] [1] [1] [3] [0..2]	104, 158, 220	104, 158, 220	0	1515											
	1570	3	coef_probs_32x32 [1] [1] [1] [4] [0..2]	66, 128, 186	66, 128, 186	0	1518											
	1573	3	coef_probs_32x32 [1] [1] [1] [5] [0..2]	55, 90, 137	55, 90, 137	0	1521											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	1576	3	coef_probs_32x32 [1] [1] [2] [0] [0..2]	111, 197, 242	111, 197, 242	0	1524											
	1579	3	coef_probs_32x32 [1] [1] [2] [1] [0..2]	46, 158, 219	46, 158, 219	0	1527											
	1582	3	coef_probs_32x32 [1] [1] [2] [2] [0..2]	9, 104, 171	9, 104, 171	0	1530											
	1585	3	coef_probs_32x32 [1] [1] [2] [3] [0..2]	2, 65, 125	2, 65, 125	0	1533											
	1588	3	coef_probs_32x32 [1] [1] [2] [4] [0..2]	1, 44, 80	1, 44, 80	0	1536											
	1591	3	coef_probs_32x32 [1] [1] [2] [5] [0..2]	1, 17, 91	1, 17, 91	0	1539											
	1594	3	coef_probs_32x32 [1] [1] [3] [0] [0..2]	104, 208, 245	104, 208, 245	0	1542											
	1597	3	coef_probs_32x32 [1] [1] [3] [1] [0..2]	39, 168, 224	39, 168, 224	0	1545											
	1600	3	coef_probs_32x32	3, 109,	3, 10	0	1548											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			[1] [1] [3] [2] [0..2]	162	9, 162													
	1603	3	coef_probs_32x32 [1] [1] [3] [3] [0..2]	1, 79, 124	1, 79, 124	0	1551											
	1606	3	coef_probs_32x32 [1] [1] [3] [4] [0..2]	1, 50, 102	1, 50, 102	0	1554											
	1609	3	coef_probs_32x32 [1] [1] [3] [5] [0..2]	1, 43, 102	1, 43, 102	0	1557											
	1612	3	coef_probs_32x32 [1] [1] [4] [0] [0..2]	84, 220, 246	84, 220, 246	0	1560											
	1615	3	coef_probs_32x32 [1] [1] [4] [1] [0..2]	31, 177, 231	31, 177, 231	0	1563											
	1618	3	coef_probs_32x32 [1] [1] [4] [2] [0..2]	2, 115, 180	2, 115, 180	0	1566											
	1621	3	coef_probs_32x32 [1] [1] [4] [3] [0..2]	1, 79, 134	1, 79, 134	0	1569											
	1624	3	coef_probs_32x32 [1] [1] [4] [4] [0..2]	1, 55, 77	1, 55, 77	0	1572											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	1627	3	coef_probs_32x32 [1] [1] [4] [5] [0..2]	1, 60, 79	1, 60, 79	0	1575											
	1630	3	coef_probs_32x32 [1] [1] [5] [0] [0..2]	43, 243, 240	43, 243, 240	0	1578											
	1633	3	coef_probs_32x32 [1] [1] [5] [1] [0..2]	8, 180, 217	8, 180, 217	0	1581											
	1636	3	coef_probs_32x32 [1] [1] [5] [2] [0..2]	1, 115, 166	1, 115, 166	0	1584											
	1639	3	coef_probs_32x32 [1] [1] [5] [3] [0..2]	1, 84, 121	1, 84, 121	0	1587											
	1642	3	coef_probs_32x32 [1] [1] [5] [4] [0..2]	1, 51, 67	1, 51, 67	0	1590											
	1645	3	coef_probs_32x32 [1] [1] [5] [5] [0..2]	1, 16, 6	1, 16, 6	0	1593											
	1648	16	DUMMY	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,	16	1596											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT		State counter EBB Address	Coefficient counter EBB Address							
												4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					0, 0, 0, 0, 0														
CL aligned	1664	3	mbskip_probs [0..2]	192, 128, 64	192, 128, 64	0	1596	12	208	MODE COUNTERS (Others)	18-23								
	1667	3	inter_mode_probs [0] [0..2]	0, 0, 0	2, 173, 34	0	1599	15			24-51								
	1670	3	inter_mode_probs [1] [0..2]	0, 0, 0	7, 145, 85	0	1602	18											
	1673	3	inter_mode_probs [2] [0..2]	0, 0, 0	7, 166, 63	0	1605	21											
	1676	3	inter_mode_probs [3] [0..2]	0, 0, 0	7, 94, 66	0	1608	24											
	1679	3	inter_mode_probs [4] [0..2]	0, 0, 0	8, 64, 46	0	1611	27											
	1682	3	inter_mode_probs [5] [0..2]	0, 0, 0	17, 81, 31	0	1614	30											
	1685	3	inter_mode_probs [6] [0..2]	0, 0, 0	25, 29, 30	0	1617	33											
	1688	2	switchable_interp_probs [0] [0..1]	0, 0	235, 16	0	1620	36			52-63								

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					2													
	1690	2	switchable_interp_probs [1] [0..1]	0, 0	36, 255	0	16	38										
	1692	2	switchable_interp_probs [2] [0..1]	0, 0	34, 3	0	16	40										
	1694	2	switchable_interp_probs [3] [0..1]	0, 0	149, 144	0	16	42										
	1696	4	intra_inter_probs [0..3]	0, 0, 0, 0	9, 102, 187, 225	0	16	44		64-71								
	1700	5	comp_inter_probs [0..4]	0, 0, 0, 0, 0	239, 183, 119, 96, 41	0	16	48		72-81								
	1705	2	single_ref_probs [0] [0..1]	0, 0	33, 16	0	16	53		82-101								
	1707	2	single_ref_probs [1] [0..1]	0, 0	77, 74	0	16	55										
	1709	2	single_ref_probs [2] [0..1]	0, 0	142, 142	0	16	57										
	1711	2	single_ref_probs [3] [0..1]	0, 0	172, 170	0	16	59										
	171	2	single_ref_pro	0, 0	23	0	16	6										

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
	3		bs [4] [0..1]		8, 247		45	1										
	1715	5	comp_ref_probs [0..4]	0, 0, 0, 0, 0	50, 126, 123, 221, 226	0	1647	63		102-111								
	1720	9	y_mode_probs [0] [0..8]	0, 0, 0, 0, 0, 0, 0, 0, 0	65, 32, 18, 144, 162, 194, 41, 51, 98	0	1652	68		112-151								
	1729	9	y_mode_probs [1] [0..8]	0, 0, 0, 0, 0, 0, 0, 0, 0	132, 68, 18, 165, 217, 196, 45, 40	0	1661	77										

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					,78													
	1738	9	y_mode_probs [2] [0..8]	0, 0, 0, 0, 0, 0, 0, 0, 0	173,80,19,176,240,193,64,35,46	0	1670	86										
	1747	9	y_mode_probs [3] [0..8]	0, 0, 0, 0, 0, 0, 0, 0, 0	221,135,38,194,248,121,96,85,29	0	1679	95										
	1756	3	partition_probs [0] [0..2]	158, 97, 94	199,122,141	0	1688	104		152-215								
	1759	3	partition_probs [1] [0..2]	93, 24, 99	147,63	0	1691	107										

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					, 159													
	1762	3	partition_probs [2] [0..2]	85, 119, 44	148, 133, 118	0	1694	110										
	1765	3	partition_probs [3] [0..2]	62, 59, 67	121, 104, 114	0	1697	113										
	1768	3	partition_probs [4] [0..2]	149, 53, 53	174, 73, 87	0	1700	116										
	1771	3	partition_probs [5] [0..2]	94, 20, 48	92, 41, 83	0	1703	119										
	1774	3	partition_probs [6] [0..2]	83, 53, 24	82, 99, 50	0	1706	122										
	1777	3	partition_probs [7] [0..2]	52, 18, 18	53, 39, 39	0	1709	125										
	1780	3	partition_probs [8] [0..2]	150, 40, 39	177, 58, 59	0	1712	128										
	1783	3	partition_probs [9] [0..2]	78, 12, 26	68, 26,	0	1715	131										

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT		State counter EBB Address	Coefficient counter EBB Address							
												4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					63														
	1786	3	partition_probs [10] [0..2]	67, 33, 11	52, 79, 25	0	1718	134											
	1789	3	partition_probs [11] [0..2]	24, 7, 5	17, 14, 12	0	1721	137											
	1792	3	partition_probs [12] [0..2]	174, 35, 49	222, 34, 30	0	1724	140											
	1795	3	partition_probs [13] [0..2]	68, 11, 27	72, 16, 44	0	1727	143											
	1798	3	partition_probs [14] [0..2]	57, 15, 9	58, 32, 12	0	1730	146											
	1801	3	partition_probs [15] [0..2]	12, 3, 3	10, 7, 6	0	1733	149											
	1804	3	mvc_joints [3]	?,?,?	?, ?, ?	0	1736	152		MV COUNTERS	216-219								
	1807	1	mv_sign [0]	0	128	0	1739	155			220-221								
	1808	10	mv_classes [0] [0..9]	0, 0, 0, 0, 0, 0, 0, 0, 0, 0	224, 144, 192, 16	0	1740	156			222-232								

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT		State counter EBB Address	Coefficient counter EBB Address							
												4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					8, 192, 176, 192, 198, 198, 245														
	1818	1	mv_class0 [0] [0..0]	0	216	0	1750	166			233-234								
	1819	10	mv_bits [0] [0..9]	0, 0, 0, 0, 0, 0, 0, 0, 0, 0	136, 140, 148, 160, 176, 192, 224, 234, 234, 240	0	1751	167			235-254								
	1829	1	mv_sign [1]	0	128	0	1761	177			255-256								
	1830	10	mv_classes [1] [0..9]	0, 0, 0, 0, 0, 0, 0, 0, 0, 0	216, 128, 176,	0	1762	178			257-267								

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					160, 176, 176, 192, 198, 198, 208													
	1840	1	mv_class0 [1] [0..0]	0	208	0	1772	188		268-269								
	1841	10	mv_bits [1] [0..9]	0, 0, 0, 0, 0, 0, 0, 0, 0, 0	136, 140, 148, 160, 176, 192, 224, 234, 234, 240	0	1773	189		270-289								
	1851	3	mv_class0_fp [0] [0] [0..2]	0, 0, 0	128, 128, 64	0	1783	199		290-297								
	1854	3	mv_class0_fp [0] [1] [0..2]	0, 0, 0	96, 11	0	1786	202										

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT		State counter EBB Address	Coefficient counter EBB Address							
												4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					2, 64														
	1863	3	mv_fp [0] [0..2]	0, 0, 0	64, 96, 64	0	1789	205			298-301								
	1857	3	mv_class0_fp [1] [0] [0..2]	0, 0, 0	128, 128, 64	0	1792	208			302-309								
	1860	3	mv_class0_fp [1] [1] [0..2]	0, 0, 0	96, 112, 64	0	1795	211											
	1866	3	mv_fp [1] [0..2]	0, 0, 0	64, 96, 64	0	1798	214			310-313								
	1869	2	mv_class0_hp [0..1]	0, 0	160, 160	0	1801	217			314-315								
	1871	2	mv_hp [0..1]	0, 0	128, 128	0	1803	219			316-317								
	1873	47	DUMMY	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,	47	1805	221											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT		State counter EBB Address	Coefficient counter EBB Address							
												4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
				0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0														
CL aligned	1920	9	uv_mode_probs [0] [0..8]	144, 11, 54, 157, 195, 130, 46, 58, 108	120, 7, 76, 176, 208, 126, 28,	0	1805	221	240	MODE COUNTERS (Others)	318-417								

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT		State counter EBB Address	Coefficient counter EBB Address							
												4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					54, 103														
	1929	9	uv_mode_probs [1] [0..8]	118, 15, 123, 148, 131, 101, 44, 93, 131	48, 12, 154, 155, 139, 90, 34, 117, 119	0	1814	230											
	1938	9	uv_mode_probs [2] [0..8]	113, 12, 23, 188, 226, 142, 26, 32, 125	67, 6, 25, 204, 243, 158, 13, 21, 96	0	1823	239											
	1947	9	uv_mode_probs [3] [0..8]	120, 11, 50, 123, 163, 135, 64,	97, 5, 44, 131,	0	1832	248											

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
				77, 103	176, 139, 48, 68, 97													
	1956	9	uv_mode_probs [4] [0..8]	113, 9, 36, 155, 111, 157, 32, 44, 161	83, 5, 42, 156, 111, 152, 26, 49, 152	0	1841	257										
	1965	9	uv_mode_probs [5] [0..8]	116, 9, 55, 176, 76, 96, 37, 61, 149	80, 5, 58, 178, 74, 83, 33, 62, 145	0	1850	266										
	1974	9	uv_mode_probs	115, 9, 28,	86,	0	1859	27										

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
			[6] [0..8]	141, 161, 167, 21, 25, 193	5, 32, 154, 192, 168, 14, 22, 163			5										
	1983	9	uv_mode_probs [7] [0..8]	120, 12, 32, 145, 195, 142, 32, 38, 86	85, 5, 32, 156, 216, 148, 19, 29, 73	0	1868	284										
	1992	9	uv_mode_probs [8] [0..8]	116, 12, 64, 120, 140, 125, 49, 115, 121	77, 7, 64, 116, 132, 122, 37, 126,	0	1877	293										

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults				Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
											4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
					120													
	2001	9	uv_mode_probs [9] [0..8]	102, 19, 66, 162, 182, 122, 35, 59, 128	101, 21, 107, 181, 192, 103, 19, 67, 125	0	1886	302										
	2010	7	seg_tree_probs [0..6]	255, 255, 255, 255, 255, 255, 255	255, 255, 255, 255, 255, 255, 255	0	1895	311										
	2017	3	seg_pred_probs [0..2]	255, 255, 255	255, 255, 255	0	1902	318										
	2020	28	DUMMY	0, 0, 0, 0, 0, 0, 0, 0, 0, 0,	0, 0, 0, 0, 0,	115	1905	321										

Alignment	New Offset	# Bytes	Description	Keyframe defaults	Inter frame defaults	Capture At DV_CNT	State counter EBB Address	Coefficient counter EBB Address							
								4x4 (KF)	4x4 (INTER)	8x8 (KF)	8x8 (INTER)	16x16 (KF)	16x16 (INTER)	32x32 (KF)	32x32 (INTER)
				0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0										
CL aligned	2048														

Stream-in formats for creating compressed header

The following memory surfaces are input to PAK for Compressed Header coding

i. Prob Diff Surface

In Probability Diff Surface, there are 1805 8-bit Probability Diffs. Each of them corresponding to a Probability Diff in Compressed Header syntax. Although, for a given compressed header, not all the Probability Diff would be coded (depends on update flag), Probability Diff Surface is fully populated with 1805 entries (1805*8 / 512 = 29 cachelines). The 1805 8-bit Probability Diffs are expected to follow Compressed Header syntax order and fully packed.

ii. Compressed Header Syntax Surface

Each of the Compressed header Coding element (described in (2)) is represented by a 4-bit field. These 4-bit fields follows Compressed Header Syntax. Each of the field has a valid, Bin_probDiff_select, Prob_select, Bin as described in the table below.

	Description
Valid	Set to 1 if this is a valid Bin OR ProbabilityDiff field to code; Set to 0 to skip coding this field
Bin_ProbDiff_select	Set to 1 if Current field is a Bin (corresponding Prob, Bin are indicated by next 2 bits); Set to 0 if Current field is Probability Diff (probability diff to be coded is located in probability surface - ReMap)
Prob_Select	If current field is Bin, set to 1 if prob is 252; set to 0 if prob is 128
Bin	if current field is Bin, this is Bin value to be encoded

Compressed Header Syntax Surface is a fixed length surface. For syntax that should not be coded, valid bit should be set to 0. Total length of Compressed Header syntax Surface has 4033 Coding elements (16132 bits in 32 cachelines):

1805 Prob Diff and Prob Update flag

4 is_coeff_updated flag (per 4x4, 8x8, 16x16, 32x32)

5 control fields (MIN (tx_mode, ALLOW_32x32), tx_mode == TX_MODE_SELECT, use_compound_pred, use_hybrid_pred)

VP9 PAK Quant Lookup Tables

Qindex	IQ_Scale						FQ_Shift						FQ_Quant					
	8b		10b		12b		8b		10b		12b		8b		10b		12b	
	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC
0	4	4	4	4	4	4	1	1	1	1	1	1	32768	32768	32768	32768	32768	32768
1	8	8	9	9	12	13	2	2	2	2	2	2	32768	32768	29127	29127	21845	20164
2	8	9	10	11	18	19	2	2	2	2	3	3	32768	29127	26214	23831	29127	27594
3	9	10	13	13	25	27	2	2	2	2	3	3	29127	26214	20164	20164	20971	19418
4	10	11	15	16	33	35	2	2	2	3	4	4	26214	23831	17476	32768	31775	29959
5	11	12	17	18	41	44	2	2	3	3	4	4	23831	21845	30840	29127	25575	23831
6	12	13	20	21	50	54	2	2	3	3	4	4	21845	20164	26214	24966	20971	19418

Qindex	IQ_Scale						FQ_Shift						FQ_Quant					
	8b		10b		12b		8b		10b		12b		8b		10b		12b	
	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC
7	12	14	22	24	60	64	2	2	3	3	4	5	21845	18724	23831	21845	17476	32768
8	13	15	25	27	70	75	2	2	3	3	5	5	20164	17476	20971	19418	29959	27962
9	14	16	28	30	80	87	2	3	3	3	5	5	18724	32768	18724	17476	26214	24105
10	15	17	31	33	91	99	2	3	3	4	5	5	17476	30840	16912	31775	23045	21183
11	16	18	34	37	103	112	3	3	4	4	5	5	32768	29127	30840	28339	20360	18724
12	17	19	37	40	115	126	3	3	4	4	5	5	30840	27594	28339	26214	18236	16644
13	18	20	40	44	127	139	3	3	4	4	5	6	29127	26214	26214	23831	16513	30174
14	19	21	43	48	140	154	3	3	4	4	6	6	27594	24966	24385	21845	29959	27235
15	19	22	47	51	153	168	3	3	4	4	6	6	27594	23831	22310	20560	27413	24966
16	20	23	50	55	166	183	3	3	4	4	6	6	26214	22795	20971	19065	25266	22919
17	21	24	53	59	180	199	3	3	4	4	6	6	24966	21845	19784	17772	23301	21076
18	22	25	57	63	194	214	3	3	4	4	6	6	23831	20971	18396	16644	21620	19599
19	23	26	60	67	208	230	3	3	4	5	6	6	22795	20164	17476	31300	20164	18236
20	24	27	64	71	222	247	3	3	5	5	6	6	21845	19418	32768	29537	18893	16980
21	25	28	68	75	237	263	3	3	5	5	6	7	20971	18724	30840	27962	17697	31895
22	26	29	71	79	251	280	3	3	5	5	6	7	20164	18078	29537	26546	16710	29959
23	26	30	75	83	266	297	3	3	5	5	7	7	20164	17476	27962	25266	31536	28244
24	27	31	78	88	281	314	3	3	5	5	7	7	19418	16912	26886	23831	29852	26715
25	28	32	82	92	296	331	3	4	5	5	7	7	18724	32768	25575	22795	28339	25343

Qindex	IQ_Scale						FQ_Shift						FQ_Quant					
	8b		10b		12b		8b		10b		12b		8b		10b		12b	
	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC
26	29	33	86	96	312	349	3	4	5	5	7	7	18078	31775	24385	21845	26886	24036
27	30	34	90	100	327	366	3	4	5	5	7	7	17476	30840	23301	20971	25653	22919
28	31	35	93	105	343	384	3	4	5	5	7	7	16912	29959	22550	19972	24456	21845
29	32	36	97	109	358	402	4	4	5	5	7	7	32768	29127	21620	19239	23431	20867
30	32	37	101	114	374	420	4	4	5	5	7	7	32768	28339	20763	18396	22429	19972
31	33	38	105	118	390	438	4	4	5	5	7	7	31775	27594	19972	17772	21509	19152
32	34	39	109	122	405	456	4	4	5	5	7	7	30840	26886	19239	17189	20712	18396
33	35	40	113	127	421	475	4	4	5	5	7	7	29959	26214	18558	16513	19925	17660
34	36	41	116	131	437	493	4	4	5	6	7	7	29127	25575	18078	32017	19195	17015
35	37	42	120	136	453	511	4	4	5	6	7	7	28339	24966	17476	30840	18517	16416
36	38	43	124	140	469	530	4	4	5	6	7	8	27594	24385	16912	29959	17886	31655
37	38	44	128	145	484	548	4	4	6	6	7	8	27594	23831	32768	28926	17331	30615
38	39	45	132	149	500	567	4	4	6	6	7	8	26886	23301	31775	28149	16777	29589
39	40	46	136	154	516	586	4	4	6	6	8	8	26214	22795	30840	27235	32513	28630
40	41	47	140	158	532	604	4	4	6	6	8	8	25575	22310	29959	26546	31536	27776
41	42	48	143	163	548	623	4	4	6	6	8	8	24966	21845	29330	25731	30615	26929
42	43	49	147	168	564	642	4	4	6	6	8	8	24385	21399	28532	24966	29746	26132
43	43	50	151	172	580	660	4	4	6	6	8	8	24385	20971	27776	24385	28926	25420
44	44	51	155	177	596	679	4	4	6	6	8	8	23831	20560	27060	23696	28149	24708

Qindex	IQ_Scale						FQ_Shift						FQ_Quant					
	8b		10b		12b		8b		10b		12b		8b		10b		12b	
	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC
45	45	52	159	181	611	698	4	4	6	6	8	8	23301	20164	26379	23172	27458	24036
46	46	53	163	186	627	716	4	4	6	6	8	8	22795	19784	25731	22550	26757	23431
47	47	54	166	190	643	735	4	4	6	6	8	8	22310	19418	25266	22075	26092	22826
48	48	55	170	195	659	753	4	4	6	6	8	8	21845	19065	24672	21509	25458	22280
49	48	56	174	199	674	772	4	4	6	6	8	8	21845	18724	24105	21076	24892	21732
50	49	57	178	204	690	791	4	4	6	6	8	8	21399	18396	23563	20560	24314	21210
51	50	58	182	208	706	809	4	4	6	6	8	8	20971	18078	23045	20164	23763	20738
52	51	59	185	213	721	828	4	4	6	6	8	8	20560	17772	22671	19691	23269	20262
53	52	60	189	217	737	846	4	4	6	6	8	8	20164	17476	22192	19328	22764	19831
54	53	61	193	222	752	865	4	4	6	6	8	8	19784	17189	21732	18893	22310	19395
55	53	62	197	226	768	884	4	4	6	6	8	8	19784	16912	21290	18558	21845	18978
56	54	63	200	231	783	902	4	4	6	6	8	8	19418	16644	20971	18157	21426	18600
57	55	64	204	235	798	920	4	5	6	6	8	8	19065	32768	20560	17848	21024	18236
58	56	65	208	240	814	939	4	5	6	6	8	8	18724	32263	20164	17476	20610	17867
59	57	66	212	244	829	957	4	5	6	6	8	8	18396	31775	19784	17189	20237	17531
60	57	67	215	249	844	976	4	5	6	6	8	8	18396	31300	19508	16844	19878	17189
61	58	68	219	253	859	994	4	5	6	6	8	8	18078	30840	19152	16578	19531	16878
62	59	69	223	258	874	1012	4	5	6	7	8	8	17772	30393	18808	32513	19195	16578
63	60	70	226	262	889	1030	4	5	6	7	8	9	17476	29959	18558	32017	18872	32577

Qindex	IQ_Scale						FQ_Shift						FQ_Quant					
	8b		10b		12b		8b		10b		12b		8b		10b		12b	
	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC
64	61	71	230	267	904	1049	4	5	6	7	8	9	17189	29537	18236	31418	18558	31987
65	62	72	233	271	919	1067	4	5	6	7	8	9	16912	29127	18001	30954	18255	31447
66	62	73	237	275	934	1085	4	5	6	7	8	9	16912	28728	17697	30504	17962	30925
67	63	74	241	280	949	1103	4	5	6	7	8	9	16644	28339	17403	29959	17678	30421
68	64	75	244	284	964	1121	5	5	6	7	8	9	32768	27962	17189	29537	17403	29932
69	65	76	248	289	978	1139	5	5	6	7	8	9	32263	27594	16912	29026	17154	29459
70	66	77	251	293	993	1157	5	5	6	7	8	9	31775	27235	16710	28630	16895	29001
71	66	78	255	297	1008	1175	5	5	6	7	8	9	31775	26886	16448	28244	16644	28556
72	67	79	259	302	1022	1193	5	5	7	7	8	9	31300	26546	32388	27776	16416	28126
73	68	80	262	306	1037	1211	5	5	7	7	9	9	30840	26214	32017	27413	32357	27708
74	69	81	266	311	1051	1229	5	5	7	7	9	9	30393	25890	31536	26973	31926	27302
75	70	82	269	315	1065	1246	5	5	7	7	9	9	29959	25575	31184	26630	31506	26929
76	70	83	273	319	1080	1264	5	5	7	7	9	9	29959	25266	30727	26296	31068	26546
77	71	84	276	324	1094	1282	5	5	7	7	9	9	29537	24966	30393	25890	30671	26173
78	72	85	280	328	1108	1299	5	5	7	7	9	9	29127	24672	29959	25575	30283	25830
79	73	86	283	332	1122	1317	5	5	7	7	9	9	28728	24385	29641	25266	29905	25477
80	74	87	287	337	1136	1335	5	5	7	7	9	9	28339	24105	29228	24892	29537	25134
81	74	88	290	341	1151	1352	5	5	7	7	9	9	28339	23831	28926	24600	29152	24818
82	75	89	293	345	1165	1370	5	5	7	7	9	9	27962	23563	28630	24314	28802	24492

Qindex	IQ_Scale						FQ_Shift						FQ_Quant					
	8b		10b		12b		8b		10b		12b		8b		10b		12b	
	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC
83	76	90	297	349	1179	1387	5	5	7	7	9	9	27594	23301	28244	24036	28460	24192
84	77	91	300	354	1192	1405	5	5	7	7	9	9	27235	23045	27962	23696	28149	23882
85	78	92	304	358	1206	1422	5	5	7	7	9	9	26886	22795	27594	23431	27822	23596
86	78	93	307	362	1220	1440	5	5	7	7	9	9	26886	22550	27324	23172	27503	23301
87	79	94	310	367	1234	1457	5	5	7	7	9	9	26546	22310	27060	22857	27191	23029
88	80	95	314	371	1248	1474	5	5	7	7	9	9	26214	22075	26715	22610	26886	22764
89	81	96	317	375	1261	1491	5	5	7	7	9	9	25890	21845	26462	22369	26609	22504
90	81	97	321	379	1275	1509	5	5	7	7	9	9	25890	21620	26132	22133	26317	22236
91	82	98	324	384	1288	1526	5	5	7	7	9	9	25575	21399	25890	21845	26051	21988
92	83	99	327	388	1302	1543	5	5	7	7	9	9	25266	21183	25653	21620	25771	21746
93	84	100	331	392	1315	1560	5	5	7	7	9	9	24966	20971	25343	21399	25516	21509
94	85	101	334	396	1329	1577	5	5	7	7	9	9	24672	20763	25115	21183	25247	21277
95	85	102	337	401	1342	1595	5	5	7	7	9	9	24672	20560	24892	20919	25003	21037
96	87	104	343	409	1368	1627	5	5	7	7	9	9	24105	20164	24456	20510	24528	20623
97	88	106	350	417	1393	1660	5	5	7	7	9	9	23831	19784	23967	20116	24087	20213
98	90	108	356	425	1419	1693	5	5	7	7	9	9	23301	19418	23563	19737	23646	19819
99	92	110	362	433	1444	1725	5	5	7	7	9	9	22795	19065	23172	19373	23237	19451
100	93	112	369	441	1469	1758	5	5	7	7	9	9	22550	18724	22733	19021	22841	19086
101	95	114	375	449	1494	1791	5	5	7	7	9	9	22075	18396	22369	18682	22459	18735

Qindex	IQ_Scale						FQ_Shift						FQ_Quant					
	8b		10b		12b		8b		10b		12b		8b		10b		12b	
	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC
102	96	116	381	458	1519	1824	5	5	7	7	9	9	21845	18078	22017	18315	22089	18396
103	98	118	387	466	1544	1856	5	5	7	7	9	9	21399	17772	21675	18001	21732	18078
104	99	120	394	474	1569	1889	5	5	7	7	9	9	21183	17476	21290	17697	21385	17763
105	101	122	400	482	1594	1922	5	5	7	7	9	9	20763	17189	20971	17403	21050	17458
106	102	124	406	490	1618	1954	5	5	7	7	9	9	20560	16912	20661	17119	20738	17172
107	104	126	412	498	1643	1987	5	5	7	7	9	9	20164	16644	20360	16844	20422	16886
108	105	128	418	506	1668	2020	5	6	7	7	9	9	19972	32768	20068	16578	20116	16611
109	107	130	424	514	1692	2052	5	6	7	8	9	10	19599	32263	19784	32640	19831	32704
110	108	132	430	523	1717	2085	5	6	7	8	9	10	19418	31775	19508	32078	19542	32186
111	110	134	436	531	1741	2118	5	6	7	8	9	10	19065	31300	19239	31595	19273	31685
112	111	136	442	539	1765	2150	5	6	7	8	9	10	18893	30840	18978	31126	19011	31213
113	113	138	448	547	1789	2183	5	6	7	8	9	10	18558	30393	18724	30671	18755	30741
114	114	140	454	555	1814	2216	5	6	7	8	9	10	18396	29959	18477	30229	18497	30283
115	116	142	460	563	1838	2248	5	6	7	8	9	10	18078	29537	18236	29799	18255	29852
116	117	144	466	571	1862	2281	5	6	7	8	9	10	17924	29127	18001	29382	18020	29420
117	118	146	472	579	1885	2313	5	6	7	8	9	10	17772	28728	17772	28976	17800	29013
118	120	148	478	588	1909	2346	5	6	7	8	9	10	17476	28339	17549	28532	17576	28605
119	121	150	484	596	1933	2378	5	6	7	8	9	10	17331	27962	17331	28149	17358	28220
120	123	152	490	604	1957	2411	5	6	7	8	9	10	17050	27594	17119	27776	17145	27834

Qindex	IQ_Scale						FQ_Shift						FQ_Quant					
	8b		10b		12b		8b		10b		12b		8b		10b		12b	
	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC
121	125	155	499	616	1992	2459	5	6	7	8	9	10	16777	27060	16810	27235	16844	27291
122	127	158	507	628	2027	2508	5	6	7	8	9	10	16513	26546	16545	26715	16553	26757
123	129	161	516	640	2061	2556	6	6	8	8	10	10	32513	26051	32513	26214	32561	26255
124	131	164	525	652	2096	2605	6	6	8	8	10	10	32017	25575	31956	25731	32017	25761
125	134	167	533	664	2130	2653	6	6	8	8	10	10	31300	25115	31476	25266	31506	25295
126	136	170	542	676	2165	2701	6	6	8	8	10	10	30840	24672	30954	24818	30997	24845
127	138	173	550	688	2199	2750	6	6	8	8	10	10	30393	24244	30504	24385	30517	24403
128	140	176	559	700	2233	2798	6	6	8	8	10	10	29959	23831	30012	23967	30053	23984
129	142	179	567	713	2267	2847	6	6	8	8	10	10	29537	23431	29589	23530	29602	23571
130	144	182	576	725	2300	2895	6	6	8	8	10	10	29127	23045	29127	23140	29177	23180
131	146	185	584	737	2334	2943	6	6	8	8	10	10	28728	22671	28728	22764	28752	22802
132	148	188	592	749	2367	2992	6	6	8	8	10	10	28339	22310	28339	22399	28351	22429
133	150	191	601	761	2400	3040	6	6	8	8	10	10	27962	21959	27915	22046	27962	22075
134	152	194	609	773	2434	3088	6	6	8	8	10	10	27594	21620	27548	21704	27571	21732
135	154	197	617	785	2467	3137	6	6	8	8	10	10	27235	21290	27191	21372	27202	21392
136	156	200	625	797	2499	3185	6	6	8	8	10	10	26886	20971	26843	21050	26854	21070
137	158	203	634	809	2532	3234	6	6	8	8	10	10	26546	20661	26462	20738	26504	20751
138	161	207	644	825	2575	3298	6	6	8	8	10	10	26051	20262	26051	20336	26061	20348
139	164	211	655	841	2618	3362	6	6	8	8	10	10	25575	19878	25614	19949	25633	19960

Qindex	IQ_Scale						FQ_Shift						FQ_Quant					
	8b		10b		12b		8b		10b		12b		8b		10b		12b	
	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC
140	166	215	666	857	2661	3426	6	6	8	8	10	10	25266	19508	25191	19576	25219	19588
141	169	219	676	873	2704	3491	6	6	8	8	10	10	24818	19152	24818	19217	24818	19223
142	172	223	687	889	2746	3555	6	6	8	8	10	10	24385	18808	24420	18872	24438	18877
143	174	227	698	905	2788	3619	6	6	8	8	10	10	24105	18477	24036	18538	24070	18543
144	177	231	708	922	2830	3684	6	6	8	8	10	10	23696	18157	23696	18196	23713	18216
145	180	235	718	938	2872	3748	6	6	8	8	10	10	23301	17848	23366	17886	23366	17905
146	182	239	729	954	2913	3812	6	6	8	8	10	10	23045	17549	23014	17586	23037	17604
147	185	243	739	970	2954	3876	6	6	8	8	10	10	22671	17260	22702	17296	22717	17313
148	187	247	749	986	2995	3941	6	6	8	8	10	10	22429	16980	22399	17015	22406	17028
149	190	251	759	1002	3036	4005	6	6	8	8	10	10	22075	16710	22104	16743	22104	16756
150	192	255	770	1018	3076	4069	6	6	8	8	10	10	21845	16448	21788	16480	21816	16492
151	195	260	782	1038	3127	4149	6	7	8	9	10	11	21509	32263	21454	32326	21461	32349
152	199	265	795	1058	3177	4230	6	7	8	9	10	11	21076	31655	21103	31714	21123	31729
153	202	270	807	1078	3226	4310	6	7	8	9	10	11	20763	31068	20789	31126	20802	31141
154	205	275	819	1098	3275	4390	6	7	8	9	10	11	20460	30504	20485	30559	20491	30573
155	208	280	831	1118	3324	4470	6	7	8	9	10	11	20164	29959	20189	30012	20189	30026
156	211	285	844	1138	3373	4550	6	7	8	9	10	11	19878	29433	19878	29485	19895	29498
157	214	290	856	1158	3421	4631	6	7	8	9	10	11	19599	28926	19599	28976	19616	28982
158	217	295	868	1178	3469	4711	6	7	8	9	10	11	19328	28435	19328	28484	19345	28490

Qindex	IQ_Scale						FQ_Shift						FQ_Quant					
	8b		10b		12b		8b		10b		12b		8b		10b		12b	
	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC
159	220	300	880	1198	3517	4791	6	7	8	9	10	11	19065	27962	19065	28008	19081	28014
160	223	305	891	1218	3565	4871	6	7	8	9	10	11	18808	27503	18829	27548	18824	27554
161	226	311	906	1242	3621	4967	6	7	8	9	10	11	18558	26973	18517	27016	18533	27021
162	230	317	920	1266	3677	5064	6	7	8	9	10	11	18236	26462	18236	26504	18250	26504
163	233	323	933	1290	3733	5160	6	7	8	9	10	11	18001	25970	17982	26011	17977	26011
164	237	329	947	1314	3788	5256	6	7	8	9	10	11	17697	25497	17716	25536	17716	25536
165	240	335	961	1338	3843	5352	6	7	8	9	10	11	17476	25040	17458	25078	17462	25078
166	243	341	975	1362	3897	5448	6	7	8	9	10	11	17260	24600	17207	24636	17220	24636
167	247	347	988	1386	3951	5544	6	7	8	9	10	11	16980	24174	16980	24209	16985	24209
168	250	353	1001	1411	4005	5641	6	7	8	9	10	11	16777	23763	16760	23780	16756	23793
169	253	359	1015	1435	4058	5737	6	7	8	9	10	11	16578	23366	16529	23382	16537	23395
170	257	366	1030	1463	4119	5849	7	7	9	9	11	11	32640	22919	32577	22935	32585	22947
171	261	373	1045	1491	4181	5961	7	7	9	9	11	11	32140	22489	32109	22504	32101	22515
172	265	380	1061	1519	4241	6073	7	7	9	9	11	11	31655	22075	31625	22089	31647	22100
173	269	387	1076	1547	4301	6185	7	7	9	9	11	11	31184	21675	31184	21690	31206	21700
174	272	394	1090	1575	4361	6297	7	7	9	9	11	11	30840	21290	30783	21304	30776	21314
175	276	401	1105	1603	4420	6410	7	7	9	9	11	11	30393	20919	30366	20932	30366	20938
176	280	408	1120	1631	4479	6522	7	7	9	9	11	11	29959	20560	29959	20572	29966	20579
177	284	416	1137	1663	4546	6650	7	7	9	9	11	11	29537	20164	29511	20177	29524	20183

Qindex	IQ_Scale						FQ_Shift						FQ_Quant					
	8b		10b		12b		8b		10b		12b		8b		10b		12b	
	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC
178	288	424	1153	1695	4612	6778	7	7	9	9	11	11	29127	19784	29101	19796	29101	19801
179	292	432	1170	1727	4677	6906	7	7	9	9	11	11	28728	19418	28679	19429	28697	19434
180	296	440	1186	1759	4742	7034	7	7	9	9	11	11	28339	19065	28292	19075	28304	19081
181	300	448	1202	1791	4807	7162	7	7	9	9	11	11	27962	18724	27915	18735	27921	18740
182	304	456	1218	1823	4871	7290	7	7	9	9	11	11	27594	18396	27548	18406	27554	18411
183	309	465	1236	1859	4942	7435	7	7	9	9	11	11	27147	18040	27147	18049	27158	18052
184	313	474	1253	1895	5013	7579	7	7	9	9	11	11	26800	17697	26779	17706	26773	17709
185	317	483	1271	1931	5083	7723	7	7	9	9	11	11	26462	17367	26400	17376	26405	17378
186	322	492	1288	1967	5153	7867	7	7	9	9	11	11	26051	17050	26051	17058	26046	17060
187	326	501	1306	2003	5222	8011	7	7	9	9	11	11	25731	16743	25692	16752	25702	16754
188	330	510	1323	2039	5291	8155	7	7	9	9	11	11	25420	16448	25362	16456	25367	16458
189	335	520	1342	2079	5367	8315	7	8	9	10	11	12	25040	32263	25003	32279	25007	32283
190	340	530	1361	2119	5442	8475	7	8	9	10	11	12	24672	31655	24654	31670	24663	31673
191	344	540	1379	2159	5517	8635	7	8	9	10	11	12	24385	31068	24332	31083	24328	31086
192	349	550	1398	2199	5591	8795	7	8	9	10	11	12	24036	30504	24001	30517	24006	30521
193	354	560	1416	2239	5665	8956	7	8	9	10	11	12	23696	29959	23696	29972	23692	29972
194	359	571	1436	2283	5745	9132	7	8	9	10	11	12	23366	29382	23366	29395	23362	29395
195	364	582	1456	2327	5825	9308	7	8	9	10	11	12	23045	28826	23045	28839	23041	28839
196	369	593	1476	2371	5905	9484	7	8	9	10	11	12	22733	28292	22733	28304	22729	28304

Qindex	IQ_Scale						FQ_Shift						FQ_Quant					
	8b		10b		12b		8b		10b		12b		8b		10b		12b	
	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC
197	374	604	1496	2415	5984	9660	7	8	9	10	11	12	22429	27776	22429	27788	22429	27788
198	379	615	1516	2459	6063	9836	7	8	9	10	11	12	22133	27280	22133	27291	22137	27291
199	384	627	1537	2507	6149	10028	7	8	9	10	11	12	21845	26757	21831	26768	21827	26768
200	389	639	1559	2555	6234	10220	7	8	9	10	11	12	21564	26255	21523	26265	21529	26265
201	395	651	1580	2603	6319	10412	7	8	9	10	11	12	21236	25771	21236	25781	21240	25781
202	400	663	1601	2651	6404	10604	7	8	9	10	11	12	20971	25305	20958	25314	20958	25314
203	406	676	1624	2703	6495	10812	7	8	9	10	11	12	20661	24818	20661	24827	20664	24827
204	411	689	1647	2755	6587	11020	7	8	9	10	11	12	20410	24350	20373	24358	20376	24358
205	417	702	1670	2807	6678	11228	7	8	9	10	11	12	20116	23899	20092	23907	20098	23907
206	423	715	1692	2859	6769	11437	7	8	9	10	11	12	19831	23464	19831	23472	19828	23470
207	429	729	1717	2915	6867	11661	7	8	9	10	11	12	19553	23014	19542	23021	19545	23019
208	435	743	1741	2971	6966	11885	7	8	9	10	11	12	19284	22580	19273	22587	19267	22586
209	441	757	1766	3027	7064	12109	7	8	9	10	11	12	19021	22162	19000	22170	19000	22168
210	447	771	1791	3083	7163	12333	7	8	9	10	11	12	18766	21760	18735	21767	18737	21765
211	454	786	1817	3143	7269	12573	7	8	9	10	11	12	18477	21345	18466	21351	18464	21350
212	461	801	1844	3203	7376	12813	7	8	9	10	11	12	18196	20945	18196	20951	18196	20950
213	467	816	1871	3263	7483	13053	7	8	9	10	11	12	17962	20560	17933	20566	17936	20565
214	475	832	1900	3327	7599	13309	7	8	9	10	11	12	17660	20164	17660	20170	17662	20169
215	482	848	1929	3391	7715	13565	7	8	9	10	11	12	17403	19784	17394	19790	17396	19788

Qindex	IQ_Scale						FQ_Shift						FQ_Quant					
	8b		10b		12b		8b		10b		12b		8b		10b		12b	
	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC
216	489	864	1958	3455	7832	13821	7	8	9	10	11	12	17154	19418	17137	19423	17137	19422
217	497	881	1990	3523	7958	14093	7	8	9	10	11	12	16878	19043	16861	19048	16865	19047
218	505	898	2021	3591	8085	14365	7	8	9	10	11	12	16611	18682	16602	18688	16600	18686
219	513	915	2054	3659	8214	14637	8	8	10	10	12	12	32704	18335	32672	18340	32680	18339
220	522	933	2088	3731	8352	14925	8	8	10	10	12	12	32140	17982	32140	17986	32140	17985
221	530	951	2123	3803	8492	15213	8	8	10	10	12	12	31655	17641	31610	17646	31610	17645
222	539	969	2159	3876	8635	15502	8	8	10	10	12	12	31126	17313	31083	17313	31086	17316
223	549	988	2197	3952	8788	15806	8	8	10	10	12	12	30559	16980	30545	16980	30545	16983
224	559	1007	2236	4028	8945	16110	8	8	10	10	12	12	30012	16660	30012	16660	30009	16662
225	569	1026	2276	4104	9104	16414	8	9	10	11	12	13	29485	32704	29485	32704	29485	32708
226	579	1046	2319	4184	9275	16734	8	9	10	11	12	13	28976	32078	28938	32078	28941	32082
227	590	1066	2363	4264	9450	17054	8	9	10	11	12	13	28435	31476	28399	31476	28405	31480
228	602	1087	2410	4348	9639	17390	8	9	10	11	12	13	27869	30868	27846	30868	27848	30872
229	614	1108	2458	4432	9832	17726	8	9	10	11	12	13	27324	30283	27302	30283	27302	30287
230	626	1129	2508	4516	10031	18062	8	9	10	11	12	13	26800	29720	26757	29720	26760	29723
231	640	1151	2561	4604	10245	18414	8	9	10	11	12	13	26214	29152	26204	29152	26201	29155
232	654	1173	2616	4692	10465	18766	8	9	10	11	12	13	25653	28605	25653	28605	25650	28608
233	668	1196	2675	4784	10702	19134	8	9	10	11	12	13	25115	28055	25087	28055	25082	28058
234	684	1219	2737	4876	10946	19502	8	9	10	11	12	13	24528	27526	24519	27526	24523	27529

Qindex	IQ_Scale						FQ_Shift						FQ_Quant					
	8b		10b		12b		8b		10b		12b		8b		10b		12b	
	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC
235	700	1243	2802	4972	11210	19886	8	9	10	11	12	13	23967	26994	23950	26994	23946	26997
236	717	1267	2871	5068	11482	20270	8	9	10	11	12	13	23399	26483	23374	26483	23378	26485
237	736	1292	2944	5168	11776	20670	8	9	10	11	12	13	22795	25970	22795	25970	22795	25973
238	755	1317	3020	5268	12081	21070	8	9	10	11	12	13	22221	25477	22221	25477	22219	25480
239	775	1343	3102	5372	12409	21486	8	9	10	11	12	13	21648	24984	21634	24984	21632	24987
240	796	1369	3188	5476	12750	21902	8	9	10	11	12	13	21076	24510	21050	24510	21053	24512
241	819	1396	3280	5584	13118	22334	8	9	10	11	12	13	20485	24036	20460	24036	20463	24038
242	843	1423	3375	5692	13501	22766	8	9	10	11	12	13	19901	23580	19884	23580	19882	23582
243	869	1451	3478	5804	13913	23214	8	9	10	11	12	13	19306	23125	19295	23125	19293	23127
244	896	1479	3586	5916	14343	23662	8	9	10	11	12	13	18724	22687	18714	22687	18715	22689
245	925	1508	3702	6032	14807	24126	8	9	10	11	12	13	18137	22250	18127	22250	18128	22252
246	955	1537	3823	6148	15290	24590	8	9	10	11	12	13	17567	21831	17553	21831	17556	21832
247	988	1567	3953	6268	15812	25070	8	9	10	11	12	13	16980	21413	16976	21413	16976	21414
248	1022	1597	4089	6388	16356	25551	8	9	10	11	12	13	16416	21010	16412	21010	16412	21011
249	1058	1628	4236	6512	16943	26047	9	9	11	11	13	13	31714	20610	31685	20610	31686	20611
250	1098	1660	4394	6640	17575	26559	9	9	11	11	13	13	30559	20213	30545	20213	30547	20214
251	1139	1692	4559	6768	18237	27071	9	9	11	11	13	13	29459	19831	29440	19831	29438	19831
252	1184	1725	4737	6900	18949	27599	9	9	11	11	13	13	28339	19451	28333	19451	28332	19452
253	1232	1759	4929	7036	19718	28143	9	9	11	11	13	13	27235	19075	27230	19075	27227	19076

Qindex	IQ_Scale						FQ_Shift						FQ_Quant					
	8b		10b		12b		8b		10b		12b		8b		10b		12b	
	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC	DC	AC
254	1282	1793	5130	7172	20521	28687	9	9	11	11	13	13	26173	18714	26163	18714	26162	18714
255	1336	1828	5347	7312	21387	29247	9	9	11	11	13	13	25115	18355	25101	18355	25102	18356

VP9 SB, CU/PU and TU Sizes - Encoder Only

CU/PU/TU Partitioning Configurations

SB size	CU size	min/max TU range
64x64	64x64	32x32..4x4
	32x32	32x32..4x4
	16x16	16x16..4x4
	8x8	8x8..4x4

PU Options for a Given CU

Current CU size	Possible CU sizes	Allowed CU/PU partition types.
64x64	64x64	2Nx2N, 2NxN, Nx2N
	32x32	2Nx2N, 2NxN, Nx2N
	16x16	2Nx2N, 2NxN, Nx2N
	8x8	2Nx2N, 2NxN, Nx2N, NxN

Definition of the VP9 CU Record Structure - Encoder Only

he following table defines the CU record structure as indirect data to the PAK Object Command. Entries are DW based (4 bytes) and cache aligned. This memory surface is pointed to by the HCP Indirect CU Object Base Address in the HCP_IND_OBJ_BASE_ADDR_STATE Command.

VP9 CU Record Structure Definition (Packed CU)

VP9 Compact CU Packet:

R0.7	31:21	**Reserved MBZ**
	20	**Modified Flag (should not be used by HW)**

		This bit is used by VME to alert kernel of modifications. SRM and FBR can modify the CU Bit Cost and CU Errors.
		0 = No modification to the CU Bit Cost and CU Error.
		1 = Modifications to the CU Bit Cost or CU Error or both.
		Note: HPM will set this to zero.
	19	**Reserved**
	18	**interpred_comp1**
		Interpred comp mode for Part1
		0: Single
		1: Compound(ref_refframe1>intra)
	17	**Reserved**
	16	**interpred_comp0**
		Interpred comp mode for Part0
		0: Single
		1: Compound(ref_refframe1>intra)
	15	**cu_pred_mode1**
		Pred mode for part1 in raster scan order
		cu_pred_mode=intra means ref_refframe0=intra
		0: Intra
		1: Inter
	14	**cu_pred_mode0**
		Pred mode for part0 in raster scan order
		cu_pred_mode=intra means ref_refframe0=intra
		0: Intra
		1: Inter
	13:12	**CU_part_mode**
		0: 2Nx2N,
		1: 2NxN,
		2: Nx2N,
		3: NxN (8x8 only)
	11:8	**intra_chroma_mode[0]**
		0:DC_PRED
		1:V_PRED
		2:H_PRED
		3:TM_PRED
		4:D45_PRED
		5:D135_PRED 6:D117_PRED 7:D153_PRED 8:D207_PRED
		9:D63_PRED

		10-15 Reserved
	7:6	**CU Size**
		0 = 8x8
		1 = 16x16
		2 = 32x32
		3 = 64x64
	5:4	**Reserved**
	3:0	**intra_mode[0]**
		Luma Intra mode for part0
		0:DC_PRED
		1:V_PRED
		2:H_PRED
		3:TM_PRED
		4:D45_PRED
		5:D135_PRED 6:D117_PRED 7:D153_PRED 8:D207_PRED
		9:D63_PRED
R0.6	31:28	**intra_chroma_mode[1]**
		Applicable for part1 of shapes > 8x8
		0:DC_PRED
		1:V_PRED
		2:H_PRED
		3:TM_PRED
		4:D45_PRED
		5:D135_PRED 6:D117_PRED 7:D153_PRED 8:D207_PRED
		9:D63_PRED
		10-15 Reserved
	27:22	**Reserved MBZ**
	21:18	**Reserved MBZ**
		Format: U4
	17:16	**Reserved MBZ**
	15:12	**intra_mode[3][3:0]**
		Luma pred mode for part3
		0:DC_PRED
		1:V_PRED
		2:H_PRED
		3:TM_PRED
		4:D45_PRED

		5:D135_PRED
		6:D117_PRED
		7:D153_PRED
		8:D207_PRED
		9:D63_PRED
	11:10	**Reserved**
	9:6	**intra_mode[2][3:0]**
		Luma pred mode for part2
		0:DC_PRED
		1:V_PRED
		2:H_PRED
		3:TM_PRED
		4:D45_PRED
		5:D135_PRED
		6:D117_PRED
		7:D153_PRED
		8:D207_PRED
		9:D63_PRED
	5:4	**Reserved**
	3:0	**intra_mode[1][3:0]**
		Luma pred mode for part1
		0:DC_PRED
		1:V_PRED
		2:H_PRED
		3:TM_PRED
		4:D45_PRED
		5:D135_PRED
		6:D117_PRED
		7:D153_PRED
		8:D207_PRED
		9:D63_PRED
R0.5	31:30	**Reserved**
	29:28	**MotionComp_flttype[1]**
		0: EIGHTTAP
		1: EIGHTTAP_SMOOTH
		2: EIGHTTAP_SHARP
		3: BILINEAR
		HW will use this filtertype if SWITCHABLE=1 in pic state;

		Used for part1 of blocks > 8x8
	27:26	**Reserved**
	25:24	**MotionComp_flttype[0]**
		0: EIGHTTAP
		1: EIGHTTAP_SMOOTH
		2: EIGHTTAP_SHARP
		3: BILINEAR
		HW will use this filtertype if SWITCHABLE=1 in pic state;
		Used for part0
	23	**Reserved**
	22:20	**SegmentIdx[1]**
		Segment 0 to 7
		SegmentID for part1 of blocks > 8x8
	19	**Reserved**
	18:16	**SegmentIdx[0]**
		Segment 0 to 7
		SegmentID for part0 of blocks > 8x8
	15	**segmentPredFlag1**
		0: Disable
		1: Enable
		Segment Prediction disable for Part1
	14	**segmentPredFlag0**
		0: Disable
		1: Enable
		Segment Prediction disable for Part0
	13:4	**Reserved**
	3:2	**TU_SIZE[1]**
		0 = 4x4
		1 = 8x8
		2 = 16x16
		3 = 32x32
		tu_size[2][1:0], tu_size[3][1:0] must be 0 (4x4)
	1:0	**TU_SIZE[0]**
		0 = 4x4
		1 = 8x8
		2 = 16x16
		3 = 32x32
		tu_size[2][1:0], tu_size[3][1:0] must be 0 (4x4)

R0.4	31:29	**QuantRound1**
		Quantization Round value for Part1 (8x8 and below shapes will have the same round value) In VDEnc mode, this parameter is set to the QuantRound0.
		0:+1/16
		1:+2/16
		2:+3/16
		3:+4/16
		4:+5/16
		5:+6/16(default)
		6:+7/16
		7:+8/16
	28:26	**QuantRound0**
		Quantization Round value for Part0 (8x8 and below shapes will have the same round value) This parameter is equivalent to the RoundingSelect in the HEVC CU packet.
		0:+1/16
		1:+2/16
		2:+3/16
		3:+4/16
		4:+5/16
		5:+6/16(default)
		6:+7/16
		7:+8/16
	25:14	**Reserved**
	13:12	**ref_refframe1[1]**
		frame1(backward)reference frame id for part0
		HW uses if SegmentReferenceEnabled=0 in segment ID command
		0:intra
		1:last
		2:golden
		3:altref
		Format = U2
	11:10	**Reserved**
	9:8	**ref_refframe1[0]**
		frame1(forward)reference frame id for part0
		HW uses if SegmentReferenceEnabled=0 in segment ID command
		0:intra

		1:last
		2:golden
		3:altref
		Format = U2
	7:6	**Reserved**
	5:4	**ref_refframe0[1]**
		frame0(forward)reference frame id for part1
		HW uses if SegmentReferenceEnabled=0 in segment ID command
		0:intra
		1:last
		2:golden
		3:altref
		Format = U2
	3:2	**Reserved**
	1:0	**ref_refframe0[0]**
		frame0(forward)reference frame id for part0
		HW uses if SegmentReferenceEnabled=0 in segment ID command
		0:intra
		1:last
		2:golden
		3:altref
		Format = U2
R0.3	31:16	**mvy_refframe1[1]**
		Frame1(Backward) MVy for part1
		Format = S13.2 (2's comp)
	15:0	**mvx_refframe1[1]**
		Frame1(Backward) MVx for part1
		Format = S13.2 (2's comp)
R0.2	31:16	**mvy_refframe1[0]**
		Frame1(Backward) MVy for part0
		Format = S13.2 (2's comp)
	15:0	**mvx_refframe1[0]**
		Frame1(Backward) MVx for part0
		Format = S13.2 (2's comp)
R0.1	31:16	**mvy_refframe0[1]**
		Frame0(Forward) MVy for part1
		Format = S13.2 (2's comp)
	15:0	**mvx_refframe0[1]**

		Frame0(Forward) MVx for part1
		Format = S13.2 (2's comp)
R0.0	31:16	**mvy_refframe0[0]**
		Frame0(Forward) MVy for part0
		Format = S13.2 (2's comp)
	15:0	**mvx_refframe0[0]**
		Frame0(Forward) MVx for part0
		Format = S13.2 (2's comp)

Quant Scale and Filter Level Table

	Decode mode	Encode BRC first pass, or no BRC	Encode HW multi-pass BRC - Subsequent Passes
Determination of base qindex	N/A	Directly from PIC_STATE, base_qindex	`final_base_qindex = clip(0,255, base_qindex (from PIC_STATE) + accumulated delta from previous passes)`
Determination of inverse quant scale of each block	Directly from SEGMENT_STATE	`final_qindex = clip(0,255, clip(0, 255, base_qindex + // from PIC_STATE segment_q_delta) + // from SEGMENT_STATE chroma or AC delta // from PIC_STATE) iq_lookup_table[final_qindex];`	`final_qindex = clip(0,255, clip(0, 255, final_base_qindex + // from above segment_q_delta) + // from SEGMENT_STATE chroma or AC delta // from PIC_STATE) iq_lookup_table[final_qindex];`
Determination of forward quant scale and shift of each block	N/A	`fq_lookup_table[final_qindex];`	`fq_lookup_table[final_qindex];`
Determination of base filter level	N/A	Directly from PIC_STATE (filter_level) `final_base_filter_level = filter_level (from PIC_STATE)`	`final_base_filter_level = clip(0, 63, filter_level (from PIC_STATE) + accumulated delta from previous passes)`
Determination of final filter level of each block	Directly from SEGMENT_STATE	`mask = final_base_filter_level > 31 ? -2 : -1; final_filter_level = clip(0,63, clip(0,63, base_filter_level + // from PIC_STATE segment_lf_delta) + // from SEGMENT_STATE) (ref_delta & mask) + // from PIC_STATE (mode_delta & mask) // from PIC_STATE );`	`mask = final_base_filter_level > 31 ? -2 : -1; final_filter_level = clip(0,63, clip(0,63, final_base_filter_level + // from above segment_lf_delta) + // from SEGMENT_STATE) (ref_delta & mask) + // from PIC_STATE (mode_delta & mask) // from PIC_STATE );`

When SW writes the uncompressed header for an encoded frame, it must downshift the mode_delta and ref_delta by 1, if the final_base_filter_level > 31.

VP9 Allowed SB Size Encoder Only

The following table details the SB size allowed and the number of records per SB for the encoder.

Allowed SB Size - Encoder Only

VP9 SB Size Allowed	Number of Records per SB
64x64	64

Note: HW will support partial SBs within a frame boundary to a minimum CU8x8 granularity

HCP PAK Data Structure

The following documents HCP PAK Data Structure.

Tile Size and CU Stream-out Records

HEVC: Streamout0 cacheline is composed of 4 quarter cachelines, each containing information on CU skip flag, coding block flag for the TUs in a PU, residual/coefficient bit count for a PU, total bit count for CU, SB exceed limit flag. A typical streamout0 cacheline, therefore, has information on statistics for 4 PUs and Super Block exceed limit flag.

Pak pipeline streamout enable bit, set by HCP_PIPE_MODE_SELECT command, enables or disables the streamout.

Programming Note				
Context:		CU level statistics		
Level	**Field**	**Width**	**Cacheline**	**Comment**
PU	PU Skip Flag	1	qcacheline[0]	Packed in Quarter Cacheline in PU format
SB	SB exceed limit	1	qcacheline[1]	Packed in Quarter Cacheline in PU format (valid on last PU of SB)
	Reserved	14	qcacheline[15:2	Reserved
PU	TU CBF Y/U/V	48	qcacheline[63:16]	Packed in Quarter Cacheline in PU format
PU	PU Coefficient Bit Count (Only residual)	18	qcacheline[81:64]	Packed in Quarter Cacheline in PU format
PU	PU Bit Count (all PU Syntax)	18	qcacheline[113:96]	Packed in Quarter Cacheline in PU format
	Reserved	14	qcacheline[127:114]	Reserved

HEVC Streamout 1: Per Tile Quarter Cacheline

Level	Field	Width	Cacheline	Comment
Tile	Tile Bit Count (header + data + tail)	32	cacheline[31:0]	
	Reserved(MBZ)	32	cacheline[63:32]	
	TilePositionX[15:0]	16	cacheline[79:64]	
	TilePositionY[15:0]	16	cacheline[95:80]	
	Reserved(MBZ)	32	cacheline[127:96]	

VP9: CU statistics record (individual PUs per record down to 8x8 only)

Fields	Bits	
Skip	3:0	Indicates Skip flag Group 4 4x4s -> 4 bits
InterMode	11:4	InterMode: 0 NEARESTMV, 1 NEARMV, 2 ZEROMV, 3NEWMV Group 4 4x4s total 8 bits
Reserved	15:12	
NZ coeff count	28:16	Number of non-zero coeffs; sum of YUV, 13bits
Reserved	31:29	
NumBitsforCoeffs	47:32	Number of Bits for coefficients per block, 16bits
NumBitsforBlock	63:48	Number of Bits in block

Definition of the CU Record Structure- Encoder Only

The following table defines the CU record structure as indirect data to the PAK Object Command. Entries are DW based (4 bytes) and cache aligned. This memory surface is pointed to by the HCP Indirect CU Object Base Address in the HCP_IND_OBJ_BASE_ADDR_STATE Command.

CU Record Structure Definition

Intel restriction max 16 TU per CU, max 256 TUs in a CU.

Definition of the CU Record Structure for VME Interface - Encoder Only

DWord	Bitfield	Definition
R0.7	31	**CU_qp sign** Indicates sign bit for QP. Must be zero for 8bit mode
	30:24	**CU_qp** Note: HPM will set this to zero. This is a pass through for FBR and SRM. Kernel needs to populate this field before calling PAK. Magnitue of CU level QP. Valid range: 0 to 51 for 8bit mode -12 to 51 for 10bit mode -24 to 51 for 12bit mode

DWord	Bitfield	Definition
		QP can change at CU level Restriction: diff_cu_qp_delta_depth must be equal to either 0 or (lcu_size - min_cu_size)
	23	**zero_out_coefficients** 0: Do not force coefficients to zero for entire CU 1: Force coefficients to zero for entire CU *This bit must be zero in VDenc mode*
	23	**zero_out_coefficients_Y** 0: Do not force coefficients to zero in Luma block 1: Force coefficients to zero in Luma block
	22	IPCM_enable If IPCM is enabled, then entire CU is IPCM predicted. Both PU and TU sizes should be same as CU size. Cu_pred_mode is ignored when IPCM is enbled. 1- enable IPCM *0-disable IPCM* Note: Supports 8b only Note: Supports 8, 10 and 12bits. Note: HW ignores this bit for RhoDomain calculations so the statics will be slightly inaccurate.
	21	**Last CU of LCU Flag** Set to 1, if the current CU is the last one inside the current LCU (for VDENC only).
	20	**Modified Flag (should not be used by HW)**
	19:18	**InterPred_IDC_ MV1**
	17:16	**InterPred_IDC_ MV0**
	15	**cu_pred_mode**
	14:12	**CU_part_mode** **Note:** NxN CU_part_mode is used by RPM only in VME. It is used to generate predicted pixels using different prediction modes per 4x4 sub-block. i.e each 4x4 sub-block can have its own prediction mode. **Note:** 2NxN and N2XN intra is only valid for VP9. VP9 supports 32x16, 16x32, 16x8 and 8x16 Intra parititions. Luma Intra Mode indicates the intra prediction mode for 4x4_0. The additional prediction modes are overloaded on R0.6 [23:0] in this case.
	11	**CU_transquant_bypass_flag**
	10:8	**Chroma Intra Mode** 0: DM (use Luma mode, from block 0 if NxN) 1 : Reserved (supposedly to be defined for LM mode) 2: Planar

<table>
<tr><th>DWord</th><th>Bitfield</th><th>Definition</th></tr>
<tr><td></td><td></td><td>3: Vertical
4: Horizontal
5: DC
Note: Also indicates Chroma Intra Mode for block0 for 4:2:2 and 4:4:4.</td></tr>
<tr><td></td><td>7:6</td><td>CU Size</td></tr>
<tr><td></td><td>5:0</td><td>Luma Intra Mode</td></tr>
<tr><td>R0.6</td><td>31:30</td><td>SCC CU Coding Mode (for VDENC only)<table><tr><th>Bit 31:30</th><th>Definition</th></tr><tr><td>00</td><td>Not IBC and Not Palette</td></tr><tr><td>01</td><td>IBC</td></tr><tr><td>10</td><td>Palette</td></tr><tr><td>11</td><td>Illegal</td></tr></table></td></tr>
<tr><td></td><td>29</td><td>Palette Transpose Flag (for VDENC only)
Set to 1, if the palette indices of the current CU are to be transposed.</td></tr>
<tr><td></td><td>28</td><td>Palette Transpose Flag (for VDENC only)
Set to 1 when there is at least one pixel is coded with escape code.</td></tr>
<tr><td></td><td>27:24</td><td>TU Count M1
Note: Intel restriction max 16 TU per CU (however spec allows up to 256 TUs).</td></tr>
<tr><td></td><td>23:22</td><td>Reserved MBZ</td></tr>
<tr><td></td><td>21:18</td><td>Rounding Select
This select is used to pick up the threshold table in the PAK for RhoDomain. It is also used for quantization.This is generated based on CU type, rounding threshold and rounding offset in the VDEnc.
Note: This Rounding Select used after Quantization only when RhoDomain is ON in VDenc Mode.
Format: U4</td></tr>
<tr><td></td><td>17:12</td><td>Luma Intra Mode 4x4_3
Final explicit Luma Intra Mode 4x4_1. Valid values 0..34
Note: CU_part_mode==NxN</td></tr>
<tr><td></td><td>11:6</td><td>Luma Intra Mode 4x4_2
Final explicit Luma Intra Mode 4x4_1. Valid values 0..34
Note: CU_part_mode==NxN</td></tr>
<tr><td></td><td>5:0</td><td>Luma Intra Mode 4x4_1</td></tr>
<tr><td>R0.5</td><td>31:0</td><td>TU Size
Note: HPM will set this to zero. This is a pass through for FBR and SRM. Kernel needs to populate this field before calling PAK.</td></tr>
<tr><td>R0.4</td><td>31:16</td><td>TU_YUV_Transform_Skip</td></tr>
</table>

DWord	Bitfield	Definition
		0: TU transform skip flag is not set (normal transform) 1: TU transform skip flag is set Note: HPM will set this to zero. This is a pass through for FBR and SRM. Kernel needs to populate this field before calling PAK. UV flags are overloaded on Y.
	15:12	**L1_MV1 RefID/0&Chroma Intra Mode1[14:12]** Format = U4/U3 overload **Chroma Intra Mode** if **cu_pred_mode=Intra of** block1 for 4:2:2 and 4:4:4
	11:8	**L1_MV0 RefID[11:8]/0&Chroma Intra Mode2[10:8]** Format = U4/U3 overload **Chroma Intra Mode** if **cu_pred_mode=Intra of** block2 for 4:2:2 and 4:4:4
	7:4	**L0_MV1 RefID[7:4]/0&Chroma Intra Mode3[6:4]** Format = U4/U3 overload **Chroma Intra Mode** if **cu_pred_mode=Intra of** block3 for 4:2:2 and 4:4:4
	3:0	**L0_MV0 RefID**
R0.3	31:16	**L1_MV1.Y**
	15:0	**L1_MV1.X**
R0.2	31:16	**L1_MV0.Y**
	15:0	**L1_MV0.X**
R0.1	31:16	**L0_MV1.Y**
	15:0	**L0_MV1.X**
R0.0	31:16	**L0_MV0.Y** **Programming Note** Intra MV Y for IBC
	15:0	**L0_MV0.X** **Programming Note** Intra MV X for IBC

R0.0 Exists if (CU_Pred_Mode=0, IntraCU)	31:27	**Reserved MBZ**
	26:24	**Chroma Intra Mode second best** DM This is the chroma Intra mode that corresponds to the "Luma Intra Mode second best" that is derived assuming the top or left neighbor is not available. 0: DM (use Luma mode, from block 0 if NxN) 1 : Reserved (supposedly to be defined for LM mode)

		2: Planar 3: Vertical *4: Horizontal* 5: DC
	23:18	**Luma Intra Mode second best 4x4_3** *Valid only when CU_part_mode==NxN.* Valid values 0..34
	17:12	**Luma Intra Mode second best 4x4_2** Valid only when CU_part_mode==NxN.
	11:6	**Luma Intra Mode second best 4x4_1** *Valid only when CU_part_mode==NxN.* Valid values 0..34
	5:0	**Luma Intra Mode second best** When slices are dynamically terminated in the PAK, this Luma Intra Mode is used for the CUs when only the left or only the top neighbor is available. *When both top and left neighbors are not available, PAK defaults to DC mode for those CUs for both Luma and Chroma Intra modes.* Valid values 0..34

Programming the CU Palette Map:

Code Name (Command Type)	Code[1:0] = Data[76:75]	End = Data[74]	Valid[1:0]= Data[73:72]	Payload[...]
REUSED FLAG	00			Data[63:0]
REUSED FLAG	00			Data[63:0]
NEW COLORS	01		Max: 2 Colors per clock. Valid[1:0]=11 or 01 Valid[1:0]=00 means No NEW COLORS at all	Data[71:36], Data[35:0] 36 bits/Pixel with 12 bits/component.
NEW COLORS ...	01	End=1 on the last New Colors		
INDEX	10			Data[29:24], Data[21:16], Data[13:8], Data[5 6 bits/INDEX (Max Palette Table size is 64). 4 INDEXes per clock in Raster order within

				4x1 after 4x1 ... till CU right edge. Escape_Map[3:0]=Data[35:32]	
INDEX ...	10	End=1 on the last Index of CU			
ESCAPE	11		Max: 2 Escapes per clock. Valid[1:0]=11 or 01 Valid[1:0]=00 means No ESCAPE at all.	Data[71:36], Data[35:0] 36 bits/Pixel with 12 bits/component Max number of ESCAPE = 25% of CU Pixels.	
ESCAPE ...	11	End=1 on the last Escape of CU			

HEVC LCU, CU, TU, and PU Sizes - Encoder Only

HEVC LCU/CU Partitioning Configurations

LCU size	min CU size	CU Depth	Hierarchical Depth=CU Depth+1
64x64	64x64	0	1
	32x32	1	2
	16x16	2	3
	8x8	3	4
32x32	32x32	0	1
	16x16	1	2
	8x8	2	3
16x16	16x16	0	1
	8x8	1	2
8x8	8x8	X	Not allowed in spec

HEVC PU Options for a Given CU

Current CU size (leaf node)	min CU sizes (Pic State)	Allowed PU partition types.
64x64 (2Nx2N) Must be a LCU	64x64	Skip : 2Nx2N Intra : 2Nx2N, NxN Inter : 2Nx2N, 2NxN, Nx2N, NxN
	32x32	Skip : 2Nx2N Intra : 2Nx2N (no NxN defined in the spec.) Inter : 2Nx2N, 2NxN, Nx2N, 2NxnU, 2NxnD, nLx2N,

Current CU size (leaf node)	min CU sizes (Pic State)	Allowed PU partition types.
		nRx2N
	16x16	Skip : 2Nx2N Intra : 2Nx2N (no NxN defined in the spec.) Inter : 2Nx2N, 2NxN, Nx2N, 2NxnU, 2NxnD, nLx2N, nRx2N
	8x8	Skip : 2Nx2N Intra : 2Nx2N (no NxN defined in the spec.) Inter : 2Nx2N, 2NxN, Nx2N, 2NxnU, 2NxnD, nLx2N, nRx2N
32x32 (2Nx2N) Can or is not a LCU	32x32	Skip : 2Nx2N Intra : 2Nx2N, NxN Inter : 2Nx2N, 2NxN, Nx2N, NxN
	16x16	Skip : 2Nx2N Intra : 2Nx2N Inter : 2Nx2N, 2NxN, Nx2N, 2NxnU, 2NxnD, nLx2N, nRx2N
	8x8	Skip : 2Nx2N Intra : 2Nx2N Inter : 2Nx2N, 2NxN, Nx2N, 2NxnU, 2NxnD, nLx2N, nRx2N
16x16 (2Nx2N) Can or is not a LCU	16x16	Skip : 2Nx2N Intra : 2Nx2N, NxN Inter : 2Nx2N, 2NxN, Nx2N, NxN
	8x8	Skip : 2Nx2N Intra : 2Nx2N Inter : 2Nx2N, 2NxN, Nx2N, 2NxnU, 2NxnD, nLx2N, nRx2N
8x8 (2Nx2N) Cannot be a LCU	8x8	Skip : 2Nx2N Intra : 2Nx2N and NxN Inter : 2Nx2N, 2NxN, Nx2N (both NxN and AMP are not

Current CU size (leaf node)	min CU sizes (Pic State)	Allowed PU partition types.
		allowed for 8x8 inter CU)

Note: In an 8x8 Inter CU NxN isn't allowed if the SPS parameter disable_inter_4x4 is 1. In Main profile currently this flag is always 1.

U.D, L and R (Up, Down, Left and Right)

n = 1/4 or 3/4 .

HEVC TU Partitioning for a Given CU

CU size	TU size	TU Depth	Max Depth=TU Depth+1	PAK supported TU sizes and corresponding number of TUs in CU
64x64	64x64	0	1	no 64x64 transform, so automatically breakdown into 4 32x32 TUs.
	32x32	1	2	number of TUs in CU = 4
	16x16	2	3	number of TUs in CU = 16
	8x8	3	4	this configuration is currently not supported.
	4x4	4	5	this configuration is currently not supported.
32x32	32x32	0	1	number of TUs in CU = 1
	16x16	1	2	number of TUs in CU = 4
	8x8	2	3	number of TUs in CU = 16
	4x4	3	4	this configuration is currently not supported.
16x16	16x16	0	1	number of TUs in CU = 1
	8x8	1	2	number of TUs in CU = 4
	4x4	2	3	number of TUs in CU = 16
8x8	8x8	0	1	number of TUs in CU = 1
	4x4	1	2	number of TUs in CU = 4

The actual level of partitioning is governed by

- MaxTUSize and MinTUSize in Pic State.
- max_transform_hierarchy_depth_inter <= 2 (intel restriction) DW4 bit 3:2 Pic State
- max_transform_hierarchy_depth_intra <= 2 (intel restriction) DW4 bit 1:0 Pic State

Allowed LCU Size - Encoder Only

The following table details the LCU size allowed and the number of records per LCU for the encoder.

LCU Size Allowed	Fixed Number of Records per LCU
64x64	64
32x32	16
16x16	4

Note: 0.5 CL per CU record in VME mode and 1 CL per CU record in extENC mode.

HEVC/VP9 PAK Frame Statistics

PAK outputs frame level statistics for RhoDomain, SSE, slice size conformance features and LCU statistics. The RhoDomain and Slice Size conformance parameters are exclusive only to HEVC. The SSE and LCU statistics are for both HEVC and VP9.

HEVC Frame Statistics

SliceSizeConformance

DWord	Bit	Description
0	31:17	**Reserved: MBZ**
	16	**Slice Overflow Occured** Format: Enable This field indicates that one or more slices in the current frame exceeded the Target size. When the actual slice size exceeds "Target slice size in Bytes", HW sets "Slice Overflow Occurred" bit in the PAK Frame Statistics.
	15:8	**MaxFrameQP** This parameter indicates the maximum CU QP in the frame. Format: U8 Valid Range is 0-63 for 10bit and 0-51 for 8bit hevc
	7:0	**MinFrameQP** This parameter indicates the minimum CU QP in the frame. Format: U8 Valid Range is 0-63 for 10bit and 0-51 for 8bit hevc
1	31:0	**Max Slice Size in Bytes** Format: U32 This parameter indicates the largest Slice in Bytes in the current frame.
2..3	31:0	**Reserved: MBZ**

VP9 Frame statistics

SSE Statistics

..33	63:48	**Reserved: MBZ**
	47:0	**Frame Luma SSE** MSB word of the Sum square Error statistics for the luma pixels in the current frame. The internal 4x4 subblock SSE is 24-bits. This is accumulated across the frame for all 4x4s and clamped to 48-bits. Format: U48
34..35	63:48	**Reserved: MBZ**
	47:0	**Frame Chroma Cb SSE** Sum square Error statistics for Cb pixels in the current frame. The internal 4x4 subblock SSE is 24-bits. This is accumulated across the frame for all 4x4s and clamped to 48-bits. Format: U48
36..37	63:48	**Reserved: MBZ**
	47:0	**Frame Chroma Cr SSE** Sum square Error statistics for Cr pixels in the current frame. The internal 4x4 subblock SSE is 24-bits. This is accumulated across the frame for all 4x4s and clamped to 48-bits. Format: U48
38	31:16	**Class0 Zone1 4X4 SUBBLKS SSE Count** This parameter indicates the count of the nu4x4 subblkser of macro-blocks in the current frame whose Sum square Error (SSE) met the Class0 Zone1 SSE threshold requirements. The output count is a multiple of 16. The value is internally » 4. Format: U16
	15:0	**Class0 Zone0 4X4 SUBBLKS SSE Count** This parameter indicates the count of the nu4x4 subblkser of macro-blocks in the current frame whose Sum square Error (SSE) met the Class0 Zone0 SSE threshold requirements. The output count is a multiple of 16. The value is internally » 4. Format: U16
39	31:16	**Max Class0 4X4 SUBBLKS SSE** The maximum macro-block sum square error for the Y+U+V pixels for the macro-blocks that were in Class 0.

		This is clamped to 16-bits. The internal 4x4 subblock SSE is 24-bits. It is » 4 before the threshold check for zone classification and Max 4x4 SSE clamping. Format: U16
	15:0	**Class0 Zone2 4X4 SUBBLKS SSE Count** This parameter indicates the count of the nu4x4 subblkser of macro-blocks in the current frame whose Sum square Error (SSE) met the Class0 Zone2 SSE threshold requirements. The output count is a multiple of 16. The value is internally » 4. Format: U16
40-55	31:0 (Each)	**SSE Statistics for Class1-8.** **Class1-8 Zone0 4X4 SUBBLKS SSE Count** **Class1-8 Zone0 4X4 SUBBLKS SSE Count** **Class1-8 Zone0 4X4 SUBBLKS SSE Count** **Max Class1-8 4X4 SUBBLKS SSE** SSE statistics for Class 1-8, see DW SSE Class 0 statistics for format.
56-63	31:0	**Reserved: MBZ**

HEVC Error Concealment

The HCP implements an error concealment policy, which is always enabled and cannot be disabled. The objective is that the HCP will always complete a frame/field workload by either decoding the bit stream normally until it finishes the workload or by concealing blocks until the slice or workload is completed. It should never be allowed to hang.

Error concealment, implemented by the HCP hardware, is configured for each slice in the HCP_BSD_OBJECT command. The following information in the HCP_BSD_OBJECT command is utilized for error concealment.

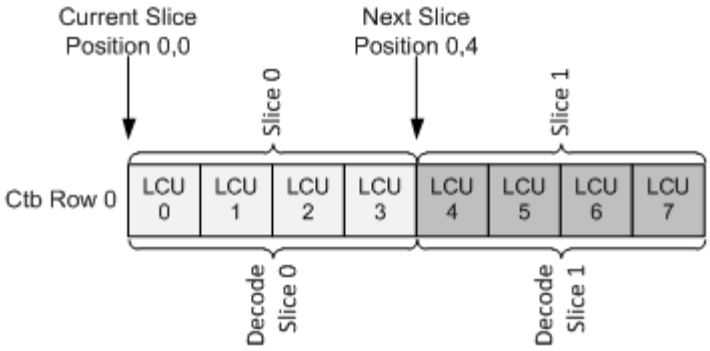
- **SliceStartCtbY, SliceStartCtbX:** The current slice position specified in Ctb coordinates.
- **NextSliceStartCtbY, NextSliceStartCtbX:** The next slice position specified in Ctb coordinates. If the current slice is the last slice in the picture, the next slice values are set to (0,0).
- **LastSliceofPic:** Indicates that the current slice is the last slice in the picture.
- **slice_type:** Indicates the picture type: I, P or B.

The host software will remove all extra slices in the picture. The HCP will not be given a workload that includes extra slices beyond the picture. The last slice in the picture will always be marked by the host software.

The host software will remove any overlapping slices in the picture. The HCP will not be given a workload that includes overlapping slices in the picture.

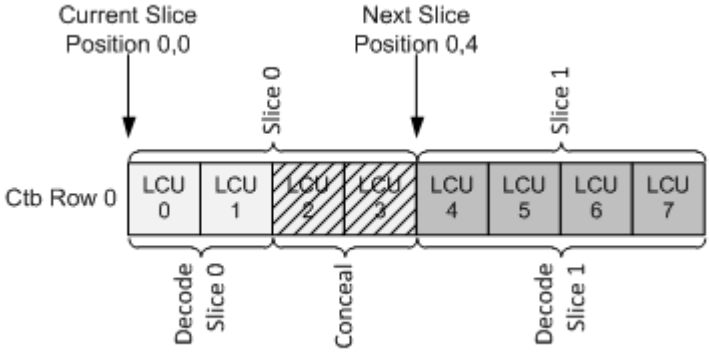
A HCP_BSD_OBJECT command will include the current slice position and the next slice position. For non-errored streams, it is guaranteed that the slice bit stream will be decoded by the HCP starting from the current slice position through to the Ctb (inclusive) adjacent to the Ctb indicated by the next slice position. HEVC Slice Decode for Non-errored Stream Cases illustrates the example of a non-errored stream decode starting with XXX.

HEVC Slice Decode for Non-errored Stream Cases



For error stream cases where the next slice position does not align itself with the last successfully decoded Ctb in the current slice, the HCP will conceal Ctbs from the last decoded Ctb in the current slice through to the last Ctb prior to the Ctb indicated by the next slice position. If the error occurs such that the current decoded Ctb cannot be decoded, the HCP will ensure that the current Ctb is written out by any means before writing out concealed Ctbs for the remaining Ctbs in the current slice. In the case of the last slice in a picture, the HCP will conceal Ctbs from the last decoded Ctb in the current slice through to the last Ctb position in the picture indicated by the resolution of the picture in the HCP_PICT_STATE command. HEVC Slice Decode for Missing Blocks in a Slice illustrates the case described.

HEVC Slice Decode for Missing Blocks in a Slice



Since the host software removes overlapping slices, the next slice position will never be equal to or less than the current slice position.

A concealed Ctb for an I-slice is constructed by the HCP specifying the Intra_Planar prediction mode for the Ctb.

A concealed Ctb for a P-slice is constructed by the HCP specifying the skip_flag.

A concealed Ctb for a B-slice is constructed by the HCP specifying the skip_flag.

HEVC Register Definitions

The Message Channel Interface is a read-only bus used to access the HCP status registers. All registers are 32 bits where reserved bits return a value of zero and subtractive-decode is used to return 0x0000 for all register holes. The Unit ID is 28h. For HCP, the address range is 0x0001E900h to 0001E9FFh.

HCP Encoder Register Read/Write

Register Name	Address	Tile-Based Engine OFF (Reads at FRAME boundary)	Tile-Based Engine ON (Reads at TILE ROW boundary)
HCP_BITSTREAM_BYTECOUNT_FRAME	8'hA0	Frame Byte Count per frame	Accumulated Frame Byte Count until current tile
HCP_BITSTREAM_BYTECOUNT_FRAME_NO_HDR	8'hA4	Frame Byte Count without header per frame	Accumulated Frame Byte Count without header until current tile
HCP_BITSTREAM_SE_BITCOUNT_FRAME	8'hA8	Syntax element bit count per frame	Accumulated Syntax element bit count until current tile
HCP_CABAC_BINCOUNT_FRAME	8'hAC	Bin count per frame	Accumulated Bin count until current tile
HCP_CABAC_INSERTION_COUNT	8'hB0	Cabac Zero Word insertion byte count per frame	Cabac Zero Word insertion byte count per frame. (only available at the last tile of the frame)
HCP_MIN_FRAME_PADDING_COUNT	8'hB4	Min Frame Padding byte count per frame	>Min Frame Padding byte count per frame. (only available at the last tile of the frame)
HCP_IMAGE_STATUS_MASK	8'hB8	Frame level Mask bits (BRC Min, BRC Max, LCU Max mask bits)	>Frame level Mask bits (BRC Min, BRC Max, LCU Max mask bits). Updates every HCP_PIC_STATE i.e. per tile
HCP_IMAGE_STATUS_CONTROL	8'hBC	Frame level Status Control bits. Cumulative Delta Qp, Frame Underflow/Overflow, LCU size exceed flag are reflected per frame level.	>Some bits are updated tile/frame level. Driver updates Cumulative Delta Qp at the start of every tile row. LCU size exceed flag updates every tile. Frame Underflow/Overflow updates only at last tile in a frame.
HCP_QP_STATUS_COUNT0	8'hC0	Min and Max Qp from HFQ per frame	Min and Max Qp per tile. (if read at TILE ROW then last TILE of the row's update will be available)
HCP_QP_STATUS_COUNT1	8'hC4	Cumulative Qp from HFQ per frame	Cumulative Qp per tile. (if read at TILE ROW then last TILE of

Register Name	Address	Tile-Based Engine OFF (Reads at FRAME boundary)	Tile-Based Engine ON (Reads at TILE ROW boundary)
			the row's update will be available)
HCP_SLICE_COUNT	8'hC8	Slice count per frame	Slice count in a frame until current tile
HCP_BITSTREAM_BYTECOUNT_TILE	8'hCC	Tile Byte Count. Since read happens at frame boundary, it will reflect tile size of the last tile in a frame	Current Tile Byte Count per tile. (if read at TILE ROW then last TILE of the row's update will be available)

Register Attributes Description

Host Register Attributes gives the defined register tags and their description.

Host Register Attributes

Tag	Name	Description
R/W	Read/Write	Bit is read and writeable.
R/SW	Read/Special Write	Bit is readable. Write is only allowed once after a reset.
RO	Read Only	Bit is only readable, but writes have no effects.
WO	Write Only	Bit is only writeable, reads return zeros.
RV	Reserved	Bit is reserved and not visible. Reads will return 0, and writes have no effect.
NA	Not Accessible	This bit is not accessible.

HCP Decoder Register Map

This documents all HEVC Decoder MMIO Registers.

HCP Decoder Register Descriptions

The HCP implements the following MMIO registers. A description of the register including its address and DWord descriptions are provided.

Registers
HCP Decode Status
HCP_CABAC_Status
HCP Last Position
HCP PMU Status
HCP Picture Checksum cIdx0
HCP Picture Checksum cIdx1

Registers
HCP Picture Checksum cIdx2

HCP Encoder Register Descriptions

Register
HCP_BIN_CT - HCP Frame BitStream BIN Count
HCP_BITSTREAM_BYTECOUNT_FRAME_NO_HEADER - Reported Bitstream Output Byte Count without header per Frame Register
HCP_BITSTREAMSE_BITCOUNT_FRAME - Reported Bitstream Output Bit Count for Syntax Elements Only
HCP_CABAC_BIN_COUNT_FRAME - Reported Bitstream Output CABAC Bin Count Register
HCP_CABAC_INSERTION_COUNT - Reported Bitstream Output CABAC Insertion Count
HCP_MINSIZE_PADDING_COUNT - Bitstream Output Minimal Size Padding Count Report Register
HCP_IMAGE_STATUS_CONTROL - HCP Image Status Control
HCP_QP_STATUS_COUNT - HCP Qp Status Count
HCP_UNIT_DONE - HCP Unit Done
HCP_IMAGE_STATUS_MASK - HCP Image Status Mask
HCP_SLICE_COUNT
Reported Bitstream Output Byte Count per Tile

Acronyms and Applicable Standards

Acronyms and Abbreviations

The table below defines acronyms and abbreviations used in this document.

Acronyms

Acronym	Meaning
AAC	Advanced Audio Coding -- part of the MPEG specification, AAC is the latest development in audio compression. It provides higher-quality audio reproduction than MPEG-1 Layer 3 (MP3), while requiring nearly 50% less data. It is defined in ISO/IEC 13818-7.
ADSL	Asymmetrical Digital Subscriber Line -- an asymmetrical DSL technology that takes advantage of the one-way nature of most multimedia communication, and provides much faster data rates for downstream (to the subscriber) then the upstream.
API	Application Programming Interface -- a set of routines used by an application program to request and carry out low-level services performed by the operating system.
ARGB	Alpha Red Green Blue -- color channel components.
ARIB	Association of Radio Industries and Business -- designated by the Ministry of Public Management, Home Affairs, Posts and Telecommunications (MPHPT) in Japan. ARIB members include broadcasters, radio equipment manufacturers, telecommunication operators, and related organizations.
ASP	Advanced Simple Profile - MPEG4-2

Acronym	Meaning
ATSC	ATSC Advanced Television Systems Committee - an organization in US that establishes and promotes technical standards for advanced television systems, such as digital television (DTV).
BDU	Bit-stream Data Unit
BIST	Built In Self Test
BPP	Bits Per Pixel
BSD	Byte Stream Decoder
CA, CAM	Conditional Access, Conditional Access Module - the removable descrambling module implemented in digital cable or satellite television system. The data flows through the module, which can have any proprietary scrambling algorithm implemented, yet maintaining system interface compatibility. The CAMs are usually provided by the operators in the TV network.
CPU	Central Processing Unit
DAA	Direct Access Arrangement
DAC	Digital-to-Analog Converter
DDA	Digital Difference Analyzer
DDS	Direct Digital Synthesizer
DPB	Decoded Picture Buffer. This buffer holds the decoded pictures for reference and for output along with the currently decoding picture. This differs from the DPB in the standard, which only holds the decoded pictures for reference.
DVB	Digital Video Broadcasting -- a set of open worldwide standards that define digital broadcasting using existing satellite, cable, and terrestrial infrastructures. It uses MPEG-2 specification as a universal foundation and expands it with DVB data structures and processes DVB-compliant digital broadcasting and equipment is widely available to consumers and is indicated with the DVB logo.
DVB-S	Satellite television DVB standards, based on QPSK and 8-DPSK modulation.
DVB-T	Terrestrial television DVB standards, based on 2k and 8k OFDM modulation.
DVD	Digital Versatile Disc
DVD-R	Recordable DVD. Since different disk formats are currently in use, including DVD-R,DVD+R, they are collectively mentioned as DVD-R in this document
DVI	Digital Visual Interface standard (EIA/CEA-861A). The standard defines a method for sending digital video signals over DVI and OpenLDI interface specifications. The standard is fully backward compatible with earlier DVI standards. New features include carrying auxiliary video information, such as aspect ratio and native video format information.
DVO	Digital Video Output - the parallel, low voltage signaling interface defined for Intel(R) video chipsets
DSL xDSL	Digital Subscriber Line - transmission of data over copper telephone lines capable of bringing high-bandwidth to subscribers. Many flavors of DSL are currently in use, which are collectively called xDSL throughout the document.
DSP	Digital Signal Processor
DST	Destination
DWord	A 32-bit word
ES	Elementary Streams -- the raw output of an encoder, containing only what is necessary for a

Acronym	Meaning
	decoder to approximate the original picture or audio.
FIFO	First in First Out
FIR	Finite Impulse Response
FPU	Floating Point Unit
FW	Firmware running on the decoder controller, as used in Volume 4 of the *Olo River Plus Silicon EAS*
IDR	Instantaneous Decoding Refresh
IEEE 1394 1394	IEEE 1394 or iLink* or FireWire* An IEEE electronics industry standard for connecting multimedia and computing Up to 63 devices can be attached to your PC via a single plug-and-socket connection.
IEEE 802.11 802.11	The Institute for Electronics and Electrical Engineers (IEEE) wireless network specification. 802.11g and 802.11a networks can transmit payload at the rates in excess 34Mbits/s and allow for the wireless transmission at distances from several dozen to several hundred feet indoors.
IF	Intermediate Frequency -- the fixed, relatively low-frequency carrier to which current programs are ported by the tuner.
GMCH	Graphics and Memory Control Hub -- a chip that connects the IA processor to memory and other components in PC.
HDD	Hard Disk Drive -- magnetic mass storage device used in media centers for audiovisual program recording.
HDMI	High Definition Multimedia Interface (HDMI). This interface is used between any audio/video source, such as a set-top box, DVD player, or A/V receiver, and an audio or video monitor, such as a DTV. HDMI supports standard, enhanced or high-definition video, plus multi-channel digital audio on a single cable. The format transmits all ATSC HDTV standards and supports eight-channel digital audio (at up to a 192kHz sampling rate), with bandwidth to spare for future enhancements.
HDTV	High-Definition Television -- HDTV specifically refers to the highest-resolution formats of the 18 total DTV formats, true HDTV is generally considered to be 1,080-line interlaced (1080i) or 720-line progressive (720p).
HSR	Hidden Surface Removal
HW	Hardware
I/F	Interface
IEEE	IEEE 32-bit Floating Point number format representation
ISP	Image Synthesis Processor -- A collective term to describe all components of the hidden surface removal operation within the PowerVR architecture.
LOD	Level Of Detail -- used in texturing calculations.
LSB	Least Significant Bit
LUT	Look-up table
MBAFF	Block Adaptive Field Frame mode
MFD	Multi-Format Decoder
MMU	Memory Management Unit
MMMC	Multi-port, Multi-channel Memory Controller
MSA	Intel Micro Signal Architecture -- microprocessor architecture combining the features of

Acronym	Meaning
	microcontroller and digital signal processor. MSA is used here as a synonym of the processor core used in Olo River Plus
MSB	Most Significant Bit
MPEG	Motion Picture Experts Group - Organization that develops standards for digital video and digital audio compression.
MPR	Inter Prediction Module
NAL	Network Abstraction Layer
NAL unit	Syntax structure in a H.264 stream
NTSC	National Television System Committee, North American 525-line analog broadcast TV standard.
NIM	Network Interface Module - the integrated tuner and digital demodulator in the (satellite) TV systems. The DVB NIMs output MPEG transport stream.
NOP	No operation
OEM	Original Equipment Manufacturer
OGL/OpenGL	Open GL application programming interface
PAL	Phase Alternation Line - TV standard used in Europe. PAL uses 625 lines per frame, a 25 frames per second update rate and YUV color encoding. The number of visible pixels for PAL video is 768 x 576.
PCI	Peripheral Component Interconnect bus, a bi-directional bus defined in PCI 2.x specification
PES	Packetized Elementary Streams -- packetized streams are the ES streams arranged in data packets with PES header starting every packet. The syntax of the ES and PES is defined in MPEG. See definition for ES.
PIP	Picture In Picture display mode
POD	Point of Deployment conditional access module -- the removable conditional access module defined in the OpenCable* specification in US.
PPS	Picture Parameter set
PTS	Presentation time stamp
PVR	Personal Video Recorder, also PDR or personal digital recorder -- an interactive TV-recording device that records programs in digital format and allows users to search for/record shows based on type (for instance all basketball games or all episodes of a particular program). Users can also pause, rewind, stop, or fast-forward live programs with only a small time lag.
PWL	Piece-wise Linear
PXD	Pixel Decoder Module
RF	Radio Frequency - usually, modulated carriers which can be directly received by the tuners of TVs or radio receivers
RISC	Reduced Instruction Set Computer
RHW	Reciprocal Homogenous W -- W is a 3-D coordinate representation like X Y Z
RSB	Row Store Buffer
RTL	Register Transfer Language/Level
SEI	Supplementary Enhancement Information

Acronym	Meaning
SIF	Semaphore Interface Module
SIMD	Single Instruction Multiple Data
SMPTE	Society of Motion Picture and Television Engineers
SOC	System on chip
SP	Simple Profile - MPEG4-2
SPS	Sequence Parameter set
SRC	Source
SDTV	Standard-Definition Television -- a digital television system that is similar to current analog TV standards in picture resolution and aspect ratio. Typical SDTV resolution is 480i or 480p.
STB	Set Top Box -- a device that effectively turns a television set into an interactive Internet device and/or allows the television to receive and decode digital television (DTV) broadcasts.
TA	Tile Accelerator
TS	MPEG-2 Transport Stream -- a sequence of 188-byte packets carrying the multi-program audiovisual data
TSP	Texture Shading Processor -- a collective term to describe all components of the texture, shading and pixel blending operations within the PowerVR architecture.
VCL	Video Coded Layer
VCXO	Voltage Controlled Crystal Oscillator
VGP/ VGP Lite	Vertex Geometry Processor
VLC	Variable length coded. This refers to the collection of coding techniques that are used in VC1, and include CABAC, CAVLC and Exp-Golomb.
VOL	Video Object Layer
VOP	Video Object Plane
WAN	Wide Area Network
WSS	Wide Screen Signaling
XDS	Extended Data Services -- data services sending data in line 21/283 of the analog NTSC TV signal
XSI	Intel(R) XScale(R) System Interconnect
X, Y, Z, W	3-D coordinate representations
YUV	YUV texture format, primarily for video formats

VP9 Register Definitions

This section describes the VP9 Register Definitions as follows:

- Register Attributes Description
- VP9 Register Map
- VP9 Encoder Register Descriptions

Register Attributes Description

Host Register Attributes gives the defined register tags and their description.

Host Register Attributes

Tag	Name	Description
R/W	Read/Write	Bit is read and writeable.
R/SW	Read/Special Write	Bit is readable. Write is only allowed once after a reset.
RO	Read Only	Bit is only readable, but writes have no effects.
WO	Write Only	Bit is only writeable, reads return zeroes.
RV	Reserved	Bit is reserved and not visible. Reads will return 0, and writes have no effect.
NA	Not Accessible	This bit is not accessible.

VP9 Encoder Register Descriptions

HCP_IMAGE_STATUS_MASK - HCP Image Status Mask

HCP_IMAGE_STATUS_CONTROL - HCP Image Status Control

HCP_UNIT_DONE - HCP Unit Done

MFX Pipe

MFC_AVC_PAK_OBJECT Command

PAK Object Inline Data Description

The Inline Data includes all the required MB encoding states, constitute part of the Slice Data syntax elements, MB Header syntax elements and their derivatives. It provides information for the following operations:

1. Forward and Inverse Transform
2. Forward and Inverse Quantization
3. Advanced Rate Control (QRC)
4. MB Parameter Construction (MPC)
5. CABAC/CAVLC encoding
6. Bit stream packing
7. Intra and inter-Prediction decoding loop
8. Internal error handling

These state/parameter values may subject to change on a per-MB basis, and must be provided in each MFC_AVC_PAK_OBJECT command. The values set for these variables are retained internally, until they are reset by hardware Asynchronous Reset or changed by the next MFC_AVC_PAK_OBJECT command.

The inline data has been designed to match the DXVA 2.0, with the exception of the starting byte (DW0:0-7) and the ending dword (DW7:0-31).

The Deblocker Filter Control flags (FilterInternalEdgesFlag, FilterTopMbEdgeFlag and FilterLeftMbEdgesFlag) are generated by H/W, which are depending on MbaffFrameFlag, CurrMbAddr, PicWidthInMbs and disable_deblocking_filter_idc states.

Current MB [x,y] address is not sent, it is assumed that the H/W will keep track of the MB count and current MB position internally.

DWord	Bit	Description
3	31	**ExtendedForm** This field specifies that **LumaIntraMode** and **RefPicSelect** are fully replicated in 4x4 and 8x8 sub-blocks respectively. This non-DXVA form is used for optimal kernel performance.
	30	Reserved: MBZ
	29:24	Reserved
	23	Reserved : MBZ (reserved for future use as ExternalMvBufFlag)
	22:20	MvFormat (Motion Vector Size). This field specifies the size and format of the output motion vectors. This field is reserved (MBZ) when the IntraMbFlag = 1. The valid encodings are: 000 = 0: No motion vector 100 = 8MV: Four 8x8 motion vector pairs 110 = 32MV: 16 4x4 motion vector pairs Others are reserved. (The following encodings are intended for future usages: 001 = 1MV: one 16x16 motion vector 010 = 2MV: One 16x16 motion vector pair 011 = 4MV: Four 8x8 motion vectors 101 = 16MV: 16 4x4 motion vectors 111 = Packed, number of MVs is given by PackedMvNum.)
	19	CbpDcY. This field specifies if the Luma DC sub-block is coded. Setting it to 0 will force PAK to zero out the Luma sub-block. Otherwise, whether the sub-block is coded will be determined by the quantization process. 1 – the 4x4 DC-only Luma sub-block of the Intra16x16 coded MB is present; it is still possible that all DC coefficients are zero.

DWord	Bit	Description
		0 – no 4x4 DC-only Luma sub-block is present; either not in Intra16x16 MB mode or all DC coefficients are zero.
	18	CbpDcU. This field specifies if the Chroma Cb DC sub-block is coded. Setting it to 0 will force PAK to zero out the Luma sub-block. Otherwise, whether the sub-block is coded will be determined by the quantization process. 1 – the 2x2 DC-only Chroma Cb sub-block of all coded MB (any type) is present; it is still possible that all DC coefficients are zero. 0 – no 2x2 DC-only Chroma Cb sub-block is present; all DC coefficients are zero.
	17	CbpDcV. This field specifies if the Chroma Cb DC sub-block is coded. Setting it to 0 will force PAK to zero out the Luma sub-block. Otherwise, whether the sub-block is coded will be determined by the quantization process. 1 – the 2x2 DC-only Chroma Cr sub-block of all coded MB (any type) is present; it is still possible that all DC coefficients are zero. 0 – no 2x2 DC-only Chroma Cr sub-block is present; all DC coefficients are zero.
	16	Reserved: MBZ (reserved for future use as ExternalResidBufFlag for turbo mode)
	15	Transform8x8Flag This field indicates that 8x8 transform is used for the macroblock. When it is set to 0, the current MB uses 4x4 transform. When it is set to 1, the current MB uses 8x8 transform. The transform_size_8x8_flag syntax element, if present in the output bitstream, is the same as this field. However, whether transform_szie_8x8_flag is present or not in the output bitstream depends on several other conditions. This field is only allowed to be set to 1 for two conditions: It must be 1 if IntraMbFlag = INTRA and IntraMbMode = INTRA_8x8 It may be 1 if IntraMbFlag = INTER and there is no sub partition size less than 8x8 Otherwise, this field must be set to 0. 0: 4x4 integer transform 1: 8x8 integer transform
	14	FieldMbFlag This field specifies the field polarity of the current macroblock, as the mb_field_decoding_flag syntax element in AVC spec. This field specifies whether current macroblock is coded as a field or frame macroblock in MBAFF mode. It is exactly the same as FIELD_PIC_FLAG syntax element in non-MBAFF mode.

DWord	Bit	Description
		0 = Frame macroblock 1 = Field macroblock
	13	IntraMbFlag This field specifies whether the current macroblock is an Intra (I) macroblock. I_PCM is considered as Intra MB. For I-picture MB (IntraPicFlag =1), this field must be set to 1. This flag must be set in consistent with the interpretation of MbType (inter or intra modes). 0: INTER (inter macroblock) 1: INTRA (intra macroblock)
	12:8	MbType5Bits This field is encoded to match with the best macroblock mode determined as described in the next section. It follows an unified encoding for inter and intra macroblocks according to AVC Spec.
	7	FieldMbPolarityFlag This field indicates the field polarity of the current macroblock. Within an MbAff frame picture, this field may be different per macroblock and is set to 1 only for the second macroblock in a MbAff pair if FieldMbFlag is set. Otherwise, it is set to 0. Within a field picture, this field is set to 1 if the current picture is the bottom field picture. Otherwise, it is set to 0. It is a constant for the whole field picture. This field is reserved and MBZ for a progressive frame picture. 0 = Current macroblock is a field macroblock from the top field 1 = Current macroblock is a field macroblock from the bottom field Programming Note: Here bits [26:24] (MbAffFieldFlag and FiedlMbPolarityFlag) match with bits [10:8] of the Media Block Read message descriptor, simplifying the programming for message generation, as when MbAffFieldFlag is “1”, kernels need to override the original “frame” surface state set for MBAFF frame picture.
	6	Reserved: MBZ
	5:4	IntraMbMode This field is provided to carry information partially overlapped with MbType. This field is only valid if IntraMbFlag = INTRA, otherwise, it is ignored by hardware..
	3	Reserved: MBZ
	2	SkipMbFlag By setting it to 1, this field forces an inter macroblock to be encoded as a skipped macroblock. It is equivalent to mb_skip_flag in AVS spec, indicating that a macroblock is inferred as a P_Skip (or

DWord	Bit	Description
		B_Skip) in a P Slice (or B Slice). Hardware honors input MVs for motion prediction and forces CBP to zero. By setting it to 0, an inter macroblock will be coded as a normal inter macroblock. The macroblock may still be coded as a skipped macroblock, according to the macroblock type conversion rules described in the later sub sections. This field can only be set to 1 for certain values of MbType. See details later. This field is only valid for an inter macroblock. Hardware ignores this field for an intra macroblock. 0 = not a skipped macroblock 1 = is coded as a skipped macroblock
	1:0	InterMbMode This field is provided to carry redundant information as that encoded in MbType. This field is only valid if IntraMbFlag =0, otherwise, it is ignored by hardware.
4	31:24	Reserved for future MbYCnt expansion.
	23:16	MbYCnt (Vertical Origin). This field specifies the vertical origin of current macroblock in the destination picture in units of macroblocks. Format = U8 in unit of macroblock.
	15:8	Reserved for future MbXCnt expansion.
	7:0	MbXCnt (Horizontal Origin). This field specifies the horizontal origin of current macroblock in the destination picture in units of macroblocks. Format = U8 in unit of macroblock.
5	31:24	Reserved for future CbpAcUV exopansion for 4.2.2. and 4.4.4 For 4.2.2, [23:16] for U(Cb), and [31:24] for C(Cr). For 4.4.4, the field [31:16] is interpreted as CbpAdJ|CbpAcV for 16 sub-blocks.
	31:16	Cbp4x4V (Coded Block Pattern Cr) Only the lower 4 bits [3:0] are valid for 4:2:0. The 4x4 Cr sub-blocks are numbered as blk0 1 bit3 2 blk2 3 bit1 0 The cbpCr bit assignment is cbpCr bit [3 - X] for sub-block_num X. 0 in a bit - indicates the corresponding 4x4 sub-block is not present (because all coefficient values are zero), or force to zero for PAK. 1 in a bit - indicates the corresponding 4x4 sub-block is present (although it is still possible to have all its coefficients be zero - bad coding).

DWord	Bit	Description
		For monochrome, this field is ignored. For 4.2.2, [23:16] for U(Cb), and [31:24] ignored. For 4.4.4, the definition is the same as for luma component: 1bit per 4x4 block.
	23:20	CbpAcV (Coded Block Pattern Cr) Only the lower 4 bits [3:0] are valid for 4:2:0. The 4x4 Cr sub-blocks are numbered as blk0 1 bit3 2 blk2 3 bit1 0 The cbpCr bit assignment is cbpCr bit [3 - X] for sub-block_num X. 0 in a bit - indicates the corresponding 4x4 sub-block is not present (because all coefficient values are zero), or force to zero for PAK. 1 in a bit - indicates the corresponding 4x4 sub-block is present (although it is still possible to have all its coefficients be zero - bad coding). For monochrome, this field is ignored.
	19:16	Cbp4x4U (Coded Block Pattern Cb) Only the lower 4 bits [3:0] are valid for 4:2:0. The 4x4 Cb sub-blocks are numbered as blk0 1 bit3 2 blk2 3 bit1 0 The cbpCb bit assignment is cbpCb bit [3 - X] for sub-block_num X. 0 in a bit - indicates the corresponding 4x4 sub-block is not present (because all coefficient values are zero), or force to zero for PAK. 1 in a bit - indicates the corresponding 4x4 sub-block is present (although it is still possible to have all its coefficients be zero - bad coding). For monochrome, this field is ignored.
	15:0	Cbp4x4Y[bit 15:0] (Coded Block Pattern Y) For 4x4 sub-block (when Transform8x8flag = 0 or in intra16x16) : 16-bit cbp, one bit for each 4x4 Luma sub-block (not including the DC 4x4 Luma block in intra16x16) in a MB. The 4x4 Luma sub-blocks are numbered as blk0 1 4 5 bit15 14 11 10 blk2 3 6 7 bit13 12 9 8 blk8 9 12 13 bit7 6 3 2

DWord	Bit	Description
		blk10 11 14 15 bit5 4 1 0 The cbpY bit assignment is cbpY bit [15 - X] for sub-block_num X. For 8x8 block (when Transform8x8flag = 1) Only the lower 4 bits [3:0] are valid; the remaining upper bits [15:4] are ignored. The 8x8 Luma blocks are numbered as blk0 1 bit3 2 blk2 3 bit1 0 The cbpY bit assignment is cbpY bit [3 - X] for block_num X. 0 in a bit - indicates the corresponding 8x8 block or 4x4 sub-block is not present (because all coefficient values are zero), or force to zero for PAK. 1 in a bit - indicates the corresponding 8x8 block or 4x4 sub-block is present (although it is still possible to have all its coefficients be zero - bad coding).
	15:0	Only the lower 4 bits [3:0] are valid for 4:2:0. The 4x4 Cb sub-blocks are numbered as blk0 1 bit3 2 blk2 3 bit1 0 The cbpCb bit assignment is cbpCb bit [3 - X] for sub-block_num X. 0 in a bit - indicates the corresponding 4x4 sub-block is not present (because all coefficient values are zero), or force to zero for PAK. 1 in a bit - indicates the corresponding 4x4 sub-block is present (although it is still possible to have all its coefficients be zero - bad coding). For monochrome, this field is ignored. For 4.2.2, [7:0] for U(Cb), and [15:8] ignored. For 4.4.4, the definition is the same as for luma component: 1bit per 4x4 block.
6	31:28	Skip8x8Pattern This field indicates whether each of the four 8x8 sub macroblocks is using the predicted MVs and will not be explicitly coded in the bitstream (the sub macroblock will be coded as direct mode). It contains four 1-bit subfields, corresponding to the 4 sub macroblocks in sequential order. The whole macroblock may be actually coded as B_Direct_16x16 or B_Skip, according to the macroblock type conversion rules described in a later sub section. This field is only valid for a B slice. It is ignored by hardware for a P slice. Hardware also ignores this field for an intra macroblock. 0 in a bit – Corresponding MVs are sent in the bitstream 1 in a bit – Corresponding MVs are not sent in the bitstream
	27	EnableCoeffClamp 1 = the magnitude of coefficients of the current MB will be clamped based on the clamping matrix

DWord	Bit	Description
		after quantization 0 = no clamping
	26	LastMbFlag 1 – the current MB is the last MB in the current Slice 0 – the current MB is not the last MB in the current SliceReserved MBZ.
	25	SkipMbConvDisable This is a per-MB level control to enable and disable skip conversion. This field is ORed with SkipConvDisable field. This field is only valid for a P or B slice. It must be zero for other slice types. Rules are provided in Section Macroblock Type Conversion Rules 0 - Enable skip type conversion for the current macroblock 1 - Disable skip type conversion for the current macroblock
	24	Reserved MBZ.
	23:16	Reserved. Ignored by HW, this field will be re-derived internally. (was QpPrimeV. For 8-bit pixel data, QpCr is the same as QpPrimeCr, and it takes on a value in the range of 0 to 51, positive integer.)
	15:8	Reserved. IIgnored by HW, this field will be re-derived internally. (Was QpPrimeU. For 8-bit pixel data, QpCb is the same as QpPrimeCb, and it takes on a value in the range of 0 to 51, positive integer.)
	7:0	QpPrimeY This is the per-MB QP value specified for the current MB. For 8-bit pixel data, QpY is the same as QpPrimeY, and it takes on a value in the range of 0 to 51, positive integer. Note: This value may differ from the actual codes, when HW QRC is on
7 to 9	31:0 Each	For intra macroblocks, definition of these fields are specified in Inline data subfields for an Intra Macroblock. For inter macroblocks, definition of these fields are specified in Inline data subfields for an Inter Macroblock .
10	31:24	MaxSizeInWord PAK should not exceed this budget accumulatively, otherwise it will trickle the PANIC mode.
	23:16	TargetSizeInWord PAK should use this budget accumulatively to decide if it needs to limit the number of non-zero

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td></td><td></td><td>coefficients.</td></tr>
<tr><td></td><td>15:0</td><td>Lambda_Or_RoundingOffset

When TQEnb=1, this 16-bit unsigned value multiplied by 2 is used as a lambda for the rate-distortion cost estimation in Trellis quantization (TQ). If the upper 4 bits are all set to 1 (0xFXXX), TQ is disabled and the regular quantizer is used. Thus, the valid range is 0~0xEFFF.

When TQEnb=0 or the upper 4 bits are all set to 1, the lower 4-bit value indicates the rounding precision (offset) for the regular quantizer

<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>0000b</td><td>RoundInterEnable, RoundInter, RoundIntraEnable, and RoundIntra defined in MFC_AVC_SLICE_STATE are used as rounding precision.</td></tr>
<tr><td>1000b</td><td>+1/16</td></tr>
<tr><td>1001b</td><td>+2/16</td></tr>
<tr><td>1010b</td><td>+3/16</td></tr>
<tr><td>1011b</td><td>+4/16</td></tr>
<tr><td>1100b</td><td>+5/16</td></tr>
<tr><td>1101b</td><td>+6/16</td></tr>
<tr><td>1110b</td><td>+7/16</td></tr>
<tr><td>1111b</td><td>+8/16</td></tr>
</table>
</td></tr>
</table>

The inline data content of Dwords 4 to 6 is defined either for intra prediction or for inter prediction, but not both.

Inline data subfields for an Intra Macroblock

Dword	Bit	Description
7	31:16	LumaIntraMode[1] Specifies the Luma Intra Prediction mode for four 4x4 sub-block of a MB, 4-bit each. See the bit assignment table later in this section.
	15:0	LumaIntraMode[0] Specifies the Luma Intra Prediction mode for four 4x4 sub-block, four 8x8 block or one intra16x16 of a MB. 4-bit per 4x4 sub-block (Transform8x8Flag=0, Mbtype=0 and intraMbFlag=1) or 8x8 block (Transform8x8Flag=1, Mbtype=0, MbFlag=1), since there are 9 intra modes. 4-bit for intra16x16 MB (Transform8x8Flag=0, Mbtype=1 to 24 and intraMbFlag=1), but only the LSBit[1:0] is valid, since there are only 4 intra modes. See the bit assignment table later in this section.
8	31:16	LumaIntraMode[3] Specifies the Luma Intra Prediction mode for four 4x4 sub-block of a MB, 4-bit each. See the bit assignment table later in this section.
	15:0	LumaIntraMode[2] Specifies the Luma Intra Prediction mode for four 4x4 sub-block of a MB, 4-bit each. See the bit assignment later in this section.
9	31:8	Reserved : MBZ (Reserved for encocder turbo mode IntraResidueDataSize, when this is not 0, optional residue data are provided to the PAK; Reserved for decoder)
	7:0	IntraStruct This field contains 6 bits for IntraPredAvailFlags[5:0] and 2 bits for ChromaIntraPredMode. The IntraPredAvailFlags[4:0] (the lower 5 bits) have already included the effect of the constrained_intra_pred_flag. See the diagram later for the definition of neighbor position around the current MB or MB pair (in MBAFF mode). 1 – IntraPredAvailFlagY, indicates the values of samples of neighbor Y can be used in intra prediction for the current MB. 0 – IntraPredAvailFlagY, indicates the values of samples of neighbor Y is not available for intra prediction of the current MB. IntraPredAvailFlag-A and -E can only be different from each other when constrained_intra_pred_flag is equal to 1 and mb_field_decoding_flag is equal to 1 and the value of the mb_field_decoding_flag for the macroblock pair to the left of the current macroblock is equal to 0 (which can only occur

Dword	Bit	Description
		when MbaffFrameFlag is equal to 1). IntraPredAvailFlag-F is used only if it is in MBAFF mode, i.e. MbaffFrameFlag = 1, the current macroblock is of frame type, i.e. MbFieldFag = 0, and the current macroblock type is Intra8x8, i.e. IntraMbFlag = INTRA, IntraMbMode = INTRA_8x8, and Transform8x8Flag = 1. In any other cases IntraPredAvailFlag-A shall be used instead.

Bits	IntraPredAvailFlags Definition
7	IntraPredAvailFlagF – F (Left 8th row (-1,7) neighbor)
6	IntraPredAvailFlagA – A (Left neighbor top half)
5	IntraPredAvailFlagE – E (Left neighbor bottom half)
4	IntraPredAvailFlagB – B (Top neighbor)
3	IntraPredAvailFlagC – C (Top right neighbor)
2	IntraPredAvailFlagD – D (Top left corner neighbor)
1:0	ChromaIntraPredMode – 2 bits to specify 1 of 4 chroma intra prediction modes, see the table in later section.

Inline data subfields for an Inter Macroblock

DWord	Bit	Description
7	31:16	Reserved : MBZ
	15:8	SubMbPredMode (Sub-Macroblock Prediction Mode): If InterMbMode is INTER8x8, this field describes the prediction mode of the sub-partitions in the four 8x8 sub-macroblock. It contains four subfields each with 2-bits, corresponding to the four 8x8 sub-macroblocks in sequential order. This field is derived from sub_mb_type for a BP_8x8 macroblock. This field is derived from MbType for a non-BP_8x8 inter macroblock, and carries redundant information as MbType). If InterMbMode is INTER16x16, INTER16x8 or INTER8x16, this field carries the prediction modes of the sub macroblock (one 16x16, two 16x8 or two 8x16). The unused bits are set to zero. Bits [1:0]: SubMbPredMode[0] Bits [3:2]: SubMbPredMode[1] Bits [5:4]: SubMbPredMode[2] Bits [7:6]: SubMbPredMode[3]
	7:0	SubMbShape (Sub Macroblock Shape) This field describes the sub-block partitioning of each sub macroblocks (four 8x8 blocks). It contains

DWord	Bit	Description
		four subfields each with 2-bits, corresponding to the 4 fixed size 8x8 sub macroblocks in sequential order. This field is provided for MB with sub_mb_type equal to BP_8x8 only (B_8x8 and P_8x8 as defined in DXVA). Otherwise, this field is ignored by hardware Bits [1:0]: SubMbShape[0] – for 8x8 Block 0 Bits [3:2]: SubMbShape[1] – for 8x8 Block 1 Bits [5:4]: SubMbShape[2] – for 8x8 Block 2 Bits [7:6]: SubMbShape[3] – for 8x8 Block 3 Blocks of the MB is numbered as follows : 01 23 Each 2-bit value [1:0] is defined as : 00 – SubMbPartWidth=8, SubMbPartHeight=8 01 – SubMbPartWidth=8, SubMbPartHeight=4 10 – SubMbPartWidth=4, SubMbPartHeight=8 11 – SubMbPartWidth=4, SubMbPartHeight=4
8	31:24	RefPicSelect[0][3] Support up to 4 reference pictures per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List0 Table.
	23:16	RefPicSelect[0][2] Support up to 4 reference pictures per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List0 Table.
	15:8	RefPicSelect[0][1] Support up to 4 reference pictures per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List0 Table.
	7:0	RefPicSelect[0][0] Support up to 4 reference pictures per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List0 Table.
9	31:24	RefPicSelect[1] [3] Support up to 4 reference pictures per L1 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List1 Table.

DWord	Bit	Description
	23:16	RefPicSelect[1][2] Support up to 4 reference pictures per L1 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List1 Table.
	15:8	RefPicSelect[1][1] Support up to 4 reference pictures per L1 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List1 Table.
	7:0	RefPicSelect[1][0] Support up to 4 reference pictures per L1 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List1 Table.

Luma Intra Prediction Modes

Luma Intra Prediction Modes (LumaIntraPredModes) is defined in Definition of LumaIntraPredModes. It is further categorized as Intra16x16PredMode (Definition of Intra16x16PredMode), Intra8x8PredMode (Definition of Intra8x8PredMode) and Intra4x4PredMode (Definition of Intra4x4PredMode), operating on 16x16, 8x8 and 4x4 block sizes, respectively. The Figure illustrates the intra prediction directions geometrically for the Intra4x4 prediction. When a macroblock is subdivided, the intra prediction is performed for the subdivision in a predetermined order. For example, Numbers of Block4x4 in a 16x16 region shows the block order for Intra4x4 prediction. And Numbers of Block4x4 in an 8x8 region or numbers of Block8x8 in a 16x16 region shows the block order of Block8x8 in a 16x16 region or Block4x4 in an 8x8 region.

Definition of LumaIntraPredModes

LumaIntraPredModes [index]		Intra16x16PredMode	Intra8x8PredMode	Intra4x4PredMode
Index	Bit	MbType = [1...24] Transform8x8Flag = 0	MbType = 0 Transform8x8Flag = 1	MbType = 0 Transform8x8Flag = 0
0	15:12	MBZ	Block8x8 3	Block4x4 3 (0_0)
	11:8	MBZ	Block8x8 2	Block4x4 2 (0_1)
	7:4	MBZ	Block8x8 1	Block4x4 1 (0_2)
	3:0	Block16x16	Block8x8 0	Block4x4 0 (0_3)
1	15:12	MBZ	MBZ	Block4x4 7 (1_0)
	11:8	MBZ	MBZ	Block4x4 6 (1_1)
	7:4	MBZ	MBZ	Block4x4 5 (1_2)
	3:0	MBZ	MBZ	Block4x4 4 (1_3)
2	15:12	MBZ	MBZ	Block4x4 11 (2_0)
	11:8	MBZ	MBZ	Block4x4 10 (2_1)

LumaIntraPredModes [index]		Intra16x16PredMode	Intra8x8PredMode	Intra4x4PredMode
Index	Bit	MbType = [1...24] Transform8x8Flag = 0	MbType = 0 Transform8x8Flag = 1	MbType = 0 Transform8x8Flag = 0
	7:4	MBZ	MBZ	Block4x4 9 (2 2)
	3:0	MBZ	MBZ	Block4x4 8 (2_3)
3	15:12	MBZ	MBZ	Block4x4 15 (3_0)
	11:8	MBZ	MBZ	Block4x4 14 (3_1)
	7:4	MBZ	MBZ	Block4x4 13 (3_2)
	3:0	MBZ	MBZ	Block4x4 12 (3_3)

Definition of Intra16x16PredMode

Intra16x16PredMode	Description
0	Intra_16x16_Vertical
1	Intra_16x16_Horizontal
2	Intra_16x16_DC
3	Intra_16x16_Plane
4 – 15	Reserved

Definition of Intra8x8PredMode

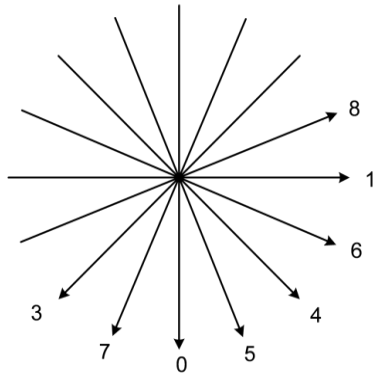
Intra8x8PredMode	Description
0	Intra_8x8_Vertical
1	Intra_8x8_Horizontal
2	Intra_8x8_DC
3	Intra_8x8_Diagonal_Down_Left
4	Intra_8x8_Diagonal_Down_Right
5	Intra_8x8_Vertical_Right
6	Intra_8x8_Horizontal_Down
7	Intra_8x8_Vertical_Left
8	Intra_8x8_Horizontal_Up
9 – 15	Reserved

Definition of Intra4x4PredMode

Intra4x4PredMode	Description
0	Intra_4x4_Vertical
1	Intra_4x4_Horizontal
2	Intra_4x4_DC
3	Intra_4x4_Diagonal_Down_Left
4	Intra_4x4_Diagonal_Down_Right

Intra4x4PredMode	Description
5	Intra_4x4_Vertical_Right
6	Intra_4x4_Horizontal_Down
7	Intra_4x4_Vertical_Left
8	Intra_4x4_Horizontal_Up
9 – 15	Reserved

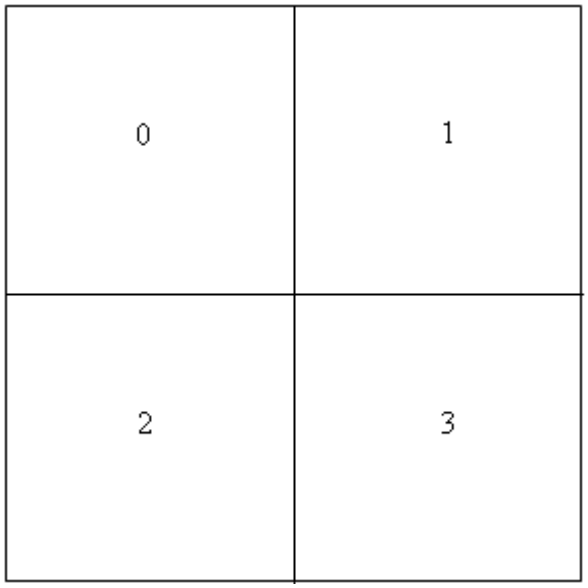
Intra_4x4 prediction mode directions



Numbers of Block4x4 in a 16x16 region

0	1	4	5
2	3	6	7
8	9	12	13
10	11	14	15

Numbers of Block4x4 in an 8x8 region or numbers of Block8x8 in a 16x16 region



Definition of Chroma Intra Prediction Mode

ChromaIntraPredMode (intra_chroma_pred_mode)	Name of intra_chroma_pred_mode
0	Intra_Chroma_DC (prediction mode)
1	Intra_Chroma_Horizontal (prediction mode)
2	Intra_Chroma_Vertical (prediction mode)
3	Intra_Chroma_Plane (prediction mode)

Reference Indices defined for each MB partition type and Bit Assignment

		frame/field MB/Picture			
MB partitioning	16x16	16x8	8x16	8x8	
RefIdxL0/1[0]	blk0	blk0	blk0	blk0	Bit 7:0
RefIdxL0/1[1]	x	blk1	blk1	blk1	Bit 15:8
RefIdxL0/1[2]	x	x	x	blk2	Bit 23:16
RefIdxL0/1[3]	x	x	x	blk3	Bit 31:24

MB Neighbor Availability in Intra-Prediction Modes (IntraPredAvailFlags)

Current MB is labelled as X. For non-MBAFF mode, 4 neighbors, A, B, C, D, are depicted in the following picture and are defined as the following.

- MB D: top left neighbor of current MB X
- MB C: top right neighbor of current MB X
- MB B: top neighbor of current MB X

- MB A: left neighbor of the current MB X

mbAddrD D (top-left)	**mbAddrB** **B** (top)	**mbAddrC** **C** (top-right)
mbAddrA A (left)	**X** **CurrMbAddrX**	**N/A**
N/A	**N/A**	**N/A**

For MBAFF mode, the current MB is labelled as X0 or X1, 4 neighbor pairs, A0/A1, B0/B1, C0/C1, D0/D1, are depicted in the following picture and are defined as the following.

- MB D0: first MB of top left neighbor MB pair of current MB pair X0/X1
- MB D1: second MB of top left neighbor MB pair of current MB pair X0/X1
- MB C0: first MB of top right neighbor MB pair of current MB pair X0/X1
- MB C1: second MB of top right neighbor MB pair of current MB pair X0/X1
- MB B0: first MB of top neighbor MB pair of current MB pari X0/X1
- MB B1: second MB of top neighbor MB pair of current MB pari X0/X1
- MB A0: first MB of left neighbor MB pair of the current MB pair X0/X1
- MB A1: second MB of left neighbor MB pair of the current MB pair X0/X1

mbAddrD D0	mbAddrB B0	mbAddrC C0
mbAddrD+1 D1	mbAddrB+1 B1	mbAddrC+1 C1
mbAddrA A0	CurrMbAddrX X0 or	N/A
mbAddrA+1 A1	CurrMbAddrX X1	N/A

For a given macroblock X (or X0/X1), the 6 neighbor availability signals, namely, A, B, C, D, E, F, are defined as the following.

- IntraPredAvailFlagF – F: (Single neighbor pixel at the left 8th row (-1,7)
- IntraPredAvailFlagA – A (Left neighbor top half pixel group)
- IntraPredAvailFlagE – E (Left neighbor bottom half pixel group)
- IntraPredAvailFlagB – B (Top neighbor pixel group)
- IntraPredAvailFlagC – C (Top right neighbor pixel group)
- IntraPredAvailFlagD – D (Top left corner neighbor pixel)

The following table depicts the generation of IntraPredAvailFlags[5:0] signals in a condensed form. It should note that for most cases only one input neighbor signal is assigned for each condition. The exception is in the four places for deriving left neighbor A and E where the neighbor is only available if left neighbors (A0 and A1) are both available (A0&A1). Also note that F takes output value very similar to that for A except the two "AND" conditions, where F is assigned to A1 instead of (A0&A1).

Definition of intra-prediction neighbor availability calculation in MBAFF mode

Output è		D		B		C		A		E		F		
Current X \ Neighbor Y		Y-Frame	Y-Field	Y-Frame	Y-Field	Y-Frame	Y-Field	Y-Frame	Y-Field	Y-Frame	Y-Field	Y-Frame	Y-Field	
X_0 (Top)	X-Frame	D_1	D_1	B_1	B_1	C_1	C_1	A_0	A_0 & A_1	A_0	A_0 & A_1	A_0	A_1	
	X-Field	D_1	D_0	B_1	B_0	C_1	C_0	A_0	A_0	A_1	A_0	A_0	A_0	
X_1 (Bottom)	X-Frame	A_0	A_1	X_0	N/A	0	0	A_1	A_0 & A_1	A_1	A_0 & A_1	A_1	A_1	
	X-Field	D_1	D_1	B_1	B_1	C_1	C_1	A_0	A_1	A_1	A_1	A_0	A_1	

In Definition of intra-prediction neighbor availability calculation in MBAFF mode, X-Frame or X-Field indicates the frame/field mode of the current MB; and Y-Frame or Y-Field indicates the corresponding neighbor MB for the given neighbor location, being upper left (D) or left (A) for example. Therefore, "Y-" takes the selected neighbor MB name as in the output cell in the same column. For example, for output D, if X1 is a frame MB, Y = A, if X1 is a field MB, Y = D.

For non-MBAFF mode, as A0=A1, B0=B1, C0=C1 and D0=D1, the neighbor assignment is degenerated into the following simple table. Here, E is assigned to the same as A and F is forced to 0.

Definition of intra-prediction neighbor availability calculation in non-MBAFF mode

Output è	D	B	C	A	E	F
X	D0	B0	C0	A0	A0	0

To further explain the neighbor assignment rules in Definition of intra-prediction neighbor availability calculation in MBAFF mode, the following table provides description for each condition. Please note that

this table is informative as it provides redundant information as in Definition of intra-prediction neighbor availability calculation in MBAFF mode.

Detailed explanation of intra-prediction neighbor availability calculation in MBAFF mode

Current MB	Current MB Field	Neighbor MB Field	Neighbor MB Select (Y=?)	Neighbor Avail Result (OUTPUT)	Description
				D	
X0	X-Frame	Y-Frame	D	D1	Top Frame MB uses [-1,-1] = D_31, thus D1 only, regardless D frame or field pair
(Top)	X-Frame	Y-Field	D	D1	
	X-Field	Y-Frame	D	D1	Top Field MB uses [-1,-2] = D_30, thus if D is frame pair, takes D1 (D1_14 pixel), and if D is field pair, takes D0 (D0_15 pixel)
	X-Field	Y-Field	D	D0	
X1	X-Frame	Y-Frame	A	A0	Bottom Frame MB uses [-1,15] = A_15, thus A0 (A0_15 pixel) if A is a frame pair, or A1 (A1_7 pixel), if A is a field pair
(Bottom)	X-Frame	Y-Field	A	A1	
	X-Field	Y-Frame	D	D1	Bottom Field MB uses [-1,-1] = D_31, thus D1 only, regardless D frame or field pair
	X-Field	Y-Field	D	D1	
				B	
X0	X-Frame	Y-Frame	B	B1	Top Frame MB uses [0…15,-1] = B_31, thus B1 only, regardless B frame or field pair
(Top)	X-Frame	Y-Field	B	B1	
	X-Field	Y-Frame	B	B1	Top Field MB uses [0…15,-2] = B_30, thus if B is frame pair, takes B1 (B1_14 row), and if B is field pair, takes B0 (B0_15 row)
	X-Field	Y-Field	B	B0	
X1	X-Frame	Y-Frame	X	X0	Bottom Frame MB uses [0…15,15], thus X0 (X0_15 row)
(Bottom)	X-Frame	Y-Field	X	n/a	Note: X0 and X1 must have the same field type, this row is n/a.
	X-Field	Y-Frame	B	B1	Bottom Field MB uses [0…15,-1] = B_31, thus B1 only, regardless B frame or field pair
	X-Field	Y-Field	B	B1	
				C	
X0	X-Frame	Y-Frame	C	C1	Top Frame MB uses [16…23,-1] = C_31, thus C1 only, regardless C frame or field pair
(Top)	X-Frame	Y-Field	C	C1	
	X-Field	Y-Frame	C	C1	Top Field MB uses [16…23,-2] = C_30, thus if C is frame pair, takes C1 (C1_14 row), and if C is field pair, takes C0 (C0_15 row)
	X-Field	Y-Field	C	C0	
X1	X-Frame	Y-Frame	n/a	0	Bottom Frame MB doesn't have left-top

Current MB	Current MB Field	Neighbor MB Field	Neighbor MB Select (Y=?)	Neighbor Avail Result (OUTPUT)	Description
				D	
(Bottom)	X-Frame	Y-Field	n/a	0	neighbor by definition, thus forced to 0
	X-Field	Y-Frame	C	C1	Bottom Field MB uses [16…23,-1] = C_31, thus C1 only, regardless C frame or field pair
	X-Field	Y-Field	C	C1	
				A	
X0	X-Frame	Y-Frame	A	A0	First Half of Top Frame MB uses [-1,0…7], thus A0 if A is a frame pair; but is only avail if both A0 and A1 are avail if A is a field pair due to the mix
(Top)	X-Frame	Y-Field	A	A0&A1	
	X-Field	Y-Frame	A	A0	First Half of Top Field MB uses [-1,0..2..4..14], thus take A0 (if A is frame pair, takes A0 even lines, and if A is field pair, takes A0 first half)
	X-Field	Y-Field	A	A0	
X1	X-Frame	Y-Frame	A	A1	First Half of Bottom Frame MB uses [-1,16…23], thus A1 if A is a frame pair; but is only avail if both A0 and A1 are avail if A is a field pair due to the mix
(Bottom)	X-Frame	Y-Field	A	A0&A1	
	X-Field	Y-Frame	A	A0	First Half of Bottom Field MB uses [-1,1..3..15], thus take A0 (if A is frame pair, takes A0 odd lines, and if A is field pair, takes A1 first half)
	X-Field	Y-Field	A	A1	
				E	
X0	X-Frame	Y-Frame	A	A0	Second Half of Top Frame MB uses [-1,8…15], thus A0 if A is a frame pair; but is only avail if both A0 and A1 are avail if A is a field pair due to the mix
(Top)	X-Frame	Y-Field	A	A0&A1	
	X-Field	Y-Frame	A	A1	Second Half of Top Field MB uses [-1,16..18..30], thus take A1 (if A is frame pair, takes A1 even lines, and if A is field pair, takes A0 second half)
	X-Field	Y-Field	A	A0	
X1	X-Frame	Y-Frame	A	A1	Second Half of Bottom Frame MB uses [-1,24…31], thus A1 if A is a frame pair; but is only avail if both A0 and A1 are avail if A is a field pair due to the mix
(Bottom)	X-Frame	Y-Field	A	A0&A1	
	X-Field	Y-Frame	A	A1	Second Half of Bottom Field MB uses [-1,17..19..31], thus takes A1 (if A is frame pair, takes A1 odd lines, and if A is field pair, takes A1 second half)
	X-Field	Y-Field	A	A1	
				F	
X0	X-Frame	Y-Frame	A	A0	Top Frame MB uses [-1,7] = A_7 (odd location), thus A0 if A is frame pair and A1 if field pair
(Top)	X-Frame	Y-Field	A	A1	

Current MB	Current MB Field	Neighbor MB Field	Neighbor MB Select (Y=?)	Neighbor Avail Result (OUTPUT)	Description
				D	
	X-Field	Y-Frame	A	A0	Top Field MB uses [-1,14] = A_14 (even location), thus A0 regardless A frame or field pair
	X-Field	Y-Field	A	A0	
X1	X-Frame	Y-Frame	A	A1	Bottom Frame MB uses [-1,23] = A_23 (odd location), thus A1 regardless A frame or field pair
(Bottom)	X-Frame	Y-Field	A	A1	
	X-Field	Y-Frame	A	A0	Bottom Field MB uses [-1,15] = A_15 (odd location), thus A0 if A is frame pair and A1 if A is field pair
	X-Field	Y-Field	A	A1	

Macroblock Type for Intra Cases

MbType follows two different tables according to whether the macroblock is an inter or intra macroblock according to IntraMbFlag.

For an intra macroblock, MbType, as defined in MbType definition for Intra Macroblock, carries redundant information as IntraMbMode. The notation I_16x16_x_y_z used in the table, 'x' is Intra16x16LumaPredMode, 'y' is ChromaCbpInd, and 'z' is LumaCbpInd, as defined in Sub field definition used by MbType for a macroblock with Intra16x16 prediction.

MbType definition for Intra Macroblock

Macroblock Type	MbType
I_4x4	0
I_8x8	0
I_16x16_0_0_0	1
I_16x16_1_0_0	2
I_16x16_2_0_0	3
I_16x16_3_0_0	4
I_16x16_0_1_0	5
I_16x16_1_1_0	6
I_16x16_2_1_0	7
I_16x16_3_1_0	8
I_16x16_0_2_0	9
I_16x16_1_2_0	Ah
I_16x16_2_2_0	Bh
I_16x16_3_2_0	Ch
I_16x16_0_0_1	Dh
I_16x16_1_0_1	Eh

I_16x16_2_0_1	Fh
I_16x16_3_0_1	10h
I_16x16_0_1_1	11h
I_16x16_1_1_1	12h
I_16x16_2_1_1	13h
I_16x16_3_1_1	14h
I_16x16_0_2_1	15h
I_16x16_1_2_1	16h
I_16x16_2_2_1	17h
I_16x16_3_2_1	18h
I_PCM	19h (used by HW)

Note: MbType here is identical as specified in DXVA 2.0.

For Intra_16x16 modes, the 5 bits of value (MbType – 1) have the following meanings.

Sub field definition used by MbType for a macroblock with Intra16x16 prediction

Bits	Description
4	LumaCbpInd – Luma Coded Block Pattern Indicator 0 means none of the luma blocks are coded. 1 means that at least one luma block is coded. 0 = SUBMODE_I16_L_0 1 = SUBMODE_I16_L_NZ In VME output, this field is forced to be 1 before adding 1 in Intra_16x16 mode.
3:2	ChromaCbpInd – Chroma Coded Block Pattern Indicator 00 means none of chroma blocks are coded. 01 means that only the chroma DC block is coded, but all AC blocks are not coded. 10 means that at least one AC chroma block is coded. 00 = SUBMODE_I16_C_0 01 = SUBMODE_I16_C_DC 10 = SUBMODE_I16_C_NZ 11 = Reserved In VME output, this field is forced to be 10 before adding 1 in Intra_16x16 mode. Programming Note: Adding 1 to MbType by VME hardware may have carry into this field. But as '11' is reserved, the carry-in doesn't propagate into bit 4 or higher. This allows software to update MbType, if desired, using the redundant LumaIntraPredModes information.
1:0	Intra16x16PredMode – Intra16x16 Prediction Mode These two bits carries redundant (identical) information as that in LumaIntraPredModes[0][0]. 0 = SUBMODE_I16_VER

Bits	Description
	1 = SUBMODE_I16_HOR 2 = SUBMODE_I16_DC 3 = SUBMODE_I16_PLANE

IntraMbMode definition

IntraMbMode [1:0]	Description	Supported by VME?	Used by PAK?
0	INTRA_16x16 (redundant with MbType)	Yes	Ignored
1	INTRA_8x8	Yes	Yes
2	INTRA_4x4	Yes	Yes
3	IPCM (redundant with MbType)	No	Ignored

As an alternative representation, MbType is logically the same as the following, except the I_PCM and I_NxN (i.e. I_4x4 and I_8x8) cases:

- 24 types of 16x16 intra modes: A+B+C+D:(1h – 18h)

MBTYPE_INTRA_16x16 1hA

- 4 Intra16x16 modes:

SUBMODE_I16_VER 0B

SUBMODE_I16_HOR 1B

SUBMODE_I16_DC 2B

SUBMODE_I16_PLN 3B

- 3 Chroma Cbp indices:

SUBMODE_I16_C_0 0C

SUBMODE_I16_C_DC 4C

SUBMODE_I16_C_NZ 8C

- 2 Luma Cbp indices:

SUBMODE_I16_L_0 0D

SUBMODE_I16_L_NZ ChD

Macroblock Type for Inter Cases

Sub-Macroblock Prediction Mode, SubMbPredMode, indicates the prediction mode for the sub-partitions. Prediction mode specifies prediction direction being forward (from L0), backward (from L1) or bi-directional (from both L0 and L1). Its meaning depends on InterMbMode. Definition of SubMbPredMode[i] provides the definition of the field.

- If InterMbMode is INTER16x16, only SubMbPredMode[0] is valid, it describes the prediction mode of the 16x16 macroblock. The other entries are set to zero by hardware.
- For AVC, SubMbPredMode[0] contains redundant information as encoded in MbType parameter.
- Note: SubMbPredMode[1]-[3] are intentionally set to zero to allow a simple LUT to derive MbType as described later.
- If InterMbMode is INTER16x8, and INTER8x16, only the first two entries SubMbPredMode[0] and SubMbPredMode[1] are valid, describing the sub-macroblock prediction mode.
- For AVC, SubMbPredMode[0]/[1] contains redundant information as encoded in MbType parameter.
- Note: SubMbPredMode[2]-[3] are intentionally set to zero to allow a simple LUT to derive MbType as described later.
- If InterMbMode is INTER8x8, each entry of SubMbPredMode describes the prediction mode of the sub-partition of an 8x8 sub-macroblock.
- For AVC, SubMbPredMode can be derived from sub_mb_type field for BP_8x8 macroblocks as defined in AVC spec.
- Note on Direct Sub-macroblock Prediction Mode: Direct prediction is not conveyed through SubMbPredMode, instead, it is carried through Direct8x8Pattern.

InterMbMode definition

MbSkipFlag	InterMbMode	Description
0	0	INTER16x16
0	1	INTER16x8
0	2	INTER8x16
0	3	INTER8x8
1	0	PSKIP/BSKIP16x16*
1	3	BSKIP
1	1, 2	Reserved
Used by PAK	Ignored by PAK	

* BSKIP16x16 is an optional non-standard but equivalent optimization.

Definition of SubMbPredMode based on InterMbMode

SubMbPredMode	INTER16x16	INTER16x8	INTER8x16	INTER8x8
Bit	MbType = [1...3]	MbType = [16h]	MbType = [4...15h]	MbType = [16h]
7:6	MBZ	MBZ	MBZ	Block8x8 3
5:4	MBZ	MBZ	MBZ	Block8x8 2
3:2	MBZ	Block16x8 1	Block8x16 1	Block8x8 1
1:0	Block16x16	Block16x8 0	Block8x16 0	Block8x8 0
	Ignored by PAK	Ignored by PAK	Ignored by PAK	Used by PAK

Definition of SubMbPredMode[i]

SubMbPredMode	Description	InterMbMode	VME Output	MvCountPred	Notes
0	Pred_L0	All	Yes	1	P or B Slice
1	Pred_L1	All	Yes	1	B Slice Only
2	BiPred	All	Yes	2	B Slice Only
3	Reserved	Reserved	Reserved	Reserved	Reserved

Sub-Macroblock Shape, SubMbShape[i], for i = 0…3, describes the shape of the sub partitions of the 8x8 sub-macroblock of a BP_8x8 macroblock. This field is only valid if InterMBMode is INTER8x8. They are defined in Definition of SubMbShape for an 8x8 region of a BP_8x8 macroblock (including BSKIP, BDIRECT). The parameters can be derived from sub_mb_type field as defined in AVC spec.

Note: These fields must be correctly set even for Direct or Skip 8x8 cases, the individual B_Direct_8x8 block is flagged by the Direct8x8Pattern variable.

Definition of SubMbShape for an 8x8 region of a BP_8x8 macroblock (including BSKIP, BDIRECT)

	Description			
SubMbShape	NumSubMbPart	SubMbPartWidth	SubMbPartHeight	MvCountShape
0	1	8	8	1
1	2	8	4	2
2	2	4	8	2
3	4	4	4	4

For an inter macroblock, MbType, carries redundant information as InterMbMode and SubMbPredMode. The next table provides the typical inter macroblock types and the following table Additional MbType definition with Direct/Skip for Inter Macroblock provides that with skip and direct modes. The definition of MbType for both P slice and B slice is the same and is equivalent to that for mb_type of a B slice in the AVC spec. As direct mode is indicated using a separate field Direct8x8Pattern, 0 is reserved for MbType.

Here, MVCount is the number of motion vectors actually encoded in the bitstream. It is informative. For a BP_8x8 or equivalent Skip/Direct macroblock, MVCount is the sum of the following term for the four 8x8 sub macroblock (with i = 0…3):

MvCountShape[i] * MvCountPred[i] * MvCountDirect[i]

where MvCountShape[i] is block count for sub macroblock [i], MvCountPred[i] is the motion vector count for each block of sub macroblock[i], and MvCountDirect[i] is the multiplier for direct mode for B Slice, indicating whether motion vectors are coded or not. It must be set to 1 for P slice. For B Slice, MvCountDirect[i] = !Direct8x8Pattern[i], which is 0 for a sub macroblock coded as direct mode and 1 otherwise.

In the tables, “DC” stands for “Don’t Care” as PAK hardware ignores these fields.

MbType definition for Inter Macroblock (and MbSkipflag = 0)

Macroblock Type	MbType	MbSkipFlag	Direct8x8Pattern	SubMbShape	SubMbPredMode	MVCount
Reserved	0	-	-	-	-	-
BP_L0_16x16	1	0	0	DC	DC	1
B_L1_16x16	2	0	0	DC	DC	1
B_Bi_16x16	3	0	0	DC	DC	2
BP_L0_L0_16x8	4	0	0	DC	DC	2
BP_L0_L0_8x16	5	0	0	DC	DC	2
B_L1_L1_16x8	6	0	0	DC	DC	2
B_L1_L1_8x16	7	0	0	DC	DC	2
B_L0_L1_16x8	8	0	0	DC	DC	2
B_L0_L1_8x16	9	0	0	DC	DC	2
B_L1_L0_16x8	0Ah	0	0	DC	DC	2
B_L1_L0_8x16	0Bh	0	0	DC	DC	2
B_L0_Bi_16x8	0Ch	0	0	DC	DC	3
B_L0_Bi_8x16	0Dh	0	0	DC	DC	3
B_L1_Bi_16x8	0Eh	0	0	DC	DC	3
B_L1_Bi_8x16	0Fh	0	0	DC	DC	3
B_Bi_L0_16x8	10h	0	0	DC	DC	3
B_Bi_L0_8x16	11h	0	0	DC	DC	3
B_Bi_L1_16x8	12h	0	0	DC	DC	3
B_Bi_L1_8x16	13h	0	0	DC	DC	3
B_Bi_Bi_16x8	14h	0	0	DC	DC	4
B_Bi_Bi_8x16	15h	0	0	DC	DC	4
BP_8x8	16h	0	!= Fh	vary	vary	Sum
Reserved	17h-1Fh	-	-	-	-	-

Additional MbType definition with Direct/Skip for Inter Macroblock

Macroblock Type	MbType	Xfrm 8x8	MbSkipFlag	Direct8x8Pattern	SubMbShape	SubMbPredMode	MvCount	Notes
P_Skip_16x16	1	-	1	DC	DC	DC	0	Skipped macroblock. Motion compensation like P_L0_16x16
B_Skip_16x16_4MVPair	16h	Vary	1	Fh	0	vary	0	Skipped macroblock. Motion compensation like B_8x8 with 8x8 subblocks, when direct_8x8_inference_flag is set to 1
B_Skip_16x16_16MVPair	16h	0	1	Fh	FFh	vary	0	Skipped macroblock. Motion compensation like B_8x8 with 4x4 subblocks, when direct_8x8_inference_flag is set to 0
B_Direct_16x16_4MVPair	16h	vary	0	Fh	0	vary	0	MbType coded as B_Direct_16x16. Motion compensation like B_8x8 with 8x8 subblocks, when direct_8x8_inference_flag is set to 1
B_Direct_16x16_16MVPair	16h	0	0	Fh	FFh	vary	0	MbType coded as B_Direct_16x16. Motion compensation like B_8x8 with 4x4 subblocks, when direct_8x8_inference_flag is set to 0

People might notice that B_DIRECT_16x16 and B_SKIP are mapped on BP_8x8 for AVC decoding interface in IT mode as the motion compensation operation for both modes are the same as BP_8x8. According to AVC Spec, motion vectors for B_DIRECT_16x16 and B_SKIP are derived from temporally co-located macroblock on an 8x8 sub macroblock basis if direct_8x8_inference_flag is set to 1 or on a 4x4 block basis if it is set to 0. For each sub macroblock or block, SubMbPredMode is derived, thus can any of the valid numbers. Motion vectors may also be different. In spatial direct mode, the motion vectors are subject to spatial neighbor macroblocks as well as co-located macroblock. The spatial prediction is based on the neighbor macroblocks, so the same spatial predicted motion vector applies to all sub macroblocks or blocks. However, under certain conditions, temporal predictor may replace (colZeroFlag) the spatial predictor for a given sub macroblock or block. Thus the motion vectors may differ.

In MbType definition for Inter Macroblock (and MbSkipflag = 0), the macroblock type names for major partitions nicely follow forms BP_MbPredMode_MbShape (like BP_L0_16x16) and B_MbPredMode0_MbPredMode1_MbShape (like B_L0_Bi_16x8). For minor partitions it is fixed as BP_MbShape as BP_8x8.

However, in Additional MbType definition with Direct/Skip for Inter Macroblock the macroblock types for Skip and Direct modes does not follow the same rule. The third field in P_Skip_16x16 or B_Direct_16x16_x indicates that “Skip” or “Direct” applies to the entire 16x16 macroblock, even though MbShape is 8x8 as

that in BP_8x8. In order to distinguish the SubMbShape being 8x8 or 4x4 for B_Skip and B_Direct, the fourth field is added. 4MVPair indicates up to 4 MV pairs are presented with SubMbShape equals to 0; and 16MVPair indicates up to 16 MV pairs are presented with SubMbShape equals to FFh. Also note that P_8x8ref0 is not specified in PAK input interface, it is up to hardware to detect and choose its packing format based on number of reference indices and reference index for the given macroblock.

Macroblock Type Conversion Rules

For improved coding efficiency the PAK hardware has the capability to convert macroblock types to use more efficiency coding modes such as DIRECT and SKIP. For an inter macroblock or a sub macroblock coded as DIRECT, no motion vector is needed in the bitstream for the macroblock or sub macroblock. If a macroblock is coded as SKIP, it only consumes one SKIP bit (no motion vector, no coefficients are coded). And infomaton about the macroblock is 'inferred' according to the rules stated in the AVC Spec.

As the input to PAK, the following signals can convey the information regarding DIRECT and SKIP:

- MbSkipFlag
- Direct8x8Pattern
- CodecBlockPattern (CbpY, CbpCb, CbpCr)

Such conversion can be enabled or disabled through the SLICE_STATE fields DirectConvDisable and SkipConvDisable as well as the in line command field MbSkipConvDisable.

A P slice doesn't support direct mode, it only supports P_Skip, which is equivalent to a 16_16_L0 prediction. Other prediction types cannot be converted to P_Skip. The following table describes the macroblock type conversion rules for a P slice. Here CBP = CbpY/CbpCb/CbpCr are the final computed results after quantization by the hardware. Note that hardware honors the input CbpY/CbpCb/CbpCr fields – if the value corresponding to a block is set to zero, the resulting CBP is also zero. The output mb_skip_flag and mb_type are the symbols coded in the bitstream as defined in the AVC spec. "DC" stands for "Don't care", "T" for "True".

Note that the internal condition of MV==MVP is subject to the precise rules stated in the AVC Spec as quoted below. Note that there are exceptions for P_Skip from the normal motion vector prediction rules.

Derivation process for luma motion vectors for skipped macroblocks in P and SP slices

This process is invoked when mb_type is equal to P_Skip.

Outputs of this process are the motion vector mvL0 and the reference index refIdxL0.

The reference index refIdxL0 for a skipped macroblock is derived as follows.

refIdxL0 = 0. (8-168)

For the derivation of the motion vector mvL0 of a P_Skip macroblock type, the following applies.

– The process specified in subclause 8.4.1.3.2 is invoked with mbPartIdx set equal to 0, subMbPartIdx set equal to 0, currSubMbType set equal to "na", and listSuffixFlag set equal to 0 as input and the output is assigned to mbAddrA, mbAddrB, mvL0A, mvL0B, refIdxL0A, and refIdxL0B.

– The variable mvL0 is specified as follows.

– If any of the following conditions are true, both components of the motion vector mvL0 are set equal to 0.

– mbAddrA is not available

– mbAddrB is not available

– refIdxL0A is equal to 0 and both components of mvL0A are equal to 0

– refIdxL0B is equal to 0 and both components of mvL0B are equal to 0

– Otherwise, the derivation process for luma motion vector prediction as specified in subclause 8.4.1.3 is invoked with mbPartIdx = 0, subMbPartIdx = 0, refIdxL0, and currSubMbType = "na" as inputs and the output is assigned to mvL0.

NOTE – The output is directly assigned to mvL0, since the predictor is equal to the actual motion vector.

Macroblock type conversion rule for an inter macroblock in a P slice

Input		Internal			Output		Notes
Macroblock Type	SkipConvDisable || SkipConvDisable	CBP	MV == MVP	MbAffSkipAllowed	mb_skip_flag	mb_type	
P_Skip_16x16	DC	DC	DC	1	1	-	Forced to P_Skip; Hardware will force CBP to zero and also ignore SkipConvDisable control. Hardware doesn't check for MV==MVP error condition
P_Skip_16x16	DC	DC	DC	0	0	0	Reverse convert to P_L0_16x16; Hardware will force CBP to zero but reversely convert MbType as P_L0_16x16 once it determines that Skip is not allowed.
BP _16x16_L0	0	0	T	1	1	-	Converted to P_Skip. Even input doesn't provide skip hint, hardware can performance the optimization by detecting CBP and MV==MVP condition.
BP _16x16_L0	0	0	T	0	0	0	Reverse back to P_L0_16x16; Hardware will reverse back to P_L0_16x16 even Skip conditions are met once it determines that Skip is not allowed.
BP _16x16_L0	1	0	T	T	0	0	Still coded as P_L0_16x16 = 0.

A B slice supports both direct and skip modes. The following table describes the macroblock type conversion rules for a B slice. Hardware does not verify MV==MVP condition for a Skip/Direct macroblock in a B Slice as no DMV is performed by hardware.

Macroblock type conversion rule for an inter macroblock in a B slice

Input			Internal			Output		Notes
Macroblock Type	SkipConvDisable || SkipConvDisable	DirectConvDisable	CBP	MV == MVP	MbAffSkipAllowed	mb_skip_flag	mb_type	
B_Skip_8x8 B_Skip_4x4	DC	DC	DC	n/a	1	1	-	Forced to B_Skip; Hardware will force CBP to zero and also ignore SkipConvDisable control.
B_Skip_8x8 B_Skip_4x4	DC	DC	DC	n/a	0	0	0	REVERSE convert to B_Direct_16x16; Hardware will force CBP to zero and also reverse convert to B_Direct_16x16 when it discovers Skip is not allowed.
B_Direct_16x16_4MVPair/16MVPair	0	0	0	n/a	1	1	-	Converted to B_Skip. Hardware first converts to B_Direct_16x16 and then further to B_Skip if CBP = 0.
B_Direct_16x16_4MVPair/16MVPair	0	0	0	n/a	0	0	0	Converted to B_Direct_16x16. Hardware first converts to B_Direct_16x16 and stop there as it discovers Skip is not allowed even CBP=0.
B_Direct_16x16_4MVPair/16MVPair	1	0	0	n/a	DC	0	0	Converted to B_Direct_16x16. Hardware converts to B_Direct_16x16 and stops there even though CBP = 0 as input disallows Skip conversion.
B_Direct_16x16_4MVPair/16MV	DC	0	NZ	n/a	DC	0	0	Converted to

Input			Internal			Output		Notes
Macroblock Type	SkipConvDisable || SkipConvDisable	DirectConvDisable	CBP	MV == MVP	MbAffSkipAllowed	mb_skip_flag	mb_type	
Pair								B_Direct_16x16. Hardware converts to B_Direct_16x16 and stops there because CBP != 0.
B_Direct_16x16_4MVPair/16MVPair	DC	1	DC	n/a	DC	0	16h	Stay as B_8x8. Hardware stays at B_8x8 and codes each sub macroblocks even all are direct.

The internal signal MbAffSkipAllowed is added to deal with a restriction on the frame/field flag (MbFieldFlag) which is unique to MBAFF. MbAffSkipAllowed is always set to 1 in non-MBAFF modes. In MBAFF mode, a macroblock pair may be both skipped only if its MbFieldFlag is the same as its available neighbor macroblock pair A or B if A or B is available (in that order), or is not 0 if A/B are both not available. Otherwise, one of the macroblocks in the pair must be coded.

To reduce the burden on software, PAK hardware handles the above restriction correctly. For the first MB in a pair, MbAffSkipAllowed is always set to 1. Therefore, hardware allows converting the first MB to Skip if skip conversion is enabled. For the second MB in a pair, hardware sets MbAffSkipAllowed to 0 if the following is true:

- The current MB Pair has different MbFieldFlag than its available neighbor A or B if A or B is available, or is not 0 if A/B are both not available
- And the first MB is coded as a SKIP (could be forced or converted)

Otherwise, it sets MbAffSkipAllowed to 1. As MbAffSkipAllowed is to 0 for the above condition, hardware will disallow Skip mode for the second MB. If the input signal forces it to Skip, hardware performs reverse-convertion to code it as P_L0_16x16 or B_Direct_16x16 with CBP = 0 for a macroblock in a P or B Slice. This means that hardware is able to correct the programming mistake by software. If the macroblock is not forced to skip, hardware simply disallows Skip conversion.

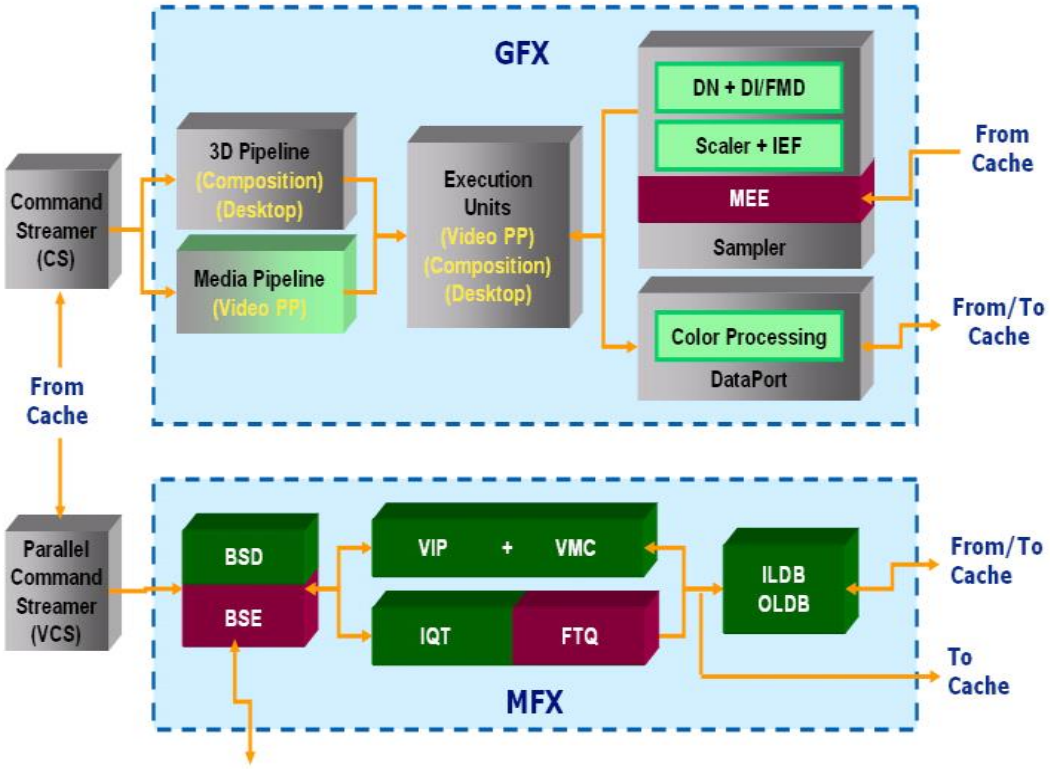
Software still has an option to disallow Skip Conversion on a per-MB basis using the MbSkipConvDisable control field in the inline command.

MFX Architecture

This section and the following sections of Media VDBOX contain the referential documentation on the Multi-Format Codecs, or MFX.

MFX Introduction

Multi-Format Codec (MFX) Engine is the hardware fixed function pipeline for decode and encoding. It includes multi-format decoding (MFD) and multi-format encoding (MFC).



MFC Overview

Multi-Format Codec (MFX) Engine is the hardware fixed function pipeline for decode and encoding. It includes multi-format decoding (MFD) and multi-format encoding (MFC).

Note: MFC only supports AVC (H.264).

Many decoding function blocks in MFD such as VIP, VMC, IQT, etc, are also used in encoding mode. Two blocks, FTQ and BSE, are encoding only.

The encoding process is partitioned across host software, the GPE engine, and the MFX engine. The generation of transport layer, sequence layer, picture layer, and slice header layer must be done in the host software. GP hardware is responsible for compressing from Slice Data Layer down to all macro-block and block layers. Specifically, GPE w/ VME acceleration is for motion vector estimation, motion estimation, and code decision.

The **VME**(*Video Motion Estimation*) is located next to all image processing units, such as DN (*denoise*) in GPE. MFX is for final bit packing and reconstructed picture generation.

MFC is operated concurrently with and independently from the GPE (3D/Media) pipeline with a separate command streamer. The two parallel engines have similar command protocol. They can be executed in

parallel with different context. For encoding, motion search, MB mode decision, and rate control are performed using GPE pipeline resources.

MFC is implemented to achieve the following objectives:

- Compliant with next generation high-definition optical video disc requirements, with sufficient performance headroom:
  - Support AVC 4:2:0 Main Profile and High Profile only (8-bit only), up to Level 4.1 resolution and up to 40 mbps bitstream. With sufficient duty cycles, higher bit rate contents can also be encoded. There is no support for Baseline, Extended, or High-10 Profiles.
- Performance requirements with MFX core frequency above 667MHz:
  - Real-time performance with 20% duty cycle or less.
  - Support concurrent decoding of two active HD bitstreams of different formats (for example, one AVC and one VC1 HD bitstream) and one active HD encoding.

As the result of this hardware partitioning, VPP and ENC are always running in GPE, and PAK is what runs exactly in MFC.

PAK - residue packing and entropy coding, including block transformation, quantization, data prediction, bitrate tuning and reference decoding. It delivers final packed bitstream and decoded key-frame reference:

- As the same as ENC, PAK is invoked on a Slice boundary; a single call of VPP can lead to multiple calls for PAK.
- Rate control is inside ENC and PAK only, not in VPP.
- PAK must always perform with reconstructed reference picture.

There is a general dependency of the three operation pipelines. Semaphores are inserted either according to frames or slices. The main CS will also be notified when the decoded reference is ready for the next frame set to be encoded. The detailed discussion will be found in a later section.

Host software is responsible for encoding the transport stream and all the sequence, picture, and slice layer/header in the bit-stream; the MFC system is responsible for compressing from Slice Data Layer down to all macro-block and block layers.

Sample Algorithmic Flow

Assuming all the hardware components are given, there are infinite usage possibilities left with intention for software to decide according to its own application needs depending upon the balanced requirement of coding speed, frame latency, power-consumption, and video quality, and depending upon the usage modes and user preferences (such as low-frame-rate-high-frame-quality vs. high-frame-rate-low-frame-quality).

The last part of this chapter, we illustrate a generic sample to show how a compression algorithm can be implemented to use our hardware.

Step 1. Application or driver initializes the encoder with desired configuration, including speed, quality, targeted bit-rate, input video info, and output format and restrictions.

Step 2. VPP - Application or driver feeds VPP one frame at a time in coded order with specified frame or field type, as well as transcoding information: motion vectors, coded complexity (i.e. bit size).

It will perform denoising and deblocking based on original and targeted bit-rate, and output additional

4 spatial variances and 2 temporal variances for each macroblock as well as the whole frame.

Step 3. ENC - Application or driver feeds ENC one coding slice buffer at a time including all VPP output. The frame level data is accessible to all slices.

a. Encoding setup unit (**ESE**) will set picture level quality parameters (including LUTs, and other costing functions) and set target bit-budget (TBB) and maximal bit-budget (MBB) to each macroblock based on rate-control (**RC**) scheme implemented. For B-frames, it will also make ME searching mode decision (either Fast, Slow or Uni-directional).
b. Loop over all macroblocks: calculate searching center (**MVP**) perform individual ME and IE (**MEE**). Multi-thread may be designed for HW according to a zigzag order for minimal dependency issue.
c. ENC make microblock level code decision (**CD**) outputs macroblock type, intra-mode, motion-vectors, distortions, as well as TBBs and MBBs.

Step 4. PAK - Application or driver feeds PAK one array of coded macroblocks covering a slice at a time, including all ENC output. Original frame buffer and reconstructed reference frame buffers are also available for PAK to access.

a. PAK may create bitstreams for all sequence, gop, picture, and slice level headers prior the first macroblock.
b. Loop over all macroblocks, accurate prediction block is constructed for either inter- or intra-predictions (**VMC** & **VIP**). If MB distortion is less than some predetermined threshold, for a B slice this step can be skipped as well as the Steps (c)-(e) and jump directly to Step (f); for a key slice the prediction calculated here will be directly used as the reference thus it jumps to Step (e) after this step.
c. Differencing the predicted block from the original block derives the residue block. Forward transformation and quantization (**FTQ**) is performed. For B slice, it will jump to Step (f) right after. For other types of slice, Steps (d) and (e) can be performed in a thread in parallel with Step (f) and beyond.
d. This is for accurate construction of reference pictures. Inverse quantization and inverse transformation (**IQT**) are performed and added to the predictions to have the decoded blocks.
e. **ILDB** is applied accordingly to the reconstructed blocks.
f. Meanwhile macroblock codes: including its configuration info (types and modes), motion info (motion vectors and reference ids), and residual info (quantized coefficients), are collected for packing (**BSE**) in the following sub-steps:
    i. Code clean-up (in **MPR**). Check and verify Mbtype and Cbps, use Skip or Zero respectively if one can. In principle, when there are equivalent codes, use the simple one.
    ii. Drop dependency (in **MPR**). Calculate relative codes from the absolute codes by associate them with neighborhood information. All neighborhood correlations are solved in this step.

iii. Unify symbols (in **SEC**). Translate relative codes into symbols, and table or context indices that are independent of the concept of syntax type.
iv. Entropy coding (**VLE**) on symbols.

g. Parsing bitstream data in RBSP form (in **VLE**), and output to application or driver.
h. By the end of each picture, write out the accurate actual data size to designate buffer for ENC to access.

Synchronization Mechanism

Encoding of a video stream can be broken down to three major steps (as explained in the previous section):

1. VPP: video-stream pre-processing
2. ENC: encoding, *that is,* code decision of inter-MVs and intra-modes
3. PAK: bit-stream packing
    a. residual calculation, transformation, and quantization
    b. code bit-stream packing
    c. reference generation of keyframes

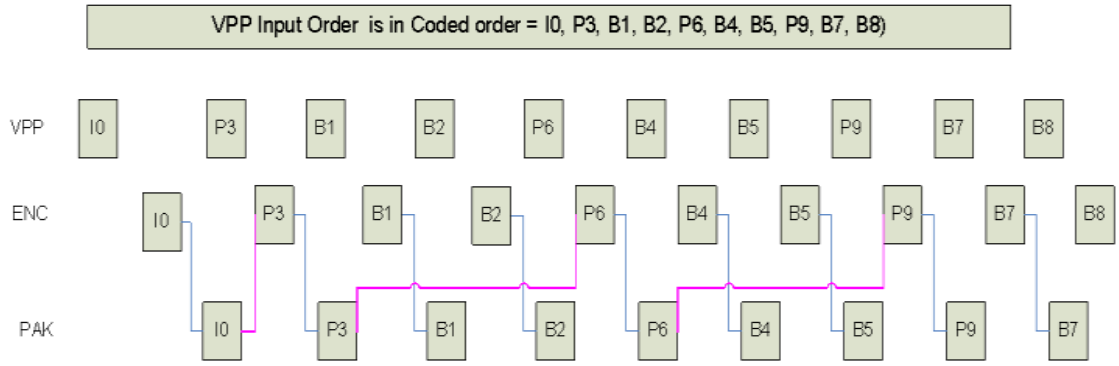
This section describes an architectural solution to map the first two steps in the GFX engine and the last step in the MFX engine. Since this mapping involves two OS-visible engines, managing them in parallel under one application is similar to the solution in earlier generations. Each engine has its own command streamers and has mechanisms to synchronize at required levels as described in the next sub-section.

Above three steps of encoding have dependencies in processing based on

i. functional pipeline order, *i.e.* on a given frame, VPP needs to be performed first, then ENC, then PAK and finally MFD (*Multi-Format Decoding*) for key reference frame generation.
ii. I-frames are key frames for P and B, they have to be first in every pipe-stage.
iii. P-frames are key frames for B frames and therefore P frames are processed first before the dependent B frames
iv. GFX Engine is time slice to work on either VPP or ENC frame as we discussed in the previous chapter.
v. PAK + MFD are executed on the same frame in the MFX engine by macro-block level pipelining within a slice. It should be noted that for the sake of simplicity, an entire frame (potentially multiple slices) are processed in the corresponding engine and no smaller granularity of switching is allowed between the functional pipeline stages.

Three steps of the encoding can be interleaved on two engines in the following way on a frame by frame basis.

Command Stream Synchronization



Restrictions

MFC implementation is subject to the following limitations.

- Context switching within MFC and with Graphics Engine occurs only at frame boundary to minimize the amount of information that needs to be tracked and maintained.

MFD Overview

When used for decoding, we also refer to the MFX Engine as the MFD Engine.

The Multi-Format Decoder (MFD) is a hardware fixed function pipeline for decoding the three-video codec format and one image compression codec format: AVC (H.264), VC-1, MPEG2, and JPEG.

- Compliant with next generation high-definition optical video disc requirements, with sufficient performance headroom:
    - Support AVC 4:2:0 Main and High (8-bit only) Profile only (no support for Baseline, Extended and High-10 Profiles), up to Level 5.1 (max 983,040 MB/s, max 36,864 MB/frame, and at most one dimension can reach 4K pixel) resolution and up to 40 mbps bitstream. With sufficient duty cycles, higher bit rate contents can also be decoded.
        - Allow a B-picture (frame or field) as a reference picture
- Support MVC 4:2:0 Stereoscopic Progressive Profile only, up to Level 5.1 (max 983,040 MB/s per view, max 36,864 MB/frame per view, and at most one dimension can reach 4K pixel) resolution and up to 40 mbps bitstream. With sufficient duty cycles, higher bit rate contents can also be decoded.
    - Support VC1 4:2:0 Simple, Main and Advanced Profiles, up to Level 4 (max 491,520 MB/s and max 16,384 MB/frame; max only one dimension will be at 4K pixel) resolution and up to 40 mbps bitstream. With sufficient duty cycles, higher bit rate contents can also be decoded.
        - Allow a B-field as a reference picture only in interlaced field decoding, no other modes.
    - Support MPEG2 HD Main Profile (4:2:0), up to High Level (1920x1152 pixels) and up to 80 mbps bitstream. With sufficient duty cycles, higher bit rate contents can also be decoded. No support for SNR and spatial-scalability.

- Does not support B-picture as a reference picture.

- Support Baseline JPEG with five choma types (4:0:0, 4:1:1, 4:2:2, 4:2:0, and 4:4:4. No support for Extended DCT-based mode, Progressive mode, Lossless mode, nor Hierarchical mode.
  - H/W support 64Kx64K, but Surface State can support only up to 16kx16k

Features	Supported	Unsupported
Coding processes	Baseline sequential mode: • 8-bit pixel precision of source images • loadable 2 AC and 2 DC Huffman tables • 3 loadable quantization matrix for Y, U, V • Interleaved and non-interleaved Scans • Single and multiple Scans	Extended DCT-based mode, Lossless, Hierarchical modes: More than 8 bit pixel resolution, progressive mode, arithmetic coding, 4 AC and 4 DC Huffman tables (extended mode), predictive process (lossless), multiple frames (hierarchical)
Number of image channels	1 for grey image 3 for Y, Cb, Cr color image	4-th channel (usually alpha blending image)
Image resolution	Arbitrary image size up to 16K * 16K	Larger than 16K * 16K (64K * 64K is max. in the JPEG standard)
Chroma subsampling ratio	Chroma 4:0:0 (grey image) Chroma 4:1:1 Chroma 4:2:0 Chroma horizontal 4:2:2 Chroma vertical 4:2:2 Chroma 4:4:4	Any other arbitrary ratio, e.g., 3:1 subsampled chroma
Additional feature (post-processing)	Image rotation: 90/180/270 degrees	

  - H/W does not impose restriction on picture frame aspect ratio, but is bounded by a max 256 MBs (4096 pixels) per dimension programmable at the H/W interface specifications.
    - For example, supporting HD video resolution 1920x1080/60i, 1920x1080/24p, 1280x720/60p

- Performance requirements with MFX core frequency above 1GHz
  - Real-time performance around 10% duty cycle
  - Support concurrently decoding of at least two active HD bitstreams of different formats (For example, one AVC and one VC1 HD bitstream)

- The parsing of transport layer and sequence layer is not performed in this hardware, and is required to be done in the host software.
- The MFD hardware pipeline is operated concurrently with and independently from the Graphics (3D/Media) pipeline with separate command streamer. The two parallel engines are designed with the similar command protocol. They can be executed in parallel with different context.
- Local storages and buffers along the hardware pipeline are kept at minimum. For example, there is no on-die row-store memory. They are resided on the system memory. MFD is designed to hide the memory access latency (in both the row stores and in the motion compensation units) in maximizing its decoding throughput.
- Support the following operating modes:
    - VLD mode - operation starts from entropy decoding of the compressed bit stream (parsing Slice Header and Slice Data Layer in AVC , Picture layer, Slice layer and MB Layer in VC-1, and MB-layer in MPEG2), all the way, to the reconstruction of display picture, including in-loop and out-loop deblocking, if any.
        - Streamout mode - a new feature of the VLD mode in assisting transcoding during decoding. Selected uncompressed data (e.g. per MB MV information) will be sent out to the EU and the ME engine for encoding into a different format or for the purpose of transcaling and transrating. In addition, the uncompressed result may continue to be processed by the rest of pipeline as in VLD mode to generate the display picture for transcoding. That is, while intermediate data are streaming out to the memory, the MFD Engine continues its decoding as usual.
        - Streamout mode - a new feature of the VLD mode in assisting transcoding during decoding. Selected uncompressed data (e.g. per MB MV information) will be sent out to the EU and the ME engine (resided on the Sampler of the 3D Gx Pipeline) for encoding into a different format or for the purpose of transcaling and transrating. In addition, the uncompressed result may continue to be processed by the rest of pipeline as in VLD mode to generate the display picture for transcoding. That is, while intermediate data are streaming out to the memory, the MFD Engine continues its decoding as usual.
        - For JPEG, only VLD mode is supported (No IT mode). Host software decodes Frame and Scan layers (down to Scan header in the JPEG bit stream syntax) and sends all the corresponding information and Scan payload to the MFD hardware pipeline.
    - IT mode - when host software has already performed all the bit stream parsing of the compressed data and packaging the uncompressed result into a specific format (as a sequence of per-MB record) stored in memory. The hardware pipeline will fetch one MB record at a time and perform the rest of the decoding process as in VLD mode
    - Host software (Application) is responsible for parsing and decoding all the transport and program layers, and all sequence layers. These parameters are passed to Driver and forwarded to H/W as needed through different STATE commands. Host software is also responsible for separating non-video data (audio, meta and user data) from sending to H/W.
        - MFD Engine is only responsible for macro-block and block layers decoding, plus certain level of header decoding. For AVC MFD starts decoding from Slice Header; for

VC1, MFD starts decoding from Picture Header, and for MPEG2 decoding starts from MB Layer only.
        - For JPEG, MFD is responsible for ECS and block layers decoding.
- Support bitstream formats (compressed video data) for each codec
    - AVC - 2 formats
    - MVC - 2 formats
        - DXVA2 MVC Short Slice Format
        - DXVA2 AVC Long Slice Format Specification (exactly the same as AVC)
    - VC1 - 2 formats
        - Fully compliant to Picture Parameter and Slice Control Parameter interface definition
- MPEG2
    - MB Layer only, according to DXVA 1 Specification
- JPEG
    - ECS Layer

The MFX codec is designed to be a stateless engine, that it does not retain any history of settings (states) for the encoding/decoding process of a picture. Hence, driver must issue the full set of MFX picture state command sequence prior to process each new picture. In addition, driver must issue the full set of Slice state command sequence prior to process a slice.

- In particularly, RC6 always happens between frame boundaries. So at the beginning of every frame, all state information needs to be programmed. There is no state information as part of media context definition.

To activate the AVC deblocker logic for incoming uncompressed 4:2:0-only video stream, one can pack the uncompressed video stream to compliant with the IPCM MB data format (including ILDB control information) and feed them into the MFD engine in IT mode. Since the MFD Engine is in IPCM mode, transformation, inter and intra processing are all inactive.

Start Code Detection and removal are done in the CPU, but the Start Code Emulation Prevention Byte is detected and removed by the front-end logic in the MFD. The bitstream format for each codec and for each mode is specified in this document.

Codec specific information are based on the following released documents from third parties:

- Draft of Version 4 of H.264/AVC (ITU-T Recommendation H.264 and ISO/IEC 14496-10 (MPEG-4 part 10) Advanced Video Coding); JVT-O205d1.doc; dated 2005-05-30
- Final Draft SMPTE Standard: VC1 Compressed Video Bitstream Format and Decoding Process, SMPTE 421M, dated 2006-1-6; PDF file.
- MPEG2 Recommendation ITU T H.262 (1995 E), ISO/IEC 13818-2: 1995 (E); doc file.
- Digital Compression and Coding of Continuous-tone Still Images, ITU-T Rec. T.81 and ISO/IEC 10918-1: Requirements and guidelines September 18 1992; itu-t81[1].pdf

MFD Memory Interface

The Memory Arbitrator follows the pre-defined arbitration policy (as indicated in the following listing P0 to P11, in which P0 is the highest priority) to select the next memory request to service, then it will perform the TLB translation (translation to physical address in memory), and make the actual request to memory.

The Memory Arbitration unit is also responsible for capturing the return data from memory (read request) and forward it to the appropriate unit along the MFD Engine.

- Read streams: (all 64B requests)
    - Commands for BSD : linear (including indirect data) (P0)
    - Indirect DMA (P1)
    - Row store for BSD: linear (P5)
    - Row store for MPR: linear (P6)
    - MC ref cache fetch : tiled (P2)
    - Intra row store: linear (P9)
    - ILDB row store: linear (P10)
- Write streams: (all 64B requests)
    - Row store write for BSD: linear and can avoid partial writes (P3)
    - Row store write for MPR: linear and can avoid partial writes (P4)
    - Intra row store write: linear and can avoid partial writes (P7)
    - ILDB row store write: linear and can avoid partial writes (P8)
    - Final dest writes: tiled and can potentially be partial, two ways to avoid these partials: 1) either write garbage and buffers are aligned or 2) read-modify writes for dribble end of line cases (P11)

MFD Codec-Specific Commands

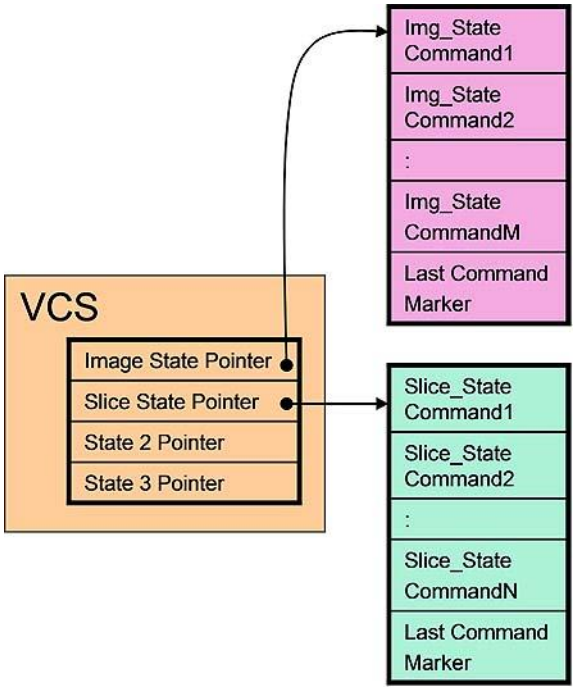
MFD hardware pipeline supports 3 different codec standards : AVC, VC1 and MPEG2. To make the interface flexible, each codec is designed with its own set of commands.

There are two categories of commands for each codec format : one set for VLD mode and one set for IT mode.

MFX State Model

The parallel video engine (PVE) supports two state delivery models: inline state model and indirect state model. For inline state model, the state commands (*_STATE) can be issued in batch buffers or ring buffers directly preceding object commands (*_OBJECT). In the indirect state model, the state commands are not placed in the batch buffers or ring buffers. Instead Indirect State Buffers provide state information (in the form of the above mentioned state commands) for the MFX pipeline. See MFX_STATE_POINTER for more details.

VCS (aka BCS) handles the difference of the two state delivery models. Therefore, the MFX pipeline always sees the state commands in both models. However, MFX hardware supports additional context save/restore of 'dynamic states'. Dynamic states are the internal signals that are persistent. This could be the CABAC context for macroblock encoding.



MFX State Model

The MFX codec is designed to be a stateless engine, that it does not retain any history of settings (states) for the encoding/decoding process of a picture. Hence, driver must issue the full set of MFX picture state command sequence prior to process each new picture. In addition, driver must issue the full set of Slice state command sequence prior to process a slice.

In particular, RC6 always happens between frame boundaries. So, at the beginning of every frame, all state information needs to be programmed. There is no state information as part of media context definition.

MFX Interruptability Model

MFX encoding and the encoding pipeline do not support interruption. All operations are frame based. Interrupts can only occur between frames; the driver will submit all the states at the beginning of each frame. Any state kept across frames is in MMIO registers that should be read between frames.

Software submits without any knowledge of where the parser head pointer is located. Also there is a non-deterministic amount of time for the new context to reach the command streamer. However, the state model for the MFX engine requires software to know exactly what state the pipeline is in at all times. This introduces cases where a preemption could occur during or after a state change without software ever knowing the state saved out to memory on the context switch.

Also, preemption is only allowed during the last macroblock in a row. Hardware cannot always perform a context switch when the new context is seen by the hardware. To avoid a switch during an invalid macroblock and to keep the state synchronized with software, there are two commands available that are used. MI_ARB_ON_OFF disables and enables preemption while MFX_WAIT ensures the context switch, if needed, preempts during macroblock execution. Below illustrates an example assuming VC1 VLD mode.

Command Ring/Batch	Notes
MI_ARB_ON_OFF = OFF	Disable preemption
S1	Inline or indirect state cmd 1
S2	Inline or indirect state cmd 2
S3	Inline or indirect state cmd 3
XXXX_OBJECT	Slice
MI_ARB_ON_OFF = ON	Enable preemption
MFX_WAIT	Allow preemption to occur while XXXX_OBJECT executes
MI_ARB_ON_OFF = OFF	Since arbitration is off again, state commands are allowed below
S4	Inline or indirect state cmd 4
S5	Inline or indirect state cmd 5
S6	Inline or indirect state cmd 6
XXXX_OBJECT	Slice
MI_ARB_ON_OFF = ON	Enable preemption
MFX_WAIT	Allow preemption to occur while XXXX_OBJECT executes
MI_ARB_ON_OFF = OFF	Since arbitration is off again, state commands are allowed below

Note that store DW commands may execute inside the preemption enabling window if needed.

Decoder Input Bitstream Formats

AVC Bitstream Formats - DXVA Short

Bitstream Buffer Address starts after the 3-byte start code, i.e. starts (and includes) at the NAL Header Byte. This byte must not be included in the Emulation Byte Detection Process.

AVC Bitstream Formats - DXVA Long

Bitstream Buffer Address starts after the 3-byte start code, i.e. starts (and includes) at the NAL Header Byte. This byte must not be included in the Emulation Byte Detection Process. Application will provide the Slice Header Skip Byte count (not including any possible Emulation Prevention Byte).

VC1 Bitstream Formats - Intel Long

Bitstream starts right at the MB layer, with any emulation byte crossing the header and MB layer being removed by application and the data length is adjusted.

MPEG2 Bitstream Formats - DXVA1

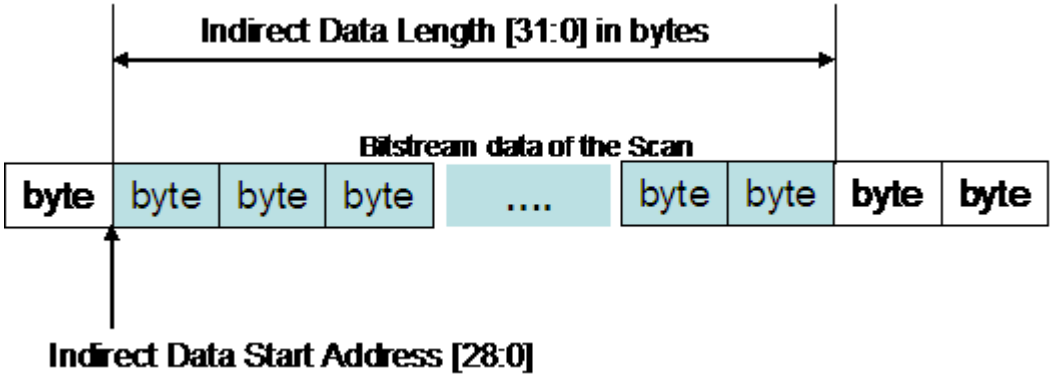
Bitstream buffer starts right at the very first MB data.

JPEG Bitstream Formats - Intel

Bitstream buffer starts right at the very first MCU data of each Scan.

The indirect data start address in MFD_JPEG_BSD_OBJECT specifies the starting Graphics Memory address of the bitstream data that follows the Scan header. It provides the byte address for the first MCU of the Scan. Different from MFD_MPEG2_BSD_OBJECT command, First MCU Bit Offset does not need to be specified because it is always set to zero.

Indirect data buffer for a Scan



The indirect data length in MFD_JPEG_BSD_OBJECT provides the length in bytes of the bitstream data for the Scan excluding Scan header. It includes the first byte of the first macroblock and the last byte of the last macroblock in the Scan. The Figure illustrates these parameters for a slice data.

Concurrent Multiple Video Stream Decoding Support

The natural place for switching across multiple streams is at the Slice boundary. Each Slice is a self-sustained unit of compressed video data and has no dependency with its neighbors (except for the Deblocking process). In addition, there is no interruptability within a Slice. However, when ILDB is invoked, the processing of some MBs will require neighbor MB information that crosses the Slice boundary. Hence, to limit the buffering requirement, in this version of hardware design, stream switching can only be performed at the picture boundary instead.

MFX Codec Commands Summary

DWord	Bit	Description
0	31:29	**Instruction Type =** GFXPIPE = 3h
	28:16	**3D Instruction Opcode =** PIPELINE_SELECT GFXPIPE[28:27 = 1h, 26:24 = 1h, 23:16 = 04h] (Single DW, Non-pipelined)
	15:1	**Reserved:** MBZ
	0	**Pipeline Select** 0: 3D pipeline is selected 1: Media pipeline is selected

Pipeline Type (28:27)	Opcode (26:24)	Sub Opcode (23:16)	Command	Definition Chapter
VC1 State				
2h	5h	0h	VC1_BSD_PIC_STATE	VC1 BSD
2h	5h	1h	Reserved	n/a
2h	5h	2h	Reserved	n/a
2h	5h	3h	VC1_BSD_BUF_BASE_STATE	VC1 BSD
2h	5h	4h	Reserved	n/a
2h	5h	5h-7h	Reserved	n/a
VC1 Object				
2h	5h	8h	VC1_BSD_OBJECT	VC1 BSD
2h	5h	9h-FFh	Reserved	n/a

Pipeline Type (28:27)	Opcode (26:24)	Sub Opcode (23:16)	Command	Definition Chapter
2h	6h	2h-7h	Reserved	N/A
Object				
2h	6h	9h-FFh	Reserved	N/A

Note that it is possible for a command to appear in both IMAGE and SLICE state buffer, e.g. QM_STATE for JPEG can be issued at frame level or scan/slice level.

Pipeline Type (28:27)	Opcode (26:24)	SubopA (23:21)	SubopB (20:16)	Command	Chapter	Recommended Indirect State Pointer Map	Interruptable?
	MFX Common	**Common**					
2h	0h	0h	0h	MFX_PIPE_MODE_SELECT	MFX	IMAGE	No
2h	0h	0h	1h	MFX_SURFACE_STATE	MFX	IMAGE	No
2h	0h	0h	2h	MFX_PIPE_BUF_ADDR_STATE	MFX	IMAGE	No
2h	0h	0h	3h	MFX_IND_OBJ_BASE_ADDR_STATE	MFX	IMAGE	No
2h	0h	0h	4h	MFX_BSP_BUF_BASE_ADDR_STATE	MFX	IMAGE	No
2h	0h	0h	6h	MFX_ STATE_POINTER	MFX	IMAGE	No

Pipeline Type (28:27)	Opcode (26:24)	SubopA (23:21)	SubopB (20:16)	Command	Chapter	Recommended Indirect State Pointer Map	Interruptable?
2h	0h	0h	7h	MFX_QM_STATE	MFX	IMAGE/SLICE	No
2h	0h	0h	8h	MFX_FQM_STATE	MFX	IMAGE	No
2h	0h	0h	9h	MFX_DBK_OBJECT	MFX	IMAGE	No
2h	0h	0h	A-1Eh	Reserved	n/a	n/a	No
	MFX Common	**Dec**					
2h	0h	1h	0-8h	Reserved	n/a	n/a	n/a
2h	0h	1h	9h	MFD_ IT_OBJECT	MFX	n/a	No
2h	0h	1h	A-1Fh	Reserved	n/a	n/a	n/a
	MFX Common	**Enc**					
2h	0h	2h	0-7Fh	Reserved	n/a	n/a	n/a
2h	0h	2h	8h	MFX_PAK_INSERT_OBJECT	MFX	n/a	No
2h	0h	2h	9h	Reserved	n/a	n/a	n/a
2h	0h	2h	Ah	MFX_STITCH_OBJECT	MFX	n/a	No
2h	0h	2h	B-1Fh	Reserved	n/a	n/a	n/a
	AVC/ MVC	**Common (State)**					
2h	1h	0h	0h	MFX_AVC_IMG_STATE	MFX	IMAGE	n/a
2h	1h	0h	1h	Reserved	n/a	n/a	n/a
2h	1h	0h	2h	MFX_AVC_DIRECTMODE_STATE	MFX	SLICE	n/a
2h	1h	0h	3h	MFX_AVC_SLICE_STATE	MFX	SLICE	n/a
2h	1h	0h	4h	MFX_AVC_REF_IDX_STATE	MFX	SLICE	n/a
2h	1h	0h	5h	MFX_AVC_WEIGHTOFFSET_STATE	MFX	SLICE	n/a
2h	1h	0h	9	Reserved	n/a	n/a	n/a
2h	1h	0h	D-1Fh	Reserved	n/a	n/a	n/a
	AVC/ MVC	**Dec**					
2h	1h	1h	0-5h	Reserved	MFX	n/a	n/a
2h	1h	1h	6h	MFD_AVC_DPB_STATE	MFX	IMAGE	n/a
2h	1h	1h	7h	MFD_AVC_SLICEADDR_OBJECT	MFX	n/a	n/a
2h	1h	1h	8h	MFD_AVC_BSD_OBJECT	MFX	n/a	No
2h	1h	1h	9-1Fh	Reserved	n/a	n/a	n/a
	AVC/ MVC	**Enc**					
2h	1h	2h	0-8h	Reserved	n/a	n/a	n/a
2h	1h	2h	9h	MFC_AVC_PAK_OBJECT	MFX	n/a	No
2h	1h	2h	A-1Fh	Reserved	n/a	n/a	n/a
	AVC/ MVC	**Extension**					

Pipeline Type (28:27)	Opcode (26:24)	SubopA (23:21)	SubopB (20:16)	Command	Chapter	Recommended Indirect State Pointer Map	Interruptable?
	VC1	Common (State)					
2h	2h	0h	0h	Reserved	n/a	n/a	n/a
2h	2h	0h	1h	MFX_VC1_PRED_PIPE_STATE	MFX	IMAGE	n/a
2h	2h	0h	2h	MFX_VC1_DIRECTMODE_STATE	MFX	SLICE	n/a
2h	2h	0h	3-1Fh	Reserved	n/a	n/a	n/a
	VC1	Dec					
2h	2h	1h	0h	MFD_VC1_SHORT_PIC_STATE	MFX	IMAGE	n/a
2h	2h	1h	1h	MFD_VC1_LONG_PIC_STATE	MFX	IMAGE	n/a
2h	2h	1h	2-7h	Reserved	n/a	n/a	n/a
2h	2h	1h	8h	MFD_VC1_BSD_OBJECT	MFX	n/a	No
2h	2h	1h	9-1Fh	Reserved	n/a	n/a	n/a
	VC1	Enc					
2h	2h	2h	0-1Fh	Reserved	n/a	n/a	n/a
	MPEG2	Common (State)					
2h	3h	0h	0h	MFX_MPEG2_PIC_STATE	MFX	IMAGE	n/a
2h	3h	0h	1-1Fh	Reserved	n/a	n/a	n/a
	MPEG2	Dec					
2h	3h	1h	1-7h	Reserved	n/a	n/a	n/a
2h	3h	1h	8h	MFD_MPEG2_BSD_OBJECT	MFX	n/a	No
2h	3h	1h	9-1Fh	Reserved	n/a	n/a	n/a
	MPEG2	Enc					
2h	3h	2h	0-2h	Reserved	n/a	n/a	n/a
2h	3h	2h	3h	MFC_MPEG2_PAK_OBJECT			
2h	3h	2h	3-8h	Reserved			
2h	3h	2h	9h	MFC_MPEG2_SLICEGROUP_STATE			
2h	3h	2h	A-1Fh	Reserved			
	VP8	Common (State)					
2h	4h	0h	0h	MFX_VP8_PIC_STATE	MFX	IMAGE	n/a
	VP8	Dec					
2h	4h	1h	8h	MFD_VP8_BSD_OBJECT	MFX	IMAGE	No
	VP8	Enc					
2h	4h	2h		Reserved			
	JPEG	Common					
2h	7h	0h	0h	MFX_JPEG_PIC_STATE	MFX	IMAGE	No
2h	7h	0h	1h	Reserved	n/a	n/a	n/a
2h	7h	0h	2h	MFX_JPEG_HUFF_TABLE_STATE	MFX	IMAGE	No

Pipeline Type (28:27)	Opcode (26:24)	SubopA (23:21)	SubopB (20:16)	Command	Chapter	Recommended Indirect State Pointer Map	Interruptable?
2h	7h	0h	3-1Fh	Reserved	n/a	n/a	n/a
	JPEG	**Common**					
2h	7h	0h	0h	MFX_JPEG_PIC_STATE	MFX	IMAGE	No
2h	7h	0h	1h	Reserved	n/a	n/a	n/a
2h	7h	0h	2h	MFX_JPEG_HUFF_TABLE_STATE	MFX	IMAGE	No
2h	7h	0h	3-1Fh	Reserved	n/a	n/a	n/a
	JPEG	**Dec**					
2h	7h	1h	1-7h	Reserved	MFX	n/a	n/a
2h	7h	1h	8h	MFD_JPEG_BSD_OBJECT	MFX	MCU	No
2h	7h	1h	9-1Fh	Reserved	MFX	n/a	n/a
	JPEG	**Enc**					
2h	7h	2h	0-1Fh	Reserved	MFX	n/a	n/a

MMIO Space Registers

Range Start	Range End	Unit owner
00002000	00002FFF	Render/Generic Media Engine
00004000	00004FFF	Render/Generic Media Graphics Memory Arbiter
00005000	0000517F	
00006000	00007FFF	Reserved
00012000	000123FF	MFX Control Engine (Video Command Streamer)
00012400	00012FFF	Media Units (VIN unit)
00014000	00014FFF	MFX Memory Arbiter
00022000	00022FFF	Blitter Engine
00024000	00024FFF	Blitter Memory Arbiter
00030000	0003FFFF	Reserved
00100000	00107FFF	Fence Registers
00140000	0017FFFF	MCHBAR (SA)

Memory Interface Command Map

04h Opcode (28:23)	MI_FLUSH

MFX Decoder Commands Sequence

The MFX codec is designed to be a stateless engine, that it does not retain any history of settings (states) for the encoding/decoding process of a picture. Hence, driver must issue the full set of MFX picture state command sequence prior to process each new picture. In addition, driver must issue the full set of Slice state command sequence prior to process a slice.

In particular, RC6 always happens between frame boundaries. So at the beginning of every frame, all state information needs to be programmed. There is no state information as part of media context definition

Examples for AVC

The following gives a sample command sequence programmed by a driver

a) For Intel or DXVA2 AVC Long Slice Bitstream Format

MFX_PIPE_MODE_SELECT

MFX_SURFACE_STATE

MFX_PIPE_BUF_ADDR_STATE

MFX_IND_OBJ_BASE_ADDR_STATE

MFX_BSP_BUF_BASE_ADDR_STATE

MFX_QM_STATE

VLD mode: MFX_AVC_PICID_STATE

MFX_AVC_IMG_STATE

MFX_AVC_DIRECTMODE_STATE

MFX_AVC_REF_IDX_STATE

MFX_AVC_WEIGHTOFFSET_STATE

MFX_AVC_SLICE_STATE

VLD mode: MFD_AVC_BSD_OBJECT

IT mode: MFD_IT_OBJECT

MI_FLUSH

b) For DXVA2 AVC Short Slice Bitstream Format (for VLD mode only)

MFX_PIPE_MODE_SELECT

MFX_SURFACE_STATE

MFX_PIPE_BUF_ADDR_STATE

MFX_IND_OBJ_BASE_ADDR_STATE

MFX_BSP_BUF_BASE_ADDR_STATE

MFD_AVC_DPB_STATE

VLD mode: MFX_AVC_PICID_STATE

MFX_AVC_IMG_STATE

MFX_QM_STATE

MFX_AVC_DIRECTMODE_STATE

VLD mode : MFD_AVC_SLICEADDR_OBJECT

VLD mode: MFD_AVC_BSD_OBJECT

VLD mode : MFD_AVC_BSD_SLICEADDR_OBJECT

VLD mode: MFD_AVC_BSD_OBJECT

... repeat these four commands N-1 times for a N-slice picture

VLD mode: MFD_AVC_BSD_OBJECT (for the last slice of the picture)

MI_FLUSH

Examples for VC1

The following gives a sample command sequence programmed by a driver

a) For Intel Proprietary Long Bitstream Format

MFX_VC1_DIRECTMODE_STATE

MFX_VC1_PRED_PIPE_STATE

MFX_VC1_LONG_PIC_STATE

VLD mode: MFD_VC1_BSD_OBJECT

IT mode: MFD_IT_OBJECT

MI_FLUSH

b) For DXVA2 VC1 Compliant Bitstream Format (for VLD mode only)

MI_FLUSH (Video Pipeline Cache invalidate = 1)

MFX_VC1_DIRECTMODE_STATE

MFX_VC1_PRED_PIPE_STATE

MFX_VC1_SHORT_PIC_STATE

VLD mode: MFD_VC1_BSD_OBJECT

MI_FLUSH

c) For DXVA2 VC1 Compliant Bitstream Format (for VLD mode only), and field pair picture

Batch buffer for top-field

states....

Slice_objs...

MI_flush

store register immediate (if VC1 short format with interlaced field pic)

MI_flush

Batch buffer for bottom field

load register immediate (if VC1 short format with interlaced field pic)

MI_flush

states....

Slice_objs...

MI_flush

Examples for JPEG

The following gives a sample command sequence programmed by a driver

Programmed once at the start of decoding

MFX_PIPE_MODE_SELECT

MFX_PIPE_SURFACE_STATE

MFX_IND_OBJ_BASE_ADDR_STATE

MFX_PIPE_BUF_ADDR_STATE

MFX_JPEG_PIC_STATE

Programmed at the start of Frame or Scan (These commands can be sent multiple times either before MFX_JPEG_PIC_STATE or before MFD_JPEG_BSD_OBJECT)

MFX_JPEG_HUFF_TABLE

MFX_QM_STATE

Programmed per Scan (These commands can be sent multiple times depending on each bit stream)

MFD_JPEG_ BSD_OBJECT

MI_FLUSH

MFX Pipe Common Commands

MFX Commands are organized into groups based on their scope of functioning. There are Pipeline Common state commands that are common to all codecs (encoder and decoder) and is applicable to the processing of one full frame/field. There are also individual codec Common state commands that are common to both encoder and decoder of that particular codec. These latter common state commands, some are applicable at the processing of one full frame/field, and some are applicable at the processing of an individual slice level.

MFX_WAIT

MFX_STATE_POINTER

MFX_PIPE_MODE_SELECT

The Encoder Pipeline Modes of Operation (Per Frame):

1. PAK Mode: VCS-command driven, setup by driver. Like the IT mode of decoder, it is executed on a per-MB basis. Hence, each PAK Object command corresponds to coding of only one MB.
    a. Normal Mode (including transcoding): receive per-MB control and data (MV, mb_type, cbp, etc.). It generates the output compressed bitstream as well as the reconstructed reference pictures, one MB at a time, for later use.

Encoder StreamOut Mode: to provide per-MB, per-Slice and per-Frame coding result and information (statistics) to the Host, Video Preprocessing Unit and ENC Unit to enhance their operations.

The Decoder Pipeline Modes of Operation (Per Frame):

1. VLD Mode: The output from the BSD (weight&offset/coeff/motion vectors record) can be sent in part (as specified) and to the remaining fixed function hardware pipeline to complete the decoding processing. The driver specifies through MFD commands of what to send out from the BSD unit and where to send the BSD output.
    a. For transcoding (including transrating and transcaling), part of the BSD output (a series of per-MB record) can be sent to memory for further processing to encode into a difference output format. This function is named as StreamOut. When StreamOut is active, not all MB information needs to be sent, only MVs and selective MB coding information.
2. IT Mode: In this mode, the BSD is not invoked. Instead host performs all the bitstream decoding and parsing; and the result are saved into memory in a specific per-MB record format. The MFD Engine VCS reads in these records one at time and finish the rest of the decoding (IT, MC, IntraPred and ILDB).

MB information is organized into two indirect data buffers, one for MVs and one for residue coefficients. As such, two indirect base address pointers are defined.

MFX_SURFACE_STATE

MFX_PIPE_BUF_ADDR_STATE

MFX_IND_OBJ_BASE_ADDR_STATE

MFX_BSP_BUF_BASE_ADDR_STATE

MFX_PAK_INSERT_OBJECT

MFX_STITCH_OBJECT

MFX_QM_STATE

Bits	31:24	23:16	15:8	7:0
Dword 1	QuantMatrix[0][3]	QuantMatrix[0][2]	QuantMatrix[0][1]	QuantMatrix[0][0]
Dword 2	QuantMatrix[0][7]	QuantMatrix[0][6]	QuantMatrix[0][5]	QuantMatrix[0][4]
Dword 3	QuantMatrix[1][3]	QuantMatrix[1][2]	QuantMatrix[1][1]	QuantMatrix[1][0]
...	...	...	...	...
Dword 16	QuantMatrix[7][7]	QuantMatrix[7][6]	QuantMatrix[7][5]	QuantMatrix[7][4]

MFX_ STATE FQM

This is a frame-level state. Reciprocal Scaling Lists are always sent from the driver regardless whether they are specified by an application or the default/flat lists are being used. This is done to save the ROM (to store the default matrices) inside the PAK Subsystem. Hence, the driver is responsible for determining the final set of scaling lists to be used for encoding the current slice, based on the AVC Spec (Fall-Back Rules A and B). For encoding, there is no need to send the qm_list_flags[i], i=0 to7 and qm_present_flag to the PAK, since Scaling Lists syntax elements are encoded above Slice Data Layer.

FQM Reciprocal Scaling Lists elements are 16-bit each, conceptually equal to 1/ScaleValue. QM matrix elements are 8-bit each, equal to ScaleValue. However, in AVC spec., the Reciprocal Scaling Lists elements are not exactly equal to one-over of the corresponding Scaling Lists elements. The numbers are adjusted to simplify hardware implementation.

For all the description below, a scaling list set contains 6 4x4 scaling lists (or forward scaling lists) and 2 8x8 scaling lists (or forward scaling lists).

In MFX PAK mode, PAK needs both forward Q scaling lists and IQ scaling lists. The IQ scaling lists are sent as in MFD in raster scan order as shown in MFX_AVC_QM_STATE. But the Forward Q scaling lists are sent in transport form, i.e. column-wise raster order (column-by-column) to simplify the H/W.

Precisely, if the reciprocal forward scaling matrix is F[4][4], then the 16 word of the matrix will be set as the following:

For JPEG encoder, 16-bit precision is used for each element 1/QM matrix. The 32 DWords are used for 64 QM elements with the following data structure:

	Bits 15:0	Bits 31:16
DWord1	1/QM[0][0]	1/QM[1][0]
DWord2	1/QM[2][0]	1/QM[3][0]
DWord3	1/QM[4][0]	1/QM[5][0]
DWord4	1/QM[6][0]	1/QM[7][0]
DWord5	1/QM[0][1]	1/QM[1][1]
DWord6	1/QM[2][1]	1/QM[3][1]
DWord7	1/QM[4][1]	1/QM[5][1]
DWord8	1/QM[6][1]	1/QM[7][1]
...		
DWord31	1/QM[4][7]	1/QM[5][7]
DWord32	1/QM[6][7]	1/QM[7][7]

Bitplane Buffer

Bitplane coding is used in seven different cases in VC-1, although not all the seven syntax elements are present in the same picture header at the same time. The following list shows which syntax elements are coded as bitplanes in each picture header:

Progressive I and BI picture headers in AP: ACPRED, OVERFLAGS

Field interlace I and BI picture headers in AP: ACPRED, OVERFLAGS

Frame interlace I and BI picture headers in AP: FIELDTX, ACPRED, OVERFLAGS

Frame interlace P picture headers in AP: SKIPMB

Progressive P picture headers in SP and MP: MVTYPEMB, SKIPMB

Progressive P picture headers in AP: MVTYPEMB, SKIPMB

Field interlace B picture headers in AP: FORWARDMB

Frame interlace B picture headers in AP: DIRECTMB, SKIPMB

Progressive B picture headers in AP: DIRECTMB, SKIPMB

Progressive B picture headers in MP: DIRECTMB, SKIPMB

There are also seven different modes of coding the bitplane information. Except when the bitplane is coded in raw mode, the bitplane is decoded by the host and provided to the hardware in the bitplane buffer.

Since at most three bitplanes are encoded in any picture header, instead of using a complete byte for signaling the values of all the seven possible bitplanes for each MB, a more efficient approach is used with each byte divided in two nibbles and each nibble carries the data of up to four bitplanes for one MB.

PictureType	Bits 3, 7	Bit 2, 6	Bits 1, 5	Bits 0, 4
I or BI	0	OVERFLAGS	ACPRED	FIELDTX
P	0	MVTYPEMB	SKIPMB	0
B	0	FORWARDMB	SKIPMB	DIRECTMB

The bytes containing the above defined nibbles are stored in the bitplane buffer in raster scan order. The bitplane buffer is a linear buffer with a buffer pitch (as defined by Bitplane Buffer Pitch field in VC1_BSD_PIC_STATE command). When the number of macroblock in a row is odd, the last byte of the row containing the last macroblock in bits 0-3. The first macroblock of the next row starts at the next pitch offset from the first macroblock of the current row.

The bitplane buffer structure must be sent once per picture only if there is one or more syntax elements coded as bitplanes in the picture header.

Video Codecs

The following sections contain the various registers for video codec support. Specifically, the codec types supported are:

Supported Codec Types
Advanced Video Coding (AVC)/ H.264/MPEG-4 Part 10 (MVC)
MPEG-2 (H.222/H.262) -- Used in Digital Video Broadcast and DVDs
VC1 -- SMPTE 421M, known informally as VC-1
VP8 -- Video compression format
JPEG and MJPEG -- A video format in which video gram or interlaced field of a digital video sequence is compressed separately as a JPEG image
Other Codec Functions

Internal Media Rowstore table - An internal Media Rowstore Storage is added to reduce memory read/write to save power. If the internal Media Rowstore exists, driver should use the storage as the following table indicates.

AVC/VC1/MPEG2/JPEG/VP8 Decoder/Encoder:

[BSD is bitstream decoder rowstore; MPR is Motion Prediction rowstore; IP is Intra Prediction rowstore; VLF is loop filter rowstore; VDE is VDENC rowstore]

Codec	Mode	Frame Width	BSD	MPR	IP	VLF	VDENC	BSD Addr	MPR Addr	IP Addr	VLF ADDR	VDENC ADDR
VDENC AVC	Frame	<= 4k	N	Y	Y	Y	Y	0	256	512	768	0
	Field/Mbaff	<= 4k	N	Y	Y	N	Y	0	512	1024	N/A	0
VC1 Dec		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
MPEG2		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
JPEG		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
VP8		<= 4k	Y	N	Y	Y	Y	0	N/A	256	512	1536

AVC (H.264)

AVC Common Commands

MFX Commands are organized into groups based on their scope of functioning. There are Pipeline Common state commands that are common to all codecs (encoder and decoder) and is applicable to the processing of one full frame/field. There are also individual codec Common state commands that are common to both encoder and decoder of that particular codec. These latter common state commands, some are applicable at the processing of one full frame/field, and some are applicable at the processing of an individual slice level.

MFX_AVC_IMG_STATE

A new command is added to support MPEG transport stream encapsulation of AVC bitstream in Encoder mode. This command should be used only when MPEG transport stream is needed.

MFX_MPEG_TS_CONTROL

MAX_QP_DELTA: Maximum QP delta is the Magnitude of QP delta between passes.

MAX_QP_DELTA is selected such that cumulative QP over all possible passes shouldn't exceed 51.

Example Configurations:

MAX Number of Passes	MAX_QP_DELTA
4	0xc
5	0xa
6	0x8
7	0x7

Commands
MFX_AVC_DIRECTMODE_STATE MFX_AVC_REF_IDX_STATE MFX_AVC_WEIGHTOFFSET_STATE

AVC Decoder Commands

These are decoder-only commands. They provide the pointer to the compressed input bitstream to be decoded.

MFD_AVC_DPB_STATE

NOTE modified from DXVA2 - The values in RefFrameList and UsedForReference_Flag are the primary means by which the H/W can determine whether the corresponding entries in RefFrameList, POCList, LTSTFrameNumList, and Non-ExistingFrame_Flag should be considered valid for use in the decoding process of the current picture or not. When RefFrameList[i] is marked to be invalid, the values of POCList[i][0], POCList[i][1], LTSTFrameNumList[i], UsedForReference_Flag[i], and Non-ExistingFrame_Flag[i] must all be equal to 0. When UsedForReference_Flag[i] = 0, the value of RefFrameList[i] must be marked invalid.

MFD_AVC_SLICEADDR

MFD_AVC_BSD_OBJECT

Inline Data Description for MFD_AVC_BSD_Object

MFD_AVC_PICID_STATE

NOTE 1: In AVC short format, PictureIDList has one-to-one corresponding to LongTermFrame_Flag list, Non-ExistingFrame_flag list, UsedForReference_Flag list, FrameNumList list in MFD_AVC_DPB_STATE.

NOTE 2: PictureIDList is only used to identify reference picture across frames. Hardware will convert the picture in the RefFrameList to PictureID before writing out DMV data and convert back to RefFrameList Index after reading out DMV data. The reference pictures and their orders in the RefFrameList can be changed across frames.

Session Decoder StreamOut Data Structure

When StreamOut is enabled, per MB intermediated decoded data (MVs, mb_type, MB qp, etc.) are sent to the memory in a fixed record format (and of fixed size). The per-MB records must be written in a strict raster order and with no gap (i.e. every MB regardless of its mb_type and slice type, must have an entry in the StreamOut buffer). Therefore, the consumer of the StreamOut data can offset into the StreamOut Buffer (**StreamOut Data Destination Base Address**) using individual MB addresses.

A StreamOut Data record format is detailed as follows:

DWord	Bit	Description
0	23	**Reserved MBZ**
	22-20	**EdgeFilterFlag** (AVC) / **OverlapSmoothFilter** (VC1)
	19-17	**CodedPatterDC** (for AVC only, **111b** for others) This field indicates whether DC coefficients are sent. 1 bit each for Y, U and V.
	16	**Reserved MBZ**
	15	**Transform8x8Flag** When it is set to 0, the current MB uses 4x4 transform. When it is set to 1, the current MB uses 8x8 transform. The transform_szie_8x8_flag syntax element, if present in the output bitstream, is the same as this field. However, whether transform_szie_8x8_flag is present or not in the output bitstream depends on several conditions: This field is only allowed to be set to 1 for two conditions: It must be 1 if **IntraMbFlag** = INTRA and **IntraMbMode** = INTRA_8x8 It may be 1 if **IntraMbFlag** = INTER and there is no sub partition size less than 8x8 Otherwise, this field must be set to 0. 0: 4x4 integer transform 1: 8x8 integer transform
	14	**MbFieldFlagMbFieldFlag** This field specifies whether current macroblock is coded as a field or frame macroblock in MBAFF mode. This field is exactly the same as FIELD_PIC_FLAG syntax element in non-MBAFF mode. Same as the mb_field_decoding_flag syntax element in AVC spec. >0 = Frame macroblock 1 = Field macroblock

DWord	Bit	Description
	13	**IntraMbFlag** This field specifies whether the current macroblock is an Intra (I) macroblock. I_PCM is considered as Intra MB. For I-picture MB (IntraPicFlag =1), this field must be set to 1. This flag must be set in consistent with the interpretation of MbType (inter or intra modes). 0: INTER (inter macroblock) 1: INTRA (intra macroblock)
	12-8	**MbType5Bits** This field is encoded to match with the best macroblock mode determined as described in the next section. It follows AVC encoding for inter and intra macroblocks.<
	7	**MbPolarity** FieldMB Polarity - vctrl_vld_top_field AVC
	6	**Reserved MBZ**
	5:4	**IntraMbMode** This field is provided to carry information partially overlapped with MbType. This field is only valid if **IntraMbFlag** = INTRA, otherwise, it is ignored by hardware.
	3	**Reserved MBZ**
	2	**MbSkipFlag** Reserved MBZ (DXVA Encoder). HW (VDSunit) doesn't have skip MB info. It sets to 1 if any of the sub-blocks is inter, uses predicted MVs, and skips sending MVs explicitly in the code stream. Currently H/W can provide this flag and is defaulted to 0 always.
	1:0	**InterMbMode** This field is provided to carry redundant information as that in MbType. It also carries additional information such as skip. This field is only valid if **IntraMbFlag** =INTER, otherwise, it is ignored by hardware.
1	31:16	**MbYCnt (Vertical Origin).** This field specifies the vertical origin of current macroblock in the destination picture in units of macroblocks. Format = U8 in unit of macroblock.
	15:0	**MbXCnt (Horizontal Origin).** This field specifies the horizontal origin of current macroblock in the destination picture in units of macroblocks. Format = U8 in unit of macroblock.
2	31	**Conceal MB Flag.** This field specifies if the current MB is a conceal MB, use in AVC/VC1/MPEG2 mode.
	30	**Last MB of the Slice Flag.** This field indicate the current MB is a last MB of the slice. Use in AVC/VC1/MPEG2 mode.
	29:24	**Reserved**
	23:20	**CbpAcV** 0 in a bit - indicates the corresponding 8x8 block or 4x4 sub-block is not present (because all

DWord	Bit	Description
		coefficient values are zero) 1 in a bit - indicates the corresponding 8x8 block or 4x4 sub-block is present (although it is still possible to have all its coefficients be zero - bad coding).
	19:16	**CbpAcU** 0 in a bit - indicates the corresponding 8x8 block or 4x4 sub-block is not present (because all coefficient values are zero) 1 in a bit - indicates the corresponding 8x8 block or 4x4 sub-block is present (although it is still possible to have all its coefficients be zero - bad coding).
	15:0	**CbpAcY** 0 in a bit - indicates the corresponding 8x8 block or 4x4 sub-block is not present (because all coefficient values are zero) 1 in a bit - indicates the corresponding 8x8 block or 4x4 sub-block is present (although it is still possible to have all its coefficients be zero - bad coding). Bit15=Y0Sub0, Bit0=Y3Sub3
3	31:28	**Skip8x8Pattern** /> This field indicates whether each of the four 8x8 sub macroblocks is using the predicted MVs and will not be explicitly coded in the bitstream (the sub macroblock will be coded as direct mode). It contains four 1-bit subfields, corresponding to the 4 sub macroblocks in sequential order. The whole macroblock may be actually coded as B_Direct_16x16 or B_Skip, according to the macroblock type conversion rules described in a later sub section. This field is only valid for a B slice. It is ignored by hardware for a P slice. Hardware also ignores this field for an intra macroblock. 0 in a bit - Corresponding MVs are sent in the bitstream 1 in a bit - Corresponding MVs are not sent in the bitstream
	27:25	**Reserved**
	24:16	**NzCoefCountMB** - all coded coefficients input including AC/DC blocks in current MB. Range 0 to 384 (9 bits)
	15:8	MbClock16 - MB compute clocks in 16-clock unit.
	7	**mbz** (AVC) **/ QScaleType** (MPEG2)
	6:0	**QpPrimeY** (AVC) **/ QScaleCode** (MPEG2) The luma quantization index. This is the per-MB QP value specified for the current MB.
4 to 6	31:0 Each	For intra macroblocks, definition of these fields are specified in 1 For inter macroblocks, definition of these fields are specified in 2
7	31:24	**Reserved**
	23:20	**MvFieldSelect** (Ref polarity top or bottom bits) for VC1 and MPEG2 vcp_vds_mvdataR[162:159] VC1 vmd_vds_mt_vert_fld_selR[3:0] MPEG2
	19:12	**Reserved**
	11:10	**SubBlockCodeType V** (If 8x8, 8x4, 4x8, 4x4 type)

DWord	Bit	Description
	9:8	**SubBlockCodeType U** (specifies 8x8, 8x4, 4x8, 4x4 type) VC1
	7:6	**SubBlockCodeType Y3** (specifies 8x8, 8x4, 4x8, 4x4 type) VC1
	5:4	**SubBlockCodeType Y2** (specifies 8x8, 8x4, 4x8, 4x4 type) VC1
	3:2	**SubBlockCodeType Y1** (specifies 8x8, 8x4, 4x8, 4x4 type) VC1
	1:0	**SubBlockCodeType Y0** (specifies 8x8, 8x4, 4x8, 4x4 type) VC1
Inter Cases		
8	31:16	**MvFwd[0].y** - y-component of the forward motion vector of the 1 8x8 or 1 4x4 subblock
	15:0	**MvFwd[0].x** - x-component of the forward motion vector of the 1 8x8 or 1 4x4 subblock
9	31:0	**MvBck[0]** - the backward motion vector of the 1 8x8 or 1 4x4 subblock
10	31:0	**MvFwd[1]** - the forward motion vector of the 2 8x8 or 4 4x4 subblock
11	31:0	**MvBck[1]** - the backward motion vector of the 2 8x8 or 4 4x4 subblock
12	31:0	**MvFwd[2]** - the forward motion vector of the 3 8x8 or 8 4x4 subblock
13	31:0	**MvBck[2]** - the backward motion vector of the 3 8x8 or 8 4x4 subblock
14	31:0	**MvFwd[3]** - the forward motion vector of the 4 8x8 or 12 4x4 subblock
15	31:0	**MvBck[3]**> - the backward motion vector of the 4th 8x8 or 12 4x4 subblock
8 to 15	31:0	**Reserved MBZ**

Inline data subfields for an Intra Macroblock

DWord	Bit	Description
4	31:16	**LumaIndraPredModes[1]** Specifies the Luma Intra Prediction mode for four 4x4 sub-block of a MB, 4-bit each. AVC: See the bit assignment table later in this section. VC1: MBZ. MPEG2: MBZ.
	15:0	**LumaIndraPredModes[0]** Specifies the Luma Intra Prediction mode for four 4x4 sub-block, four 8x8 block or one intra16x16 of a MB. 4-bit per 4x4 sub-block (Transform8x8Flag=0, Mbtype=0 and intraMbFlag=1) or 8x8 block (Transform8x8Flag=1, Mbtype=0, MbFlag=1), since there are 9 intra modes. 4-bit for intra16x16 MB (Transform8x8Flag=0, Mbtype=1 to 24 and intraMbFlag=1), but only the LSBit[1:0] is valid, since there are only 4 intra modes.

DWord	Bit	Description
		AVC: See the bit assignment table later in this section. VC1: MBZ. MPEG2: MBZ.
5 AVC INTRA	31:16	**LumaIndraPredModes[3]** Specifies the Luma Intra Prediction mode for four 4x4 sub-block of a MB, 4-bit each. AVC: See the bit assignment table later in this section. VC1: MBZ. MPEG2: MBZ.
	15:0	**LumaIndraPredModes[2]** Specifies the Luma Intra Prediction mode for four 4x4 sub-block of a MB, 4-bit each. AVC: See the bit assignment later in this section. VC1: MBZ. MPEG2: MBZ.
6	31:8	Reserved (Reserved for encoder turbo mode **IntraResidueDataSize**, when this is not 0, optional residue data are provided to the PAK; Reserved for decoder)
	7:0	**MbIntraStruct** The IntraPredAvailFlags[4:0] have already included the effect of the constrained_intra_pred_flag. See the diagram later for the definition of neighbors position around the current MB or MB pair (in MBAFF mode). 1 - IntraPredAvailFlagX, indicates the values of samples of neighbor X can be used in intra prediction for the current MB. 0 - IntraPredAvailFlagX, indicates the values of samples of neighbor X is not available for intra prediction of the current MB. IntraPredAvailFlag-A and -E can only be different from each other when constrained_intra_pred_flag is equal to 1 and mb_field_decoding_flag is equal to 1 and the value of the mb_field_decoding_flag for the macroblock pair to the left of the current macroblock is equal to 0 (which can only occur when MbaffFrameFlag is equal to 1). IntraPredAvailFlag-F is used only if ○ it is in MBAFF mode, i.e. **MbaffFrameFlag** = 1, ○ the current macroblock is of frame type**,** i.e. **MbFieldFag** = 0, and ○ the current macroblock type is Intra8x8, that is, **IntraMbFlag** = INTRA, **IntraMbMode** = INTRA_8x8, and **Transform8x8Flag** = 1. In any other cases IntraPredAvailFlag-A shall be used instead.

DWord	Bit	Description
		<table><tr><th>Bits</th><th>IntraPredAvailFlags[4:0] Definition</th></tr><tr><td>7</td><td>**IntraPredAvailFlagF - F** (Left 8th row (-1,7) neighbor)</td></tr><tr><td>6</td><td>**IntraPredAvailFlagA - A** (Left neighbor top half)</td></tr><tr><td>5</td><td>**IntraPredAvailFlagE - E** (Left neighbor bottom half)</td></tr><tr><td>4</td><td>**IntraPredAvailFlagB - B** (Top neighbor)</td></tr><tr><td>3</td><td>**IntraPredAvailFlagC - C** (Top right neighbor)</td></tr><tr><td>2</td><td>**IntraPredAvailFlagD - D** (Top left corner neighbor)</td></tr><tr><td>1:0</td><td>**ChromaIntraPredMode -** 2 bits to specify 1 of 4 chroma intra prediction mode, see the table in later section.</td></tr></table>

Inline data subfields for an Inter Macroblock

DWord	Bit	Description
4	31:24	Reserved: MBZ (DXVA Decoder)
	23:16	Reserved: MBZ (DXVA Decoder)
	15:8	**SubMbPredModes**[bit 7:0] (Sub Macroblock Prediction Mode) This field describes the prediction mode of the sub macroblocks (four 8x8 blocks). It contains four subfields each with 2-bits, corresponding to the 4 fixed size 8x8 sub macroblocks in sequential order. This field is provided for MB with sub_mb_type equal to BP_8x8 only (B_8x8 and P_8x8 as defined in DXVA) This field is derived from MbType for a non-BP_8x8 inter macroblock, and carries redundant information as MbType) Bits [1:0]: SubMbPredMode[0] - for 8x8 Block 0 Bits [3:2]: SubMbPredMode[1] - for 8x8 Block 1 Bits [5:4]: SubMbPredMode[2] - for 8x8 Block 2 Bits [7:6]: SubMbPredMode[3] - for 8x8 Block 3 Blocks of the MB is numbered as follows : 0 1 2 3 Each 2-bit value [1:0] is defined as : 00 - Pred_L0

DWord	Bit	Description
		01 - Pred_L1 10 - BiPred **For VC1:** Bits [1:0]: "00"= Y0 Forward only, "01"= Y0 Backward only, "10"= Y0 Bi direction Bits [3:2]: SubMbPredMode[1] - for 8x8 Block 1 Bits [5:4]: SubMbPredMode[2] - for 8x8 Block 2 Bits [7:6]: SubMbPredMode[3] - for 8x8 Block 3
	7:0	**SubMbShape**[bit 7:0] (Sub Macroblock Shape) This field describes the sub-block partitioning of each sub macroblocks (four 8x8 blocks). It contains four subfields each with 2-bits, corresponding to the 4 fixed size 8x8 sub macroblocks in sequential order. This field is provided for MB with sub_mb_type equal to BP_8x8 only (B_8x8 and P_8x8 as defined in DXVA) This field is forced to 0 for a non-BP_8x8 inter macroblock, and effectively carries redundant information as MbType). Bits [1:0]: SubMbShape[0] - for 8x8 Block 0 Bits [3:2]: SubMbShape[1] - for 8x8 Block 1 Bits [5:4]: SubMbShape[2] - for 8x8 Block 2 Bits [7:6]: SubMbShape[3] - for 8x8 Block 3 Blocks of the MB is numbered as follows : 0 1 2 3 Each 2-bit value [1:0] is defined as : 00 - SubMbPartWidth=8, SubMbPartHeight=8 01 - SubMbPartWidth=8, SubMbPartHeight=4 10 - SubMbPartWidth=4, SubMbPartHeight=8 11 - SubMbPartWidth=4, SubMbPartHeight=4 For VC-1, This field indicates the transformation types used for luma components, 2 bits for each 8x8.
5	31:24	**Frame Store ID L0[3]** Support up to 4 Frame store ID per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the frame Store ID into the Reference Picture List0 Table. Bit 7: **Must Be One**: (This is reserved for control fields in future extension, when reference index are

DWord	Bit	Description
		generated instead of frame store ID) 1: indicate it is in Frame store ID format. 0: indicate it is in Reference Index format. Bit 6:5: reserved MBZ Bit 4:0: Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in intel implementation)
	23:16	**Frame Store ID L0[2]** Support up to 4 Frame store ID per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the frame Store ID into the Reference Picture List0 Table. Bit 7: **Must Be One**: (This is reserved for control fields in future extension, when reference index are generated instead of frame store ID) 1: indicate it is in Frame store ID format. 0: indicate it is in Reference Index format. Bit 6:5: reserved MBZ Bit 4:0: Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in intel implementation)
	15:8	**Frame Store ID L0[1]** Support up to 4 Frame store ID per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the frame Store ID into the Reference Picture List0 Table. Bit 7: **Must Be One**: (This is reserved for control fields in future extension, when reference index are generated instead of frame store ID) 1: indicate it is in Frame store ID format. 0: indicate it is in Reference Index format. Bit 6:5: reserved MBZ Bit 4:0: Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in intel implementation).
	7:0	**Frame Store ID L0[0]** Support up to 4 Frame store ID per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the frame Store ID into the Reference Picture List0 Table. Bit 7: **Must Be One**: (This is reserved for control fields in future extension, when reference index are generated instead of frame store ID) 1: indicate it is in Frame store ID format. 0: indicate it is in Reference Index format.

DWord	Bit	Description
		Bit 6:5: reserved MBZ Bit 4:0: Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in intel implementation)
6	31:24	**Frame Store ID L1[3]** Support up to 4 Frame store ID per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the frame Store ID into the Reference Picture List0 Table. Bit 7: **Must Be One**: (This is reserved for control fields in future extension, when reference index are generated instead of frame store ID) 1: indicate it is in Frame store ID format. 0: indicate it is in Reference Index format. Bit 6:5: reserved MBZ Bit 4:0: Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in intel implementation)
	23:16	**Frame Store ID L1[2]** Support up to 4 Frame store ID per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the frame Store ID into the Reference Picture List0 Table. Bit 7: **Must Be One**: (This is reserved for control fields in future extension, when reference index are generated instead of frame store ID) 1: indicate it is in Frame store ID format. 0: indicate it is in Reference Index format. Bit 6:5: reserved MBZ Bit 4:0: Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in intel implementation)
	15:8	**Frame Store ID L1[1]** Support up to 4 Frame store ID per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the frame Store ID into the Reference Picture List0 Table. Bit 7: **Must Be One**: (This is reserved for control fields in future extension, when reference index are generated instead of frame store ID) 1: indicate it is in Frame store ID format. 0: indicate it is in Reference Index format. Bit 6:5: reserved MBZ Bit 4:0: Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in intel implementation)

DWord	Bit	Description
	7:0	**Frame Store ID L1[0]** Support up to 4 Frame store ID per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the frame Store ID into the Reference Picture List0 Table. Bit 7: **Must Be One**: (This is reserved for control fields in future extension, when reference index are generated instead of frame store ID) 1: indicate it is in Frame store ID format. 0: indicate it is in Reference Index format. Bit 6:5: reserved MBZ Bit 4:0: Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in intel implementation)

AVC Encoder PAK Commands

Each PAK Commands is composed of a command op-code DW and one or more command data DWs (inline data). The size of each command is specified as part of the op-code DW. Most of the commands have fixed size, except some are allowed to be of variable length.

There is an inherent order of executing MFC PAK commands that driver must follow.

MFC_AVC_PAK_OBJECT

PAK Object Inline Data Description

The Inline Data includes all the required MB encoding states, constitute part of the Slice Data syntax elements, MB Header syntax elements and their derivatives. It provides information for the following operations:

1. Forward and Inverse Transform
2. Forward and Inverse Quantization
3. Advanced Rate Control (QRC)
4. MB Parameter Construction (MPC)
5. CABAC/CAVLC encoding
6. Bit stream packing
7. Intra and inter-Prediction decoding loop
8. Internal error handling

These state/parameter values may subject to change on a per-MB basis, and must be provided in each MFC_AVC_PAK_OBJECT command. The values set for these variables are retained internally, until they are reset by hardware Asynchronous Reset or changed by the next MFC_AVC_PAK_OBJECT command.

The inline data has been designed to match the DXVA 2.0, with the exception of the starting byte (DW0:0-7) and the ending dword (DW7:0-31).

The Deblocker Filter Control flags (FilterInternalEdgesFlag, FilterTopMbEdgeFlag and FilterLeftMbEdgesFlag) are generated by H/W, which are depending on MbaffFrameFlag, CurrMbAddr, PicWidthInMbs and disable_deblocking_filter_idc states.

Current MB [x,y] address is not sent, it is assumed that the H/W will keep track of the MB count and current MB position internally.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>3</td><td>30</td><td>Reserved: MBZ</td></tr>
<tr><td></td><td>19</td><td>CbpDcY. This field specifies if the Luma DC sub-block is coded. Setting it to 0 will force PAK to zero out the Luma sub-block. Otherwise, whether the sub-block is coded will be determined by the quantization process.

1 - the 4x4 DC-only Luma sub-block of the Intra16x16 coded MB is present; it is still possible that all DC coefficients are zero.

0 - no 4x4 DC-only Luma sub-block is present; either not in Intra16x16 MB mode or all DC coefficients are zero.

<table><tr><th colspan="2">Programming Note</th></tr><tr><td>Context:</td><td>PAK Object Inline Data Description - CbpDcY</td></tr><tr><td colspan="2">When Reference Mb: IPCM or inferred IPCM, current mb: base mode flag = 1: TcoeffLvlPredFlag = 1 ; all bits in CbpDcY, CbpDcU, CbpDcV, Cbp4x4Y[15:0], Cbp4x4V[15:0] and Cbp4x4U[15:0] must set to 1's.</td></tr></table></td></tr>
<tr><td></td><td>18</td><td>CbpDcU. This field specifies if the Chroma Cb DC sub-block is coded. Setting it to 0 will force PAK to zero out the Luma sub-block. Otherwise, whether the sub-block is coded will be determined by the quantization process.

1 - the 2x2 DC-only Chroma Cb sub-block of all coded MB (any type) is present; it is still possible that all DC coefficients are zero.

0 - no 2x2 DC-only Chroma Cb sub-block is present; all DC coefficients are zero.

<table><tr><th colspan="2">Programming Note</th></tr><tr><td>Context:</td><td>PAK Object Inline Data Description</td></tr><tr><td colspan="2">When Reference Mb: IPCM or inferred IPCM, current mb: base mode flag = 1: TcoeffLvlPredFlag = 1 ; all bits in CbpDcY, CbpDcU, CbpDcV, Cbp4x4Y[15:0], Cbp4x4V[15:0] and Cbp4x4U[15:0] must set to 1's.</td></tr></table></td></tr>
<tr><td></td><td>17</td><td>CbpDcV. This field specifies if the Chroma Cb DC sub-block is coded. Setting it to 0 will force PAK to zero out the Luma sub-block. Otherwise, whether the sub-block is coded will be determined by the quantization process.

1 - the 2x2 DC-only Chroma Cr sub-block of all coded MB (any type) is present; it is still possible that all DC coefficients are zero.

0 - no 2x2 DC-only Chroma Cr sub-block is present; all DC coefficients are zero.</td></tr>
</table>

DWord	Bit	Description
		Programming Note **Context:** PAK Object Inline Data Description When Reference Mb: IPCM or inferred IPCM, current mb: base mode flag = 1: TcoeffLvlPredFlag = 1 ; all bits in CbpDcY, CbpDcU, CbpDcV, Cbp4x4Y[15:0], Cbp4x4V[15:0] and Cbp4x4U[15:0] must set to 1's.
	16	**Reserved: MBZ** (reserved for future use as **ExternalResidBufFlag** for turbo mode)
	15	**Transform8x8Flag** This field indicates that 8x8 transform is used for the macroblock. When it is set to 0, the current MB uses 4x4 transform. When it is set to 1, the current MB uses 8x8 transform. The transform_size_8x8_flag syntax element, if present in the output bitstream, is the same as this field. However, whether transform_szie_8x8_flag is present or not in the output bitstream depends on several other conditions. This field is only allowed to be set to 1 for two conditions: It must be 1 if **IntraMbFlag** = INTRA and **IntraMbMode** = INTRA_8x8 It may be 1 if **IntraMbFlag** = INTER and there is no sub partition size less than 8x8 Otherwise, this field must be set to 0. **Programming Note** **Context:** PAK Inline Object Data Description When SvcSliceState TcoeffLvlPredFlag=1, and AvcImgState EntropyCodingFlag is 1(CABAC), this field cannot be 1. 0: 4x4 integer transform 1: 8x8 integer transform
	14	**FieldMbFlag** This field specifies the field polarity of the current macroblock, as the mb_field_decoding_flag syntax element in AVC spec. This field specifies whether current macroblock is coded as a field or frame macroblock in MBAFF mode. It is exactly the same as FIELD_PIC_FLAG syntax element in non-MBAFF mode. 0 = Frame macroblock 1 = Field macroblock
	13	**IntraMbFlag** This field specifies whether the current macroblock is an Intra (I) macroblock. I_PCM is considered as Intra MB.

DWord	Bit	Description
		For I-picture MB (IntraPicFlag =1), this field must be set to 1. This flag must be set in consistent with the interpretation of MbType (inter or intra modes). 0: INTER (inter macroblock) 1: INTRA (intra macroblock)
	12:8	**MbType5Bits** This field is encoded to match with the best macroblock mode determined as described in the next section. It follows a unified encoding for inter and intra macroblocks according to AVC Spec.
	7	**FieldMbPolarityFlag** This field indicates the field polarity of the current macroblock. Within an MbAff frame picture, this field may be different per macroblock and is set to 1 only for the second macroblock in a MbAff pair if FieldMbFlag is set. Otherwise, it is set to 0. Within a field picture, this field is set to 1 if the current picture is the bottom field picture. Otherwise, it is set to 0. It is a constant for the whole field picture. This field is reserved and MBZ for a progressive frame picture. 0 = Current macroblock is a field macroblock from the **top** field 1 = Current macroblock is a field macroblock from the **bottom** field *Programming Note: Here bits [26:24] (MbAffFieldFlag and FiedlMbPolarityFlag) match with bits [10:8] of the Media Block Read message descriptor, simplifying the programming for message generation, as when MbAffFieldFlag is "1", kernels need to override the original "frame" surface state set for MBAFF frame picture.*
	6	MB Reserved: Inter MB converted to IPCM. This field is for HW purposes only. SW should not use it.
	5:4	**IntraMbMode** This field is provided to carry information partially overlapped with MbType. This field is only valid if **IntraMbFlag** = INTRA, otherwise, it is ignored by hardware..
	3	Reserved: MBZ
	2	**SkipMbFlag** By setting it to 1, this field forces an inter macroblock to be encoded as a skipped macroblock. It is equivalent to mb_skip_flag in AVS spec, indicating that a macroblock is inferred as a P_Skip (or B_Skip) in a P Slice (or B Slice). Hardware honors input MVs for motion prediction and forces CBP to zero. By setting it to 0, an inter macroblock will be coded as a normal inter macroblock. The macroblock may still be coded as a skipped macroblock, according to the macroblock type conversion rules

DWord	Bit	Description
		described in the later sub sections. This field can only be set to 1 for certain values of MbType. See details later. This field is only valid for an inter macroblock. For intra MB (bit 13 of this DW set to one), this bit must be set to zero. 0 = not a skipped macroblock 1 = is coded as a skipped macroblock
	1:0	**InterMbMode** This field is provided to carry redundant information as that encoded in MbType. This field is only valid if **IntraMbFlag** =0, otherwise, it is ignored by hardware.
4	31:16	Cbp4x4Y[bit 15:0] (Coded Block Pattern Y) For 4x4 sub-block (when Transform8x8flag = 0 or in intra16x16) : 16-bit cbp, one bit for each 4x4 Luma sub-block (not including the DC 4x4 Luma block in intra16x16) in a MB. The 4x4 Luma sub-blocks are numbered as blk0 1 4 5 bit15 14 11 10 lk2 3 6 7 bit13 12 9 8 blk8 9 12 13 bit7 6 3 2 blk10 11 14 15 bit5 4 1 0 The cbpY bit assignment is cbpY bit [15 - X] for sub-block_num X. For 8x8 block (when Transform8x8flag = 1) Only the lower 4 bits [3:0] are valid; the remaining upper bits [15:4] are ignored. The 8x8 Luma blocks are numbered as blk0 1 bit3 2 blk2 3 bit1 0 The cbpY bit assignment is cbpY bit [3 - X] for block_num X. 0 in a bit - indicates the corresponding 8x8 block or 4x4 sub-block is not present (because all coefficient values are zero), or force to zero for PAK. 1 in a bit - indicates the corresponding 8x8 block or 4x4 sub-block is present (although it is still possible to have all its coefficients be zero - bad coding).

DWord	Bit	Description
		Programming Note **Context:** PAK Object Inline Data Description When Reference Mb: IPCM or inferred IPCM, current mb: base mode flag = 1: TcoeffLvlPredFlag = 1 ; all bits in CbpDcY, CbpDcU, CbpDcV, Cbp4x4Y[15:0], Cbp4x4V[15:0] and Cbp4x4U[15:0] must set to 1's.
4	15:8	MbYCnt (Vertical Origin). This field specifies the vertical origin of current macroblock in the destination picture in units of macroblocks. Format = U8 in unit of macroblock.
	7:0	MbXCnt (Horizontal Origin). This field specifies the horizontal origin of current macroblock in the destination picture in units of macroblocks. Format = U8 in unit of macroblock.
5	31:16	Cbp4x4V (Coded Block Pattern Cr) Only the lower 4 bits [3:0] are valid for 4:2:0. The 4x4 Cr sub-blocks are numbered as blk0 1 bit3 2 blk2 3 bit1 0 The cbpCr bit assignment is cbpCr bit [3 - X] for sub-block_num X. 0 in a bit - indicates the corresponding 4x4 sub-block is not present (because all coefficient values are zero), or force to zero for PAK. 1 in a bit - indicates the corresponding 4x4 sub-block is present (although it is still possible to have all its coefficients be zero - bad coding). For monochrome, this field is ignored. For 4.2.2, [23:16] for U(Cb), and [31:24] ignored. For 4.4.4, the definition is the same as for luma component: 1bit per 4x4 block. **Programming Note** **Context:** PAK Object Inline Data Description When Reference Mb: IPCM or inferred IPCM, current mb: base mode flag = 1: TcoeffLvlPredFlag = 1 ; all bits in CbpDcY, CbpDcU, CbpDcV, Cbp4x4Y[15:0], Cbp4x4V[15:0] and Cbp4x4U[15:0] must set to 1's.
5	15:0	Cbp4x4U (Coded Block Pattern Cb) Only the lower 4 bits [3:0] are valid for 4:2:0. The 4x4 Cb sub-blocks are numbered as blk0 1 bit3 2 blk2 3 bit1 0

DWord	Bit	Description
		The cbpCb bit assignment is cbpCb bit [3 - X] for sub-block_num X. 0 in a bit - indicates the corresponding 4x4 sub-block is not present (because all coefficient values are zero), or force to zero for PAK. 1 in a bit - indicates the corresponding 4x4 sub-block is present (although it is still possible to have all its coefficients be zero - bad coding). For monochrome, this field is ignored. For 4.2.2, [7:0] for U(Cb), and [15:8] ignored. For 4.4.4, the definition is the same as for luma component: 1bit per 4x4 block. **Programming Note** **Context:** PAK Object Inline Data Description When Reference Mb: IPCM or inferred IPCM, current mb: base mode flag = 1: TcoeffLvlPredFlag = 1 ; all bits in CbpDcY, CbpDcU, CbpDcV, Cbp4x4Y[15:0], Cbp4x4V[15:0] and Cbp4x4U[15:0] must set to 1's.
6	31:28	**Skip8x8Pattern** This field indicates whether each of the four 8x8 sub macroblocks is using the predicted MVs and will not be explicitly coded in the bitstream (the sub macroblock will be coded as direct mode). It contains four 1-bit subfields, corresponding to the 4 sub macroblocks in sequential order. The whole macroblock may be actually coded as B_Direct_16x16 or B_Skip, according to the macroblock type conversion rules described in a later sub section. This field is only valid for a B slice. It is ignored by hardware for a P slice. Hardware also ignores this field for an intra macroblock. 0 in a bit - Corresponding MVs are sent in the bitstream 1 in a bit - Corresponding MVs are not sent in the bitstream
	27	**EnableCoeffClamp** 1 = the magnitude of coefficients of the current MB will be clamped based on the clamping matrix after quantization 0 = no clamping
	26	**LastMbFlag** 1 - the current MB is the last MB in the current Slice 0 - the current MB is not the last MB in the current Slice - Reserved MBZ.
	25	**SkipMbConvDisable** This is a per-MB level control to enable and disable skip conversion. This field is ORed with SkipConvDisable field. This field is only valid for a P or B slice. It must be zero for other slice types. Rules are provided in Section Macroblock Type Conversion Rules.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td></td><td></td><td>0 - Enable skip type conversion for the current macroblock
1 - Disable skip type conversion for the current macroblock</td></tr>
<tr><td></td><td>24</td><td>Reserved MBZ.</td></tr>
<tr><td></td><td>23:16</td><td>Reserved. Ignored by HW, this field will be re-derived internally.
(was QpPrimeV. For 8-bit pixel data, QpCr is the same as QpPrimeCr, and it takes on a value in the range of 0 to 51, positive integer.)</td></tr>
<tr><td></td><td>15:8</td><td>Reserved. Ignored by HW, this field will be re-derived internally.
(Was QpPrimeU. For 8-bit pixel data, QpCb is the same as QpPrimeCb, and it takes on a value in the range of 0 to 51, positive integer.)</td></tr>
<tr><td></td><td>7:0</td><td>QpPrimeY
This is the per-MB QP value specified for the current MB.
For 8-bit pixel data, QpY is the same as QpPrimeY, and it takes on a value in the range of 0 to 51, positive integer.
<table><tr><th colspan="2">Programming Note</th></tr><tr><td>Context:</td><td>PAK Object Inline Data Description</td></tr><tr><td colspan="2">This value may differ from the actual codes, when HW QRC is on</td></tr></table></td></tr>
<tr><td>7 .. 9</td><td>31:0
Each</td><td>For intra macroblocks, definition of these fields are specified in Inline data subfields for an Intra Macroblock.
For inter macroblocks, definition of these fields are specified in Inline data subfields for an Inter Macroblock.</td></tr>
<tr><td>10</td><td>31:24</td><td>MaxSizeInWord
PAK should not exceed this budget accumulatively, otherwise it will trickle the PANIC mode.</td></tr>
<tr><td></td><td>23:16</td><td>TargetSizeInWord
PAK should use this budget accumulatively to decide if it needs to limit the number of non-zero coefficients.</td></tr>
<tr><td></td><td>15:0</td><td>Lambda_Or_RoundingOffset
When TQEnb=1, in MFX_AVC_IMG_STATE, this 16-bit unsigned value multiplied by 2 is used as a lambda for the rate-distortion cost estimation in Trellis quantization (TQ). If the upper 4 bits are all set to 1 (0xFXXX), TQ is disabled and the regular quantizer is used. Thus, the valid range is 0~0xEFFF. When TQ is enabled per MB, the TQR in MFC_AVC_IMG_STATE is used for rounding quantization coefficients.
When TQEnb=0 or the upper 4 bits are all set to 1, the lower 4-bit value indicates the rounding precision (offset) for the regular quantizer. The values ranging 0001b ~ 0111 are reserved.</td></tr>
</table>

DWord	Bit	Description
		<table><tr><th>Value</th><th>Name</th></tr><tr><td>0000b</td><td>**RoundInterEnable, RoundInter, RoundIntraEnable,** and **RoundIntra** defined in MFC_AVC_SLICE_STATE are used as rounding precision.</td></tr><tr><td>1000b</td><td>+1/16</td></tr><tr><td>1001b</td><td>+2/16</td></tr><tr><td>1010b</td><td>+3/16</td></tr><tr><td>1011b</td><td>+4/16</td></tr><tr><td>1100b</td><td>+5/16</td></tr><tr><td>1101b</td><td>+6/16</td></tr><tr><td>1110b</td><td>+7/16</td></tr><tr><td>1111b</td><td>+8/16</td></tr></table>**Format: U16**

VDEnc Mode Inline data (For PAK Standalone validation)

12	31:16	**MV Y** The value of the y component of this motion vector for FWD block 0.
	15:0	**MV X** The value of the x component of this motion vector for FWD block 0.
13	31:16	**MV Y** The value of the y component of this motion vector for FWD block 1.
	15:0	**MV X** The value of the x component of this motion vector for FWD block 1.
14	31:16	**MV Y** The value of the y component of this motion vector for FWD block 2.
	15:0	**MV X** The value of the x component of this motion vector for FWD block 2.
15	31:16	**MV Y** The value of the y component of this motion vector for FWD block 3.
	15:0	**MV X** The value of the x component of this motion vector for FWD block 3.

16	31:16	**MV Y** The value of the y component of this motion vector for BWD block 0.
	15:0	**MV X** The value of the x component of this motion vector for BWD block 0.
17	31:16	**MV Y** The value of the y component of this motion vector for BWD block 1.
	15:0	**MV X** The value of the x component of this motion vector for BWD block 1.
18	31:16	**MV Y** The value of the y component of this motion vector for BWD block 2.
	15:0	**MV X** The value of the x component of this motion vector for BWD block 2.
19	31:16	**MV Y** The value of the y component of this motion vector for BWD block 3.
	15:0	**MV X** The value of the x component of this motion vector for BWD block 3.
20	31:16	**LumaIntraMode[1]** Specifies the Luma Intra Prediction mode for four 4x4 sub-block of a MB, 4-bit each.
	15:0	**LumaIntraMode[0]** Specifies the Luma Intra Prediction mode for four 4x4 sub-block, four 8x8 block or one intra16x16 of a MB. 4-bit per 4x4 sub-block (Transform8x8Flag=0, Mbtype=0 and intraMbFlag=1) or 8x8 block (Transform8x8Flag=1, Mbtype=0, MbFlag=1), since there are 9 intra modes. 4-bit for intra16x16 MB (Transform8x8Flag=0, Mbtype=1 to 24 and intraMbFlag=1), but only the LSBit[1:0] is valid, since there are only 4 intra modes.
21	31:16	**LumaIntraMode[3]** Specifies the Luma Intra Prediction mode for four 4x4 sub-block of a MB, 4-bit each.
	15:0	**LumaIntraMode[2]**

		Specifies the Luma Intra Prediction mode for four 4x4 sub-block of a MB, 4-bit each.
22	31:16	**Minimal Distortion** This field contains the overall distortion for the source block associated with the winning MbType, which could be one of intra or inter modes.
	15:0	**SkipRawDistortion** This field contains Skip Raw Distortion which may be used by software to further refine the skip decision.
23	31:16	**InterRawDistortion** This field provides the Inter Raw Distortion (Sad/Haar) for the current macroblock.
	15:0	**BestIntraRawDistortion** This field contains the best IntraRawDistortion (Sad/Haar) for the current macroblock. The IntraMBMode will indicate if this is a16x16/8x8/4x4 distortion.

Inline data for LumaIntraMode

	0 or 1	0	0	1	1
ExtendedForm	Intra4x4	Intra8x8	Intra16x16	Intra8x8	Intra16x16
DW4 - 31:28	Block 7	-	-	-	Block 0
DW4 - 27:24	Block 6	-	-	-	Block 0
DW4 - 23:20	Block 5	-	-	-	Block 0
DW4 - 19:16	Block 4	-	-	-	Block 0
DW4- 15:12	Block 3	-	-	-	Block 0
DW4 - 11:8	Block 2	-	-	-	Block 0
DW4 - 7:4	Block 1	-	-	-	Block 0
DW4 - 3:0	Block 0	-	-	-	Block 0
DW5 - 31:28	Block 15	-	-	-	Block 0
DW5 - 27:24	Block 14	-	-	-	Block 0
DW5 - 23:20	Block 13	-	-	-	Block 0
DW5 - 19:16	Block 12	-	-	-	Block 0
DW5 - 15:12	Block 11	-	-	-	Block 0
DW5- 11:8	Block 10	-	-	-	Block 0
DW5 - 7:4	Block 9	-	-	-	Block 0
DW5 - 3:0	Block 8	-	-	-	Block 0

vctrl_pred_mode[63:0]	(vctrl_it_lumaintrapredmode3[15:0] & vctrl_it_lumaintrapredmode2[15:0] & vctrl_it_lumaintrapredmode1[15:0] & vctrl_it_lumaintrapredmode0[15:0]) : vctrl_pred_mode_noextend[63:0]
vctrl_pred_mode_noextend[63:0]	(vctrl_INTRA_vld_16x16mode & vctrl_it_Transform8x8Flag) ? vctrl_pred_mode_noextend_4x4[63:0] : vctrl_pred_mode_noextend_16x16[63:0] : vctrl_pred_mode_noextend_8x8[63:0] : vctrl_pred_mode_noextend_4x4[63:0]
vctrl_pred_mode_noextend_16x16[63:0]	vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0] & vctrl_it_lumaintrapredmode0[3:0]
vctrl_pred_mode_noextend_8x8[63:0]	"h000" & vctrl_it_lumaintrapredmode0[15:12] & "h000" & vctrl_it_lumaintrapredmode0[11:8] & "h000" & vctrl_it_lumaintrapredmode0[7:4] & "h000" & vctrl_it_lumaintrapredmode0[3:0]
vctrl_pred_mode_noextend_4x4[63:0]	vctrl_it_lumaintrapredmode3[15:0] & vctrl_it_lumaintrapredmode2[15:0] & vctrl_it_lumaintrapredmode1[15:0] & vctrl_it_lumaintrapredmode0[15:0]

Inline data for RefPicSelect

	0	0	0	0 or 1	1	1	1
ExtendedForm	16x16	16x8	8x16	8x8	16x16	16x8	8x16
DW8 - 31:24	-	-	-	L0 blk3	L0 blk0	-	L0 blk1
DW8 - 23:16	-	-	-	L0 blk2	L0 blk0	-	L0 blk0
DW8 - 15:8	-	L0 blk1	L0 blk1	L0 blk1	L0 blk0	-	L0 blk1
DW8 - 7:0	L0 blk0	L0 blk0	L0 blk0	L0 blk0	L0 blk0	-	L0 blk0
DW9 - 31:24	-	-	-	L1 blk3	L1 blk0	-	L1 blk1
DW9 - 23:16	-	-	-	L1 blk2	L1 blk0	-	L1 blk0
DW9 - 15:8	-	L1 blk1	L1 blk1	L1 blk1	L1 blk0	-	L1 blk1
DW9 - 7:0	L1 blk0	L1 blk0	L1 blk0	L1 blk0	L1 blk0	-	L1 blk0

The inline data content of Dwords 4 to 6 is defined either for intra prediction or for inter prediction, but not both.

Inline data subfields for an Intra Macroblock

Dword	Bit	Description
7	31:16	**LumaIntraMode[1]** Specifies the Luma Intra Prediction mode for four 4x4 sub-block of a MB, 4-bit each. See the bit assignment table later in this section.
	15:0	**LumaIntraMode[0]** Specifies the Luma Intra Prediction mode for four 4x4 sub-block, four 8x8 block or one intra16x16 of a MB. 4-bit per 4x4 sub-block (Transform8x8Flag=0, Mbtype=0 and intraMbFlag=1) or 8x8 block (Transform8x8Flag=1, Mbtype=0, MbFlag=1), since there are 9 intra modes. 4-bit for intra16x16 MB (Transform8x8Flag=0, Mbtype=1 to 24 and intraMbFlag=1), but only the LSBit[1:0] is valid, since there are only 4 intra modes. See the bit assignment table later in this section.
8	31:16	**LumaIntraMode[3]** Specifies the Luma Intra Prediction mode for four 4x4 sub-block of a MB, 4-bit each. See the bit assignment table later in this section.
	15:0	**LumaIntraMode[2]** Specifies the Luma Intra Prediction mode for four 4x4 sub-block of a MB, 4-bit each. See the bit assignment later in this section.

<table>
<tr><th>Dword</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="2">9</td><td>31:8</td><td>Reserved: MBZ
(Reserved for encocder turbo mode IntraResidueDataSize, when this is not 0, optional residue data are provided to the PAK; Reserved for decoder)</td></tr>
<tr><td>7:0</td><td>IntraStruct

This field contains 6 bits for IntraPredAvailFlags[5:0] and 2 bits for ChromaIntraPredMode. The IntraPredAvailFlags[4:0] (the lower 5 bits) have already included the effect of the constrained_intra_pred_flag. See the diagram later for the definition of neighbor position around the current MB or MB pair (in MBAFF mode).

1 - IntraPredAvailFlagY, indicates the values of samples of neighbor Y can be used in intra prediction for the current MB.

0 - IntraPredAvailFlagY, indicates the values of samples of neighbor Y is not available for intra prediction of the current MB.

IntraPredAvailFlag-A and -E can only be different from each other when constrained_intra_pred_flag is equal to 1 and mb_field_decoding_flag is equal to 1 and the value of the mb_field_decoding_flag for the macroblock pair to the left of the current macroblock is equal to 0 (which can only occur when MbaffFrameFlag is equal to 1).

IntraPredAvailFlag-F is used only if

• It is in MBAFF mode, that is, MbaffFrameFlag = 1
• The current macroblock is of frame type, that is, MbFieldFag = 0
• The current macroblock type is Intra8x8, that is, IntraMbFlag = INTRA, IntraMbMode = INTRA_8x8, and Transform8x8Flag = 1

In any other cases IntraPredAvailFlag-A shall be used instead.

<table>
<tr><th>Bits</th><th>IntraPredAvailFlags Definition</th></tr>
<tr><td>7</td><td>IntraPredAvailFlagF - F (Left 8th row (-1,7) neighbor)</td></tr>
<tr><td>6</td><td>IntraPredAvailFlagA - A (Left neighbor top half)</td></tr>
<tr><td>5</td><td>IntraPredAvailFlagE - E (Left neighbor bottom half)</td></tr>
<tr><td>4</td><td>IntraPredAvailFlagB - B (Top neighbor)</td></tr>
<tr><td>3</td><td>IntraPredAvailFlagC - C (Top right neighbor)</td></tr>
<tr><td>2</td><td>IntraPredAvailFlagD - D (Top left corner neighbor)</td></tr>
<tr><td>1:0</td><td>ChromaIntraPredMode - 2 bits to specify 1 of 4 chroma intra prediction modes, see the table in later section.</td></tr>
</table>
</td></tr>
</table>

Inline data subfields for an Inter Macroblock

DWord	Bit	Description
7	31:16	**Reserved: MBZ**
	15:8	**SubMbPredMode (Sub-Macroblock Prediction Mode):** If **InterMbMode** is INTER8x8, this field describes the prediction mode of the sub-partitions in the four 8x8 sub-macroblock. It contains four subfields each with 2-bits, corresponding to the four 8x8 sub-macroblocks in sequential order. This field is derived from sub_mb_type for a BP_8x8 macroblock. This field is derived from **MbType** for a non-BP_8x8 inter macroblock, and carries redundant information as **MbType**). If **InterMbMode** is INTER16x16, INTER16x8 or INTER8x16, this field carries the prediction modes of the sub macroblock (one 16x16, two 16x8 or two 8x16). The unused bits are set to zero. Bits [1:0]: SubMbPredMode[0] Bits [3:2]: SubMbPredMode[1] Bits [5:4]: SubMbPredMode[2] Bits [7:6]: SubMbPredMode[3]
	7:0	**SubMbShape** (Sub Macroblock Shape) This field describes the sub-block partitioning of each sub macroblocks (four 8x8 blocks). It contains four subfields each with 2-bits, corresponding to the 4 fixed size 8x8 sub macroblocks in sequential order. This field is provided for MB with sub_mb_type equal to BP_8x8 only (B_8x8 and P_8x8 as defined in DXVA). Otherwise, this field is ignored by hardware Bits [1:0]: SubMbShape[0] - for 8x8 Block 0 Bits [3:2]: SubMbShape[1] - for 8x8 Block 1 Bits [5:4]: SubMbShape[2] - for 8x8 Block 2 Bits [7:6]: SubMbShape[3] - for 8x8 Block 3 Blocks of the MB is numbered as follows : 01 23 Each 2-bit value [1:0] is defined as : 00 - SubMbPartWidth=8, SubMbPartHeight=8 01 - SubMbPartWidth=8, SubMbPartHeight=4 10 - SubMbPartWidth=4, SubMbPartHeight=8 11 - SubMbPartWidth=4, SubMbPartHeight=4
8	31:24	**RefPicSelect[0][3]**

DWord	Bit	Description
		Support up to 4 reference pictures per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List0 Table.
	23:16	**RefPicSelect[0][2]** Support up to 4 reference pictures per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List0 Table.
	15:8	**RefPicSelect[0][1]** Support up to 4 reference pictures per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List0 Table.
	7:0	**RefPicSelect[0][0]** Support up to 4 reference pictures per L0 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List0 Table.
9	31:24	**RefPicSelect[1] [3]** Support up to 4 reference pictures per L1 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List1 Table. For P- picture these bits must be set to zero.
	23:16	**RefPicSelect[1][2]** Support up to 4 reference pictures per L1 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List1 Table. For P- picture these bits must be set to zero.
	15:8	**RefPicSelect[1][1]** Support up to 4 reference pictures per L1 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List1 Table. For P- picture these bits must be set to zero.
	7:0	**RefPicSelect[1][0]** Support up to 4 reference pictures per L1 direction, one per MB partition, if exists. See details in later section. This field specifies the reference index into the Reference Picture List1 Table. For P- picture these bits must be set to zero.

Luma Intra Prediction Modes

Luma Intra Prediction Modes (LumaIntraPredModes) is defined in Definition of LumaIntraPredModes. It is further categorized as Intra16x16PredMode, Intra8x8PredMode and Intra4x4PredMode, operating on 16x16, 8x8 and 4x4 block sizes, respectively. illustrates the intra prediction directions geometrically for

the Intra4x4 prediction. When a macroblock is subdivided, the intra prediction is performed for the subdivision in a predetermined order. For example, Numbers of Block4x4 in a 16x16 region shows the block order for Intra4x4 prediction, and Numbers of Block4x4 in an 8x8 region or numbers of Block8x8 in a 16x16 region shows the block order of Block8x8 in a 16x16 region or Block4x4 in an 8x8 region.

Definition of LumaIntraPredModes

LumaIntraPredModes [index]		Intra16x16PredMode	Intra8x8PredMode	Intra4x4PredMode
Index	Bit	MbType = [1...24] Transform8x8Flag = 0	MbType = 0 Transform8x8Flag = 1	MbType = 0 Transform8x8Flag = 0
0	15:12	MBZ	**Block8x8 3**	**Block4x4 3 (0_0)**
	11:8	MBZ	**Block8x8 2**	**Block4x4 2 (0_1)**
	7:4	MBZ	**Block8x8 1**	**Block4x4 1 (0_2)**
	3:0	**Block16x16**	**Block8x8 0**	**Block4x4 0 (0_3)**
1	15:12	MBZ	MBZ	**Block4x4 7 (1_0)**
	11:8	MBZ	MBZ	**Block4x4 6 (1_1)**
	7:4	MBZ	MBZ	**Block4x4 5 (1_2)**
	3:0	MBZ	MBZ	**Block4x4 4 (1_3)**
2	15:12	MBZ	MBZ	**Block4x4 11 (2_0)**
	11:8	MBZ	MBZ	**Block4x4 10 (2_1)**
	7:4	MBZ	MBZ	**Block4x4 9 (2 2)**
	3:0	MBZ	MBZ	**Block4x4 8 (2_3)**
3	15:12	MBZ	MBZ	**Block4x4 15 (3_0)**
	11:8	MBZ	MBZ	**Block4x4 14 (3_1)**
	7:4	MBZ	MBZ	**Block4x4 13 (3_2)**
	3:0	MBZ	MBZ	**Block4x4 12 (3_3)**

Definition of Intra16x16PredMode

Intra16x16PredMode	Description
0	**Intra_16x16_Vertical**
1	**Intra_16x16_Horizontal**
2	**Intra_16x16_DC**
3	**Intra_16x16_Plane**
4 - 15	Reserved

Definition of Intra8x8PredMode

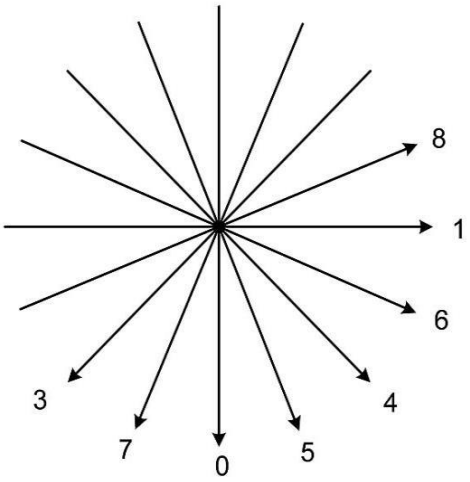
Intra8x8PredMode	Description
0	**Intra_8x8_Vertical**
1	**Intra_8x8_Horizontal**
2	**Intra_8x8_DC**
3	**Intra_8x8_Diagonal_Down_Left**
4	**Intra_8x8_Diagonal_Down_Right**
5	**Intra_8x8_Vertical_Right**
6	**Intra_8x8_Horizontal_Down**
7	**Intra_8x8_Vertical_Left**
8	**Intra_8x8_Horizontal_Up**
9 - 15	Reserved

Definition of Intra4x4PredMode

Intra4x4PredMode	Description
0	**Intra_4x4_Vertical**
1	**Intra_4x4_Horizontal**
2	**Intra_4x4_DC**

Intra4x4PredMode	Description
3	**Intra_4x4_Diagonal_Down_Left**
4	**Intra_4x4_Diagonal_Down_Right**
5	**Intra_4x4_Vertical_Right**
6	**Intra_4x4_Horizontal_Down**
7	**Intra_4x4_Vertical_Left**
8	**Intra_4x4_Horizontal_Up**
9 - 15	Reserved

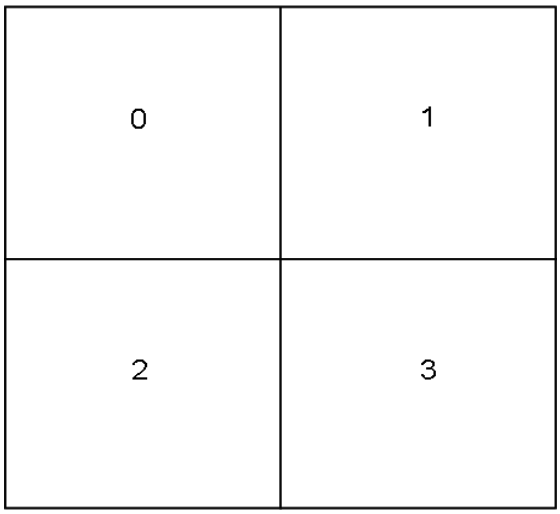
Intra_4x4 prediction mode directions



Numbers of Block4x4 in a 16x16 region

0	1	4	5
2	3	6	7
8	9	12	13
10	11	14	15

Numbers of Block4x4 in an 8x8 region or numbers of Block8x8 in a 16x16 region



Definition of Chroma Intra Prediction Mode

ChromaIntraPredMode (intra_chroma_pred_mode)	Name of intra_chroma_pred_mode
0	Intra_Chroma_DC (prediction mode)
1	Intra_Chroma_Horizontal (prediction mode)
2	Intra_Chroma_Vertical (prediction mode)
3	Intra_Chroma_Plane (prediction mode)

Reference Indices Defined for Each MB Partition Type and Bit Assignment

	frame/field MB/Picture				
MB partitioning	16x16	16x8	8x16	8x8	
RefIdxL0/1[0]	blk0	blk0	blk0	blk0	Bit 7:0
RefIdxL0/1[1]	x	blk1	blk1	blk1	Bit 15:8
RefIdxL0/1[2]	x	x	x	blk2	Bit 23:16
RefIdxL0/1[3]	x	x	x	blk3	Bit 31:24

MB Neighbor Availability in Intra-Prediction Modes (IntraPredAvailFlags)

Current MB is labelled as X. For non-MBAFF mode, 4 neighbors, A, B, C, D, are depicted in the following picture and are defined as the following.

- MB D: top left neighbor of current MB X
- MB C: top right neighbor of current MB X
- MB B: top neighbor of current MB X
- MB A: left neighbor of the current MB X

mbAddrD D (top-left)	**mbAddrB** B (top)	**mbAddrC** C (top-right)
mbAddrA A (left)	**CurrMbAddrX** X	N/A
N/A	N/A	N/A

For MBAFF mode, the current MB is labelled as X0 or X1, 4 neighbor pairs, A0/A1, B0/B1, C0/C1, D0/D1, are depicted in the following picture and are defined as the following.

- MB D0: first MB of top left neighbor MB pair of current MB pair X0/X1
- MB D1: second MB of top left neighbor MB pair of current MB pair X0/X1
- MB C0: first MB of top right neighbor MB pair of current MB pair X0/X1
- MB C1: second MB of top right neighbor MB pair of current MB pair X0/X1
- MB B0: first MB of top neighbor MB pair of current MB pari X0/X1
- MB B1: second MB of top neighbor MB pair of current MB pari X0/X1
- MB A0: first MB of left neighbor MB pair of the current MB pair X0/X1
- MB A1: second MB of left neighbor MB pair of the current MB pair X0/X1

mbAddrD D0	**mbAddrB** B0	**mbAddrC** C0
mbAddrD+1 D1	**mbAddrB+1** B1	**mbAddrC+1** C1
mbAddrA A0	**CurrMbAddrX** X0 or	N/A
mbAddrA+1 A1	**CurrMbAddrX** X1	N/A

For a given macroblock X (or X0/X1), the 6 neighbor availability signals, namely, A, B, C, D, E, F, are defined as the following.

- IntraPredAvailFlagF - F (Single neighbor pixel at the left 8th row (-1,7)

- IntraPredAvailFlagA - A (Left neighbor top half pixel group)
- IntraPredAvailFlagE - E (Left neighbor bottom half pixel group)
- IntraPredAvailFlagB - B (Top neighbor pixel group)
- IntraPredAvailFlagC - C (Top right neighbor pixel group)
- IntraPredAvailFlagD - D (Top left corner neighbor pixel)

The following table depicts the generation of IntraPredAvailFlags[5:0] signals in a condensed form. It should note that for most cases only one input neighbor signal is assigned for each condition. The exception is in the four places for deriving left neighbor A and E where the neighbor is only available if left neighbors (A0 and A1) are both available (A0&A1). Also note that F takes output value very similar to that for A except the two "AND" conditions, where F is assigned to A1 instead of (A0&A1).

Table: Definition of intra-prediction neighbor availability calculation in MBAFF mode

Output =>		D		B		C		A		E		F	
Current X \ Neighbor Y		Y-Frame	Y-Field	Y-Frame	Y-Field	Y-Frame	Y-Field	Y-Frame	Y-Field	Y-Frame	Y-Field	Y-Frame	Y-Field
X_0 (Top)	X-Frame	D_1	D_1	B_1	B_1	C_1	C_1	A_0	A_0 & A_1	A_0	A_0 & A_1	A_0	A_1
	X-Field	D_1	D_0	B_1	B_0	C_1	C_0	A_0	A_0	A_1	A_0	A_0	A_0
X_1 (Bottom)	X-Frame	A_0	A_1	X_0	N/A	0	0	A_1	A_0 & A_1	A_1	A_0 & A_1	A_1	A_1
	X-Field	D_1	D_1	B_1	B_1	C_1	C_1	A_0	A_1	A_1	A_1	A_0	A_1

In the table below, Definition of intra-prediction neighbor availability calculation in MBAFF mode, *X-Frame* or *X-Field* indicates the frame/field mode of the current MB; and *Y-Frame* or *Y-Field* indicates the corresponding neighbor MB for the given neighbor location, being upper left (D) or left (A) for example. Therefore, "Y-" takes the selected neighbor MB name as in the output cell in the same column. For example, for output D, if X1 is a frame MB, Y = A, if X1 is a field MB, Y = D.

For non-MBAFF mode, as A0=A1, B0=B1, C0=C1 and D0=D1, the neighbor assignment is degenerated into the following simple table. Here, E is assigned to the same as A and F is forced to 0.

Table: Definition of intra-prediction neighbor availability calculation in non-MBAFF mode

Output =>	D	B	C	A	E	F
X	D0	B0	C0	A0	A0	0

To further explain the neighbor assignment rules in Definition of intra-prediction neighbor availability calculation in MBAFF mode, the following table provides description for each condition. Please note that this table is **informative** as it provides redundant information as in Definition of intra-prediction neighbor availability calculation in MBAFF mode.

Table: Detailed explanation of intra-prediction neighbor availability calculation in MBAFF mode

Current MB	Current MB Field	Neighbor MB Field	Neighbor MB Select (Y=?)	Neighbor Avail Result (OUTPUT)	Description
			D		
X0 (Top)	X-Frame	Y-Frame	D	D1	Top Frame MB uses [-1,-1] = D_31, thus D1 only, regardless D frame or field pair
	X-Frame	Y-Field	D	D1	
	X-Field	Y-Frame	D	D1	Top Field MB uses [-1,-2] = D_30, thus if D is frame pair, takes D1 (D1_14 pixel), and if D is field pair, takes D0 (D0_15 pixel)
	X-Field	Y-Field	D	D0	
X1 (Bottom)	X-Frame	Y-Frame	A	A0	Bottom Frame MB uses [-1,15] = A_15, thus A0 (A0_15 pixel) if A is a frame pair, or A1 (A1_7 pixel), if A is a field pair
	X-Frame	Y-Field	A	A1	
	X-Field	Y-Frame	D	D1	Bottom Field MB uses [-1,-1] = D_31, thus D1 only, regardless D frame or field pair
	X-Field	Y-Field	D	D1	
			B		
X0 (Top)	X-Frame	Y-Frame	B	B1	Top Frame MB uses [0...15,-1] = B_31, thus B1 only, regardless B frame or field pair
	X-Frame	Y-Field	B	B1	
	X-Field	Y-Frame	B	B1	Top Field MB uses [0...15,-2] = B_30, thus if B is frame pair, takes B1 (B1_14 row), and if B is field pair, takes B0 (B0_15 row)
	X-Field	Y-Field	B	B0	
X1 (Bottom)	X-Frame	Y-Frame	X	X0	Bottom Frame MB uses [0...15,15], thus X0 (X0_15 row)
	X-Frame	Y-Field	X	n/a	**Note:** X0 and X1 must have the same field type, this row is n/a.
	X-Field	Y-Frame	B	B1	Bottom Field MB uses [0...15,-1] = B_31, thus B1 only, regardless B frame or field pair
	X-Field	Y-Field	B	B1	
			C		
X0 (Top)	X-Frame	Y-Frame	C	C1	Top Frame MB uses [16...23,-1] = C_31, thus C1 only, regardless C frame or field pair
	X-Frame	Y-Field	C	C1	
	X-Field	Y-Frame	C	C1	Top Field MB uses [16...23,-2] = C_30, thus if C is frame pair, takes C1 (C1_14 row), and if C is field pair, takes C0 (C0_15 row)
	X-Field	Y-Field	C	C0	
X1 (Bottom)	X-Frame	Y-Frame	n/a	0	Bottom Frame MB doesn't have left-top neighbor by definition, thus forced to 0
	X-Frame	Y-Field	n/a	0	
	X-Field	Y-Frame	C	C1	Bottom Field MB uses [16...23,-1] = C_31, thus C1 only, regardless C frame or field pair
	X-Field	Y-Field	C	C1	
			A		
X0 (Top)	X-Frame	Y-Frame	A	A0	First Half of Top Frame MB uses [-1,0...7], thus A0 if A is a frame pair; but is only avail if both A0 and A1 are avail if A is a field pair
	X-Frame	Y-Field	A	A0&A1	

Current MB	Current MB Field	Neighbor MB Field	Neighbor MB Select (Y=?)	Neighbor Avail Result (OUTPUT)	Description
			D		
					due to the mix
	X-Field	Y-Frame	A	A0	First Half of Top Field MB uses [-1,0..2..4..14], thus take A0 (if A is frame pair, takes A0 even lines, and if A is field pair, takes A0 first half)
	X-Field	Y-Field	A	A0	
X1 (Bottom)	X-Frame	Y-Frame	A	A1	First Half of Bottom Frame MB uses [-1,16...23], thus A1 if A is a frame pair; but is only avail if both A0 and A1 are avail if A is a field pair due to the mix
	X-Frame	Y-Field	A	A0&A1	
	X-Field	Y-Frame	A	A0	First Half of Bottom Field MB uses [-1,1..3..15], thus take A0 (if A is frame pair, takes A0 odd lines, and if A is field pair, takes A1 first half)
	X-Field	Y-Field	A	A1	
			E		
X0 (Top)	X-Frame	Y-Frame	A	A0	Second Half of Top Frame MB uses [-1,8...15], thus A0 if A is a frame pair; but is only avail if both A0 and A1 are avail if A is a field pair due to the mix
	X-Frame	Y-Field	A	A0&A1	
	X-Field	Y-Frame	A	A1	Second Half of Top Field MB uses [-1,16..18..30], thus take A1 (if A is frame pair, takes A1 even lines, and if A is field pair, takes A0 second half)
	X-Field	Y-Field	A	A0	
X1 (Bottom)	X-Frame	Y-Frame	A	A1	Second Half of Bottom Frame MB uses [-1,24...31], thus A1 if A is a frame pair; but is only avail if both A0 and A1 are avail if A is a field pair due to the mix
	X-Frame	Y-Field	A	A0&A1	
	X-Field	Y-Frame	A	A1	Second Half of Bottom Field MB uses [-1,17..19..31], thus takes A1 (if A is frame pair, takes A1 odd lines, and if A is field pair, takes A1 second half)
	X-Field	Y-Field	A	A1	
			F		
X0 (Top)	X-Frame	Y-Frame	A	A0	Top Frame MB uses [-1,7] = A_7 (odd location), thus A0 if A is frame pair and A1 if field pair
	X-Frame	Y-Field	A	A1	
	X-Field	Y-Frame	A	A0	Top Field MB uses [-1,14] = A_14 (even location), thus A0 regardless A frame or field pair
	X-Field	Y-Field	A	A0	
X1 (Bottom)	X-Frame	Y-Frame	A	A1	Bottom Frame MB uses [-1,23] = A_23 (odd location), thus A1 regardless A frame or field pair
	X-Frame	Y-Field	A	A1	
	X-Field	Y-Frame	A	A0	Bottom Field MB uses [-1,15] = A_15 (odd

Current MB	Current MB Field	Neighbor MB Field	Neighbor MB Select (Y=?)	Neighbor Avail Result (OUTPUT)	Description
D					
	X-Field	Y-Field	A	A1	location), thus A0 if A is frame pair and A1 if A is field pair

Macroblock Type for Intra Cases

MbType follows two different tables according to whether the macroblock is an inter or intra macroblock according to IntraMbFlag.

For an intra macroblock, MbType, as defined in MbType definition for Intra Macroblock, carries redundant information as IntraMbMode. The notation I_16x16_x_y_z used in the table, 'x' is Intra16x16LumaPredMode, 'y' is ChromaCbpInd, and 'z' is LumaCbpInd, as defined in Sub field definition used by MbType for a macroblock with Intra16x16 prediction.

MbType definition for Intra Macroblock

Macroblock Type	MbType
I_4x4	0
I_8x8	0
I_16x16_0_0_0	1
I_16x16_1_0_0	2
I_16x16_2_0_0	3
I_16x16_3_0_0	4
I_16x16_0_1_0	5
I_16x16_1_1_0	6
I_16x16_2_1_0	7
I_16x16_3_1_0	8
I_16x16_0_2_0	9
I_16x16_1_2_0	Ah
I_16x16_2_2_0	Bh
I_16x16_3_2_0	Ch

Macroblock Type	MbType
I_16x16_0_0_1	Dh
I_16x16_1_0_1	Eh
I_16x16_2_0_1	Fh
I_16x16_3_0_1	10h
I_16x16_0_1_1	11h
I_16x16_1_1_1	12h
I_16x16_2_1_1	13h
I_16x16_3_1_1	14h
I_16x16_0_2_1	15h
I_16x16_1_2_1	16h
I_16x16_2_2_1	17h
I_16x16_3_2_1	18h
I_PCM	19h (used by HW)

Note: MbType here is identical as specified in DXVA 2.0.

For Intra_16x16 modes, the 5 bits of value (MbType - 1) have the following meanings.

Sub field definition used by MbType for a macroblock with Intra16x16 prediction

Bits	Description
4	**LumaCbpInd** - Luma Coded Block Pattern Indicator 0 means none of the luma blocks are coded. 1 means that at least one luma block is coded. 0 = SUBMODE_I16_L_0 1 = SUBMODE_I16_L_NZ In VME output, this field is forced to be 1 before adding 1 in Intra_16x16 mode.
3:2	**ChromaCbpInd** - Chroma Coded Block Pattern Indicator 00 means none of chroma blocks are coded. 01 means that only the chroma DC block is coded, but all AC blocks are not coded. 10 means that at least one AC

Bits	Description
	chroma block is coded. 00 = SUBMODE_I16_C_0 01 = SUBMODE_I16_C_DC 10 = SUBMODE_I16_C_NZ 11 = Reserved In VME output, this field is forced to be 10 before adding 1 in Intra_16x16 mode. *Programming Note: Adding 1 to MbType by VME hardware may have carry in to this field. But as '11' is reserved, the carry-in doesn't propagate into bit 4 or higher. This allows software to update MbType, if desired, using the redundant LumaIntraPredModes information.*
1:0	**Intra16x16PredMode** - Intra16x16 Prediction Mode These two bits carries redundant (identical) information as that in LumaIntraPredModes[0][0]. 0 = SUBMODE_I16_VER 1 = SUBMODE_I16_HOR 2 = SUBMODE_I16_DC 3 = SUBMODE_I16_PLANE

IntraMbMode definition

IntraMbMode [1:0]	Description	Supported by VME?	Used by PAK?
0	**INTRA_16x16 (redundant with MbType)**	Yes	Ignored
1	**INTRA_8x8**	Yes	Yes
2	**INTRA_4x4**	Yes	Yes
3	**IPCM (redundant with MbType)**	No	Ignored

As an alternative representation, MbType is logically the same as the following, except the I_PCM and I_NxN (i.e. I_4x4 and I_8x8) cases:

- 24 types of 16x16 intra modes: **A**+**B**+**C**+**D**: (1h - 18h)

MBTYPE_INTRA_16x16 1h A

  - 4 Intra16x16 modes:

SUBMODE_I16_VER 0 B

SUBMODE_I16_HOR 1 B

SUBMODE_I16_DC	2	B
SUBMODE_I16_PLN	3	B

- 3 Chroma Cbp indices:

SUBMODE_I16_C_0	0	C
SUBMODE_I16_C_DC	4	C
SUBMODE_I16_C_NZ	8	C

- 2 Luma Cbp indices:

SUBMODE_I16_L_0	0	D
SUBMODE_I16_L_NZ	Ch	D

Macroblock Type for Inter Cases

Sub-Macroblock Prediction Mode, SubMbPredMode, indicates the prediction mode for the sub-partitions. Prediction mode specifies prediction direction being forward (from L0), backward (from L1) or bi-directional (from both L0 and L1). Its meaning depends on InterMbMode. Definition of SubMbPredMode[i] provides the definition of the field.

- If InterMbMode is INTER16x16, only SubMbPredMode[0] is valid, it describes the prediction mode of the 16x16 macroblock. The other entries are set to zero by hardware.
    - For AVC, SubMbPredMode[0] contains redundant information as encoded in MbType parameter.
    - *Note: SubMbPredMode[1]-[3] are intentionally set to zero to allow a simple LUT to derive MbType as described later.*
- If InterMbMode is INTER16x8, and INTER8x16, only the first two entries SubMbPredMode[0] and SubMbPredMode[1] are valid, describing the sub-macroblock prediction mode.
    - For AVC, SubMbPredMode[0]/[1] contains redundant information as encoded in MbType parameter.
    - *Note: SubMbPredMode[2]-[3] are intentionally set to zero to allow a simple LUT to derive MbType as described later.*
- If InterMbMode is INTER8x8, each entry of SubMbPredMode describes the prediction mode of the sub-partition of an 8x8 sub-macroblock.
    - For AVC, SubMbPredMode can be derived from sub_mb_type field for BP_8x8 macroblocks as defined in AVC spec.
    - *Note on Direct Sub-macroblock Prediction Mode: Direct prediction is not conveyed through SubMbPredMode, instead, it is carried through Direct8x8Pattern.*

InterMbMode definition

MbSkipFlag	InterMbMode	Description
0	0	**INTER16x16**
0	1	**INTER16x8**
0	2	**INTER8x16**
0	3	**INTER8x8**
1	0	**PSKIP/BSKIP16x16***
1	3	**BSKIP**
1	1, 2	Reserved
Used by PAK	Ignored by PAK	

* BSKIP16x16 is an optional non-standard but equivalent optimization.

Definition of SubMbPredMode based on InterMbMode

SubMbPredMode	INTER16x16	INTER16x8	INTER8x16	INTER8x8
Bit	**MbType = [1...3]**	**MbType = [16h]**	**MbType = [4...15h]**	**MbType = [16h]**
7:6	MBZ	MBZ	MBZ	**Block8x8 3**
5:4	MBZ	MBZ	MBZ	**Block8x8 2**
3:2	MBZ	**Block16x8 1**	**Block8x16 1**	**Block8x8 1**
1:0	**Block16x16**	**Block16x8 0**	**Block8x16 0**	**Block8x8 0**
	Ignored by PAK	**Ignored by PAK**	**Ignored by PAK**	**Used by PAK**

Definition of SubMbPredMode[i]

SubMbPredMode	Description	InterMbMode	VME Output	MvCountPred	Notes
0	Pred_L0	All	Yes	1	P or B Slice
1	Pred_L1	All	Yes	1	B Slice Only
2	BiPred	All	Yes	2	B Slice Only
3	Reserved	Reserved	Reserved	Reserved	Reserved

Sub-Macroblock Shape, SubMbShape[i], for i = 0...3, describes the shape of the sub partitions of the 8x8 sub-macroblock of a BP_8x8 macroblock. This field is only valid if InterMBMode is INTER8x8. They are defined in Definition of SubMbShape for an 8x8 region of a BP_8x8 macroblock (including BSKIP, BDIRECT). The parameters can be derived from *sub_mb_type* field as defined in AVC spec.

Note: These fields must be correctly set even for **Direct** or **Skip** 8x8 cases, the individual B_Direct_8x8 block is flagged by the **Direct8x8Pattern** variable.

Definition of SubMbShape for an 8x8 region of a BP_8x8 macroblock (including BSKIP, BDIRECT)

	Description			
SubMbShape	NumSubMbPart	SubMbPartWidth	SubMbPartHeight	MvCountShape
0	1	8	8	1
1	2	8	4	2
2	2	4	8	2
3	4	4	4	4

For an inter macroblock, MbType, carries redundant information as InterMbMode and SubMbPredMode. MbType definition for Inter Macroblock (and MbSkipflag = 0) provides the typical inter macroblock types and Additional MbType definition with Direct/Skip for Inter Macroblock provides that with skip and direct modes. The definition of MbType for both P slice and B slice is the same and is equivalent to that for mb_type of a B slice in the AVC spec. As direct mode is indicated using a separate field Direct8x8Pattern, 0 is reserved for MbType.

Here, MVCount is the number of motion vectors actually encoded in the bitstream. It is informative. For a BP_8x8 or equivalent Skip/Direct macroblock, MVCount is the sum of the following term for the four 8x8 sub macroblock (with i = 0...3):

MvCountShape[i] * MvCountPred[i] * MvCountDirect[i]

where MvCountShape[i] is block count for sub macroblock [i], MvCountPred[i] is the motion vector count for each block of sub macroblock[i], and MvCountDirect[i] is the multipler for direct mode for B Slice, indicating whether motion vectors are coded or not. It must be set to 1 for P slice. For B Slice, MvCountDirect[i] = !Direct8x8Pattern[i], which is 0 for a sub macroblock coded as direct mode and 1 otherwise.

In the tables, "DC" stands for "Don't Care" as PAK hardware ignores these fields.

MbType definition for Inter Macroblock (and MbSkipflag = 0)

Macroblock Type	MbType	MbSkipFlag	Direct8x8Pattern	SubMbShape	SubMbPredMode	MVCount
Reserved	0	-	-	-	-	-
BP_L0_16x16	1	0	0	DC	DC	1
B_L1_16x16	2	0	0	DC	DC	1
B_Bi_16x16	3	0	0	DC	DC	2
BP_L0_L0_16x8	4	0	0	DC	DC	2
BP_L0_L0_8x16	5	0	0	DC	DC	2
B_L1_L1_16x8	6	0	0	DC	DC	2
B_L1_L1_8x16	7	0	0	DC	DC	2
B_L0_L1_16x8	8	0	0	DC	DC	2

Macroblock Type	MbType	MbSkipFlag	Direct8x8Pattern	SubMbShape	SubMbPredMode	MVCount
B_L0_L1_8x16	9	0	0	DC	DC	2
B_L1_L0_16x8	0Ah	0	0	DC	DC	2
B_L1_L0_8x16	0Bh	0	0	DC	DC	2
B_L0_Bi_16x8	0Ch	0	0	DC	DC	3
B_L0_Bi_8x16	0Dh	0	0	DC	DC	3
B_L1_Bi_16x8	0Eh	0	0	DC	DC	3
B_L1_Bi_8x16	0Fh	0	0	DC	DC	3
B_Bi_L0_16x8	10h	0	0	DC	DC	3
B_Bi_L0_8x16	11h	0	0	DC	DC	3
B_Bi_L1_16x8	12h	0	0	DC	DC	3
B_Bi_L1_8x16	13h	0	0	DC	DC	3
B_Bi_Bi_16x8	14h	0	0	DC	DC	4
B_Bi_Bi_8x16	15h	0	0	DC	DC	4
BP_8x8	16h	0	!= Fh	vary	vary	Sum
Reserved	17h-1Fh	-	-	-	-	-

Additional MbType definition with Direct/Skip for Inter Macroblock

Macroblock Type	Mb Type	Xfrm 8x8	MbSkip Flag	Direct8x8 Pattern	SubMb Shape	SubMb PredMode	MvCount	Notes
P_Skip_16x16	1	-	1	DC	DC	DC	0	Skipped macroblock. Motion compensation like P_L0_16x16
B_Skip_16x16_4MVPair	16h	vary	1	Fh	0	vary	0	Skipped macroblock. Motion compensation like B_8x8 with 8x8 subblocks, when **direct_8x8_inference_flag** is set to 1
B_Skip_16x16_16MVPair	16h	0	1	Fh	FFh	vary	0	Skipped macroblock. Motion compensation like B_8x8 with 4x4 subblocks, when **direct_8x8_inference_flag** is set to 0
B_Direct_16x16_4MVPair	16h	vary	0	Fh	0	vary	0	MbType coded as B_Direct_16x16. Motion compensation like B_8x8 with 8x8 subblocks, when

Macroblock Type	Mb Type	Xfrm 8x8	MbSkip Flag	Direct8x8 Pattern	SubMb Shape	SubMb PredMode	MvCount	Notes
								direct_8x8_inference_flag is set to 1
B_Direct_16x16_16MVPair	16h	0	0	Fh	FFh	vary	0	MbType coded as B_Direct_16x16. Motion compensation like B_8x8 with 4x4 subblocks, when **direct_8x8_inference_flag** is set to 0

People might notice that B_DIRECT_16x16 and B_SKIP are mapped on BP_8x8 for AVC decoding interface in IT mode as the motion compensation operation for both modes are the same as BP_8x8. According to AVC Spec, motion vectors for B_DIRECT_16x16 and B_SKIP are derived from temporally co-located macroblock on an 8x8 sub macroblock basis if direct_8x8_inference_flag is set to 1 or on a 4x4 block basis if it is set to 0. For each sub macroblock or block, SubMbPredMode is derived, thus can any of the valid numbers. Motion vectors may also be different. In spatial direct mode, the motion vectors are subject to spatial neighbor macroblocks as well as co-located macroblock. The spatial prediction is based on the neighbor macroblocks, so the same spatial predicted motion vector applies to all sub macroblocks or blocks. However, under certain conditions, temporal predictor may replace (colZeroFlag) the spatial predictor for a given sub macroblock or block. Thus the motion vectors may differ.

In MbType definition for Inter Macroblock (and MbSkipflag = 0), the macroblock type names for major partitions nicely follow forms *BP_MbPredMode_MbShape* (like BP_L0_16x16) and *B_MbPredMode0_MbPredMode1_MbShape* (like B_L0_Bi_16x8). For minor partitions it is fixed as *BP_MbShape* as BP_8x8.

However, in Additional MbType definition with Direct/Skip for Inter Macroblock the macroblock types for Skip and Direct modes does not follow the same rule. The third field in P_Skip_16x16 or B_Direct_16x16_x indicates that "Skip" or "Direct" applies to the entire 16x16 macroblock, even though MbShape is 8x8 as that in BP_8x8. In order to distinguish the SubMbShape being 8x8 or 4x4 for B_Skip and B_Direct, the fourth field is added. 4MVP**air** indicates upto 4 MV pairs are presented with SubMbShape equals to 0; and 16MVP**air** indicates up to 16 MV pairs are presented with SubMbShape equals to FFh.Also note that P_8x8ref0 is not specified in PAK input interface, it is up to hardware to detect and choose its packing format based on number of reference indices and reference index for the given macroblock.

Macroblock Type Conversion Rules

For improved coding efficiency the PAK hardware has the capability to convert macroblock types to use more efficiency coding modes such as DIRECT and SKIP. For an inter macroblock or a sub macroblock coded as DIRECT, no motion vector is needed in the bitstream for the macroblock or sub macroblock. If a macroblock is coded as SKIP, it only consumes one SKIP bit (no motion vector, no coefficients are coded). And information about the macroblock is 'inferred' according to the rules stated in the AVC Spec.

As the input to PAK, the following signals can convey the information regarding DIRECT and SKIP:

- MbSkipFlag
- Direct8x8Pattern
- CodecBlockPattern (CbpY, CbpCb, CbpCr)

Such conversion can be enabled or disabled through the SLICE_STATE fields DirectConvDisable and SkipConvDisable as well as the in line command field MbSkipConvDisable.

A P slice doesn't support direct mode, it only supports P_Skip, which is equivalent to a 16_16_L0 prediction. Other prediction types cannot be converted to P_Skip. The following table describes the macroblock type conversion rules for a P slice. Here CBP = CbpY/CbpCb/CbpCr are the final computed results after quantization by the hardware. Note that hardware honors the input CbpY/CbpCb/CbpCr fields - if the value corresponding to a block is set to zero, the resulting CBP is also zero. The output mb_skip_flag and mb_type are the symbols coded in the bitstream as defined in the AVC spec. *DC* stands for *Don't care*, *T* for *True*.

Note that the internal condition of MV==MVP is subject to the precise rules stated in the AVC Spec as quoted below. Note that there are exceptions for P_Skip from the normal motion vector prediction rules.

Derivation process for luma motion vectors for skipped macroblocks in P and SP slices

This process is invoked when mb_type is equal to P_Skip.

Outputs of this process are the motion vector mvL0 and the reference index refIdxL0.

The reference index refIdxL0 for a skipped macroblock is derived as follows.

refIdxL0 = 0. (8-168)

For the derivation of the motion vector mvL0 of a P_Skip macroblock type, the following applies.

- The process specified in subclause 8.4.1.3.2 is invoked with mbPartIdx set equal to 0, subMbPartIdx set equal to 0, currSubMbType set equal to "na", and listSuffixFlag set equal to 0 as input and the output is assigned to mbAddrA, mbAddrB, mvL0A, mvL0B, refIdxL0A, and refIdxL0B.

- The variable mvL0 is specified as follows.

- If any of the following conditions are true, both components of the motion vector mvL0 are set equal to 0.

- mbAddrA is not available

- mbAddrB is not available

- refIdxL0A is equal to 0 and both components of mvL0A are equal to 0

- refIdxL0B is equal to 0 and both components of mvL0B are equal to 0

- Otherwise, the derivation process for luma motion vector prediction as specified in subclause 8.4.1.3 is invoked with mbPartIdx = 0, subMbPartIdx = 0, refIdxL0, and currSubMbType = "na" as inputs and the output is assigned to mvL0.

NOTE - The output is directly assigned to mvL0, since the predictor is equal to the actual motion vector.

Macroblock type conversion rule for an inter macroblock in a P slice

Input		Internal			Output		Notes
Macroblock Type	SkipConvDisable || SkipConvDisable	CBP	MV == MVP	MbAffSkipAllowed	mb_skip_flag	mb_type	
P_Skip_16x16	DC	DC	DC	1	1	-	Forced to P_Skip; Hardware will force CBP to zero and also ignore SkipConvDisable control. Hardware doesn't check for MV==MVP error condition
P_Skip_16x16	DC	DC	DC	0	0	0	Reverse convert to P_L0_16x16; Hardware will force CBP to zero but reversely convert MbType as P_L0_16x16 once it determines that Skip is not allowed.
BP _16x16_L0	0	0	T	1	1	-	Converted to P_Skip. Even input doesn't provide skip hint, hardware can performance the optimization by detecting CBP and MV==MVP condition.
BP _16x16_L0	0	0	T	0	0	0	Reverse back to P_L0_16x16; Hardware will reverse back to P_L0_16x16 even Skip conditions are met once it determines that Skip is not allowed.
BP _16x16_L0	1	0	T	T	0	0	Still coded as P_L0_16x16 = 0.

A B slice supports both direct and skip modes. The following table describes the macroblock type conversion rules for a B slice. Hardware does not verify MV==MVP condition for a Skip/Direct macroblock in a B Slice as no DMV is performed by hardware.

Macroblock type conversion rule for an inter macroblock in a B slice

Input			Internal			Output		Notes
Macroblock Type	SkipConvDisable || SkipConvDisable	DirectConvDisable	CBP	MV == MVP	MbAffSkipAllowed	mb_skip_flag	mb_type	
B_Skip_8x8 B_Skip_4x4	DC	DC	DC	n/a	1	1	-	Forced to B_Skip; Hardware will force CBP to zero and also ignore SkipConvDisable control.
B_Skip_8x8 B_Skip_4x4	DC	DC	DC	n/a	0	0	0	REVERSE convert to B_Direct_16x16; Hardware will force CBP to zero and also reverse convert to B_Direct_16x16 when it discovers Skip is not allowed.
B_Direct_16x16_4MVPair/16MVPair	0	0	0	n/a	1	1	-	Converted to B_Skip. Hardware first converts to B_Direct_16x16 and then further to B_Skip if CBP = 0.
B_Direct_16x16_4MVPair/16MVPair	0	0	0	n/a	0	0	0	Converted to B_Direct_16x16. Hardware first converts to B_Direct_16x16 and stop there as it discovers Skip is not

Input			Internal			Output		Notes
Macroblock Type	SkipConvDisable || SkipConvDisable	DirectConvDisable	CBP	MV == MVP	MbAffSkipAllowed	mb_skip_flag	mb_type	
								allowed even CBP=0.
B_Direct_16x16_4MVPair/16MVPair	1	0	0	n/a	DC	0	0	Converted to B_Direct_16x16. Hardware converts to B_Direct_16x16 and stops there even though CBP = 0 as input disallows Skip conversion.
B_Direct_16x16_4MVPair/16MVPair	DC	0	NZ	n/a	DC	0	0	Converted to B_Direct_16x16. Hardware converts to B_Direct_16x16 and stops there because CBP != 0.
B_Direct_16x16_4MVPair/16MVPair	DC	1	DC	n/a	DC	0	16h	Stay as B_8x8. Hardware stays at B_8x8 and codes each sub macroblocks even all are direct.

The internal signal MbAffSkipAllowed is added to deal with a restriction on the frame/field flag (MbFieldFlag) which is unique to MBAFF. MbAffSkipAllowed is always set to 1 in non-MBAFF modes. In MBAFF mode, a macroblock pair may be both skipped only if its MbFieldFlag is the same as its available neighbor macroblock pair A or B if A or B is available (in that order), or is not 0 if A/B are both not available. Otherwise, one of the macroblocks in the pair must be coded.

To reduce the burden on software, PAK hardware handles the above restriction correctly. For the first MB in a pair, MbAffSkipAllowed is always set to 1. Therefore, hardware allows converting the first MB to Skip if skip conversion is enabled. For the second MB in a pair, hardware sets MbAffSkipAllowed to 0 if the following is true:

- The current MB Pair has different MbFieldFlag than its available neighbor A or B if A or B is available, or is not 0 if A/B are both not available
- And the first MB is coded as a SKIP (could be forced or converted)

Otherwise, it sets MbAffSkipAllowed to 1. As MbAffSkipAllowed is to 0 for the above condition, hardware will disallow Skip mode for the second MB. If the input signal forces it to Skip, hardware performs reverse-conversion to code it as P_L0_16x16 or B_Direct_16x16 with CBP = 0 for a macroblock in a P or B Slice. This means that hardware is able to correct the programming mistake by software. If the macroblock is not forced to skip, hardware simply disallows Skip conversion.

Software still has an option to disallow Skip Conversion on a per-MB basis using the MbSkipConvDisable control field in the inline command.

Indirect Data Description

For each macroblock, an ENC-PAK data set consists of two types of data blocks: indirect **MV data block** and **inline MB information**.

The indirect MV data block may be in two modes: **unpackedmode** and **packed-size mode**.

Unpacked Motion Vector Data Block

Unpacked Motion Vector Data Block

In the **unpacked** mode, motion vectors are expanded (or duplicated) to either bidirectional 8x8 8MV major partition format, or bidirectional 4x4 32MV format. Thus either 32 bytes or 128 bytes is assigned to each MB.

Motion Vector block contains motion vectors in an intermediate format that is partially expanded according to the sub- macroblock size. During the expansion, a place that does not contain a motion vector is filled by replicating the relevant motion vector according to the following motion vector replication rules. If the relevant motion vector doesn't exist (for the given L0 or L1), it is zero filled.

Motion Vector Replication Rules:

- Rule #1
    - #1.1: For L0 MV, for any sub-macroblock or sub-partition where there is at least one motion vector
        - If L0 inter prediction exists, the corresponding L0 MV is used
        - Else it must be zero
    - #1.2: For L1 MV, for any sub-macroblock or sub-partition where there is at least one motion vector
        - If L1 inter prediction exists, the corresponding L1 MV is used
        - Else it must be zero

- For a macroblock with a 16x16, 16x8 or 8x16 sub-macroblock, MvSize = 8. The eight MV fields follow Rule #1.
    - The 16x16 is broken down into 4 8x8 sub-macroblocks. The 16x16 MVs (after rule #1) are replicated into all 8x8 blocks.
    - For an 8x16 partition, each 8x16 is broken down into 2 8x8 stacking vertically. The 8x16 MVs (after rule #1) are replicated into both 8x8 blocks.
    - For a 16x8 partition, each 16x8 is broken down into 2 8x8 stacking horizontally. The 16x8 MVs (after rule #1) are replicated into both 8x8 blocks.
- For macroblock with sub-macroblock of 8x8 without minor partition (SubMbShape[0...3] = 0), MvSize = 8, (e.g. mb_type equal to P_8x8, P_8x8ref0, or B_8x8)
    - There is no motion vector replication
- For macroblock with sub-macroblock of 8x8 with at least one minor partition (if any SubMbShape[i] != 0), MvSize = 32, (e.g. mb_type equal to P_8x8, P_8x8ref0, or B_8x8)
    - For an 8x8 sub-partition, the 8x8 MVs (after rule #1) is replicated into all the four 4x4 blocks.
    - For an 4x8 sub-partition within an 8x8 partition, each 4x8 is broken down into 2 4x4 stacking vertically. The 4x8 MVs (after rule #1) are replicated into both 4x4 blocks.
    - For an 8x4 sub-partition within an 8x8 partition, each 8x4 is broken down into 2 4x4 stacking horizontally. The 8x4 MVs (after rule #1) are replicated into both 4x4 blocks.
    - For a 4x4 sub-partition within an 8x8 partition, each 4x4 has its own MVs (after rule #1).

Motion Vector block and MvSize

	DWord	Bit	MvSize	
			8	32
W1.0		31:16	**MV_Y0_L0.y**	**MV_Y0_0_L0.y**
		15:0	**MV_Y0_L0.x**	**MV_Y0_0_L0.x**
W1.1		31:16	**MV_Y0_L1.y**	**MV_Y0_0_L1.y**
		15:0	**MV_Y0_L1.x**	**MV_Y0_0_L1.x**
W1.2		31:0	**MV_Y1_L0**	**MV_Y0_1_L0**
W1.3		31:0	**MV_Y1_L1**	**MV_Y0_1_L1**
W1.4		31:0	**MV_Y2_L0**	**MV_Y0_2_L1**
W1.5		31:0	**MV_Y2_L1**	**MV_Y0_2_L0**

	DWord	Bit	MvSize	
			8	32
W1.6		31:0	**MV_Y3_L0**	**MV_Y0_3_L0**
W1.7		31:0	**MV_Y3_L1**	**MV_Y0_3_L1**
W2.0		31:0	**n/a**	**MV_Y1_0_L1**
W2.1		31:0	**n/a**	**MV_Y1_0_L0**
W2.2		31:0	**n/a**	**MV_Y1_1_L1**
W2.3		31:0	**n/a**	**MV_Y1_1_L0**
W2.4		31:0	**n/a**	**MV_Y1_2_L1**
W2.5		31:0	**n/a**	**MV_Y1_2_L0**
W2.6		31:0	**n/a**	**MV_Y1_3_L0**
W2.7		31:0	**n/a**	**MV_Y1_3_L1**
W3.0		31:0	**n/a**	**MV_Y2_0_L1**
W3.1		31:0	**n/a**	**MV_Y2_0_L0**
W3.2		31:0	**n/a**	**MV_Y2_1_L1**
W3.3		31:0	**n/a**	**MV_Y2_1_L0**
W3.4		31:0	**n/a**	**MV_Y2_2_L1**
W3.5		31:0	**n/a**	**MV_Y2_2_L0**
W3.6		31:0	**n/a**	**MV_Y2_3_L0**
W3.7		31:0	**n/a**	**MV_Y2_3_L1**
W4.0		31:0	**n/a**	**MV_Y3_0_L1**
W4.1		31:0	**n/a**	**MV_Y3_0_L0**
W4.2		31:0	**n/a**	**MV_Y3_1_L1**

	DWord	Bit	MvSize	
			8	32
W4.3		31:0	**n/a**	**MV_Y3_1_L0**
W4.4		31:0	**n/a**	**MV_Y3_2_L1**
W4.5		31:0	**n/a**	**MV_Y3_2_L0**
W4.6		31:0	**n/a**	**MV_Y3_3_L0**
W4.7		31:0	**n/a**	**MV_Y3_3_L1**

The motion vector(s) for a given sub-macroblock or a sub-partition are uniquely placed in the output message as shown by the non-duplicate fields in Motion Vector duplication by sub-macroblocks for a 16x16 macroblock, whereas the 8x8 column is for 4x(8x8) partition without minor shape and Motion Vector duplication by sub-partitions for the first 8x8 sub-macroblock Y0 if any Y0-Y3 contains minor shape (Y1_ to Y3_ have the same format in W2 to W4).

MV_Yx_L0 and MV_Yx_L1 may be present individually or both. If one is not present, the corresponding field must be zero. Subsequently, the duplicated fields will be zero as well.

Motion Vector duplication by sub-macroblocks for a 16x16 macroblock, whereas the 8x8 column is for 4x(8x8) partition without minor shape

DWord	Bit	16x16	16x8	8x16	8x8
W1.0	31:16	**MV_Y0_L1 (A)**	**MV_Y0_L1 (A)**	**MV_Y0_L1**	**MV_Y0_L1**
	15:0	**MV_Y0_L0 (A)**	**MV_Y0_L0 (A)**	**MV_Y0_L0**	**MV_Y0_L0**
W1.1	31:16	**Duplicate (A)**	**Duplicate (A)**	**MV_Y1_L1**	**MV_Y1_L1**
	15:0	**Duplicate (A)**	**Duplicate (A)**	**MV_Y1_L0**	**MV_Y1_L0**
W1.2	31:16	**Duplicate (A)**	**MV_Y2_L1 (B)**	**Duplicate (A)**	**MV_Y2_L1**
	15:0	**Duplicate (A)**	**MV_Y2_L0 (B)**	**Duplicate (A)**	**MV_Y2_L0**

DWord	Bit	16x16	16x8	8x16	8x8
W1.3	31:16	**Duplicate (A)**	**Duplicate (B)**	**Duplicate (B)**	**MV_Y3_L1**
	15:0	**Duplicate (A)**	**Duplicate (B)**	**Duplicate (B)**	**MV_Y3_L0**

Motion Vector duplication by sub-partitions for the first 8x8 sub-macroblock Y0 if any Y0-Y3 contains minor shape (Y1_ to Y3_ have the same format in W2 to W4)

DWord	Bit	8x8	8x4	4x8	4x4
W1.0	31:16	**MV_Y0_L1**	**MV_Y0_0_L1 (A)**	**MV_Y0_0_L1 (A)**	**MV_Y0_0_L1**
	15:0	**MV_Y0_L0**	**MV_Y0_0_L0 (A)**	**MV_Y0_0_L0 (A)**	**MV_Y0_0_L0**
W1.1	31:16	**Duplicate (A)**	**Duplicate (A)**	**MV_Y0_1_L1 (B)**	**MV_Y0_1_L1**
	15:0	**Duplicate (A)**	**Duplicate (A)**	**MV_Y0_1_L0 (B)**	**MV_Y0_1_L0**
W1.2	31:16	**Duplicate (A)**	**MV_Y0_2_L1 (B)**	**Duplicate (A)**	**MV_Y0_2_L1**
	15:0	**Duplicate (A)**	**MV_Y0_2_L0 (B)**	**Duplicate (A)**	**MV_Y0_2_L0**
W1.3	31:16	**Duplicate (A)**	**Duplicate (B)**	**Duplicate (B)**	**MV_Y0_3_L0**
	15:0	**Duplicate (A)**	**Duplicate (B)**	**Duplicate (B)**	**MV_Y0_3_L1**

Packed-Size Motion Vector Data Block

In the packed case, no redundant motion vectors are sent. So the number of motion vectors sent, as specified by **MvQuantity** is the same as the motion vectors that will be packed (**MvPacked**).

The following tables are for information only. Fields like MvQuantity and MvPacked are not required interface fields.

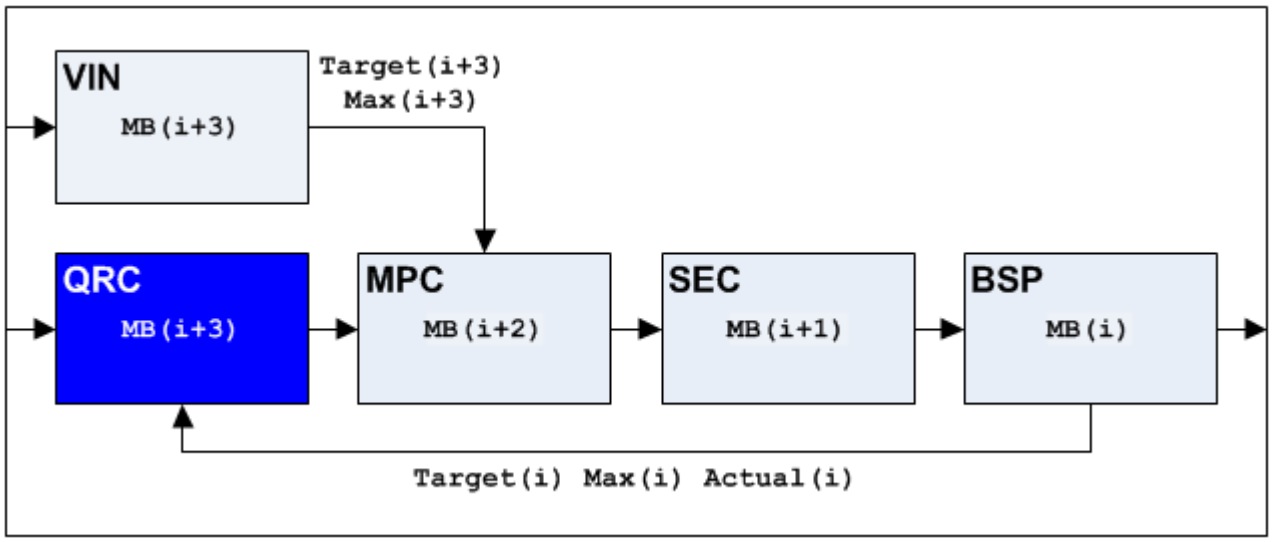
MbSkipFlag	MbType	Description	Mv Quantity	MvSize	(Minimal MvSize)
1	1	P_Skip_16x16	0	8	1
0	1	BP_L0_16x16	1	8	1
0	2	B_L1_16x16	1	8	1
0	3	B_Bi_16x16	2	8	2
0	4	BP_L0_L0_16x8	2	8	4
0	5	BP_L0_L0_8x16	2	8	4
0	6	B_L1_L1_16x8	2	8	8
0	7	B_L1_L1_8x16	2	8	8
0	8	B_L0_L1_16x8	2	8	8
0	9	B_L0_L1_8x16	2	8	8
0	0Ah	B_L1_L0_16x8	2	8	8
0	0Bh	B_L1_L0_8x16	2	8	8
0	0Ch	B_L0_Bi_16x8	3	8	8
0	0Dh	B_L0_Bi_8x16	3	8	8
0	0Eh	B_L1_Bi_16x8	3	8	8
0	0Fh	B_L1_Bi_8x16	3	8	8
0	10h	B_Bi_L0_16x8	3	8	8
0	11h	B_Bi_L0_8x16	3	8	8
0	12h	B_Bi_L1_16x8	3	8	8
0	13h	B_Bi_L1_8x16	3	8	8
0	14h	B_Bi_Bi_16x8	4	8	8
0	15h	B_Bi_Bi_8x16	4	8	8
0	16h	**BP**_8x8	^34	8 or 32	8 or 32

When MbType = 22, BP_8x8, take the sum of four individual 8x8 subblocks

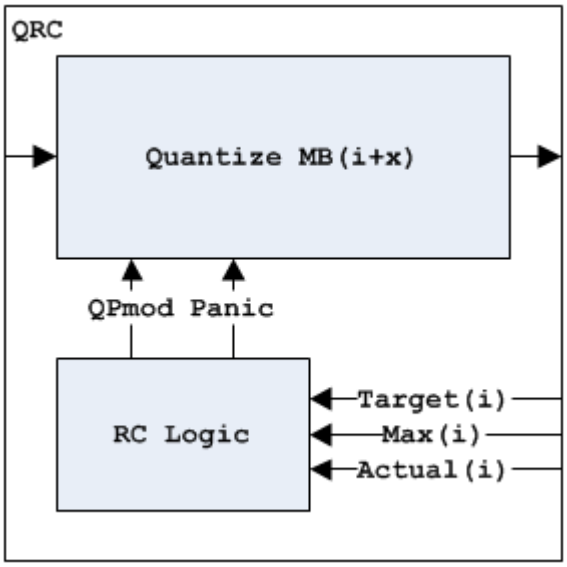
Direct8x8Pattern	SubMb Shape	SubMb PredMode	Description	Mv Quantity	Mv Size	(Min MvSize)
OR	OR	OR		ADD	ADD	ADD
1	0	0	P_Skip_8x8 B_Direct_L0_8x8 (B-Skip_ L0_8x8)	0	2	1
1	0	1	B_Direct_L1_8x8 (B-Skip_ L1_8x8)	0	2	1
1	0	2	B_Direct_Bi_8x8 (B-Skip_ Bi_8x8)	0	2	2
1	3	0	P_Skip_4x4 B_Direct_L0_4x4 (B-Skip_ L0_4x4)	0	8	4
1	3	1	B_Direct_L1_4x4 (B-Skip_ L1_4x4)	0	8	4
1	3	2	B_Direct_Bi_4x4 (B-Skip_ Bi_4x4)	0	8	8
0	0	0	BP_L0_8x8	1	2	1
0	0	1	B_L1_8x8	1	2	1
0	0	2	B_BI_8x8	2	2	2
0	1	0	BP_L0_8x4	2	8	4
0	1	1	B_L1_8x4	2	8	4
0	1	2	B_BI_8x4	4	8	8
0	2	0	BP_L0_4x8	2	8	4
0	2	1	B_L1_4x8	2	8	4
0	2	2	B_BI_4x8	4	8	8
0	3	0	BP_L0_4x4	4	8	4
0	3	1	B_L1_4x4	4	8	4
0	3	2	B_BI_4x4	8	8	8

Macroblock Level Rate Control

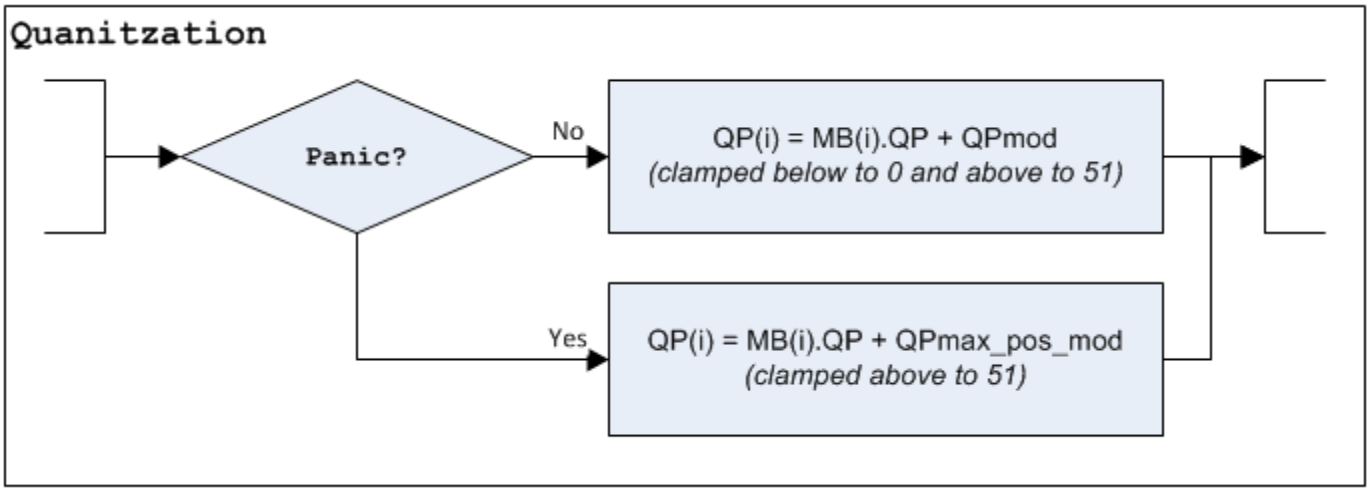
The QRC (Quantization Rate Control) unit receives data from BSP (Bit Serial Packer) and VIN (Video In) and generates adjustments to QP values across macroblocks.



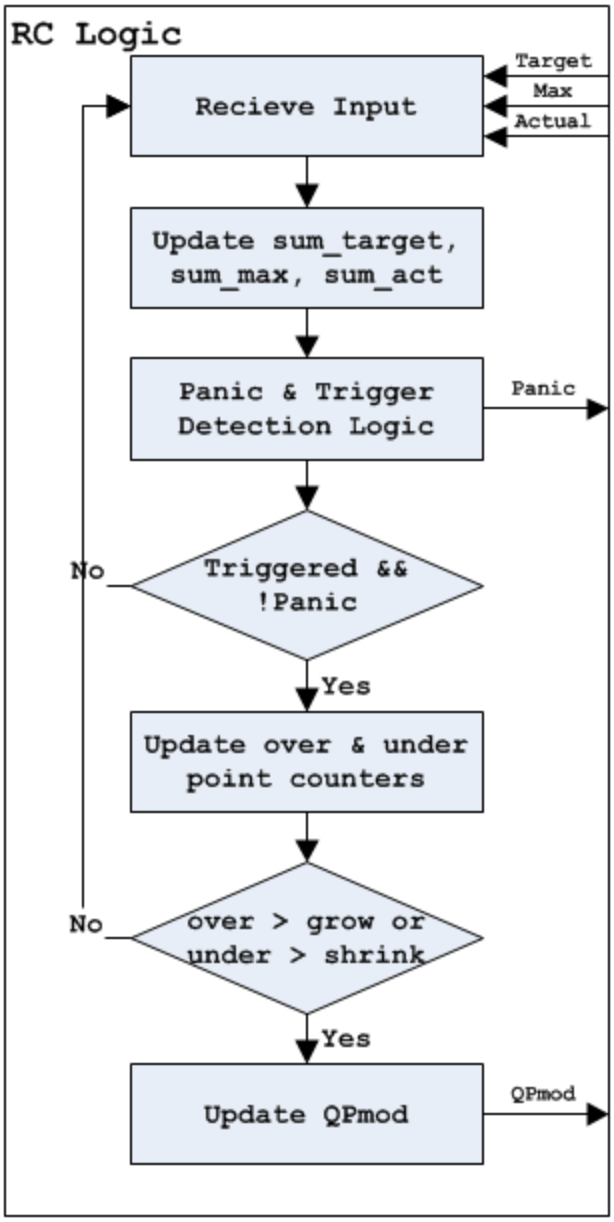
QRC can be logically partitioned into two units as shown below.



Macroblock level rate control is handled by the RC logic and the quantization logic.



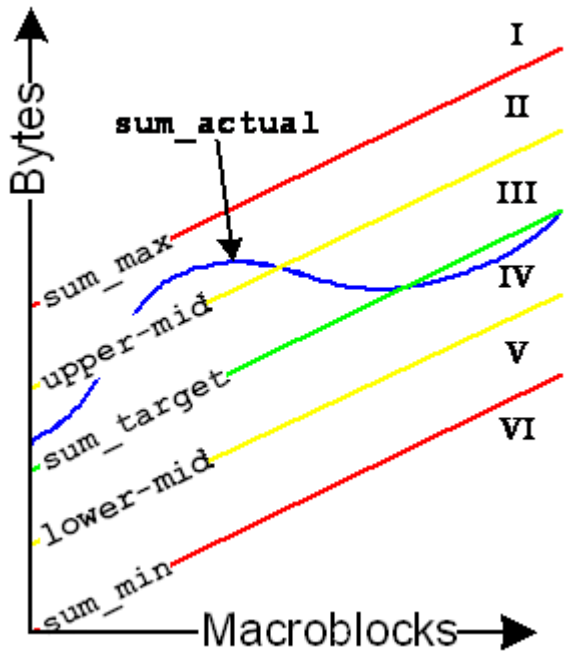
The signals QPmod and panic are generated by the RC logic based on data feedback from BSP. A flowchart of the RC logic is given below.



Theory of Operation Overview

BSP will generate a byte estimate for each macroblock packed. Additionally, the user will specify a target and max size per macroblock. The running sum of these signals (actual, target, max) creates "curves" which are used to identify when QP adjustments are necessary (see figure below). Three more curves are symmetrically generated by QRC (upper_midpt, lower_midpt, sum_min) from target and max. The values of target and max are specified by the user will dictate the shape of these curves.

The difference between sum_actual and sum_target (called 'bytediff') identifies the margin of error between the target and actual sizes. The difference between the current bytediff and the previously calculated bytediff represents the rate of change in this margin over time. The sign of this rate is used to identify if the correction is trending in the appropriate direction (towards bytediff = 0).



QPmod

Each macroblock will have a requested QP (which could vary across macroblocks or remain constant). QPmod is to be added to the QP requested. QPmod will be positive when the target was under-predicted and negative when the target is over-predicted.

QPmod is incremented or decremented when internal counters (called 'over' and 'under') reach tripping points (called 'grow' and 'shrink'). For each MB processed and based on which region (1-6) sum_actual falls in, various amounts of points are added to either counters. If over exceeds grow, QPmod is incremented whereas if under exceeds shrink, QPmod is decremented.

To dampen the effect of repeated changes in the same direction, an increase in resistance for that direction and decrease in resistance for the complementary direction occurs (called 'grow_resistance' and 'shrink_resistance'). This resistance is added to grow or shrink, which then requires more points to trip the next correction in that direction.

The user can specify guard-bands that limit the amount QPmod can be modified. QPmod cannot exceed QPmax_pos_mod or become less than -QPmax_neg_mod_abs.

Triggering

The RC unit begins to modify QPmod occurs only when it is triggered.

Three levels of triggering exist: always, gentle, loose. Always means that RC will be active once sum_actual reaches regions 3 or 4. Gentle will trigger RC once sum_actual reaches regions 2 or 5. Loose waits to trigger RC when sum_actual reaches regions 1 or 6.

RC will deactivate (triggered = false) once sum_actual begins to track sum_target over a series of macroblocks. Specifically, the sign of the rate of change for bytediff is monitored over a window of

macroblocks. When the sum of these signs over the window falls within a tolerance value (called 'stable'), triggered will reset to false.

Panic

When enabled, panic mode will occur whenever sum_actual reaches region 1 and will remain so until sum_actual reaches region 4. When panicking, all macroblocks will be quantized with QP = MB(n).QP + QPmax_pos_mod, clamped to 51.

User Controls

This unit achieves a large flexibility by allowing the user to define various key parameters. At the per-macroblock level, the values of target and max are specified and will create various shapes of curves that sum_actual will be compared against.

Per-slice, the user can specify the triggering sensitivity and the tolerance required to disable the trigger. Additionally, the user can enable panic detection.

The point values assigned to each of the 6 regions are exposed to the user which allow for asymmetrical control for over and under predictions amongst other things. Additionally, the user can specify the initial values of grow and shrink along with the resistance values applied when correction is invoked.

Lastly, the maximum and minimum values for QPmod are also exposed to the user.

AVC Encoder MBAFF Support

1. **Algorithm**

   Prediction of current macroblock motion vector is possible from neighboring macroblocks mbAddrA/mbAddrD/mbAddrB/mbAddrC/mbAddrA+1/mbAddrD+1/mbAddrB+1/mbAddrC+1. The selection of these macroblocks depends on coding type(field/frame) of current macroblock pair and the coding of neighboring macroblock pair.

   Selection of these macroblock pairs is described in detail in following sections.

   1.1 Selection of Top LeftMB pair: The selection of Top Left MB pair depends on coding type of current and also top left macroblock pair.

   1.2 Selection of LeftMB pair: The selection of Left MB pair depends on coding type of current and also left macroblock pair.

   1.3 Selection of Top MB pair: The selection of Top MB pair depends on coding type of current and also top macroblock pair.

   1.4 Selection of Top RightMB pair: The selection of Top Right MB pair depends on coding type of current and also top right macroblock pair.

   1.5 Motion Vector and refIdx Scaling: Motion vectors and reference index of neighboring macroblocks (mbAddrA/mbAddrB/mbAddrC/mbAddrD) should be scaled before using them into prediction equations. Again, the scaling depends on coding type of current and neighboring macroblock pair which is described as follows,

- If the current macroblock is a field macroblock and the macroblock mbAddrN is a frame macroblock ...

```
mvLXN[ 1 ] = mvLXN[ 1 ] / 2 (8-214)
refIdxLXN = refIdxLXN * 2   (8-215)
```

- Otherwise, if the current macroblock is a frame macroblock and the macroblock mbAddrN is a field macroblock ...

```
mvLXN[ 1 ] = mvLXN[ 1 ] * 2 (8-216)
refIdxLXN = refIdxLXN / 2   (8-217)
```

- Otherwise, the vertical motion vector component mvLXN[1] and the reference index refIdxLXN remain unchanged.

MPEG-2

This topic is currently under development.

MPEG2 Common Commands

MFX Commands are organized into groups based on their scope of functioning. There are Pipeline Common state commands that are common to all codecs (encoder and decoder) and is applicable to the processing of one full frame/field. There are also individual codec Common state commands that are common to both encoder and decoder of that particular codec. These latter common state commands, some are applicable at the processing of one full frame/field, and some are applicable at the processing of an individual slice level.

MFX_MPEG2_PIC_STATE

MPEG2 Decoder Commands

These are decoder-only commands. They provide the pointer to the compressed input bitstream to be decoded.

MFD_MPEG2_BSD_OBJECT

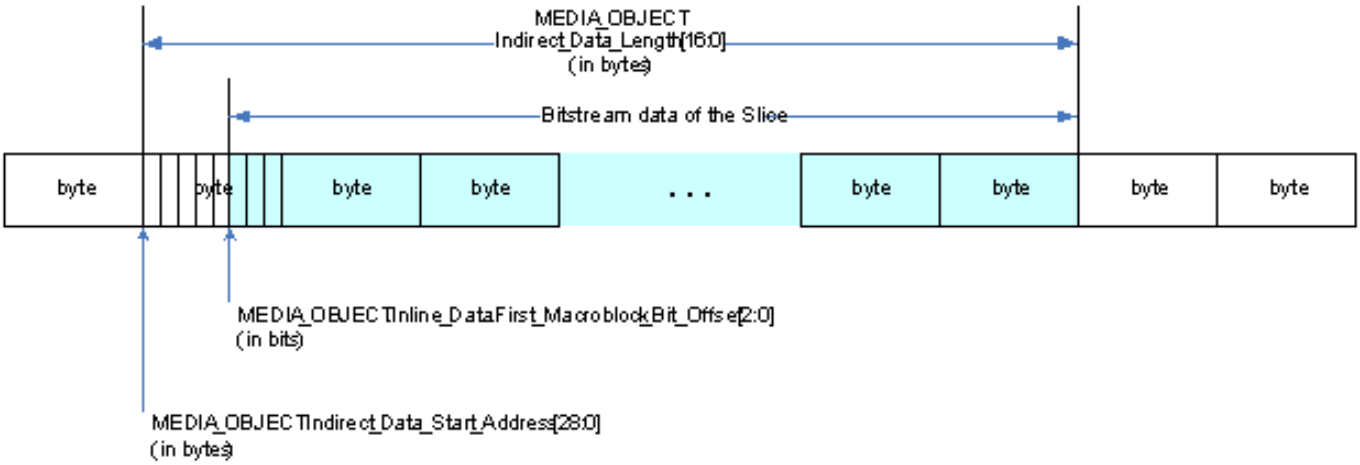
MFD_MPE2_BSD_OJBECT Inline Data Description

Indirect Data Description

The indirect data start address in MFD_MPEG2_BSD_OBJECT specifies the starting Graphics Memory address of the bitstream data that follows the slice header. It provides the byte address for the first macroblock of the slice. Together with the First Macroblock Bit Offset field in the inline data, it provides the bit location of the macroblock within the compressed bitstream.

The indirect data length in MFD_MPEG2_BSD_OBJECT provides the length in bytes of the bitstream data for this slice. It includes the first byte of the first macroblock and the last **non-zero** byte of the last macroblock in the slice. Specifically, the zero-padding bytes (if present) and the next start-code are excluded. Hardware ignores the contents after the last non-zero byte. The image below, Indirect data buffer for a slice illustrates these parameters for a slice data.

Indirect data buffer for a slice



MPEG2 Encoder PAK Commands

The MFC_MPEG2_PAK_INSERT_OBJECT Command is identical to the MFC_AVC_PAK_INSERT_OBJECT command as described in this document.

The MFC_MPEG2_STITCH_OBJECT Command is identical as MFC_AVC_STITCH_OBJECT command as described in this document.

MFC_MPEG2_SLICEGROUP_STATE

MFC_MPEG2_PAK_OBJECT

PAK Object Inline Data Description - MPEG-2

The Inline Data includes all the required MB encoding states, constitute part of the Slice Data syntax elements, MB Header syntax elements and their derivatives. It provides information for the following operations:

1. Forward and Inverse Transform
2. Forward and Inverse Quantization
3. Advanced Rate Control (QRC)
4. MB Parameter Construction (MPC)
5. VLC encoding
6. Bit stream packing
7. Internal error handling

These state/parameter values may subject to change on a per-MB basis, and must be provided in each MFC_MPEG2_PAK_OBJECT command. The values set for these variables are retained internally, until they are reset by hardware Asynchronous Reset or changed by the next MFC_MPEG2_PAK_OBJECT command.

The inline data has been designed to match AVC MB structure for efficient transcoding.

Current MB [x,y] address is not sent, it is assumed that the H/W will keep track of the MB count and current MB position internally.

DWord	Bit	Description
1	31:27	Reserved: MBZ
	22-20	**MvFormat (Motion Vector Size)**. This field specifies the size and format of the input motion vectors. This field is reserved (MBZ) when the **IntraMbFlag** = 1. The valid encodings are: **011 = Unpacked: Two motion vector pairs** Others are reserved. *(The following encodings are intended for other formats:* *001 = 1MV: one 16x16 motion vector* *010 = 2MV: One 16x16 motion vector pair* *011 = 4MV: Four 8x8 motion vectors, or Two 16x8 motion vector pairs* *100 = 8MV: Four 8x8 motion vector pairs* *101 = 16MV: 16 4x4 motion vectors* *110 = 32MV: 16 4x4 motion vector pairs* *111 = Packed, number of MVs is given by **packedMvNum**.)*
	19	**CbpDcY.** This field specifies if the Luma DC coded. Must be 1 for MPEG-2.
	18	**CbpDcU.** This field specifies if the Chroma Cb DC coded. Must be 1 for MPEG-2**.**
	17	**CbpDcV.** This field specifies if the Chroma Cb DC coded. Must be 1 for MPEG-2**.**
	16	Reserved: MBZ
	15	**TransformFlag** Used to indicate transformation type for MPEG-2. 0 = Frame DCT transformation 1 = Field DCT transformation
	14	**FieldMbFlag** For MPEG-2, this flag is set to 1 if either the picture is in field type or the MB is INTER of field type, i.e. split into two 16x8 field blocks.

DWord	Bit	Description
	13	**IntraMbFlag** This field specifies whether the current macroblock is an Intra (I) macroblock. For I-picture MB (IntraPicFlag =1), this field must be set to 1. This flag must be set in consistent with the interpretation of MbType (inter or intra modes). 0: INTER (inter macroblock) 1: INTRA (intra macroblock)
	12:8	**MbType** This field is encoded to match with the best macroblock mode determined as described in the next section. It follows an unified encoding for inter and intra macroblocks according to MFX Encoding reference as shown in Figure A.
	7:3	Reserved : MBZ
	2	**SkipMbFlag** By setting it to 1, this field forces an inter macroblock to be encoded as a skipped macroblock. It is equivalent to mb_skip_flag in AVS spec, Hardware honors input MVs for motion prediction and forces CBP to zero. By setting it to 0, an inter macroblock will be coded as a normal inter macroblock. The macroblock may still be coded as a skipped macroblock, according to the macroblock type conversion rules described in the later sub sections. This field can only be set to 1 for certain values of MbType. See details later. This field is only valid for an inter macroblock. Hardware ignores this field for an intra macroblock. 0 = not a skipped macroblock 1 = is coded as a skipped macroblock **Note:** When this flag is set to 1, the correct MVs are assumed for HW decoder to generate decoded reconstruction frame.
	1:0	**InterMbMode** This field is provided to carry redundant information as that encoded in MbType. This field is only valid if **IntraMbFlag** =0, otherwise, it is ignored by hardware.
2	31:16	**MbYCnt (Vertical Origin).** This field specifies the vertical origin of current macroblock in the destination picture in units of macroblocks. Format = U16 in unit of macroblock.
	15:0	**MbXCnt (Horizontal Origin).** This field specifies the horizontal origin of current macroblock in the destination picture in units of macroblocks.

DWord	Bit	Description
		Format = U16 in unit of macroblock.
3	31:24	**MaxSizeInWord** PAK should not exceed this budget accumulatively, otherwise it will trickle the PANIC mode.
	23:16	**TargetSizeInWord** PAK should use this budget accumulatively to decide if it needs to limit the number of non-zero coefficients.
	15:13	**MBZ**
	12:0	**Cbp -** Coded Block Pattern. This field specifies whether blocks are present or not. Format = 6-bit mask (or 8-bit, & 12-bit, for 422 and 444). Bit 11: Y0Bit 10: Y1Bit 9: Y2Bit 8: Y3 Bit 7: Cb4Bit 6: Cr5Bits 0-5: MBZ
4	31	**LastMbInSlice -** the last MB in a slice.
	30	**FirstMbInSlice -** the first slice in a slice, it requires slice header insertion.
	29:28	**MBZ**
	27	**EnableCoeffClamp** 1 = the magnitude of coefficients of the current MB will be clamped based on the clamping matrix after quantization 0 = no clamping
	26	**LastMbInSG** 1 - the current MB is the last MB in the current slice group.
	25	**MbSkipConvDisable** This is a per-MB level control to enable and disable skip conversion. This field is ORed with SkipConvDisable field. This field is only valid for a P or B slice. It must be zero for other slice types. Rules are provided in Macroblock Type Conversion Rules. 0 - Enable skip type conversion for the current macroblock 1 - Disable skip type conversion for the current macroblock
	24	**FirstMbInSG** 1 - the current MB is the last MB in the current slice group.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="4"></td><td>23:20</td><td>MBZ</td></tr>
<tr><td>19:16</td><td>MvFieldSelect - Motion Vertical Field Select. A bit-wise representation of a long [2][2] array as defined in Section 6.3.17.2 of the ISO/IEC 13818-2 (see also Section 7.6.4).
<table>
<tr><th>Bit</th><th>MVector[r]</th><th>MVector[s]</th><th>MotionVerticalFieldSelect Index</th></tr>
<tr><td>16</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>17</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>18</td><td>1</td><td>0</td><td>2</td></tr>
<tr><td>19</td><td>1</td><td>1</td><td>3</td></tr>
</table>
Format = MC_MotionVerticalFieldSelect.
0 = The prediction is taken from the top reference field.
1 = The prediction is taken from the bottom reference field.</td></tr>
<tr><td>15:5</td><td>MBZ Reserved</td></tr>
<tr><td>4:0</td><td>QpScaleCode</td></tr>
<tr><td rowspan="2">5</td><td>31:16</td><td>MV[0][0].y - the y coordinate of the first forward MV
if Mv[0][0] n/a:
if Mv[1][0] available, it MUST be set to the same value as Mv[1][0].
else it MUST be set to the value 0</td></tr>
<tr><td>15:0</td><td>MV[0][0].x - the x coordinate of the first forward MV
if Mv[0][0] n/a:
if Mv[1][0] available, it MUST be set to the same value as Mv[1][0].
else it MUST be set to the value 0</td></tr>
<tr><td>6</td><td>31:0</td><td>MV[1][0] - the first backward MV
if Mv[1][0] n/a: it MUST be set to the same value as Mv[0][0]</td></tr>
<tr><td>7</td><td>31:0</td><td>MV[0][1] - the second forward MV
if Mv[0][1] n/a:
if Mv[1][1] available, it MUST be set to the same value as Mv[1][1].
else it MUST be set to the same value as Mv[0][0]</td></tr>
<tr><td>8</td><td>31:0</td><td>MV[1][1] - the second backward MV</td></tr>
</table>

DWord	Bit	Description
		if Mv[1][1] n/a: it MUST be set to the same value as Mv[1][0]

The mapping between MPEG-2 spec and MfxMbCode can be achieved according to the following:

1) Renamed variables with identical meaning:

MPEG-2 Spec	MFX API	Value
macroblock_quant	**MbQuantPresent**	0 or 1
macroblock_intra	**IntraMbFlag**	0 or 1
dct_type	**Transform8x8Flag**	0 or 1
macroblock_pattern	**Cbp8x8**	remapped

2) Macroblock type remapping:

		PRM Entry					MPEG-2 Spec				
Frame Type	Mb Type	Intra Mb Flag	Skip Mb Flag	Mb Type 5Bits	Field Mb Flag	Inter Mb Mode	macroblock_intra	motion_type_bit0	motion_type_bit1	motion_forward	motion_backward
IPB	Intra	1	0	1Ah	0/1	-	1	-	-	-	-
P **B** B	Skip	0	1	01h 02h 03h	0/1	0	0	-	-	1 0 1	0 1 1
P	0-MV*	0	0	01h	0/1	0	0	-	-	0	0
P Frame	Frame type	0	0	01h	0	0	0	0	1	1	0
P Frame	Field type	0	0	04h	1	1	0	1	0	1	0
P Frame	dual prime	0	0	19h	0	0	0	1	1	1	0

		PRM Entry					MPEG-2 Spec				
Frame Type	Mb Type	Intra Mb Flag	Skip Mb Flag	Mb Type 5Bits	Field Mb Flag	Inter Mb Mode	macroblock_intra	motion_type_bit0	motion_type_bit1	motion_forward	motion_backward
P Field	One 16x16	0	0	01h	1	0	0	1	0	1	0
P Field	Two 16x8	0	0	04h	1	1	0	0	1	1	0
P Field	dual prime	0	0	19h	1	0	0	1	1	1	0
B Frame	Frame type	0	0	01h 02h 03h	0	0	0	0	1	1 0 1	0 1 1
B Frame	Field type	0	0	04h 06h 14h	1	1	0	1	0	1 0 1	0 1 1
B Field	One 16x16	0	0	01h 02h 03h	1	0	0	1	0	1 0 1	0 1 1
B Field	Two 16x8	0	0	04h 06h 14h	1	1	0	0	1	1 0 1	0 1 1

- Notice that there is no special way to indicate 0 motion vector case for P frame. It is for PAK to handle internally by checking up the motion vector values.
- Notice also, the MbType5bits is adapted from AVC DXVA macroblock types. It may seems awkward from MPEG-2 perspective, but provides a common VME interface for us for simpler HW design and help the advanced transcoding solution.

MFX HW Interface and DXVA Conversion

Map DXVA to HW BSpec

Location		HW BSPEC	
BYTE	Dword	MPEG-2	DXVA
DW0			
0		MbMode	
0.0-1	0[0-1]	InterMbMode	see (A)
0.2	0[2]	SkipMbFlag	<-MBskipsFollowing
0.3	0[3]	mbz	
0.4-0.5	0[4-5]	IntraMbMode	IntraMacroblock
0.6	0[6]	mbz	
0.7	0[7]	FieldMbPolarity	derived
1		MbType	
1.0-1.4	0[8-12]	MbType5Bits	see (A)
1.5	0[13]	IntraMbFlag	IntraMacroblock
1.6	0[14]	FieldMbFlag	see (A)
1.7	0[15]	TransformFlag	FieldResidual
2		MbFlag	
2.0	0[16]	ResidDataFlag	HostResDiff
2.1	0[17]	CbpDcV	PAK control
2.2	0[18]	CbpDcU	PAK control
2.3	0[19]	CbpDcY	PAK control
2.4-2.6	0[20-22]	MvFormat	= 3, derived
2.7	0[23]	mbz	
3	0[24-31]	PackedMvNum	see (A)
DW1			
4-5	1[0-15]	MbXCnt	wMBaddress
6-7	1[16-31]	MbYCnt	wMBaddress
DW2			
8	2[0-7]		bNumCoef[0]
8.0-8.5	2[0-5]	mbz	
8.6-8.7	2[6-7]	CbpAcUV	PAK control
9	2[8-11]	CbpAcY	PAK control
	2[12-15]	mbz	
10	2[16-23]	TargetedSzInWord	

Location		HW	
		BSPEC	
BYTE	Dword	MPEG-2	DXVA
11	2[24-31]	**MaxSzInWord**	
DW3			
12		**Qscale**	derived
12.0-6	3[0-6]	**QScaleCode**	
12.7	3[7]	**QScaleType**	
13	3[8-15]	**mbz**	
14	3[16-19]	**MvFieldSelect**	MvertFieldSel
	3[20-23]	**mbz**	
15		**MbExtFlag**	
15.0	3[24]	**mbz**	
15.1	3[25]	**SkipMvConvDisable**	
15.2	3[26]	**LastMbFlag**	PAK control
15.3	3[27]	**EnableCoeffClamp**	PAK control
15.4-5	3[28-29]	**MbScanMethod**	MBscanMethod
15.6	3[30]	**NewSliceFlag**	PAK control
15.7	3[31]	**EndSliceFlag**	PAK control
DW4-7			
16-32	4-7[all]	**MV[2][2][2]**	MVector[4][2]

(A): Set **InterMbMode, MbType5bits, FieldMbFlag**, and **PackedMvNum** from DXVA parameters:

```
 if(IntraMacroblock) return (TYPE_INTRA);
 else if(MotionType==3){ // dual prime
    MbType5bits = 0x19; FieldMbFlag = 0; InterMbMode = 0; PackedMvNum = 2;     return
(DUAL_PRIME);
 }
 else{
    IsFieldFrame = a PicState derivative;     switch(MotionType+IsFieldFrame{
       case 1: // Two 16x8 field in Frame Frame
       case 3: // Two 16x8 field in Field Frame
          FieldMbFlag = 1; InterMbMode= 1;          switch(MotionForward |Motionbackward «1)){
             case 1:
                MbType5bits = 4; PackedMvNum = 2;                break;
             case 2:
                MbType5bits = 6; PackedMvNum = 2;                break;
             case 3:
                MbType5bits = 0x14; PackedMvNum = 4;                break;
          }
          break;
       case 2: // 16x16 block in either case
          FieldMbFlag = IsFieldFrame; InterMbMode = 0;
switch(MotionForward|(Motionbackward«1)){
             case 1:
                MbType5bits = 1; PackedMvNum = 1;                break;
             case 2:
                MbType5bits = 2; PackedMvNum = 1;                break;
             case 3:
```

```
                MbType5bits = 3; PackedMvNum = 2;                    break;
            }
            break;
        }
    }
```

Map HW Bspec to DXVA

Location	BSPEC	
BYTE	**DXVA**	**MPEG-2**
0-1	wMBaddress	**= MbYCnt*MbW + MbXCnt**
2-3	wMBtype	
2.0	IntraMacroblock	**= IntraMbFlag**
2.1	MotionForward	see (B)
2.2	MotionBackward	see (B)
2.3	Motion4MV	VC-1 only, MBZ for Mpeg-2
2.4	Reserved	
2.5	FieldResidual	**= TranformFlag**
2.6-2.7	MBscanMethod	**= MbScanMethod**
3.0-3.1	MotionType	see (B)
3.2	HostResDiff	**= ResidDataFlag**
3.3	Reserved	
3.4-3.7	MvertFieldSel	**= MvFieldSelect**
4	MBskipsFollowing	count **SkipMbFlag**
5-7	MBdataLocation	n/a
8-9	wPatternCode	**= CbpAcY|UV**
10-15	bNumCoef[6]	n/a
16-32	MVector[4][2]	**= MV[2][2][2]**

(B): Set **MBtype** and **MotionType** from Bspec interface

```
if(MbIntraFlag) return (TYPE_INTRA);
else {
    if(MbType5Bits&8) { // dual prime
        MotionForward  = 1;
        MotionBackward = 0;
        MotionType     = 3;
        return (DUAL_PRIME);
    }
    else {
        // redundant: InterMbMode = !!(MbType5Bits&4);
        if(InterMbMode) {
            MotionForward  = !(MbType5Bits&2);
            MotionBackward = !!(MbType5Bits&0x12);
        }
```

```
        else {
            MotionForward  = (MbType5Bits&1);
            MotionBackward = !!(MbType5Bits&2);
        }
        MotionType = 2-(InterMbMode^FieldMbFlag);

        // equivalently the 2 bits are:
        // MotionType0 = (InterMbMode^FieldMbFlag);
        // MotionType1 = ~MotionType0;

        return (TYPE_INTER);
    }
}
```

Video Codec VC-1

This section describes support for the open video compression standard VC-1, which is the common name for SMPTE 421M approved on April 3, 2006.

VC1 Common Commands

MFX Commands are organized into groups based on their scope of functioning. There are Pipeline Common state commands that are common to all codecs (encoder and decoder) and is applicable to the processing of one full frame/field. There are also individual codec Common state commands that are common to both encoder and decoder of that particular codec. These latter common state commands, some are applicable at the processing of one full frame/field, and some are applicable at the processing of an individual slice level.

MFX_VC1_PRED_PIPE_STATE

MFX_VC1_DIRECTMODE_STATE

VC1 Decoder Commands

These are decoder-only commands. They provide the pointer to the compressed input bitstream to be decoded.

MFD_VC1_LONG_PIC_STATE

AltPQuantConfig and **AltPQuantEdgeMask** are derived based on the following variables: *DQUANT*, *DQUANTFRM*, *DQPROFILE*, *DQSBEDGE*, *DQDBEDGE*, and *DQBILEVEL* defined in the VC1 standard, as shown in the following table.

Definition of AltPQuantConfig and AltPQuantEdgeMask

Inputs						Outputs		Description
DQUANT	DQUANT FRM	DQ PROFILE	DQDB EDGE	DQSB EDGE	DQBI LEVEL	AltPQuant Config	AltPQuant EdgeMask	
0	-	-	-	-	-	00b	0000b	No **AltPQuant**
1	0	-	-	-	-	00b	0000b	No **AltPQuant**
1	1	11b	-	-	0	10b	0000b	All MBs are different with *MQDIFF* and *ABSMQ*
1	1	11b	-	-	1	11b	0000b	All MBs may switch with 1-bit *MQDIFF*
2	-	-	-	-	-	01b	1111b	All edge MBs
1	1	00b	-	-	-	01b	1111b	All edge MBs
1	1	01b	00b	-	-	01b	0011b	Left and top MBs
1	1	01b	01b	-	-	01b	0110b	Top and right MBs
1	1	01b	10b	-	-	01b	1100b	Right and bottom MBs
1	1	01b	11b	-	-	01b	1001b	Bottom and left MBs
1	1	10b	-	00b	-	01b	0001b	Left MBs
1	1	10b	-	01b	-	01b	0010b	Top MBs
1	1	10b	-	10b	-	01b	0100b	Right MBs
1	1	10b	-	11b	-	01b	1000b	Bottom MBs

MFD_VC1_SHORT_PIC_STATE

Intel HW does not use the MVMODE and MVMODE2 provided at the revised DXVA2 VC1 VLD interface, instead, HW will decode them directly from the bitstream picture header.

MFD_VC1_BSD_OBJECT

For VC1, a slice/picture is always started with MB x position equal to 0. Hence, no need to include in the Object Command.

Handling Emulation Bytes

In general, VC1 BSD unit is capable of handling emulation prevention bytes. However, there is a corner case that requires host software's intervention. Host software needs to overwrite the emulation byte if it overlaps the macroblock layer decode and there is not enough information for the hardware to detect the emulation byte.

The emulation bytes might have an overlap between the picture states and the first macroblock data. If the emulation bytes are 0x00 **0x000x03** 0x00 and the macroblock data starts in the middle of byte1 (**0x00**), then the host software needs to overwrite the **0x03** byte location with the previous byte (**0x00**) and change the byte offset accordingly. The hardware wouldn't know what the 1st byte was and will miss this **0x03** removal.

JPEG and MJPEG

JPEG Decoder Commands

Following are JPEG Decoder Commands:

MFD_JPEG_BSD_OBJECT

MFX_JPEG_PIC_STATE command is used for both encoding and decoding. Note the duplicate bits and the "Exists If" rows that specify what the bits represent for Encoder and Decoder.

MFX_JPEG_PIC_STATE

For JPEG decoding, the following program note is informative.

For **Rotation**, it is important to note that rotation of 90 or 270 degrees also requires exchanging **FrameWidthInBlksMinus1** with **FrameHeightInBlksMinus1** in the command. In addition, the rotation of 90 or 270 degrees also requires transportation of the quantization matrix will be transposed into the position (y, x).

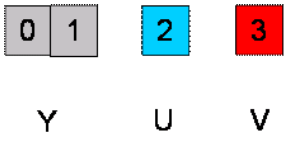
Chroma type is determined by the values of horizontal and vertical sampling factors of the components (*Hi* and Vi where *i* is a component id) in the Frame header as shown in the following table.

	H1	H2	H3	V1	V2	V3
0: YUV400	r	Not available	Not available	r	Not available	Not available
1: YUV420	2	1	1	2	1	1
2: YUV422H_2Y	2	1	1	1	1	1
3: YUV444	1	1	1	1	1	1
4: YUV411	4	1	1	1	1	1
5: YUV422V_2Y	1	1	1	2	1	1
6: YUV422H_4Y	2	1	1	2	2	2
7: YUV422V_4Y	2	2	2	2	1	1

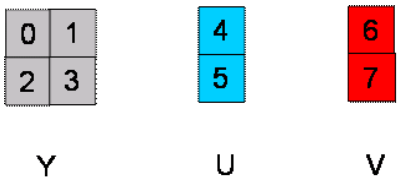
For YUV400, the value of *V1* can be 1, 2, or 3 and will be same as the value of *H1*, and the Minimum coded unit (MCU) is one 8x8 block. For the other chroma formats, if non-interleaved data, the MCU is one 8x8 block. For interleaved data, the MCU is the sequence of block units defined by the sampling factors of the components.

For example, the following figures show the MCU structures of interleaved data and the decoding order of blocks in the MCU:

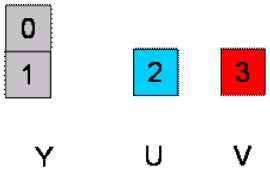
422H_2Y



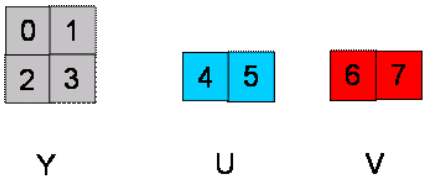
422H_4Y



422V_2Y

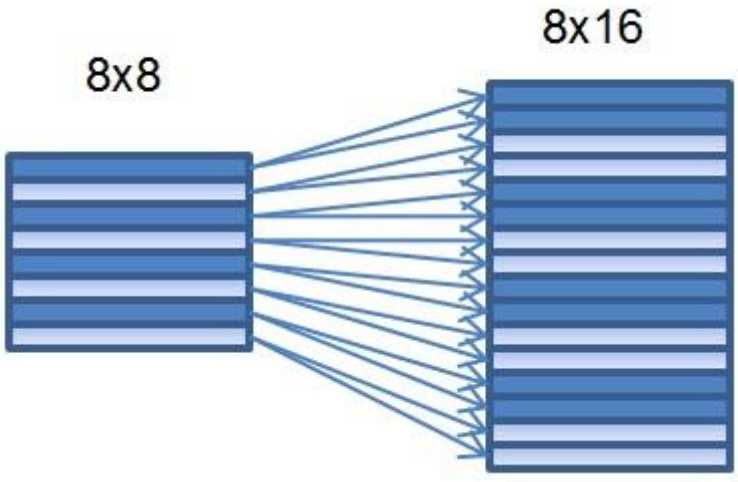


422V_4Y

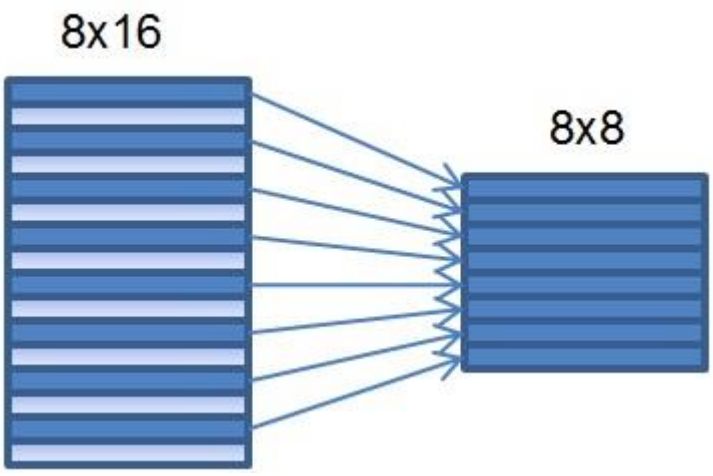


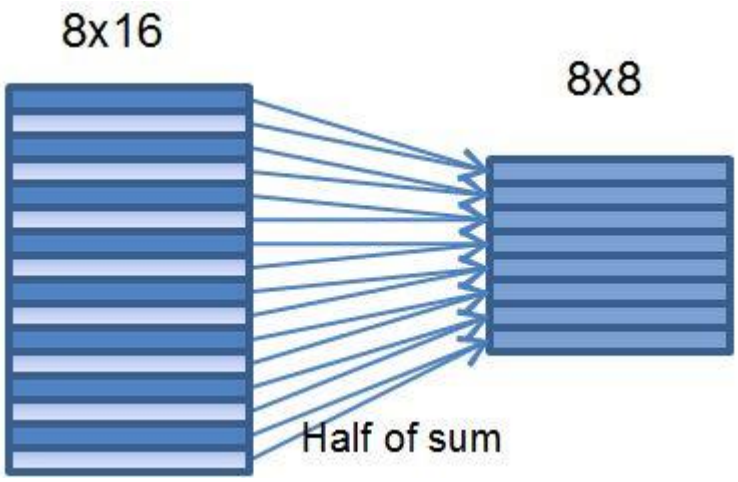
If picture width *X* in the Frame header is not a multiple of 8, the decoding process needs to extend the number of columns to complete the right-most sample blocks. If the component is to be interleaved, the decoding process needs to extend the number of samples by one or more additional blocks so that the number of blocks is an integer multiple of *Hi*. In other words, "The number of blocks in width" in the table should be an integer multiple of (8x*H1*). Similarly, if picture height *Y* in the Frame header is not a multiple of 8, the decoding process needs to extend the number of lines to complete bottom-most block-row. If the component is to be interleaved, the decoding process also needs to extend the number of lines by one or more additional block-rows so that the number of block-row is an integer multiple of (8x*V1*). For example, if non-interleaved YUV411 with *X*=270, then "The number of blocks in width" shall be (270 + 7) / 8 = 34, where "/" is integer division. Therefore, **FrameWidthInBlksMinus1** is set to 33. However, for interleaved data, "The number of blocks in width" shall be ((270 + 31) / 32) x 4 = 36. Therefore, **FrameWidthInBlksMinus1** is set to 35.

VertUpSamplingEnb is used to convert an input chroma420 to an output chroma422 in the surface format of YUY2 or UYVY. To enable this flag, the input should be interleaved Scan, InputFormatYUV should be set to YUV420, and OutputFormatYUV should be set to YUY2 or UYVY. Vertical 2:1 up-sampling is only applied to chroma blocks where each line of 8x8 block pixels is replicated to make 8x16 U/V blocks. For example:



VertDownSamplingEnb is used to convert an input chroma422 to an output chroma420 in the surface format NV12. To enable this flag, the input should be interleaved Scan, InputFormatYUV should be set to YUV422H_2Y or YUV422H_4Y, and OutputFormatYUV should be set to NV12. Combined with AvgDownSampling flag, the following table and figures show the down-sampling methods.

VertDownSamplingEnb	AvgDownSampling	Down-Sampling Methods
0	0 or 1	No down-sampling.
1	0	Drop every other line: 
1	1	Average vertically neighboring two pixels:

VertDownSamplingEnb	AvgDownSampling	Down-Sampling Methods
		

The recent history forJPEG Decoder Commands are described in the following:

- If the InputFormat is YUV400 or YUV444 or YUV411, then output cannot be NV12, YUY2 or UYVY, it has to be planar. But for 420 and 422 InputFormat, there's a choice of having Planar, NV12, YUY2 or UYVY OutputFormat. And the surface state should be programmed accordingly.
- Refer "Output Format YUV" field for more details.

MFX_JPEG_HUFF_TABLE_STATE

JPEG Encoder Commands

JPEG Encoder Command Sequence:

Commands
MFX_PIPE_MODE_SELECT
MFX_SURFACE_STATE
MFX_PIPE_BUF_ADDR_STATE
MFX_IND_OBJ_BASE_ADDR_STATE
MFX_JPEG_PIC_STATE
MFX_FQM_STATE (One each for Luma, CB and CR)
MFC_JPEG_ HUFF_TABLE_STATE(Huffman table 0 and 1 need two commands to be issued).
MFC_JPEG_SCAN_OBJECT
MFX_PAK_INSERT_OBJECT (Multiple commands can be given based on the need)

Following are JPEG Encoder Commands:

MFX_JPEG_PIC_STATE command is used for both encoding and decoding. Note the duplicate bits and the "Exists If" rows that specify what the bits represent for Encoder and Decoder.

MFX_JPEG_PIC_STATE

Programming Note: For completion of partial MCUs in JPEG encoding, it is important to note the following:

If the image's dimensions are not an exact multiple of the MCU size, the encoded data should include padding to round up to the next complete MCU, which is called completion of partial MCU. If the number of lines is not aligned with MCU structure (not a multiple of MCU size, i.e. 8, 16, 32), the encoding process needs to extend the number of lines to complete the bottom-most MCU-row. Similarly, if the number of samples per line is not aligned with MCU structure, the encoding process needs to extend the number of columns to complete the right-most sample MCUs. JPEG standard recommends that any incomplete MCUs be completed by replication of the right-most column and the bottom line of each component Y, U, and V.

The following equations are used to set the command for encoding partial MCUs.

FrameWidthInBlksMinus1 = (((*X* + (*H*1 *8 -1)) / (*H*1 *8)) * *H*1) - 1

FrameHeightInBlksMinus1 = (((*Y* + (*V*1*8 -1)) / (*V*1*8)) * *V*1) - 1

```
For YUV400,
    PixelsInHoriLastMCU = X % 8
    PixelsInVertLastMCU = Y % 8

For YUV420,
    PixelsInHoriLastMCU = X % 16 if X % 2 = 0, ((X % 16) + 1) % 16 if X % 2 = 1
    PixelsInVertLastMCU = Y % 16 if Y % 2 = 0, ((Y % 16) + 1) % 16 if Y % 2 = 1

For YUV422H_2Y,
    PixelsInHoriLastMCU = X % 16 if X % 2 = 0, ((X % 16) + 1) % 16 if X % 2 = 1
    PixelsInVertLastMCU = Y % 8

    X: the number of samples per line in Y-image

    Y: the number of lines in Y-image

    H1: horizontal sampling factor of Y-image in the Frame header

    V1:  vertical sampling factor of Y-image in the Frame header
```

Note that PixelsInHoriLastMCU=0 does not mean the num of pixels in the right-most MCUs = 0, but does mean that the right-most MCUs are fully filled with pixels, i.e., not a partial MCU.

For example, for input image dimension 17x26 pixels and an interleaved Scan, the following equations and the table show how to set the command for each OutputMcuStructure.

	YUV400	YUV420	YUV422H_2Y
MCU size of Y	8x8	16x16	16x8
MCU size of U and V	8x8	8x8	8x8
*H*1 and *V*1	1, 1	2, 2	2, 1
FrameWidthInBlksMinus1	2	3	3
FrameHeightInBlksMinus1	3	3	3
PixelsInHoriLastMCU	1	2	2

	YUV400	YUV420	YUV422H_2Y
PixelsInVertLastMCU	2	10	2

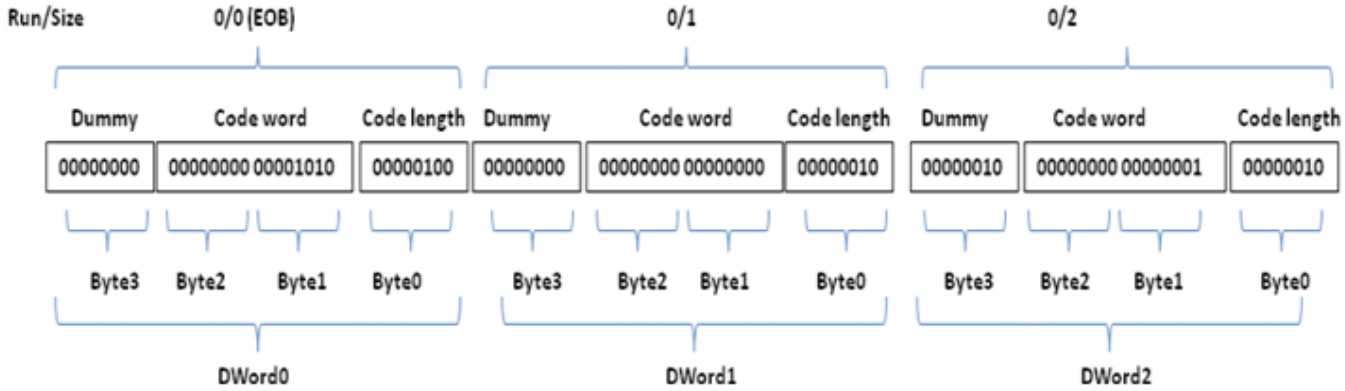
MFC_JPEG_SCAN_OBJECT

The JPEG standard Table K.5 shows the real table of code length and code word as follows:

MFC_JPEG_ HUFF_TABLE_STATE

Run/Size	Code length	Code word
0/0 (EOB)	4	1010
0/1	2	00
0/2	2	01
0/3	3	100
0/4	4	1011
0/5	5	11010
0/6	7	1111000
0/7	8	11111000
0/8	10	1111110110
0/9	16	1111111110000010
0/A	16	1111111110000011

It is not necessary to send Run/size in the command as driver will send the increasing order of run/size. Each symbol aligns to a DWord with the following byte structure. Each DWord (a group of 4 bytes) contains Byte0 for Code length, Byte1 and Byte2 for Code word, and Byte3 for dummy.



Driver will program to always send 12 pairs of Code length and Code Word in DC coefficient table and 162 pairs in AC coefficient table. When a Huffman table contains valid full entries of Run/Size, all the Code word and Code length will not be zero. If a Huffman table is customized or optimized, the table can contain smaller set of Code length and Code Word, i.e., the number of entries of the real Huffman table will be less than 12 for DC, or less than 162 for AC. For the customized Huffman table, driver will set the missing entry (Run/Size) to Code length = 0 and Code word = 0.

MFX_PAK_INSERT_OBJECT

More Decoder and Encoder

MFD IT Mode Decode Commands

These are decoder-only commands to support the IT-mode specified in DXVA interface.

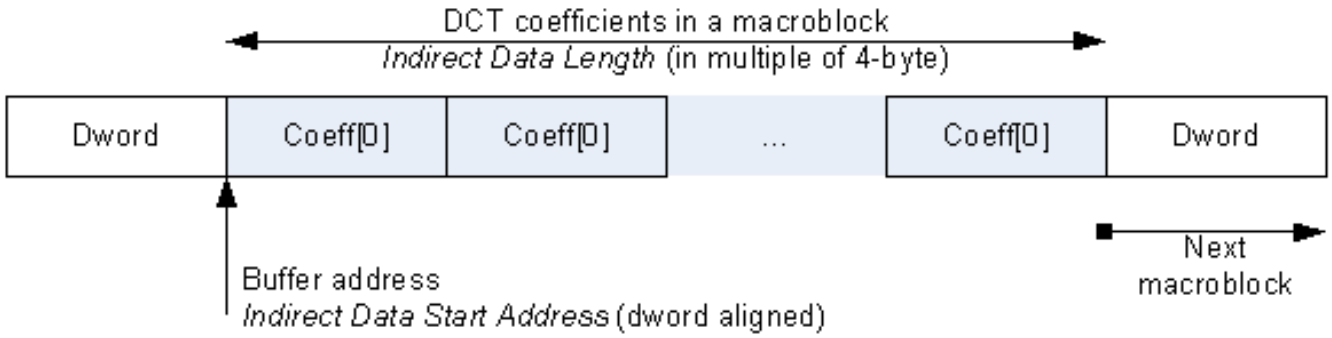
MFD_IT_OBJECT

Common Indirect IT-COEFF Data Structure

Transform-domain residual data block in AVC-IT, VC1-IT and MPEG2-IT mode follows the same data structure.

The indirect IT-COEFF data start address in MFD_IT_OBJECT command specifies the doubleword aligned address of the first non-zero DCT coefficient of the first block of the macroblock. Only the non-zero coefficients are present in the data buffer and they are packed in the 8x8 block sequence of Y0, Y1, Y2, Y3, Cb4 and Cr5, as shown in Structure of the IDCT Compressed Data Buffer. When an 8x8 block is further subdivided into 4x4 subblocks, the coefficients, if present, are organized in the subblock order. The smallest subblock division is referred to as a **transform block**. The indirect IT-COEFF data length in the command includes all the non-zero coefficients for the macroblock. It must be doubleword aligned.

Structure of the IDCT Compressed Data Buffer



Each non-zero coefficient in the indirect data buffer is contained in a doubleword-size data structure consisting of the coefficient index, end of block (EOB) flag and the fixed-point coefficient value in 2's compliment form. As shown in Structure of a transform-domain residue unit, *index* is the row major 'raster' index of the coefficient **within a transform block** (*please note that it is not converted to 8x8 block basis*). A coefficient is a 16-bit value in 2's complement.

Structure of a transform-domain residue unit

DWord	Bit	Description
0	31:16	**Transform-Domain Residual (coefficient) Value.** This field contains the value of the non-zero transform-domain residual data in 2's compliment.
	15:7	Reserved: MBZ
	6:1	**Index.** This field specifies the raster-scan address (raw address) of the coefficient within the transform block. For a coefficient at Cartesian location (row, column) = (y, x) in a transform block of width W, Index is equal to (y * W + x). For example, coefficient at location (row, column) = (0, 0) in a 4x4 transform block has an index of 0; that at (2, 3) has an index of 2*4 + 3 = 11. The valid range of this field depends on the size of the transform block. Format = U6 Range = [0, 63]
	0	**EOB (End of Block).** This field indicates whether the transform-domain residue is the last one of the current transform block.

Allowed transform block dimensions per coding standard

Transform Block Dimension	AVC	VC1	MPEG2
8x8	Yes	Yes	Yes
8x4	No	Yes	No
4x8	No	Yes	No
4x4	Yes	Yes	No

For AVC, there is intra16x16 mode, in which the DC Luma coefficients of all 4x4 sub-blocks within the current MB are sent separately in its own 4x4 Luma block. As such, only 15 coefficients remains in each of the 16 4x4 Luma blocks.

Inline Data Description in AVC-IT Mode

The Inline Data includes all the required MB decoding states, extracted primarily from the Slice Data, MB Header and their derivatives. It provides information for the following operations:

1. Inverse Quantization
2. Inverse Transform
3. Intra and inter-Prediction decoding operations
4. Internal error handling

IT Mode supports only packed MV data as specified in the DXVA spec.

These state/parameter values may subject to change on a per-MB basis, and must be provided in each MFD_IT_OBJECT command. The values set for these variables are retained internally, until they are reset by hardware Asynchronous Reset or changed by the next MFC_AVC_PAK_OBJECT command.

The inline data has been designed to match the DXVA 2.0, with the exception of the starting byte (DW0:0-7) and the ending dword (DW7:0-31).

The Deblocker Filter Control flags (FilterInternalEdgesFlag, FilterTopMbEdgeFlag and FilterLeftMbEdgesFlag) are generated by H/W, which are depending on MbaffFrameFlag, CurrMbAddr, PicWidthInMbs and disable_deblocking_filter_idc states.

Current MB [x,y] address is not sent, it is assumed that the H/W will keep track of the MB count and current MB position internally.

DWord	Bit	Description
0	31:24	**MvQuantity** Specify the number of MVs (in unit of motion vector, 4 bytes each) to be fetched for motion compensation operation. Decoder IT mode only supports packed MV format (DXVA). This field specifies the exact number of MVs present for the current MB. For a P-Skip MB, there is still 1 MV being sent (Skip MV is sent explicitly); for a B-Direct/Skip MB, there are 2 MVs being sent. For an Intra-MB, MvQuantity is set to 0. MvQuantity = 0, signifies there is no MV indirect data for the current MB. This field must be set in consistent with **Indirect MV Data Length**, so as not to exceed its bound Unsigned.
	23:20	Reserved MBZ (DXVA)
	19	**DcBlockCodedYFlag** 1 - the 4x4 DC-only Luma sub-block of the Intra16x16 coded MB is present; it is still possible that all DC coefficients are zero. 0 - no 4x4 DC-only Luma sub-block is present; either not in Intra16x16 MB mode or all DC coefficients are zero.
	18	**DcBlockCodedCbFlag** For 4:2:0 case : 1 - the 2x2 DC-only Chroma Cb sub-block of all coded MB (any type) is present; it is still possible that all DC coefficients are zero. 0 - no 2x2 DC-only Chroma Cb sub-block is present; all DC coefficients are zero.
	17	**DcBlockCodedCrFlag**

DWord	Bit	Description
		For 4:2:0 case : 1 - the 2x2 DC-only Chroma Cr sub-block of all coded MB (any type) is present; it is still possible that all DC coefficients are zero. 0 - no 2x2 DC-only Chroma Cr sub-block is present; all DC coefficients are zero.
	16	Reserved MBZ (DXVA)
	15	**Transform8x8Flag** 0: indicates the current MB is coded with 4x4 transform and therefore the luma residuals are presented in 4x4 blocks. 1: indicates the current MB is coded with 8x8 transform and therefore the luma residuals are presented in 8x8 blocks. Same as the transform_szie_8x8_flag syntax element in AVC spec.
	14	**MbFieldFlag** This field specifies whether current macroblock is coded as a field or frame macroblock in MBAFF mode. 1 = Field macroblock 0 = Frame macroblock This field is exactly the same as FIELD_PIC_FLAG syntax element in non-MBAFF mode. Same as the mb_field_decoding_flag syntax element in AVC spec.
	13	**IntraMbFlag** This field specifies whether the current macroblock is an Intra (I) macroblock. 0 - not an intra MB 1 - is an intra MB I_PCM is considered as Intra MB. For I-picture MB (IntraPicFlag =1), this field must set to 1. This flag must be set in consistent with the interpretation of MbType (inter or intra modes).
	12:8	**MbType** This field carries the Macroblock Type. The meaning depends on IntraMbFlag. If IntraMbFlag is 1, this field is the intra macroblock type as defined in MbType definition for Intra Macroblock . If IntraMbFlag is 0, this field is the inter macroblock type as defined in the first two columns of MbType definition for Inter Macroblock (and MbSkipflag = 0). All macroblock types in a P Slice are mapped into the corresponding types in a B Slice. Skip and Direct modes are converted into its corresponding processing modes.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td></td><td></td><td>Programming note: It is exactly matched with that of DXVA 2.0.</td></tr>
<tr><td></td><td>7</td><td>FieldMbPolarityFlag
This field indicates the field polarity of the current macroblock.
Within a MbAff frame picture, this field may be different per macroblock and is set to 1 only for the second macroblock in a MbAff pair if FieldMbFlag is set. Otherwise, it is set to 0.
Within a field picture, this field is set to 1 if the current picture is the bottom field picture. Otherwise, it is set to 0. It is a constant for the whole field picture.
This field is only valid for MBAFF frame picture. It is reserved and set to 0 for a progressive frame picture or a field picture.
0 = Current macroblock is a field macroblock from the top field (first in a MBAFF pair)
1 = Current macroblock is a field macroblock from the bottom field (second in a MBAFF pair)</td></tr>
<tr><td></td><td>6</td><td>IsLastMB
1 - the current MB is the last MB in the current Slice
0 - the current MB is not the last MB in the current Slice</td></tr>
<tr><td></td><td>5-4</td><td>Reserved MBZ (Intel encoder)</td></tr>
<tr><td></td><td>3:0</td><td>Reserved MBZ (DXVA Decoder)</td></tr>
<tr><td>1</td><td>31:16</td><td>CbpY[bit 15:0] (Coded Block Pattern Y)
For 4x4 sub-block (when Transform8x8flag = 0 or in intra16x16) :
16-bit cbp, one bit for each 4x4 Luma sub-block (not including the DC 4x4 Luma block in intra16x16) in a MB. The 4x4 Luma sub-blocks are numbered as
<table>
<tr><td>blk0</td><td>1</td><td>4</td><td>5</td><td>bit15</td><td>14</td><td>11</td><td>10</td></tr>
<tr><td>blk2</td><td>3</td><td>6</td><td>7</td><td>bit13</td><td>12</td><td>9</td><td>8</td></tr>
<tr><td>blk8</td><td>9</td><td>12</td><td>13</td><td>bit7</td><td>6</td><td>3</td><td>2</td></tr>
<tr><td>blk10</td><td>11</td><td>14</td><td>15</td><td>bit 5</td><td>4</td><td>1</td><td>0</td></tr>
</table>
The cbpY bit assignment is cbpY bit [15 - X] for sub-block_num X.
For 8x8 block (when Transform8x8flag = 1)
Only the lower 4 bits [3:0] are valid; the remaining upper bits [15:4] are ignored. The 8x8 Luma blocks are numbered as
<table>
<tr><td>blk0</td><td>1</td><td>bit3</td><td>2</td></tr>
<tr><td>blk2</td><td>3</td><td>bit1</td><td>0</td></tr>
</table>
The cbpY bit assignment is cbpY bit [3 - X] for block_num X.
0 in a bit - indicates the corresponding 8x8 block or 4x4 sub-block is not present (because all coefficient values are zero)</td></tr>
</table>

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td></td><td></td><td>1 in a bit - indicates the corresponding 8x8 block or 4x4 sub-block is present (although it is still possible to have all its coefficients be zero - bad coding).</td></tr>
<tr><td></td><td>15:8</td><td>VertOrigin (Vertical Origin). This field specifies the vertical origin of current macroblock in the destination picture in units of macroblocks.

For field macroblock pair in MBAFF frame, the vertical origins for both macroblocks should be set as if they were located in corresponding field pictures. For example, for field macroblock pair originated at (16, 64) pixel location in an MBAFF frame picture, the Vertical Origin for both macroblocks should be set as 2 (macroblocks). Whether the current macroblock is the first/second (top/bottom) in a MBAFF pair is specified by FieldMbPolarityFlag.

The macroblocks with (VertOrigin, HorzOrigin) must be delivered in the strict order as coded in the bitstream (raster order for progressive frame or field pictures and MBAFF pair order for MBAFF pictures). No gap is allowed. Otherwise, hardware behavior is undefined.

Format = U8 in unit of macroblock.</td></tr>
<tr><td></td><td>7:0</td><td>HorzOrigin (Horizontal Origin). This field specifies the horizontal origin of current macroblock in the destination picture in units of macroblocks.

Format = U8 in unit of macroblock.</td></tr>
<tr><td>2</td><td>31:16</td><td>CbpCr (Coded Block Pattern Cr 4:2:0-only)

Only the lower 4 bits [3:0] are valid; the remaining upper bits [15:4] are ignored (only valid for 4:2:2 and 4:4:4). The 4x4 Chroma Cr sub-blocks are numbered as
<table><tr><td>blk0</td><td>1</td><td>bit3</td><td>2</td></tr><tr><td>blk2</td><td>3</td><td>bit1</td><td>0</td></tr></table>
The cbpCr bit assignment is cbpCr bit [3 - X] for sub-block_num X.

0 in a bit - indicates the corresponding 4x4 sub-block is not present (because all coefficient values are zero)

1 in a bit - indicates the corresponding 4x4 sub-block is present (although it is still possible to have all its coefficients be zero - bad coding).

For monochrome, this field is ignored.</td></tr>
<tr><td></td><td>15-0</td><td>CbpCb (Coded Block Pattern Cb 4:2:0-only)

Only the lower 4 bits [3:0] are valid; the remaining upper bits [15:4] are ignored (only valid for 4:2:2 and 4:4:4). The 4x4 Chroma Cb sub-blocks are numbered as
<table><tr><td>blk0</td><td>1</td><td>bit3</td><td>2</td></tr><tr><td>blk2</td><td>3</td><td>bit1</td><td>0</td></tr></table>
The cbpCb bit assignment is cbpCb bit [3 - X] for sub-block_num X.

0 in a bit - indicates the corresponding 4x4 sub-block is not present (because all coefficient</td></tr>
</table>

DWord	Bit	Description
		values are zero) 1 in a bit - indicates the corresponding 4x4 sub-block is present (although it is still possible to have all its coefficients be zero - bad coding). For monochrome, this field is ignored.
3	31:24	**Reserved** MBz
	23:16	**QpPrimeCr** Driver is responsible for deriving the QpPrimeCr from QpPrimeY. For 8-bit pixel data, QpCr is the same as QpPrimeCr, and it takes on a value in the range of 0 to 51, positive integer.
	15:8	**QpPrimeCb** Driver is responsible for deriving the QpPrimeCb from QpPrimeY. For 8-bit pixel data, QpCb is the same as QpPrimeCb, and it takes on a value in the range of 0 to 51, positive integer.
	7:0	**QpPrimeY** This is the per-MB QP value specified for the current MB. For 8-bit pixel data, QpY is the same as QpPrimeY, and it takes on a value in the range of 0 to 51, positive integer.
4 to 6	31:0 Each	For intra macroblocks, definition of these fields are specified in Inline data subfields for an Intra Macroblock For inter macroblocks, definition of these fields are specified in Inline data subfields for an Inter Macroblock

Indirect Data Format in AVC-IT Mode

Indirect data in AVC-IT mode consist of Motion Vectors, Transform-domain Residue (Coefficient) and ILDB control data. All three data records have variable size. Size of each Motion Vector record is determined by the MvQuantity value as shown in Indirect MV record size in AVC-IT mode. ILDB control record is fixed at the same size for all MBs in a picture. Coefficient data record is variable size per MB, since it may only consist of non-zero coefficients.

Each MV is represented in 4 bytes, in the form of

- Lower 2 bytes : horizontal MVx component in q-pel units
- Upper 2 bytes : vertical MVy component in q-pel units
- Integer distance is measured in unit of samples in the frame or field grid position.
- Chroma MVs are not sent and are derived in the H/W.

Indirect MV record size in AVC-IT mode

Macroblock Type	MVQuant
BP_L0_16x16	1
B_L1_16x16	1
B_Bi_16x16	2
BP_L0_L0_16x8	2
BP_L0_L0_8x16	2
B_L1_L1_16x8	2
B_L1_L1_8x16	2
B_L0_L1_16x8	2
B_L0_L1_8x16	2
B_L1_L0_16x8	2
B_L1_L0_8x16	2
B_L0_Bi_16x8	3
B_L0_Bi_8x16	3
B_L1_Bi_16x8	3
B_L1_Bi_8x16	3
B_Bi_L0_16x8	3
B_Bi_L0_8x16	3
B_Bi_L1_16x8	3
B_Bi_L1_8x16	3
B_Bi_Bi_16x8	4
B_Bi_Bi_8x16	4
BP_8x8	Sum

For macroblock type of BP_8x8, MvQuant takes the sum of value MvQ[i] of the four individual 8x8 sub macroblocks.

SubMbShape[i]	SubMbPredMode[i]	Description	MvQ[i]
0	0	BP_L0_8x8	1
0	1	B_L1_8x8	1
0	2	B_BI_8x8	2
1	0	BP_L0_8x4	2
1	1	B_L1_8x4	2
1	2	B_BI_8x4	4
2	0	BP_L0_4x8	2
2	1	B_L1_4x8	2
2	2	B_BI_4x8	4
3	0	BP_L0_4x4	4
3	1	B_L1_4x4	4
3	2	B_BI_4x4	8

Indirect data Deblocking Filter Control block in AVC-IT mode:

AVC Deblocker Control Data record has a fixed size for each MB in a picture and is 12 Dwords in size.

DWord	Bit	Description
0	31:24	Reserved : MBZ (DXVA Decoder)
	23	**FilterTopMbEdgeFlag**
	22	**FilterLeftMbEdgeFlag**
	21	**FilterInternal4x4EdgesFlag**
	20	**FilterInternal8x8EdgesFlag**
	19	**FieldModeAboveMbFlag**
	18	**FieldModeLeftMbFlag**
	17	**FieldModeCurrentMbFlag**
	16	**MbaffFrameFlag** (DXVA Decoder reserved bit)
	15:8	**VertOrigin** Current MB y position (address)
	7:0	**HorzOrigin** Current MB x position (address)

DWord	Bit	Description
1	31:30	**bS_h13** 2-bit boundary strength for internal top horiz 4-pixel edge 3
	29:28	**bS_h12** 2-bit boundary strength for internal top horiz 4-pixel edge 2
	27:26	**bS_h11** 2-bit boundary strength for internal top horiz 4-pixel edge 1
	25:24	**bS_h10** 2-bit boundary strength for internal top horiz 4-pixel edge 0
	23:22	**bS_v33** 2-bit boundary strength for internal right vert 4-pixel edge 3
	21:20	**bS_v23** 2-bit boundary strength for internal right vert 4-pixel edge 2
	19:18	**bS_v13** 2-bit boundary strength for internal right vert 4-pixel edge 1
	17:16	**bS_v03** 2-bit boundary strength for internal right vert 4-pixel edge 0
	15:14	**bS_v32** 2-bit boundary strength for internal mid vert 4-pixel edge 3
	13:12	**bS_v22** 2-bit boundary strength for internal mid vert 4-pixel edge 2
	11:10	**bS_v12** 2-bit boundary strength for internal mid vert 4-pixel edge 1
	9:8	**bS_v02** 2-bit boundary strength for internal mid vert 4-pixel edge 0
	7:6	**bS_v31** 2-bit boundary strength for internal left vert 4-pixel edge 3
	5:4	**bS_v21** 2-bit boundary strength for internal left vert 4-pixel edge 2
	3:2	**bS_v11** 2-bit boundary strength for internal left vert 4-pixel edge 1
	1:0	**bS_v01** 2-bit boundary strength for internal left vert 4-pixel edge 0
2	31:28	**bS_v30_0** 4-bit boundary strength for Left0 4-pixel edge 3 (MSbit is wasted)
	17:24	**bS_v20_0** 4-bit boundary strength for Left0 4-pixel edge 2 (MSbit is wasted)
	23:20	**bS_v10_0** 4-bit boundary strength for Left0 4-pixel edge 1 (MSbit is wasted)
	19:16	**bS_v00_0** 4-bit boundary strength for Left0 4-pixel edge 0 (MSbit is wasted)
	15:14	**bS_h33** 2-bit boundary strength for internal bot horiz 4-pixel edge 3

DWord	Bit	Description
	13:12	**bS_h32** 2-bit boundary strength for internal bot horiz 4-pixel edge 2
	11:10	**bS_h31** 2-bit boundary strength for internal bot horiz 4-pixel edge 1
	9:8	**bS_h30** 2-bit boundary strength for internal bot horiz 4-pixel edge 0
	7:6	**bS_h23** 2-bit boundary strength for internal mid horiz 4-pixel edge 3
	5:4	**bS_h22** 2-bit boundary strength for internal mid horiz 4-pixel edge 2
	3:2	**bS_h21** 2-bit boundary strength for internal mid horiz 4-pixel edge 1
	1:0	**bS_h20** 2-bit boundary strength for internal mid horiz 4-pixel edge 0
3	31:28	**bS_h03_0** 4-bit boundary strength for Top0 4-pixel edge 3 (MSbit is wasted)
	27:24	**bS_h02_0** 4-bit boundary strength for Top0 4-pixel edge 2 (MSbit is wasted)
	23:20	**bS_h01_0** 4-bit boundary strength for Top0 4-pixel edge 1 (MSbit is wasted)
	19:16	**bS_h00_0** 4-bit boundary strength for Top0 4-pixel edge 0 (MSbit is wasted)
	15:12	**bS_v03** 4-bit boundary strength for Left1 4-pixel edge 3 (MSbit is wasted)
	11:8	**bS_v02** 4-bit boundary strength for Left1 4-pixel edge 2 (MSbit is wasted)
	7:4	**bS_v01** 4-bit boundary strength for Left1 4-pixel edge 1 (MSbit is wasted)
	3:0	**bS_v00** 4-bit boundary strength for Left1 4-pixel edge 0 (MSbit is wasted)
4	31:24	**bIndexBinternal_Y** Internal index B for Y
	23:16	**bIndexAinternal_Y** Internal index A for Y
	15:12	**bS_h03_1** 4-bit boundary strength for Top1 4-pixel edge 3 (MSbit is wasted)
	11:8	**bS_h02_1** 4-bit boundary strength for Top1 4-pixel edge 2 (MSbit is wasted)
	7:4	**bS_h01_1** 4-bit boundary strength for Top1 4-pixel edge 1 (MSbit is wasted)
	3:0	**bS_h00_1** 4-bit boundary strength for Top1 4-pixel edge 0 (MSbit is wasted)

DWord	Bit	Description
5	31:24	**bIndexBleft1_Y**
	23:16	**bIndexAleft1_Y**
	15:8	**bIndexBleft0_Y**
	7:0	**bIndexAleft0_Y**
6	31:24	**bIndexBtop1_Y**
	23:16	**bIndexAtop1_Y**
	15:8	**bIndexBtop0_Y**
	7:0	**bIndexAtop0_Y**
7	31:24	**bIndexBleft0_Cb**
	23:16	**bIndexAleft0_Cb**
	15:8	**bIndexBinternal_Cb**
	7:0	**bIndexAinternal_Cb**
8	31:24	**bIndexBtop0_Cb**
	23:16	**bIndexAtop0_Cb**
	15:8	**bIndexBleft1_Cb**
	7:0	**bIndexAleft1_Cb**
9	31:24	**bIndexBinternal_Cr**
	23:16	**bIndexAinternal_Cr**
	15:8	**bIndexBtop1_Cb**
	7:0	**bIndexAtop1_Cb**
10	31:24	**bIndexBleft1_Cr**

DWord	Bit	Description
	23:16	**bIndexAleft1_Cr**
	15:8	**bIndexBleft0_Cr**
	7:0	**bIndexAleft0_Cr**
11	31:24	**bIndexBtop1_Cr**
	23:16	**bIndexAtop1_Cr**
	15:8	**bIndexBtop0_Cr**
	7:0	**bIndexAtop0_Cr**

Inline Data Description in VC1-IT Mode

<table>
<tr><th>DWord</th><th>Bits</th><th>Description</th></tr>
<tr><td>+0</td><td>31:28</td><td>MvFieldSelect. A bit-wise representation indicating which field in the reference frame is used as the reference field for current field. It's only used in decoding interlaced pictures.

This field is valid for non-intra macroblock only.

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>28</td><td>Forward predict of current frame/field or TOP field of interlace frame, or block 0 in 4MV mode.</td></tr>
<tr><td>29</td><td>Backward predict of current frame/field or TOP field of interlace frame, or forward predict for block 1 in 4MV mode.</td></tr>
<tr><td>30</td><td>Forward predict of BOTTOM field of interlace frame, or block 2 in 4MV mode.</td></tr>
<tr><td>31</td><td>Backward predict of BOTTOM field of interlace frame, or forward predict for block 3 in 4MV mode.</td></tr>
</table>

Each corresponding bit has the following indication.

0 = The prediction is taken from the top reference field.

1 = The prediction is taken from the bottom reference field.</td></tr>
<tr><td></td><td>27</td><td>Reserved. MBZ</td></tr>
<tr><td></td><td>26</td><td>MvFieldSelectChroma . This field specifies the polarity of reference field for chroma blocks when their motion vector is derived in Motion4MV mode for interlaced (field) picture.

Non-intra macroblock only. This field is derived from MvFieldSelect.

0 = The prediction is taken from the top reference field.</td></tr>
</table>

DWord	Bits	Description
		1 = The prediction is taken from the bottom reference field.
	25:24	**MotionType - Motion Type** For frame picture, a macroblock may only be either 00 or 10. For interlace picture, a macroblock may be of any motion types. It can be 01 if and only if DctType is 1. This field is 00 if and only if IntraMacroblock is 1. 00 = Intra 01 = Field Motion. 10 = Frame Motion or no motion. Others = Reserved.
	23	**Reserved.** MBZ
	22	**MvSwitch.** This field specifies whether the prediction needs to be switched from forward to backward or vice versa for single directional prediction for top and bottom fields of interlace frame B macroblocks. 0 = No directional prediction switch from top field to bottom field 1 = Switch directional prediction from top field to bottom field
	21	**DctType.** This field specifies whether the residual data is coded as field residual or frame residual for interlaced picture. This field can be 1 only if MotionType is 00 (intra) or 01 (field motion). For progressive picture, this field must be set to '0', i.e. all macrobalcoks are frame macroblock. 0 = Frame residual type. 1 = Field residual type.
	20	**OverlapTransform.** This field indicates whether overlap smoothing filter should be performed on I-block boundaries. 0 = No overlap smoothing filter. 1 = Overlap smoothing filter performed.
	19	**Motion4MV.** This field indicates whether current macroblock a progressive P picture uses 4 motion vectors, one for each luminance block. It's only valid for progressive P-picture decoding. Otherwise, it is reserved and MBZ. For example, with MotionForward is 0, this field must also be set to 0. 0 = 1MV-mode. 1 = 4MV-mode.

DWord	Bits	Description
	18	**MotionBackward.** This field specifies whether the backward motion vector is active for B-picture. This field must be 0 if Motion4MV is 1 (no backward motion vector in 4MV-mode). 0 = No backward motion vector. 1 = Use backward motion vector(s).
	17	**MotionForward.** This field specifies whether the forward motion vector is active for P and B pictures. 0 = No forward motion vector. 1 = Use forward motion vector(s).
	16	**IntraMacroblock.** This field specifies if the current macroblock is intra-coded. When set, Coded Block Pattern is ignored and no prediction is performed (i.e., no motion vectors are used). For field motion, this field indicates whether the top field of the macroblock is coded as intra. 0 = Non-intra macroblock. 1 = Intra macroblock.
	15:12	**LumaIntra8x8Flag - Luma Intra 8x8 Flag** This field specifies whether each of the four 8x8 luminance blocks are intra or inter coded when Motion4MV is set to 4MV-Mode. Each bit corresponds to one block. "0" indicates the block is inter coded and '1' indicates the block is intra coded. When Motion4MV is not 4MV-Mode, this field is reserved and MBZ. Bit 15: Y0 Bit 14: Y1 Bit 13: Y2 Bit 12: Y3
	11:6	**CBP - Coded Block Pattern** This field specifies whether the 8x8 residue blocks in the macroblock are present or not. Each bit corresponds to one block. "0" indicates residue block isn't present, "1" indicates residue block is present. Note: For each block in an intra-coded macroblock or an intra-coded block in a P macroblock in 4MV-Mode, the corresponding CBP must be 1. Subsequently, there must be at least one coefficient (this coefficient might be zero) in the indirect data buffer associated with the bock (i.e. residue block must be present). Bit 11: Y0 Bit 10: Y1

DWord	Bits	Description
		Bit 9: Y2 Bit 8: Y3 Bit 7: Cb4 Bit 6: Cr5
	5	**ChromaIntraFlag - Derived Chroma Intra Flag** This field specifies whether the chroma blocks should be treated as intra blocks based on motion vector derivation process in 4MV mode. 0 = Chroma blocks are not coded as intra. 1 = Chroma blocks are coded as intra
	4	**LastRowFlag - Last Row Flag** This field indicates that the current macroblock belongs to the last row of the picture. This field may be used by the kernel to manage pixel output when overlap transform is on. 0 = Not in the last row 1 = In the last row
	3	**LastMBInRow - This field indicates the last MB in row flag.**
	2:0	**Reserved.** MBZ
+1	32:26	**Reserved.** MBZ
	25:24	**OSEdgeMaskChroma** This field contains the overscan edge mask for the Chroma blocks. The left edge masks are hardware and the top edge masks are used by the kernel software. Bit 24: Top edge of block Cb/Cr Bit 25: Left edge of block Cb/Cr
	23:16	**OSEdgeMaskLuma** This field contains the overscan edge mask for the Luma blocks. The left edge masks are hardware and the top edge masks are used by the kernel software. Bit 16: Top edge of block Y0 Bit 17: Top edge of block Y1 Bit 18: Top edge of block Y2 Bit 19: Top edge of block Y3

DWord	Bits	Description
		Bit 20: Left edge of block Y0 Bit 21: Left edge of block Y1 Bit 22: Left edge of block Y2 Bit 23: Left edge of block Y3 *Programming Note: In order to create 8 predication bits from each edge mask bit, software may first create a 0, 1 vector by using a shr instruction with a step shift vector like 0, 1, 2, 3 (e.g. using immediate of type :v. Then each 0 or 1 of the LSB can be repeated by an and instruction to create 8 bits to the flag register. Alternatively, this can be achieved with one and instruction using a CURBE constant map of bit 0 and bit 1 mask.*
	15:8	**VertOrigin - Vertical Origin** In unit of macroblocks relative to the current picture (frame or field).
	7:0	**HorzOrigin - Horizontal Origin** In unit of macroblocks.
+2	31:16	**MotionVector[0].Vert**
	15:0	**MotionVector[0].Horz**
+3	31:0	**MotionVector[1]**
+4	31:0	**MotionVector[2]**
+5	31:0	**MotionVector[3]**
+6	31:0	**MotionVectorChroma** This field is not valid for a field motion in an interlaced frame picture where 4 MVs for chroma blocks. Notes: This field is derived from MotionVector[3:0] as described in the following section.

<table>
<tr><th>DWord</th><th>Bits</th><th>Description</th></tr>
<tr><td>+7</td><td>31:24</td><td>Subblock Code for Y3
The following subblock coding definition applies to all 6 subblock coding bytes. Bits 7:6 are reserved.
<table>
<tr><th colspan="2">Subblock Partitioning (Bits [1:0]) Specify Transform uses for an 8x8 block</th><th colspan="4">Subblock Present (0 means not present, 1 means present)</th></tr>
<tr><th>Bits [1:0]</th><th>Meaning</th><th>Bit 2</th><th>Bit 3</th><th>Bit 4</th><th>Bit 5</th></tr>
<tr><td>00</td><td>Single 8x8 block (sb0)</td><td>Sb0</td><td>Don't care</td><td>Don't care</td><td>Don't care</td></tr>
<tr><td>01</td><td>Two 8x4 subblocks (sb0-1)</td><td>Sb1 (bot)</td><td>Sb0 (top)</td><td>Don't care</td><td>Don't care</td></tr>
<tr><td>10</td><td>Two 4x8 subblocks (sb0-1)</td><td>Sb1 (right)</td><td>Sb0 (left)</td><td>Don't care</td><td>Don't care</td></tr>
<tr><td>11</td><td>Four 4x4 subblocks (sb0-3)</td><td>Sb3 (lower right)</td><td>Sb2 (lower left)</td><td>Sb1 (upper right)</td><td>Sb0 (upper left)</td></tr>
</table></td></tr>
<tr><td></td><td>23:16</td><td>Subblock Code for Y2</td></tr>
<tr><td></td><td>15:8</td><td>Subblock Code for Y1</td></tr>
<tr><td></td><td>7:0</td><td>Subblock Code for Y0</td></tr>
<tr><td>+8</td><td>31:16</td><td>Reserved. MBZ</td></tr>
<tr><td></td><td>15:8</td><td>Subblock Code for Cr</td></tr>
<tr><td></td><td>7:0</td><td>Subblock Code for Cb</td></tr>
<tr><td>+9</td><td>31:24</td><td>ILDB control data for block Y3</td></tr>
<tr><td></td><td>23:16</td><td>ILDB control data for block Y2</td></tr>
<tr><td></td><td>15:8</td><td>ILDB control data for block Y1</td></tr>
<tr><td></td><td>7:0</td><td>ILDB control data for block Y0</td></tr>
<tr><td>+10</td><td>31:16</td><td>Reserved</td></tr>
<tr><td></td><td>15:8</td><td>ILDB control data for Cr block</td></tr>
<tr><td></td><td>7:0</td><td>ILDB control data for Cb block</td></tr>
</table>

Indirect Data Format in VC1-IT Mode

VC1-IT mode only contains IT-COEFF indirect data which is described in Common Indirect IT-COEFF Data Structure.

Inline Data Description in MPEG2-IT Mode

The content in this command is similar to that in the MEDIA_OBJECT command in IS mode described in the Media Chapter.

Each MFD_IT_OBJECT command corresponds to the processing of one macroblock. Macroblock parameters are passed in as inline data and the non-zero DCT coefficient data for the macroblock is passed in as indirect data.

Inline data in MPEG2-IT Mode depicts the inline data format. Inline data starts at dword 7 of MFD_IT_OBJECT command. There are 7 dwords total.

Inline data in MPEG2-IT Mode

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>+0</td><td>31:28</td><td>Motion Vertical Field Select. A bit-wise representation of a long [2][2] array as defined in Section 6.3.17.2 of the ISO/IEC 13818-2 (see also Section 7.6.4).
<table>
<tr><th>Bit</th><th>MVector[r]</th><th>MVector[s]</th><th>MotionVerticalFieldSelect Index</th></tr>
<tr><td>28</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>29</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>30</td><td>1</td><td>0</td><td>2</td></tr>
<tr><td>31</td><td>1</td><td>1</td><td>3</td></tr>
</table>
Format = MC_MotionVerticalFieldSelect.
0 = The prediction is taken from the top reference field.
1 = The prediction is taken from the bottom reference field.</td></tr>
<tr><td></td><td>27</td><td>Reserved (was Second Field)</td></tr>
<tr><td></td><td>26</td><td>Reserved. (HWMC mode)</td></tr>
</table>

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td></td><td>25:24</td><td>Motion Type. When combined with the destination picture type (field or frame) this Motion Type field indicates the type of motion to be applied to the macroblock. See ISO/IEC 13818-2 Section 6.3.17.1, Tables 6-17, 6-18. In particular, the device supports dual-prime motion prediction (11) in both frame and field picture type.
Format = MC_MotionType
<table>
<tr><th>Value</th><th>Destination = Frame
Picture_Structure = 11</th><th>Destination = Field
Picture_Structure != 11</th></tr>
<tr><td>'00'</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>'01'</td><td>Field</td><td>Field</td></tr>
<tr><td>'10'</td><td>Frame</td><td>16x8</td></tr>
<tr><td>'11'</td><td>Dual-Prime</td><td>Dual-Prime</td></tr>
</table></td></tr>
<tr><td></td><td>23:22</td><td>Reserved. (Scan method)</td></tr>
<tr><td></td><td>21</td><td>DCT Type. This field specifies the DCT type of the current macroblock. The kernel should ignore this field when processing Cb/Cr data. See ISO/IEC 13818-2 Section 6.3.17.1. This field is zero if Coded Block Pattern is also zero (no coded blocks present).
0 = MC_FRAME_DCT (Macroblock is frame DCT coded).
1 = MC_FIELD_DCT (Macroblock is field DCT coded).</td></tr>
<tr><td></td><td>20</td><td>Reserved (was Overlap Transform - H261 Loop Filter).</td></tr>
<tr><td></td><td>19</td><td>Reserved (was 4MV Mode - H263/WMV)</td></tr>
<tr><td></td><td>18</td><td>Macroblock Motion Backward. This field specifies if the backward motion vector is active. See ISO/IEC 13818-2 Tables B-2 through B-4.
0 = No backward motion vector.
1 = Use backward motion vector(s).</td></tr>
<tr><td></td><td>17</td><td>Macroblock Motion Forward. This field specifies if the forward motion vector is active. See ISO/IEC 13818-2 Tables B-2 through B-4.
0 = No forward motion vector.
1 = Use forward motion vector(s).</td></tr>
<tr><td></td><td>16</td><td>Macroblock Intra Type. This field specifies if the current macroblock is intra-coded. When set, Coded Block Pattern is ignored and no prediction is performed (i.e., no motion vectors are used). See ISO/IEC 13818-2 Tables B-2 through B-4.
0 = Non-intra macroblock.
1 = Intra macroblock.</td></tr>
<tr><td></td><td>15:12</td><td>Reserved : MBZ</td></tr>
</table>

DWord	Bit	Description
	11:6	**Coded Block Pattern.** This field specifies whether blocks are present or not. Format = 6-bit mask. Bit 11: Y0 Bit 10: Y1 Bit 9: Y2 Bit 8: Y3 Bit 7: Cb4 Bit 6: Cr5
	5:4	Reserved. (Quantization Scale Code)
	3	LastMBInRow - This field indicates the last MB in each row.
	2:0	Reserved: MBZ
+1	31:16	**Reserved : MBZ**
	15:8	**VertOrigin - Vertical Origin** In unit of macroblocks relative to the current picture (frame or field).
	7:0	**HorzOrigin - Horizontal Origin** In unit of macroblocks.
+2	31:16	**Motion Vectors - Field 0, Forward, Vertical Component.** Each vector component is a 16-bit two's-complement value. The vector is relative to the current macroblock location. According to ISO/IEC 13818-2 Table 8, the valid range of each vector component is [-2048, +2047.5], implying a format of s11.1. However, it should be noted that motion vector values are sign extended to 16 bits.
	15:0	**Motion Vectors - Field 0, Forward, Horizontal Component**
+3	31:16	**Motion Vectors - Field 0, Backward, Vertical Component**
	15:0	**Motion Vectors - Field 0, Backward, Horizontal Component**
+4	31:16	**Motion Vectors - Field 1, Forward, Vertical Component**
	15:0	**Motion Vectors - Field 1, Forward, Horizontal Component**
+5	31:16	**Motion Vectors - Field 1, Backward, Vertical Component**
	15:0	**Motion Vectors - Field 1, Backward, Horizontal Component**

Indirect Data Format in MPEG2-IT Mode

MPEG2-IT mode only contains IT-COEFF indirect data which is described in Section Common Indirect IT-COEFF Data Structure.

MFX Deblocking Commands

Following are MFX Deblocking Commands:

MFX_DBK_OBJECT

MFX Error Handling

Encoder StreamOut Mode Data Structure Definition

When StreamOut is enabled, per MB (and/or per Slice, per Picture) intermediated coding data (for example, bit allocated for each MB, and so on) are sent to the memory in a fixed record format (and of fixed size) from the PAK. The per-MB records must be written in a strict raster order and with no gap (that is, every MB regardless of its mb_type and slice type, must have an entry in the StreamOut buffer). Therefore, the consumer of the StreamOut data can offset into the StreamOut Buffer (**StreamOut Data Destination Base Address**) using individual MB addresses.

Adding per macroblock stream out for PAK is for the following purposes:

- Immediate multi-pass PAK (without host or EU intervention)
  - 3200-bit conformance
  - Re-quantization
- Providing information for host for offline processing
- Providing information for updated QP's

The description for the fixed format PAK streamout record:

Streamout Pointer: Use the existing streamout pointer and enabler

Per Macroblock Information (a fixed size structure)

DWord	Bit	Description
0	31:24	**MbQpY** - Actual QPY used by the macroblock.
	23:16	**MbClock16** - MB compute clocks in 16-clock unit.
	15:8	Reserved: MBZ
	7:4	Reserved: MBZ (future conformance flags)
	3	Reserved
	2	**MbRcFlag:** MB level Rate control flag(pass through) The same value as **RateControlCounterEnable** of MFX_AVC_SLICE_STATE Command
	1	**MbInterConfFlag:** MB level InterMB conformance flag to trigger mutli-pass 1- if total Bit Count of an inter macroblock is more than Inter Conformance Max size limit in the MFX_AVC_IMG_STATE Command

DWord	Bit	Description
	0	**MbIntraConfFlag:** MB level IntraMB conformance flag to trigger mutli-pass 1- if total Bit Count of an intra macroblock is more than Intra Conformance Max size limit in the MFX_AVC_IMG_STATE Command
1	31:29	Reserved
	28:16	**MbBits:** Total Bit Count for the macroblock
	15:12	Reserved
	12:0	**MbHdrBits:** Header Bit count (bit count due to Pre-coefficient data) for the macroblock
2	31:27	Reserved
	26:0	**Cbp:** Coded Block Pattern of sub-blocks
3	31:30	Reserved
	29	**IntraMBFlag**
	28:24	**MBType5Bits**
	23:17	Reserved
	16	**ClampFlag:** Coefficient clamping flag for RC (Status) 1 - Indicates if clamping of any coefficient is done on the macroblock for Rate Control
	15:0	Reserved (future QRC stat output)

PAK Frame Statistics StreamOut

The following frame statistics are written to memory at the conclusion of a frame. If Multipass occurs, these values are overwritten by the end of any subsequent passes of the current frame (hence it contains only the final pass statistics).

The streamout is done to the MB streamout surface, starting at the next CL boundary. If MB streamout is disabled, Frame level streamout starts with 0 offset.

MFX_PAK_FRAME_STATISTICS		
Source:	**VideoCS**	
Length Bias:	**2**	
DWord	**Bit**	**Description**
0	31:16	Reserved : MBZ
	15:0	SumSliceHeader - Report the total size (in bits) of all slice headers inserted into the bitstream for this frame.
1	31:0	SumMBHeader - Report the total size (in bits) of all MB headers (non coeff bits) inserted into the bitstream for this frame.
2	31:0	SumNZC - Report the total number of nonzero coefficients after quantization.
3	31:0	Reserved: MBZ

MFX_PAK_FRAME_STATISTICS		
4	31:16	IntraMB16x16 - Count of # of MB's that were of type Intra 16x16
	15:0	IntraMB8x8 - Count of # of MB's that were of type Intra 8x8
5	31:16	IntraMB4x4 - Count of # of MB's that were of type Intra 4x4
	15:0	InterMB16x16 - Count of # of MB's that were of type Inter 16x16
6	31:16	InterMB16x8 - Count of # of MB's that were of type Inter 16x8
	15:0	InterMB8x16 - Count of # of MB's that were of type Inter 8x16
7	31:16	InterMB8x8 - Count of # of MB's that were of type Inter 8x8
	15:0	InterSkip16x16 - Count of # of MB's that were of type Inter 16x16 skip
8:49	31:0	RhoDomainStats - Each DW contains 1 of the 42 registers containing the raw Rho Domain coefficient metrics. DW 8 is QP 10 and DW 49 is QP51.
50	31:0	Reserved: MBZ

PAK Multi-Pass

Multi-Pass PAK Usages:

- Intra MB 3200-bit conformance
- Inter MB Re-quantization
- Frame level Re-quantization

How to Enable Multi-Pass PAK?

- Using the existing conditional batch buffer execution capability to skip/execute the second pass
    - How to dynamically change the condition?
        - Defined one error condition register with a mask. Do HW status page update at the end of the first pass. 0 means all OK, non-zero means there is an error condition, requiring second pass. Mask is used by the host to control what kind of multi-pass is intended.
        - For example, one error bit is 3200-bit conformance violation. Another error bit is the total bit count exceeds (too much or too little) the target range (need to define the target range in the state).
        - **The logic perfectly fits in the conditional batch buffer control logic that VCS has today in GT. There is no additional logic need to be added in VCS to support media functionality.** (**Batch Buffer Skip:** This field only takes effect if Compare Semaphore is set and the value at **Semaphore Address is NOT greater than the Semaphore Data Dword)**.
- Adding a picture level state command to enable and control the behavior of the second pass PAK
    - How to control the re-PAK? Added 3 conformance flags (error registers) in the per-MB streamout. Then the error control is based on the error register and the mask defined in

picture level states. There are 8 register flags defined out of which only the 3200-bit case has usage model defined for today. The rest are left for future usage.

Issues and Limitations:

- There is no programmable engine in MFX for flexible control: Therefore, whatever we have defined must consider flexibility

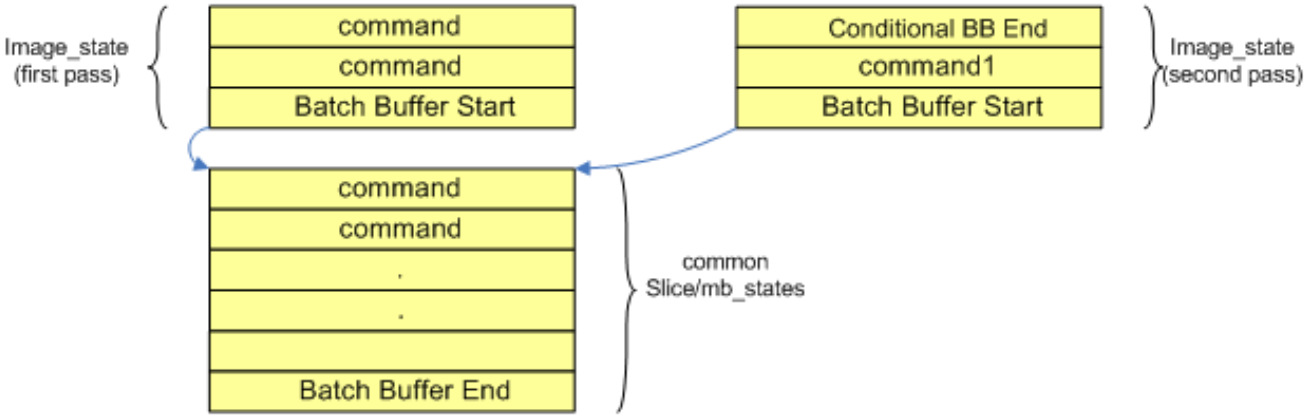
Following 2 MI packets are used inside VCS without any change to support Multipass-PAK behaviour.

- MI_Conditional_Batch_Buffer_End
- Memory Interface Registers

Driver Usage

Driver places Image states in one batch buffer and all slice level and macroblock level states into another batch buffer and link 2 batch buffers. Also replicate Image states with multipass changes in another batch buffer link them to slice/macroblock batch buffer. In this way, only Image states are replicated but not the slice/macroblock states. The image states includes all buffers defined at image(indirectMV, original pixel buffer, etc). Following changes are needed in the Multipass Image State,

- **Reset- Stream-Out Enable(disable stream out in the second pass)**
- **Set- MacroblockStatEnable (enable reading of macroblock status buffer)**
- **Reset- 3200-bit conformance (do not report 3200-bit conformance)**



Define Conditional Batch Buffer End for CS/VCSVINunit

Programming Reference

Monochrome Picture Processing

Monochrome picture is specified using the Surface State with Surface Format of 12. Therefore, MFX hardware, in either decode or encode mode, does not generate any read or write traffic for U/V components. The motivation for this bandwidth optimization is that monochrome video coding might be used for wireless display.

For Encoder:

1. No read in UV original components
2. Processing UV component - no
3. Reconstructed UV component reference picture - no
4. Filter UV component - no

For Decoder:

1. VLD mode: There is no color component coming out of the decoding pipeline in Monochrome mode and so no processing and not writing output.
2. IT mode: There is no color component in the coefficient buffer, and so no processing and not writing output.

Context Switch

There is no pre-emption for the BCS pipeline; hence every command buffer is required to contain all the states setup (preamble). Specifically, CPU cannot interrupt the BCS-BSD pipe, to stop the operation in the middle of decoding a bitstream data.

Switch of contexts can only be performed at picture boundary.

No state needs to be saved.

PMSI Support

Pipeline Flush

Implicit flush for AVC and VC1 is performed at the end of Slice : for MPEG2 is done when a new image/picture command is issued. Because MPEG2 a slice can be one MB, no point to flush. MPEG2 will snoop the next command if it is an img_state command.

Explicit flush MI (1 bit to do media pipeline vs Gx pipeline) flush and cache flush (switch reference frame) - MI flush has bit to do cache flush. MI flush is for driver synchronization.

MMIO Interface

A set of registers are defined and accessible through MMIO interface to serve multiple purposes:

- Use for system configuration
- For accessing Performance counters

The following is the table for all the MMIO addresses for MFX.

Decoder Registers

Following are Decoder Registers:

Registers
MFD_ERROR_STATUS - MFD Error Status
AVC CAVLC
AVC CABAC
VC1
MPEG2
JPEG
MFD_PICTURE_PARAM - MFD Picture Parameter
MFX_STATUS_FLAGS - MFX Pipeline Status Flags
MFX_MB_COUNT - MFX Frame Macroblock Count
MFX_SE-BIN_CT - MFX Frame BitStream SE/BIN Count

Encoder Registers

Following are the Encoder Registers:

Register
MFC_VIN_AVD_ERROR_CNTR - MFC_AVC Bitstream Decoding Front-End Parsing Logic Error Counter.
MFC_BITSTREAM_BYTECOUNT_FRAME - Reported Bitstream Output Byte Count per Frame Register
MFC_BITSTREAM_SE_BITCOUNT_FRAME - Reported Bitstream Output Bit Count for Syntax Elements Only Register
MFC_AVC_CABAC_BIN_COUNT_FRAME - Reported Bitstream Output CABAC Bin Count Register
AVC_CABAC_INSERTION_COUNT - MFC_AVC_CABAC_INSERTION_COUNT
MFC_AVC_MINSIZE_PADDING_COUNT - Bitstream Output Minimal Size Padding Count Report Register
MFC_IMAGE_STATUS_MASK - MFC Image Status Mask
MFC_IMAGE_STATUS_CONTROL - MFC Image Status Control
MFC_QUP_CT - MFC QP Status Count
MFC_BITSTREAM_BYTECOUNT_SLICE - Bitstream Output Byte Count Per Slice Report Register
MFC_BITSTREAM_SE_BITCOUNT_SLICE - Bitstream Output Bit Count for the last Syntax Element Report Register
MFX_PAK_ERROR Register
MFX_PAK_WARNING Register

Register
MFX_VP8_CNTRL_MASK - Reported BitRateControl parameter Mask
MFX_VP8_CNTRL_STATUS - Reported BitRateControl parameter Status
MFX_VP8_FRM_BYTE_CNT - Reported Final Bitstream Byte Count
MFX_VP8_FRM_ZERO_PAD - Reported Frame Zero Padding Byte Count
MFX_VP8_BRC_DQindex - Reported BitRateControl DeltaQindex
MFX_VP8_BRC_DLoopFilter - Reported BitRateControl DeltaLoopFilter
MFX_VP8_BRC_CumulativeDQindex01 - Reported BitRateControl CumulativeDeltaQindex and Qindex 01
MFX_VP8_BRC_CumulativeDQindex23 - Reported BitRateControl CumulativeDeltaQindex and Qindex 23
MFX_VP8_BRC_CumulativeDLoopFilter01 - Reported BitRateControl CumulativeDeltaLoopFilter and LoopFilter 01
MFX_VP8_BRC_CumulativeDLoopFilter23 - Reported BitRateControl CumulativeDeltaLoopFilter and LoopFilter 23
MFX_VP8_BRC_Convergence_Status - Reported BitRateControl Convergence Status

MMIO Interface

A set of registers are defined and accessible through MMIO interface to serve multiple purposes:

- Use as Status register for Bit Rate Control
- Use for Context Switch in Multipass

Register Name	Description	Register Type	Address Offset	Dec/Enc
MFX_VP8_CNTRL_MASK	BitRateControl parameter Mask register	RO	12900	Enc
MFX_VP8_CNTRL_STATUS	BitRateControl parameter Status register	RO	12904	Enc
MFX_VP8_FRM_BYTE_CNT	Final Bitstream Byte count	RO	12908	Enc
MFX_VP8_FRM_ZERO_PAD	Final Bitstream Zero Padding Byte count	RO	1290B	Enc
MFX_VP8_BRC_DQindex	BitRateControl Delta Qindex	RO	12910	Enc
MFX_VP8_BRC_DLoopFilter	BitRateControl Delta LoopFilter	RO	12914	Enc
MFX_VP8_BRC_CumulativeDQindex01	BitRateControl Cumulative Delta Qindex for Seg0/1	RW	12918	Enc
MFX_VP8_BRC_CumulativeDQindex23	BitRateControl Cumulative Delta Qindex for Seg2/3	RW	1291C	Enc
MFX_VP8_BRC_CumulativeDLoopFilter01	BitRateControl Cumulative Delta LoopFilter for Seg0/1	RW	12920	Enc
MFX_VP8_BRC_CumulativeDLoopFilter23	BitRateControl Cumulative Delta LoopFilter for Seg2/3	RW	12924	Enc
MFX_VP8_BRC_Convergence_Status	BitRateControl Convergence Status	RW	12928	Enc

The following registers are the same as above except they have a different Address Offset. They are used if the second VDbox (VP8 Encoder) exists.

Register Name	Description	Register Type	Address Offset	Dec/Enc
MFX_VP8_CNTRL_MASK	BitRateControl parameter Mask register	RO	1C900	Enc
MFX_VP8_CNTRL_STATUS	BitRateControl parameter Status register	RO	1C904	Enc
MFX_VP8_FRM_BYTE_CNT	Final Bitstream Byte count	RO	1C908	Enc
MFX_VP8_FRM_ZERO_PAD	Final Bitstream Zero Padding Byte count	RO	1C90B	Enc
MFX_VP8_BRC_DQindex	BitRateControl Delta Qindex	RO	1C910	Enc
MFX_VP8_BRC_DLoopFilter	BitRateControl Delta LoopFilter	RO	1C914	Enc
MFX_VP8_BRC_CumulativeDQindex01	BitRateControl Cumulative Delta Qindex for Seg0/1	RW	1C918	Enc
MFX_VP8_BRC_CumulativeDQindex23	BitRateControl Cumulative Delta Qindex for Seg2/3	RW	1C91C	Enc
MFX_VP8_BRC_CumulativeDLoopFilter01	BitRateControl Cumulative Delta LoopFilter for Seg0/1	RW	1C920	Enc
MFX_VP8_BRC_CumulativeDLoopFilter23	BitRateControl Cumulative Delta LoopFilter for Seg2/3	RW	1C924	Enc
MFX_VP8_BRC_Convergence_Status	BitRateControl Convergence Status	RW	1C928	Enc

Row Store Sizes and Allocations

	AVC	VC1	MPEG2	JPEG	IT	ENC	SEC ENC
vin_vmx_pixcoefind_ addr[31:6]	Bitstream	Bitstream	Bitstream	Bitstream	VDS COEF	Orig Pix	BSP data
vin_vmx_mvbsdrs_ addr[31:6]	VAD BSD		VMD RS		VDS MV	MPC MV	
vin_vmx_mpcildbmpr_ addr[31:6]	VAM MPR				VDS ILDB	MPC RS	
vin_vmx_dmv*_ addr[31:6]	VAM DMV	VCP DMV					
vin_vmx_bp_addr [31:0]		VCP BP					

	Write		Surf Size
	Read		

MPEG2 VLD Decoding Mode :

use BSD Row Store only, and

MPEG2 IT Decoding Mode :

MPEG2 IT mode does not need row-store

JPEG VLD Decoding Mode : no row store is needed

VDBOX Registers

This section describes the VDBOX Command Memory Interface registers.

MMIO Ranges

MMIO ranges for media are described in this section. The base address of MFX(x), VCS(x), VECS(x), HEVC(x) are modified.

HEVC MMIO is split into two ranges as HEVC is split into frontend and Backend. The x value can range from 0 through 7.

The address offset is defined in hierarchical manner. Each VDBOX has 16KB of MMIO address range and is allocated as shown in the table below. Unallocated address with-in 16KB space would be claimed by HEVCFE for writes and read zeros.

Offset Address of Each Engines:

UNIT	Address	Size
VCS (range 0)	0x0000 - 0x07FF	2 KB
MFX Pipe (VIN)	0x0800 - 0x0FFF	2 KB
VCS (range 1)	0x1000 - 0x1FFF	4 KB
HEVC Pipe (HWM)	0x2800 - 0x2AFF	750 B
AVP Pipe (AWM)	0x2B00 - 0x2CFF	500 B
VDENC	0x2D00-0x2DFF	1KB
Reserved	0x2E00-0x3EFF	4096B
CFCFG	0x3F00-0x3FFF	128B
SCR (no mmio space but MsgCh endpoints)		
Total allocation:		12.5 KB

VDBOX and VEBOX Offset Table:

Media Boxes	Base Address	Offset Range	Size	Media sliceid[2:0]	Media subsliceid[1:0]
VDBOX0	0x1C_0000	0x0000 - 0x3FFF	16KB	000	00
VDBOX1	0x1C_4000	0x0000 - 0x3FFF	16KB	000	01
VEBOX0	0x1C_8000	0x0000 - 0x3FFF	16KB	000	00
VDBOX2	0x1D_0000	0x0000 - 0x3FFF	16KB	001	00
VDBOX3	0x1D_4000	0x0000 - 0x3FFF	16KB	001	01
VEBOX1	0x1D_8000	0x0000 - 0x3FFF	16KB	001	00
VDBOX4	0x1E_0000	0x0000 - 0x3FFF	16KB	010	00
VDBOX5	0x1E_4000	0x0000 - 0x3FFF	16KB	010	01
VEBOX2	0x1E_8000	0x0000 - 0x3FFF	16KB	010	00
VDBOX6	0x1F_0000	0x0000 - 0x3FFF	16KB	011	00
VDBOX7	0x1F_4000	0x0000 - 0x3FFF	16KB	011	01
VEBOX3	0x1F_8000	0x0000 - 0x3FFF	16KB	011	00

Media VEBOX

This chapter describes the VEBOX Media Engine.

Media VEBOX Introduction

The VEBOX is an independent pipe with a variety of image enhancement functions.

The following sections are contained in Media VEBOX:

Feature
Denoise
Deinterlacer
Image Enhancement/Color Processing (IECP)
Capture Pipe
VEBOX State
VEBOX Surface State
VEB DI IECP Commands
Command Stream Backend - Video
Video Enhancement Engine Functions

The IECP consists of these functions:

Feature
STD - Skin Tone Detection detects colors which might represent skin.
STE - Skin Tone Enhancement modifies colors marked by STD.
GCC - Gamut Compression
ACE - Automatic Contrast Enhancement changes luma values to enhance contrast.
TCC - Total Color Control allows UV values to be modified to adjust color saturation.
ProcAmp - implements the ProcAmp DDI functions to modify the brightness, contrast, hue, and saturation.
CSC - Color Space Conversion
GEE - Gamut Expansion and Color Correction in Linear RGB Space

Programming Note
The input and output dimensions are restricted to 16K for VEBOX DN/DI/IECP/Capture Pipe.

VEBOX State and Primitive Commands

Every engine can have internal state that can be common and reused across the data entities it processes instead of reloading for every data entity.

There are two kinds of state information:

1. Surface state or state of the input and output data containers.
2. Engine state or the architectural state of the processing unit.

For example, in the case of DN/DI, architectural state information such as denoise filter strength can be the same across frames. This section gives the details of both the surface state and engine state.

Each frame should have these commands, in this order:

1. VEBOX_State
2. VEBOX_Surface_state for input & output
3. VEB_DI_IECP

Alternatively, VEBOX_Tiling_Convert can be used instead of VEB_DI_IECP.

VEBOX State

This chapter discusses various commands that control the internal functions of the VEBOX. The following commands are covered:

Command
DN/DI State Table Contents
VEBOX_IECP_STATE
VEBOX_FORWARD_GAMMA_CORRECTION_STATE
VEBOX_STATE
VEBOX_Ch_Dir_Filter_Coefficient

DN-DI State Table Contents

This section contains tables that describe the state commands that are used by the Denoise and Deinterlacer functions.

VEBOX_DNDI_STATE

VEBOX_IECP_STATE

For all piecewise linear functions in the following table, the control points must be monotonically increasing (increasing continuously) from the lowest control point to the highest. Functions which have bias/correction values associated with each control point have the additional restriction that any control points which have the same value must also have the same bias/correction value. The piecewise linear functions include:

- For Skin Tone Detection:
    - Y_point_4 to Y_point_0
    - P3L to P0L
    - P3U to P0U
    - SATP3 to SATP1
    - HUEP3 to HUEP1
    - SATP3_DARK to SATP1_DARK
    - HUEP3_DARK to HUEP1_DARK
- For ACE:
    - Ymax, Y10 to Y1 and Ymin
    - There is no state variable to set the bias for Ymin and Ymax. The biases for these two points are equal to the control point values: B0 = Ymin and B11 = Ymax. That means that if control points adjacent to Ymin and Ymax have the same value as Ymin/Ymax then the biases must also be equal to the Ymin/Ymax control points based on the restriction mentioned above.
- Forward Gamma correction
    - Gamma correction table (1K points & correction values)
- Gamut Expansion:
    - Gamma Correction (256 points & correction values)

Inverse Gamma Correction (256 points & correction values)

Command
VEBOX_STD_STE_STATE
VEBOX_ACE_LACE_STATE
VEBOX_TCC_STATE
VEBOX_PROCAMP_STATE
VEBOX_CSC_STATE
VEBOX_ALPHA_AOI_STATE
VEBOX_CCM_STATE
VEBOX_FRONT_END_CSC_STATE
VEBOX_GAMUT_CONTROL_STATE
Gamut_Expansion_Gamma_Correction
VEBOX_VERTEX_TABLE
VEBOX_CAPTURE_PIPE_STATE
VEBOX_FORWARD_GAMMA_CORRECTION_STATE
VEBOX_RGB_TO_GAMMA_CORRECTION

VEBOX Surface State

VEBOX_SURFACE_STATE

Surface Format Restrictions

The surface formats and tiling allowed are restricted, depending on which function is consuming or producing the surface.

Surface Format Restrictions

FourCC Code	Format	DN/DI Input	DN/DI Output	IECP Input	IECP Output	Capture Output	Scalar Input/Output
YUYV	YCRCB_NORMAL (4:2:2)	X	X	X	X	X	X
VYUY	YCRCB_SwapUVY (4:2:2)	X	X	X	X	X	X
YVYU	YCRCB_SwapUV (4:2:2)	X	X	X	X	X	X
UYVY	YCRCB_SwapY (4:2:2)	X	X	X	X	X	X
Y8	Y8 Monochrome	X	X	X	X	X	X
NV12	NV12 (4:2:0 with interleaved U/V)	X	X	X	X	X	X
AYUV	4:4:4 with Alpha (8-bit per channel)			X	X	X	X
Y216	4:2:2 packed 16-bit			X	X	X	X
Y416	4:4:4 packed 16-bit			X	X	X	X
Y410	4:4:4 packed 10-bit				X	X	X
P216	4:2:2 planar 16-bit			X	X	X	X
P016	4:2:0 planar 16-bit			X	X	X	X
Y16	Y16 Monochrome	X	X	X	X	X	X
	RGBA 10:10:10:2				X	X	
	RGBA 8:8:8:8	Spatial DN		X	X	X	
	RGBA 16:16:16:16	Spatial DN		X	X	X	
	BGRA 8:8:8:8				X	X	
Tiling							
	Tile Y	X	X	X	X	X	X
	Tile X	X	X	X	X	X	X
	Linear	X	X	X	X	X	X

Surface Format Restrictions

FourCC Code	Format	DN Input/Output	DI Input/Output	IECP Input	IECP Output	Capture Output	Scalar Input/Output
YUYV	YCRCB_NORMAL (4:2:2)	X	X	X	X	X	X
VYUY	YCRCB_SwapUVY (4:2:2)	X	X	X	X	X	X
YVYU	YCRCB_SwapUV (4:2:2)	X	X	X	X	X	X
UYVY	YCRCB_SwapY (4:2:2)	X	X	X	X	X	X
Y8	Y8 Monochrome	X	X	X	X	X	X
NV12	NV12 (4:2:0 with interleaved U/V)	X	X	X	X	X	X
AYUV	4:4:4 with Alpha (8-bit per channel)	X	Output only	X	X	X	X
Y216	4:2:2 packed 16-bit	X	X	X	X	X	X
Y416	4:4:4 packed 16-bit	X	Output only	X	X	X	X
Y410	4:4:4 packed 10-bit	X	Output only	X	X	X	X
P216	4:2:2 planar 16-bit	X	X	X	X	X	X
P016	4:2:0 planar 16-bit	X	X	X	X	X	X
Y16	Y16 Monochrome	X	X	X	X	X	X
	RGBA 10:10:10:2				X	X	
	RGBA 8:8:8:8	Spatial DN		X	X	X	
	RGBA 16:16:16:16	Spatial DN		X	X	X	
	BGRA 8:8:8:8				X	X	
Tiling							
	Tile Y	X	X	X	X	X	X
	Tile X	X	X	X	X	X	X
	Linear	X	X	X	X	X	X

Surface Formats - Feature Notes

Feature
Surfaces are 4 kb aligned, chroma X offset is cache line aligned (16 byte).
If Y8/Y16 is used as the input format, it must also be used for the output format (chroma is not created by VEBOX).
If IECP and either DN or DI are enabled at the same time, it is possible to select any input that is legal for DN/DI and any output which is legal for IECP. The only exception is that if DN or DI are enabled, the IECP is not able to output P216 and P016.
16-bit data from IECP or DN is rounded when converting to 8-bit output formats.
High Speed Bypass has the same format limitations as IECP Input/Output, but the surface formats for the input and output must be the same.
Capture Input is only linear Bayer Surface Format.
Input formats for Demosaic, White Balance, Vignette and Black Level Correction must be linear Bayer.
For capture pipe, we **can** support the combination of DN and P216 and P016. For capture pipe with a P016 output the U/V output is not an average of the 4 component pixels, but the U/V for pixel 4 (the lower right pixel of the 4).
Output format Y410 is supported for DN, DI and DM modes only with IECP enabled. In non IECP cases, default of "0xFFFF" is sent if output format requires alpha.
For DN 444 input formats interlaced input content is not supported
If IECP and either DN or DI are enabled at the same time, it is possible to select any input that is legal for DN/DI and any output which is legal for IECP.

SFC

This chapter describes the SFC Media Engine.

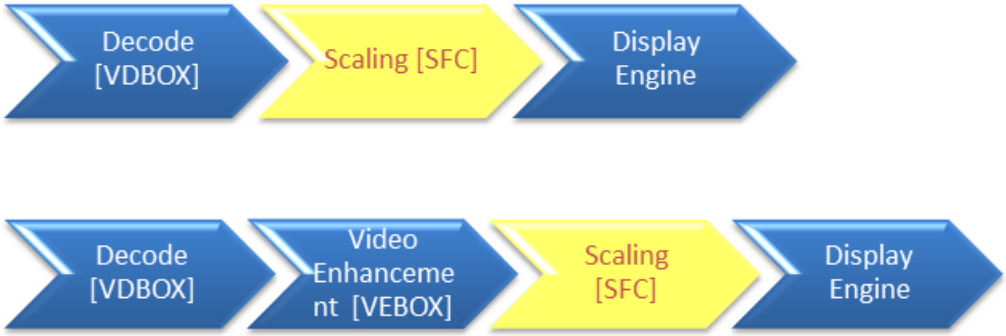
SFC Overview

Scaler & Format Converter (SFC) pipeline is a multi-format scaling engine to accelerate several media usages and achieve ultra low-power video playback.

Ultra Low-Power Usages

Several power saving techniques are brought into consideration and incorporated into the architecture of SFC pipeline: offloading from EU to fixed function to reduce Cdyn power, memory traffic reduction to lower IO and DDR power, and native surface support between acceleration engines.

Two ultra low-power playback modes are introduced to achieve sub 1-Watt solution: Decode->Scale-> Display (sprite) and Decode-> Image Enhancement-> Scale-> Display (sprite).



- In these two usages, SFC is fed by the decoder (VDBOX) and image enhancer (VEBOX) directly instead of writing to memory and read back from memory. A direct data bus is added between VD-to-SFC and VE-to-SFC. SFC will also include an internal store buffer to capture overlap pixel data between column/rows. In another word, the only IO traffic to DDR is the final scaled surface writes. All input and intermediate traffics related to SFC engine are confined inside GT and not expose to external components.
- EU-less usage: SFC is a fixed function engine architects to run concurrently along VDBOX or VEBOX. i.e. Decode and scaling will be happening at the same time, or Image enhancement and scaling will be occurring at the same time. It saves power by offloading the scaling workload off the media render engine to this dedicated engine which is much smaller.
- In both cases, scaling operation is the last processing step before final pixels are presented by the display engine. SFC is designed to generate output format native to display engine. This reduces the memory traffic caused by elimination of the extra memory copy used to convert the format incompatibility between engines. In addition, SFC supports 90 degree clockwise rotation of the final pixel surface for tablet space.

- SFC pipeline is chained together with VDBOX and VEBOX with direct interface and ability to run concurrently. VDBOX/VEBOX sends control parameters and pixel data directly to SFC through direct interface. This help reduces the IO and package power by eliminating the traffic to memory, and allows VD/VE to run concurrently along with SFC pipeline. SFC pipeline is a shared resource that can be called and accessed by VDBOX or VEBOX. A lock must be placed and granted with an acknowledgement ahead of transferring data to SFC. On completion, the lock must be removed to free up the shared resource (SFC).

SFC Commands Definition

This section contains definitions for commands used with the scaler and format converter (SFC). These commands are sent from the VDBOX/VEBOX to the SFC pipeline.

SFC_AVS_LUMA_Coeff_Table

SFC_AVS_CHROMA_Coeff_Table

SFC_AVS_STATE

SFC_FRAME_START

SFC_LOCK

SFC_STATE

Intel® Iris® Xe and UHD Graphics Open Source

Programmer's Reference Manual

For the 2020-2021 11th Generation Intel Xeon®, Core™, Celeron®, Pentium® Gold Processors based on the "Tiger Lake" Platform

Volume 12: Display Engine

December 2021, Revision 1.0

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Table of Contents

Display **1**

Terminology 1

VGA and Extended VGA Registers 3

General Control and Status Registers 4

Sequencer Registers 8

Graphics Controller Registers 12

Attribute Controller Registers 24

VGA Color Palette Registers 30

CRT Controller Register 34

Display Audio Codec Verbs 55

Block Diagram 55

Codec Node Hierarchy 55

Programming 57

North Display Engine Registers 110

Xe_D Overview 110

Mode Set 122

Clocks 168

Shared Functions 197

Central Power 208

Pipe 221

Planes 279

DSC 308

Transcoder 316

Audio 376

DisplayPort Transport 390

Digital Display Interface 390

Global Time Code (GTC) 419

Display Watermark Programming 421

Watermark Calculations 421

Memory Values 428

Retrieve Memory Latency Data 428

Memory Latency Data Definition 428

SAGV Block Time 429

Display

Terminology

Term	Description
DP	DisplayPort
SST, DP SST	DisplayPort Single Stream Transport
MST, DP MST	DisplayPort Multi Stream Transport

Register Access Field	Description	Implementation
R/W (Read/Write)	The value written into this register will control hardware and is the same value that will be read.	Write data is stored. Read is from the stored data. Stored value is used to control hardware.
Reserved	Unused register bit. Don't assume a value for these bits. Writes have no effect.	Write data is ignored. Read is zero.
MBZ (Must Be Zero)	Always write a zero to this register.	May be implemented as Reserved or as R/W.
PBC (Preserve Bit Contents)	Software must write the original value back to this bit. This allows new features to be added using these bits.	May be implemented as Reserved or as R/W.
Read Only	The read value is determined by hardware. Writes to this bit have no effect.	Write data is ignored. Read is from a status signal or some other internal source.
Write Only	The value written into this register will control hardware. Reads return zero.	Write data is stored. Read is zero. Stored value is used to control hardware.
R/W Clear (Read/Write Clear)	Sticky status bit. Hardware will set the bit, software can clear it with a write of 1b.	Internal hardware events set a sticky bit. Read is from the sticky bit. A write of 1b clears the sticky bit.
R/W Set (Read/Write Set)	Sticky status bit. Software can set the bit with a write of 1b. Hardware will clear the bit.	A write of 1b sets a sticky bit. Internal hardware events clear a sticky bit. Read is from the sticky bit.
Double Buffered	Write when desired and the written value will take effect at the time of the update point specified in the 'Double Buffer Update Point'	Two stages of registers used. Write data is stored into first

Register Access Field	Description	Implementation
	parameter. Reads will return the written value, which is not necessarily the value being currently used to control hardware. Some double-buffered registers have a corresponding "LIVE" read only register that provides the value being use to control hardware. Some have a specific arming sequence where a write to another register, specified in the 'Double Buffer Armed By' parameter, is required before the update can take place. Once the armed by register is written to, the written values of all registers controlled by that arming will take effect at the time of the double buffer update point. This is used to ensure atomic updates of several registers. Note: Once armed, by write to the armed by register, the registers controlled by this arming should not be changed until the double buffer update point is reached. If changed, this will disarm the sequence and will require another write to the armed by register to get it to the armed status again.	stage. Read is from the first stage stored data. First stage stored value is transferred to second stage storage at the double buffer update point. Second stage stored value is used to control hardware. Arm/disarm logic may be used for some registers to control the double buffer update point.
Write/Read Status	The value written into this register will control hardware. The read value is determined by hardware.	Write data is stored. Stored value is used to control hardware. Read is from a status signal or some other internal source.

VGA and Extended VGA Registers

This section describes the registers and the functional operation notations for the observable registers in the VGA section. This functionality is provided as a means for support of legacy applications and operating systems.

It is important to note that these registers in general have the desired effects only when running VGA display modes. The main exceptions to this are the palette interface which allows real mode DOS applications and full screen VGA applications under an OS control running in high resolution (non-VGA) modes to access the palette through the VGA register mechanisms and the use of the ST01 status bits that determine when the VGA enters display enable and sync periods. Other exceptions include the register bits that control the memory accesses through the A000:0000 and B000:0000 memory segments which are used during operating system emulation of VGA for "DOS box" applications.

Some of the functions of the VGA are enabled or defeated through the programming of the VGA control register bits that are located in the MMIO register space.

Given the legacy nature of this function, it has been adapted to the changing environment that it must operate within. The three most notable changes are the addition of high resolution display mode support, new operating system support, and the use of fixed resolution display devices (such as LCD panels). Additional control bits in the PCI Config space will affect the ability to access the registers and memory aperture associated with VGA.

Mode of Operation	VGA Disable	VGA Display	VGA Registers	Palette (VGA)	VGA Memory	VGA Banking
VGA DOS	No	Yes	Yes	Yes	Yes	No
HiRes DOS	Yes	No	Yes	Yes	No	Yes
Fullscreen DOS	Yes/No	No/Yes	Yes	Yes	Yes	Yes
DOS Emulation	Yes	No	Yes	Yes	Yes	Yes

VGA Display Mode	Dot Clock Select	Dot Clock Range	132 Column Text Support	9-Dot Disable Support	Main Use
Native	VGA Clock Select	25/28 MHz	No	No	Analog CRT (VGA connector)
Centered	Fixed at display Requirements	Product Specific	No	Yes	Digital Display
Upper Left Corner	Fixed at display Requirements	Product Specific	No	Yes	Internal Panel

Native, Centered, and Upper Left Corner support varies from product to product.

Even in the native VGA display operational modes, not all combinations of bit settings result in functional operating modes. VGA display modes have the restriction that they can be used only when all other display planes are disabled.

These registers are accessed via I/O space. The I/O space resides in the PCI compatibility hole and uses only the addresses that were part of the original VGA I/O space (which includes EGA and MDA emulation). Accesses to the VGA I/O addresses are steered to the proper bus and rely on proper setup of bridge registers. Extended VGA registers such as GR10 and GR11 use additional indexes for the already defined I/O addresses. VGA register accesses are allowed as 8 or 16 bit naturally aligned transactions only. Word transactions must have the least significant bit of the address set to zero. DWORD I/O operations should not be performed on these registers.

Some products may support access to these registers through MMIO. The access method varies and is documented elsewhere.

General Control and Status Registers

The setup, enable, and general registers are all directly accessible by the CPU. A sub indexing scheme is not used to read from and write to these registers.

Various bits in these registers provide control over the real-time status of the horizontal sync signal, the horizontal retrace interval, the vertical sync signal, and the vertical retrace interval. The horizontal retrace interval is the period during the drawing of each scan line containing active video data, when the active video data is not being displayed. This period includes the horizontal front and back porches, and the horizontal sync pulse. The horizontal retrace interval is always longer than the horizontal sync pulse. The vertical retrace interval is the period during which the scan lines not containing active video data are drawn. This includes the vertical front porch, back porch, and the vertical sync pulse. The vertical retrace interval is normally longer than the vertical sync pulse.

ST00 - Input Status 0

Address: 3C2h

Default: 00h

Attributes: Read Only

Bit	Descriptions
7	**CRT Interrupt Pending.** This bit is here for EGA compatibility and **will always return zero**. The generation of interrupts was originally enabled, through bits [4,5] of the Vertical Retrace End Register (CR11). This ability to generate interrupts at the start of the vertical retrace interval is a feature that is typically unused by DOS software and therefore is only supported through other means for use under a operating system support. 0 = CRT (vertical retrace interval) interrupt is not pending. 1 = CRT (vertical retrace interval) interrupt is pending
6:5	**Reserved.** Read as 0s.
4	**RGB Comparator / Sense.** This bit is here for compatibility and **will always return one**. Monitor detection must be done be done through the programming of hotplug registers in the MMIO space. 0 = Below threshold 1 = Above threshold
3:0	**Reserved.** Read as 0s.

ST01 - Input Status 1

Address: 3BAh/3DAh

Default: 00h

Attributes: Read Only

The address selection is dependent on CGA or MDA emulation mode as selected via the MSR register.

Bit	Descriptions
7	**Reserved (as per VGA specification).** Read as 0s.
6	**Reserved.** Read as 0.
5:4	**Video Feedback 1, 0.** These bits are connected to 2 of the 8 color bits sent to the palette. Bits 4 and 5 of the Color Plane Enable Register (AR12) selects which two of the 8 possible color bits become connected to these 2 bits of this register. These bits exist for EGA compatibility.
3	**Vertical Retrace/Video.** 0 = VSYNC inactive (Indicates that a vertical retrace interval is not taking place). 1 = VSYNC active (Indicates that a vertical retrace interval is taking place). VGA pixel generation is not locked to the display output but is loosely coupled. A VSYNC indication may not occur during the actual VSYNC going to the display but during the VSYNC that is generated as part of the VGA pixel generation. The exact relationship will vary with the VGA display operational mode. This status bit will remain active when the VGA is disabled and the device is running in high resolution modes (non-VGA) to allow for applications that (now incorrectly) use these status registers bits. In this case, the status will come from the pipe that the VGA is assigned to. Bits 4 and 5 of the Vertical Retrace End Register (CR11) previously could program this bit to generate an interrupt at the start of the vertical retrace interval. This ability to generate interrupts at the start of the vertical retrace interval is a feature that is largely unused by legacy software. Interrupts are not supported through the VGA register bits.
2:1	**Reserved.** Read as 0s.
0	**Display Enable Output.** Display Enable is a status bit (bit 0) in VGA Input Status Register 1 that indicates when either a horizontal retrace interval or a vertical retrace interval is taking place. This bit was used with the EGA graphics system (and the ones that preceded it, including MDA and CGA). In those cases, it was important to check the status of this bit to ensure that one or the other retrace intervals was taking place before reading from or writing to the frame buffer. In these earlier systems, reading from or writing to the frame buffer at times outside the retrace intervals meant that the CRT controller would be denied access to the frame buffer. Those behaviors resulted in either "snow" or a flickering display. This bit provides compatibility with software designed for those early graphics controllers. This bit is currently used in DOS applications that access the palette to prevent the sparkle associated with read and write accesses to the palette RAM with the same address on the same clock cycle. **This status bit remains active when the VGA display is disabled and the device is running in high resolution modes (non-VGA) to allow for applications that (now considered incorrect) use these status registers bits. In this case, the status will come from the pipe that the VGA is assigned to. When in panel fitting VGA or centered VGA operation, the meaning of these bits will not be consistent with native VGA timings.** 0 = Active display data is being sent to the display. Neither a horizontal retrace interval or a vertical retrace interval is currently taking place. 1 = Either a horizontal retrace interval (horizontal blanking) or a vertical retrace interval (vertical blanking) is

Bit	Descriptions
	currently taking place.

FCR - Feature Control

Address: 3BAh/3DAh - Write; 3CAh - Read

Default: 00h

Attributes: Read/Write

The address used for writes is dependent on CGA or MDA emulation mode as selected via the MSR register. In the original EGA, bits 0 and 1 were used as part of the feature connector interface. Feature connector is not supported in these devices and those bits will always read as zero.

Bit	Descriptions
7:4	**Reserved.** Read as 0.
3	**VSYNC Control.** This bit is provided for compatibility only and has no other function. Reads and writes to this bit have no effect other than to change the value of this bit. The previous definition of this bit selected the output on the VSYNC pin. 0 = Was used to set VSYNC output on the VSYNC pin (default). 1 = Was used to set the logical 'OR' of VSYNC and Display Enable output on the VSYNC pin. This capability was not typically very useful.
2:0	**Reserved.** Read as 0.

MSR - Miscellaneous Output

Address: 3C2h - Write; 3CCh - Read

Default: 00h

Attributes: Read/Write

Bit	Descriptions
7	**CRT VSYNC Polarity.** This is a legacy function that is used in native VGA modes. For most cases, sync polarity will be controlled by the port control bits. The VGA settings can be optionally selected for compatibility with the original VGA when used in the VGA native mode. Sync polarity was used in VGA to signal the monitor how many lines of active display are being generated. 0 = Positive Polarity (default). 1 = Negative Polarity.
6	**CRT HSYNC Polarity.** This is a legacy function that is used in native VGA modes. For most cases, sync polarity will be controlled by the port control bits. The VGA settings can be optionally selected for compatibility with

Bit	Descriptions
	the original VGA when used in the VGA native mode. 0 = Positive Polarity (default). 1 = Negative Polarity
5	**Page Select.** In Odd/Even Memory Map Mode 1 (GR6), this bit selects the upper or lower 64 KB page in display memory for CPU access: 0 = Upper page (default) 1 = Lower page. Selects between two 64KB pages of frame buffer memory during standard VGA odd/even modes (modes 0h through 5h). Bit 1 of register GR06 can also program this bit in other modes. This bit is would normally set to 1 by the software.
4	**Reserved.** Read as 0.
3:2	**Clock Select.** These bits can select the dot clock source for the CRT interface. The bits should be used to select the dot clock in standard native VGA modes only. When in the centering or upper left corner modes, these bits should be set to have no effect on the clock rate. The actual frequencies that these bits select, if they have any affect at all, is programmable through the PLL MMIO registers. 00 = CLK0, 25.175 MHz (for standard VGA modes with 640 pixel (8-dot) horizontal resolution) (default) 01 = CLK1, 28.322 MHz. (for standard VGA modes with 720 pixel (9-dot) horizontal resolution) 10 = Was used to select an external clock (now unused) 11 = Reserved
1	**A0000-BFFFFh Memory Access Enable.** VGA Compatibility bit enables access to video memory (frame buffer) at A0000-BFFFFh. When disabled, accesses to VGA memory are blocked in this region. This bit is independent of and does not block CPU access to the video linear frame buffer at other addresses. 0 = Prevent CPU access to memory/registers/ROM through the A0000-BFFFF VGA memory aperture (default). 1 = Allow CPU access to memory/registers/ROM through the A0000-BFFFF VGA memory aperture. This memory must be mapped as UC by the CPU.
0	**I/O Address Select.** This bit selects 3Bxh or 3Dxh as the I/O address for the CRT Controller registers, the Feature Control Register (FCR), and Input Status Register 1 (ST01). Presently ignored (whole range is claimed), but will "ignore" 3Bx for color configuration or 3Dx for monochrome. It is typical in AGP chipsets to shadow this bit and properly steer I/O cycles to the proper bus for operation where a MDA exists on another bus such as ISA. 0 = Select 3Bxh I/O address (MDA emulation) (default). 1 = Select 3Dxh I/O address (CGA emulation).

In standard VGA modes using the analog VGA connector, bits 7 and 6 indicate which of the three standard VGA vertical resolutions the standard VGA display should use. Extended modes, including those with a vertical resolution of 480 scan lines, may use a setting of 0 for both of these bits. Different

connector standards and timing standards specify the proper use of sync polarity. This setting was "reserved" in the VGA standard.

Analog CRT Display Sync Polarities

V	H	Display	Horizontal Frequency	Vertical Frequency
P	P	200 Line	15.7 KHz	60 Hz
N	P	350 Line	21.8 KHz	60 Hz
P	N	400 Line	31.5 KHz	70 Hz
N	N	480 Line	31.5 KHz	60 Hz

Sequencer Registers

To access registers the VGA Sequencer Index register (SRX) at address 3C4h is written with the index of the desired register. Then the desired register is accessed through the data port for the sequencer registers at address 3C5.

SRX - Sequencer Index

Address: 3C4h

Default: 00h

Attributes: Read/Write

Bit	Description
7:3	**Reserved.** Read as 0s.
2:0	**Sequencer Index.** This field contains a 3-bit Sequencer Index value used to access sequencer data registers at indices 0 through 7.

SR00 - Sequencer Reset

Address: 3C5h(Index=00h)

Default: 00h

Attributes: Read/Write

Bit	Descriptions
7:2	**Reserved.** Read as 0.
1	**Reserved.** Reserved for VGA compatibility (was reset).
0	**Reserved.** Reserved for VGA compatibility. (was reset)

SR01 - Clocking Mode

Address: 3C5h (Index=01h)

Default: 00h

Attributes: Read/Write

Bit	Descriptions
7:6	**Reserved.** Read as 0s.
5	**Screen Off.** 0 = Normal Operation (default). 1 = Disables video output (blanks the screen) and turns off display data fetches. Synchronization pulses to the display, however, are maintained. Setting this bit to 1 had been used as a way to more rapidly update and improve CPU access performance to the frame buffer during VGA modes. In non-VGA modes (VGA Disable=1), this bit has no effect. Before the VGA is disabled through the MMIO VGA control register, this bit should be set to stop the memory accesses from the display. The following sequence must be used when disabling the VGA plane. 1. Write SR01 to set bit 5 = 1 to disable video output. 2. Wait for 100us. 3. Disable the VGA plane via Bit 31 of the MMIO VGA control register (location found in the MMIO display register programming specification).
4	**Shift 4.** 0 = Load video shift registers every 1 or 2 character clocks (depending on bit 2 of this register) (default). 1 = Load shift registers every 4th character clock.
3	**Dot Clock Divide.** Setting this bit to 1 stretches doubles all horizontal timing periods that are specified in the VGA horizontal CRTC registers. This bit is used in standard VGA 40-column text modes to stretch timings to create horizontal resolutions of either 320 or 360 pixels (as opposed to 640 or 720 pixels, normally used in standard VGA 80-column text modes). The effect of this is that there will actually be twice the number of pixels sent to the display per line. 0 = Pixel clock is left unaltered (used for 640 (720) pixel modes); (default). 1 = Pixel clock divided by 2 (used for 320 (360) pixel modes).
2	**Shift Load.** Bit 4 of this register must be 0 for this bit to be effective. 0 = Load video data shift registers every character clock (default). 1 = Load video data shift registers every other character clock.
1	**Reserved.** Read as 0.
0	**8/9 Dot Clocks.** This bit determines whether a character clock is 8 or 9 dot clocks long if clock doubling is disabled and 16 or 18 clocks if it is. This also changes the interpretation of the pixel panning values (see chart). An additional control bit determines if this bit is to be ignored and 8-dot characters are to be used always. The 9-dot disable would be used when doubling the horizontal pixels on a 1280 wide display or non-doubling on a 640 wide display. Panning however will occur according to the expected outcome. 0 = 9 dot clocks (9 horizontal pixels) per character in text modes with a horizontal resolution of 720 pixels. 1 = 8 dot clocks (8 horizontal pixels) per character in text or graphics modes with a horizontal resolution of

Bit	Descriptions
	640 pixels.

SR02 - Plane/Map Mask

Address: 3C5h (Index=02h)

Default: 00h

Attributes: Read/Write

Bit	Descriptions
7:4	**Reserved.** Read as 0s.
3:0	**Memory Planes [3:0] Processor Write Access Enable.** In both the Odd/Even Mode and the Chain 4 Mode, these bits still control access to the corresponding color plane. 0 = Disable. 1 = Enable. This register is referred to in the VGA standard as the Map Mask Register.

SR03 - Character Font

Address: 3C5h (index=03h)

Default: 00h

Attributes: Read/Write

In text modes, bit 3 of the video data's attribute byte normally controls the foreground intensity. This bit may be redefined to control switching between character sets. This latter function is enabled whenever there is a difference in the values of the Character Font Select A and the Character Font Select B bits. If the two values are the same, the character select function is disabled and attribute bit 3 controls the foreground intensity.

Bit 1 of the Memory Mode Register (SR04) must be set to 1 for the character font select function of this register to be active. Otherwise, only character maps 0 and 4 are available.

Bit	Descriptions
7:6	**Reserved.** Read as 0s.
3:2,5	**Character Map Select Bits for Character Map B.** These three bits are used to select the character map (character generator tables) to be used as the secondary character set (font). The numbering of the maps is not sequential.

Bit [3:2,5]	Map Number	Table Location
00,0	0	1st 8KB of plane 2 at offset 0 (default)
00,1	4	2nd 8KB of plane 2 at offset 8K
01,0	1	3rd 8KB of plane 2 at offset 16K
01,1	5	4th 8KB of plane 2 at offset 24K
10,0	2	5th 8KB of plane 2 at offset 32K

Bit	Descriptions
	10,1 | 6 | 6th 8KB of plane 2 at offset 40K 11,0 | 3 | 7th 8KB of plane 2 at offset 48K 11,1 | 7 | 8th 8KB of plane 2 at offset 56K
1:0,4	**Character Map Select Bits for Character Map A.** These three bits are used to select the character map (character generator tables) to be used as the primary character set (font). The numbering of the maps is not sequential. Bit [1:0,4] | Map Number | Table Location 00,0 | 0 | 1st 8KB of plane 2 at offset 0 (default) 00,1 | 4 | 2nd 8KB of plane 2 at offset 8K 01,0 | 1 | 3rd 8KB of plane 2 at offset 16K 01,1 | 5 | 4th 8KB of plane 2 at offset 24K 10,0 | 2 | 5th 8KB of plane 2 at offset 32K 10,1 | 6 | 6th 8KB of plane 2 at offset 40K 11,0 | 3 | 7th 8KB of plane 2 at offset 48K 11,1 | 7 | 8th 8KB of plane 2 at offset 56K

SR04 - Memory Mode Register

Address: 3C5h (index=04h)

Default: 00h

Attributes: Read/Write

Bit	Description
7:4	**Reserved.** Read as 0.
3	**Chain 4 Mode.** The selections made by this bit affect both CPU Read and write accesses to the frame buffer. 0 = The manner in which the frame buffer memory is mapped is determined by the setting of bit 2 of this register (default). 1 = The frame buffer memory is mapped in such a way that the function of address bits 0 and 1 are altered so that they select planes 0 through 3. This setting is used in mode x13 to allow all four planes to be accessed via sequential addresses.
2	**Odd/Even Mode.** Bit 3 of this register must be set to 0 for this bit to be effective. The selections made by this bit affect only non-paged CPU accesses to the frame buffer through the VGA aperture. 0 = The frame buffer memory is mapped in such a way that the function of address bit 0 such that even addresses select planes 0 and 2 and odd addresses select planes 1 and 3 (default). 1 = Addresses sequentially access data within a bit map, and the choice of which map is accessed is made according to the value of the Plane Mask Register (SR02).
1	**Extended Memory Enable.** This bit must be set to 1 to enable the selection and use of character maps in plane 2 via the Character Map Select Register (SR03). 0 = Disable CPU accesses to more than the first 64KB of VGA standard memory (default). 1 = Enable CPU accesses to the rest of the 256KB total VGA memory beyond the first 64KB.

Bit	Description
0	**Reserved.** Read as 0.

SR07 - Horizontal Character Counter Reset

Address: 3C5h (index=07h)

Default: 00h

Attributes: Read/Write

For standard VGAs, writing this register (with any data) causes the horizontal character counter to be held in reset (the character counter output will remain 0). It remained in reset until a write occurred to any other sequencer register location with SRX set to an index of 0 through 6. In this implementation that sequence has no such special effect.

The vertical line counter is clocked by a signal derived from the horizontal display enable (which does not occur if the horizontal counter is held in reset). Therefore, if a write occurs to this register during the vertical retrace interval, both the horizontal and vertical counters will be set to 0. A write to any other sequencer register location (with SRX set to an index of 0 through 6) may then be used to start both counters with reasonable synchronization to an external event via software control. Although this was a standard VGA register, it was not documented.

Bit	Description
7:0	**Horizontal Character Counter.**

Graphics Controller Registers

Accesses to the registers of the VGA Graphics Controller are done through the use of address 3CEh written with the index of the desired register. Then the desired register is accessed through the data port for the graphics controller registers at address 3CFh. Indexes 10 and 11 must only be accessed through the I/O space.

GRX - GRX Graphics Controller Index Register

Address: 3CEh

Default:
000UUUUUb (U=Undefined)
Attributes:
Read/Write

Bit	Description
7:5	**Reserved.** Read as 0.
4:0	**Graphics Controller Register Index.** This field selects any one of the graphics controller registers (GR00-GR18) to be accessed via the data port at address 3CFh.

GR00 - Set/Reset Register

Address: 3CFh (index=00h)

Default: 0Uh (U=Undefined)

Attributes:
Read/Write

Bit	Description
7:4	**Reserved.** Read as 0.
3:0	**Set/Reset Plane [3:0].** When the Write Mode bits (bits 0 and 1) of the Graphics Mode Register (GR05) are set to select Write Mode 0, all 8 bits of each byte of each memory plane are set to either 1 or 0 as specified in the corresponding bit in this register, if the corresponding bit in the Enable Set/Reset Register (GR01) is set to 1. When the Write Mode bits (bits 0 and 1) of the Graphics Mode Register (GR05) are set to select Write Mode 3, all CPU data written to the frame buffer is rotated, then logically ANDed with the contents of the Bit Mask Register (GR08), and then treated as the addressed data's bit mask, while value of these four bits of this register are treated as the color value.

GR01 - Enable Set/Reset Register

Address: 3CFh (Index=01h)

Default: 0Uh (U=Undefined)

Attributes:
Read/Write

Bit	Description
7:4	**Reserved.** Read as 0.
3:0	**Enable Set/Reset Plane [3:0].** This register works in conjunction with the Set/Reset Register (GR00). The Write Mode bits (bits 0 and 1) must be set for Write Mode 0 for this register to have any effect. 0 = The corresponding memory plane can be read from or written to by the CPU without any special bitwise operations taking place. 1 = The corresponding memory plane is set to 0 or 1 as specified in the Set/Reset Register (GR00).

GR02 - Color Compare Register

Address: 3CFh (Index=02h)

Default: 0Uh (U=Undefined)

Attributes:
Read/Write

Bit	Description
7:4	**Reserved.** Read as 0.
3:0	**Color Compare Plane [3:0].** When the Read Mode bit (bit 3) of the Graphics Mode Register (GR05) is set to select Read Mode 1, all 8 bits of each byte of each of the 4 memory planes of the frame buffer corresponding to the address from which a CPU read access is being performed are compared to the corresponding bits in this register (if the corresponding bit in the Color Don't Care Register (GR07) is set to 1). The value that the CPU receives from the read access is an 8-bit value that shows the result of this comparison, wherein value of 1 in a given bit position indicates that all of the corresponding bits in the bytes across all of the memory planes that were included in the comparison had the same value as their memory plane's respective bits in this register.

GR03 - Data Rotate Register

Address: 3CFh (Index=03h)

Default: 0Uh (U=Undefined)

Attributes:
Read/Write

Bit	Description
7:5	**Reserved.** Read as 0.
4:3	**Function Select.** These bits specify the logical function (if any) to be performed on data that is meant to be written to the frame buffer (using the contents of the memory read latch) just before it is actually stored in the frame buffer at the intended address location. 00 = Data being written to the frame buffer remains unchanged, and is simply stored in the frame buffer. 01 = Data being written to the frame buffer is logically ANDed with the data in the memory read latch before it is actually stored in the frame buffer. 10 = Data being written to the frame buffer is logically ORed with the data in the memory read latch before it is actually stored in the frame buffer. 11 = Data being written to the frame buffer is logically XORed with the data in the

Bit	Description
	memory read latch before it is actually stored in the frame buffer.
2:0	**Rotate Count.** These bits specify the number of bits to the right to rotate any data that is meant to be written to the frame buffer just before it is actually stored in the frame buffer at the intended address location.

GR04 - Read Plane Select Register

Address: 3CFh (Index=04h)

Default: 0Uh (U=Undefined)

Attributes:
Read/Write

Bit	Description
7:2	**Reserved.** Read as 0.
1:0	**Read Plane Select.** These two bits select the memory plane from which the CPU reads data in Read Mode 0. In Odd/Even Mode, bit 0 of this register is ignored. In Chain 4 Mode, both bits 1 and 0 of this register are ignored. The four memory planes are selected as follows: 00 = Plane 0 01 = Plane 1 10 = Plane 2 11 = Plane 3 These two bits also select which of the four memory read latches may be read via the Memory read Latch Data Register (CR22). The choice of memory read latch corresponds to the choice of plane specified in the table above. The Memory Read Latch Data register and this additional function served by 2 bits are features of the VGA standard that were never documented.

GR05 - Graphics Mode Register

Address: 3CFh (Index=05h)

Default: 0UUU U0UUb (U=Undefined)

Attributes:
Read/Write

Bit	Description
7	**Reserved.** Read as 0.

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>6:5</td><td>
Shift Register Control. In standard VGA modes, pixel data is transferred from the 4 graphics memory planes to the palette via a set of 4 serial output bits. These 2 bits of this register control the format in which data in the 4 memory planes is serialized for these transfers to the palette.

Bits [6:5]=00

One bit of data at a time from parallel bytes in each of the 4 memory planes is transferred to the palette via the 4 serial output bits, with 1 of each of the serial output bits corresponding to a memory plane. This provides a 4-bit value on each transfer for 1 pixel, making possible a choice of 1 of 16 colors per pixel.

Serial
<table>
<tr><th>Out</th><th>1st Xfer</th><th>2nd Xfer</th><th>3rd Xfer</th><th>4th Xfer</th><th>5th Xfer</th><th>6th Xfer</th><th>7th Xfer</th><th>8th Xfer</th></tr>
<tr><td>Bit 3</td><td>plane3 bit7</td><td>plane3 bit6</td><td>plane3 bit5</td><td>plane3 bit4</td><td>plane3 bit3</td><td>plane3 bit2</td><td>plane3 bit1</td><td>plane3 bit0</td></tr>
<tr><td>Bit 2</td><td>plane2 bit7</td><td>plane2 bit6</td><td>plane2 bit5</td><td>plane2 bit4</td><td>plane2 bit3</td><td>plane2 bit2</td><td>plane2 bit1</td><td>plane2 bit0</td></tr>
<tr><td>Bit 1</td><td>plane1 bit7</td><td>plane1 bit6</td><td>plane1 bit5</td><td>plane1 bit4</td><td>plane1 bit3</td><td>plane1 bit2</td><td>plane1 bit1</td><td>plane1 bit0</td></tr>
<tr><td>Bit 0</td><td>plane0 bit7</td><td>plane0 bit6</td><td>plane0 bit5</td><td>plane0 bit4</td><td>plane0 bit3</td><td>plane0 bit2</td><td>plane0 bit1</td><td>plane0 bit0</td></tr>
</table>

Bits [6:5]=01

Two bits of data at a time from parallel bytes in each of the 4 memory planes are transferred to the palette in a pattern that alternates per byte between memory planes 0 and 2, and memory planes 1 and 3. First the even-numbered and odd-numbered bits of a byte in memory plane 0 are transferred via serial output bits 0 and 1, respectively, while the even-numbered and odd-numbered bits of a byte in memory plane 2 are transferred via serial output bits 2 and 3. Next, the even-numbered and odd-numbered bits of a byte in memory plane 1 are transferred via serial output bits 0 and 1, respectively, while the even-numbered and odd-numbered bits of memory plane 3 are transferred via serial out bits 1 and 3. This provides a pair of 2-bit values (one 2-bit value for each of 2 pixels) on each transfer, making possible a choice of 1 of 4 colors per pixel.

Serial
<table>
<tr><th>Out</th><th>1st Xfer</th><th>2nd Xfer</th><th>3rd Xfer</th><th>4th Xfer</th><th>5th Xfer</th><th>6th Xfer</th><th>7th Xfer</th><th>8th Xfer</th></tr>
<tr><td>Bit 3</td><td>plane2 bit7</td><td>plane2 bit5</td><td>plane2 bit3</td><td>plane2 bit1</td><td>plane3 bi7t</td><td>plane3 bit5</td><td>plane3 bit3</td><td>plane3 bit1</td></tr>
</table>
</td></tr>
</table>

Bit	Description
	Bit 2 plane2 bit6 plane2 bit4 plane2 bit2 plane2 bit0 plane3 bit6 plane3 bit4 plane3 bit2 plane3 bit0 Bit 1 plane0 bit7 plane0 bit5 plane0 bit3 plane0 bit1 plane1 bit7 plane1 bit5 plane1 bit3 plane1 bit1 Bit 0 plane0 bit6 plane0 bit4 plane0 bit2 plane0 bit0 plane1 bit6 plane1 bit4 plane1 bit2 plane1 bit0 This alternating pattern is meant to accommodate the use of the Odd/Even mode of organizing the 4 memory planes, which is used by standard VGA modes 2h and 3h. **Bits [6:5]=1x** Four bits of data at a time from parallel bytes in each of the 4 memory planes are transferred to the palette in a pattern that iterates per byte through memory planes 0 through 3. First the 4 most significant bits of a byte in memory plane 0 are transferred via the 4 serial output bits, followed by the 4 least significant bits of the same byte. Next, the same transfers occur from the parallel byte in memory planes 1, 2 and lastly, 3. Each transfer provides either the upper or lower half of an 8 bit value for the color for each pixel, making possible a choice of 1 of 256 colors per pixel. This is the setting used in mode x13. Serial Out 1st Xfer 2nd Xfer 3rd Xfer 4th Xfer 5th Xfer 6th Xfer 7th Xfer 8th Xfer Bit 3 plane0 bit7 plane0 bit3 plane1 bit7 plane1 bit3 plane2 bit7 plane2 bit3 plane3 bit7 plane3 b3it Bit 2 plane0 bit6 plane0 bit2 plane1 bit6 plane1 bit2 plane2 bit6 plane2 bit2 plane3 bit6 plane3 bit2 Bit 1 plane0 bit5 plane0 bit1 plane1 bit5 plane1 bit1 plane2 bit5 plane2 bit1 plane3 bit5 plane3 bit1 Bit 0 plane0 bit4 plane0 bit0 plane1 bit4 plane1 bit0 plane2 bit4 plane2 bit0 plane3 bit4 plane3 bit0 This pattern is meant to accommodate mode 13h, a standard VGA 256-color graphics mode.
4	**Odd/Even Mode.** 0 = Addresses sequentially access data within a bit map, and the choice of which map is accessed is made according to the value of the Plane Mask Register (SR02). 1 = The frame buffer is mapped in such a way that the function of

Bit	Description
	address bit 0 is such that even addresses select memory planes 0 and 2 and odd addresses select memory planes 1 and 3. This works in a way that is the inverse of (and is normally set to be the opposite of) bit 2 of the Memory Mode Register (SR02).
3	**Read Mode.** 0 = During a CPU read from the frame buffer, the value returned to the CPU is data from the memory plane selected by bits 1 and 0 of the Read Plane Select Register (GR04). 1 = During a CPU read from the frame buffer, all 8 bits of the byte in each of the 4 memory planes corresponding to the address from which a CPU read access is being performed are compared to the corresponding bits in this register (if the corresponding bit in the Color Don't Care Register (GR07) is set to 1). The value that the CPU receives from the read access is an 8-bit value that shows the result of this comparison. A value of 1 in a given bit position indicates that all of the corresponding bits in the bytes across all 4 of the memory planes that were included in the comparison had the same value as their memory plane's respective bits in this register.
2	**Reserved.** Read as 0.
1:0	**Write Mode.** 00 = Write Mode 0 - During a CPU write to the frame buffer, the addressed byte in each of the 4 memory planes is written with the CPU write data after it has been rotated by the number of counts specified in the Data Rotate Register

Bit	Description
	(GR03). If, however, the bit(s) in the Enable Set/Reset Register (GR01) corresponding to one or more of the memory planes is set to 1, then those memory planes will be written to with the data stored in the corresponding bits in the Set/Reset Register (GR00). 01 = Write Mode 1 - During a CPU write to the frame buffer, the addressed byte in each of the 4 memory planes is written to with the data stored in the memory read latches. (The memory read latches stores an unaltered copy of the data last read from any location in the frame buffer.) 10 = Write Mode 2 - During a CPU write to the frame buffer, the least significant 4 data bits of the CPU write data is treated as the color value for the pixels in the addressed byte in all 4 memory planes. The 8 bits of the Bit Mask Register (GR08) are used to selectively enable or disable the ability to write to the corresponding bit in each of the 4 memory planes that correspond to a given pixel. A setting of 0 in a bit in the Bit Mask Register at a given bit position causes the bits in the corresponding bit positions in the addressed byte in all 4 memory planes to be written with value of their counterparts in the memory read latches. A setting of 1 in a Bit Mask Register at a given bit position causes the bits in the corresponding bit positions in the addressed byte in all 4 memory planes to be written with the 4 bits taken from the CPU write data to thereby cause the pixel corresponding to these bits to be set to the color value. 11 = Write Mode 3 - During a CPU write to the frame buffer, the CPU write data is logically ANDed with the contents of the Bit Mask Register (GR08). The result of this ANDing is treated as the bit mask used in writing the contents of the Set/Reset Register (GR00) are written to addressed byte in all 4 memory planes.

GR06 - Miscellaneous Register

Address: 3CFh (Index=06h)

Default: 0Uh (U=Undefined)

Attributes:
Read/Write

Bit	Description

Bit	Description
7:4	**Reserved.** Read as 0s.
3:2	**Memory Map Mode.** These 2 bits control the mapping of the VGA address range for frame buffer into the CPU address space as follows: 00 = A0000h - BFFFFh 01 = A0000h - AFFFFh 10 = B0000h - B7FFFh 11 = B8000h - BFFFFh This function is used in standard VGA modes, extended VGA modes (132 column text), and in non-VGA modes (hi-res). 132 column text modes are no longer supported. VGA aperture memory accesses are also controlled by the PCI configuration Memory Enable bit and MSR<1>. For accesses using GR10 and GR11 to paged VGA RAM or to device MMIO registers, set these bits to 01 to select the (A0000-AFFFF) range. The CPU must map this memory as uncacheable (UC).
1	**Chain Odd/Even.** This bit provides the ability to alter the interpretation of address bit A0, so that it may be used in selecting between the odd-numbered memory planes (planes 1 and 3) and the even-numbered memory planes (planes 0 and 2). 0 = A0 functions normally. 1 = A0 is switched with a high order address bit, in terms of how it is used in address decoding. The result is that A0 is used to determine which memory plane is being accessed (A0=0 for planes 0 and 2 and A0=1 for planes 1 and 3).
0	**Graphics/Text Mode.** This is one of two bits that are used to determine if the VGA is operating in text or graphics modes. The other bit is in AR10[0], these two bits need to be programmed in a consistent manner to achieve the proper results. 0 = Text mode. 1 = Graphics mode.

GR07 - Color Don't Care Register

Address: 3CFh (Index=07h)

Default: 0Uh (U=Undefined)

Attributes:
Read/Write

Bit	Description
7:4	**Reserved.** Read as 0.
3:0	**Ignore Color Plane [3:0].** These bits have effect only when bit 3 of the Graphics Mode Register (GR05) is set to 1 to select read mode 1. 0 = The corresponding bit in the Color Compare Register (GR02) will not be included in color comparisons. 1 = The corresponding bit in the Color Compare Register (GR02) is used in color comparisons.

GR08 - Bit Mask Register

Address: 3CFh (Index=08h)

Default: Undefined

Attributes:
Read/Write

Bit	Description
7:0	**Bit Mask.** 0 = The corresponding bit in each of the 4 memory planes is written to with the corresponding bit in the memory read latches. 1 = Manipulation of the corresponding bit in each of the 4 memory planes via other mechanisms is enabled. This bit mask applies to any writes to the addressed byte of any or all of the 4 memory planes, simultaneously. This bit mask is applicable to any data written into the frame buffer by the CPU, including data that is also subject to rotation, logical functions (AND, OR, XOR), and Set/Reset. To perform a proper read-modify-write cycle into frame buffer, each byte must first be read from the frame buffer by the CPU (and this will cause it to be stored in the memory read latches), this Bit Mask Register must be set, and the new data then written into the frame buffer by the CPU.

GR10 - Address Mapping

Address: 3CFh (Index=10h)

Default: 00h

Attributes:
Read/Write

This register must only be accessed using I/O operations.

Bit	Description

Bit	Description
7:4	**Page Select Extension - Unused** These bits form the upper bits of a 12-bit page selection value. When combined with the GR11 <7:0> bits they define the offset into stolen memory to the 64KB page that is accessible via the VGA Memory paging mechanism. These bits are ignored.
3	**Reserved**
2:1	**Paging Map Target.** When paging is enabled, these bits determine the target for data cycle accesses through the VGA memory aperture. VGA graphics memory starts from the base of graphics data stolen memory defined in the PCI configuration BDSM register. VGA display uses the first four 64KB pages of VGA graphics memory. 00 = VGA Graphics Memory 01 = Reserved 10 = Reserved 11 = Reserved
0	**Page Mapping Enable.** This mode allows the mapping of the VGA memory address space. Once this is enabled, no VGA memory address swizzle will be performed, addresses are directly mapped to memory. A single paging register is used to map the 64KB [A0000:AFFFF] window. An internal address is generated using GR11 as the address lines extension to the lower address lines of the access A[15:2]. When mapping is enabled, the B0000:BFFFF area must be disabled using GR06<3:2>=01. The use of addresses in the A0000-BFFFF range require that both the graphics device PCI configuration memory enable and MSR<1> be enabled. 0 = Disable (default) 1 = Enable

GR11 - Page Selector

Address: 3CFh (Index=11h)

Default: 00h

Attributes:
Read/Write

This register must only be accessed using I/O operations.

Bit	Description
7	**Reserved**
6:0	**Page Select.** When concatenated with the GR10<7:4> bits, selects a 64KB window within target area when Page Mapping is enabled (GR10[0]=1). This requires that the graphics device PCI configuration space memory enable, the GR06<3:2> bits to be 01 (select A0000-AFFFF only), and the MSR<1:1> bit to be set. This register provides the Address[22:16] bits for the access. VGA paging of frame buffer memory is for non-VGA packed modes only and should not be enabled when using basic VGA modes.

GR18 - Software Flags

Address: 3CFh (Index=18h)

Default: 00h

Attributes:
Read/Write

Bit	Description
7:0	**Software Flags.** Used as scratch pad space in video BIOS. These bits are separate from the bits which appear in the MMIO space. They are used specifically by the SMI BIOS which does not have access to MMIO at the time they are required. These register bits have no effect on H/W operation.

Attribute Controller Registers

Unlike the other sets of indexed registers, the attribute controller registers are not accessed through a scheme employing entirely separate index and data ports. Address 3C0h is used both as the read and write for the index register, and as the write address for the data port. Address 3C1h is the read address for the data port.

To write to the attribute controller registers, the index of the desired register must be written to address 3C0h, and then the data is written to the very same address. A flip-flop alternates with each write to address 3C0h to change its function from writing the index to writing the actual data, and back again. This flip-flop may be deliberately set so that address 3C0h is set to write to the index (which provides a way to set it to a known state) by performing a read operation from Input Status Register 1 (ST01) at address 3BAh or 3DAh, depending on whether the graphics system has been set to emulate an MDA or a CGA as per MSR[0].

To read from the attribute controller registers, the index of the desired register must be written to address 3C0h, and then the data is read from address 3C1h. A read operation from address 3C1h does not reset the flip-flop to writing to the index. Only a write to 3C0h or a read from 3BAh or 3DAh, as described above, will toggle the flip-flop back to writing to the index.

ARX - Attribute Controller Index Register

Address: 3C0h

Default:
00UU UUUUb (U=Undefined)
Attributes:
Read/Write

Bit	Description
7:6	**Reserved.** Read as 0s.
5	**Video Enable.** In the VGA standard, this is called the "Palette Address Source" bit. Clearing this bit will cause the VGA display data to become all 00 index values. For the default palette, this will cause a black screen. The video timing signals continue. Another control bit will turn video off and stop the data fetches. 0 = Disable. Attribute controller color registers (AR[00:0F]) can be accessed by the CPU. 1 = Enable. Attribute controller color registers (AR[00:0F]) are inaccessible by the CPU.
4:0	**Attribute Controller Register Index.** These five bits are used to select any one of the attribute controller registers (AR[00:14]), to be accessed.

AR[00:0F] - Palette Registers [0:F]

Address: Read at 3C1h and Write at 3C0h; (index=00h-0Fh)

Default: 00UU UUUUb (U=Undefined)

Attributes:
Read/Write

Bit	Description
7:6	**Reserved.** Read as 0.
5:0	**Palette Bits P[5:0].** In each of these 16 registers, these are the lower 6 of 8 bits that are used to map either text attributes or pixel color input values (for modes that use 16 colors) to the 256 possible colors available to be selected in the palette. Bits 3 and 2 of the Color Select Register (AR14) supply bits P7 and P6 for the values contained in all 16 of these registers. Bits 1 and 0 of the Color Select Register (AR14) can also replace bits P5 and P4 for the values contained in all 16 of these registers, if bit 7 of the Mode Control Register (AR10) is set to 1.

AR10 - Mode Control Register

Address: Read at 3C1h and Write at 3C0h; (index=10h)

Default: UUh (U=Undefined)

Attributes:
Read/Write

Bit	Description
7	**Palette Bits P5, P4 Select.** **0** = P5 and P4 for each of the 16 selected colors (for modes that use 16 colors) are individually provided by bits 5 and 4 of their corresponding Palette Registers (AR[00:0F]). **1** = P5 and P4 for all 16 of the selected colors (for modes that use 16 colors) are provided by bits 1 and 0 of Color Select Register (AR14).
6	**Pixel Width/Clock Select.** **0** = Six bits of video data (translated from 4 bits via the palette) are output every dot clock. **1** = Two sets of 4 bits of data are assembled to generate 8 bits of video data which is output every other dot clock, and the Palette Registers (AR[00:0F]) are bypassed. This bit is set to 0 for all of the standard VGA modes, except mode 13h.
5	**Pixel Panning Compatibility.** **0** = Scroll both the upper and lower screen regions horizontally as specified in the Pixel Panning Register (AR13). **1** = Scroll only the upper screen region horizontally as specified in the Pixel Panning Register (AR13). This bit has application only when split-screen mode is being used, where the display area is divided into distinct upper and lower regions which function somewhat like separate displays.

Bit	Description
4	**Reserved.** Read as 0.
3	**Enable Blinking/Select Background Intensity.** **0** = Disables blinking in graphics modes, and for text modes, sets bit 7 of the character attribute bytes to control background intensity, instead of blinking. **1** = Enables blinking in graphics modes and for text modes, sets bit 7 of the character attribute bytes to control blinking, instead of background intensity. The blinking rate is derived by dividing the VSYNC signal. The Blink Rate Control field of the VGA control register defines the blinking rate.
2	**Enable Line Graphics Character Code.** **0** = Every 9th pixel of a horizontal line (i.e., the last pixel of each horizontal line of each 9-pixel wide character box) is assigned the same attributes as the background of the character of which the given pixel is a part. **1** = Every 9th pixel of a horizontal line (i.e., the last pixel of each horizontal line of each 9-pixel wide character box) is assigned the same attributes as the 8th pixel if the character of which the given pixel is a part. This setting is intended to accommodate the line-drawing characters of the PC's extended ASCII character set -- characters with an extended ASCII code in the range of B0h to DFh. In some literature describing the VGA standard, the range of extended ASCII codes that are said to include the line-drawing characters is mistakenly specified as C0h to DFh, rather than the correct range of B0h to DFh.
1	**Select Display Type.** **0** = Attribute bytes in text modes are interpreted as they would be for a color display. **1** = Attribute bytes in text modes are interpreted as they would be for a monochrome display.
0	**Graphics/Alphanumeric Mode.** This bit (along with GR06[0]) select either graphics mode or text mode. These two bits must be programmed in a consistent manner to achieve the desired results. **0** = Alphanumeric (text) mode. **1** = Graphics mode.

AR11 - Overscan Color Register

Address: Read at 3C1h and Write at 3C0h; (index=11h)

Default: UUh (U=Undefined)

Attributes:
Read/Write

Bit	Description
7:0	**Overscan.** These 8 bits select the overscan (border) color index value. The actual border color will be determined by the contents of the palette at the selected index. The border color is displayed between the end of active and the beginning of blank or the end of blank and the beginning of active on CRT type devices driven from the DAC output port. For native VGA modes on digital display ports, some devices have the option of including the border in the active region or not, depending on a control bit in the port control register. For centered VGA modes, the VGA control register determines if the border is included in the centered region or not. For monochrome displays, this value should be set to 00h.

AR12 - Memory Plane Enable Register

Address: Read at 3C1h and Write at 3C0h; (index=12h)

Default: 00UU UUUUb (U=Undefined)

Attributes:
Read/Write

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7:6</td><td>Reserved. Read as 0.</td></tr>
<tr><td>5:4</td><td>Video Status Mux. These 2 bits are used to select 2 of the 8 possible palette bits (P7-P0) to be made available to be read via bits 5 and 4 of the Input Status Register 1 (ST01). The table below shows the possible choices.
<table>
<tr><th>Bit [5:4]</th><th>ST01 Bit 5</th><th>ST01 Bit 4</th></tr>
<tr><td>00</td><td>P2 (default)</td><td>P0 (default)</td></tr>
<tr><td>01</td><td>P5</td><td>P4</td></tr>
<tr><td>10</td><td>P3</td><td>P1</td></tr>
<tr><td>11</td><td>P7</td><td>P6</td></tr>
</table>
These bits are typically unused by current software; they are provided for EGA compatibility.</td></tr>
<tr><td>3:0</td><td>Enable Plane [3:0]. These 4 bits individually enable the use of each of the 4 memory planes in providing 1 of the 4 bits used in video output to select 1 of 16 possible</td></tr>
</table>

Bit	Description
	colors from the palette to be displayed. 0 = Disable the use of the corresponding memory plane in video output to select colors, forcing the bit that the corresponding memory plane would have provided to a value of 0. 1 = Enable the use of the corresponding memory plane in video output to select colors. AR12 is referred to in the VGA standard as the Color Plane Enable Register.

AR13 - Horizontal Pixel Panning Register

Address: Read at 3C1h and Write at 3C0h; (index=13h)

Default: 0Uh (U=Undefined)

Attributes:
Read/Write

Bit	Description
7:4	**Reserved.**
3:0	**Horizontal Pixel Shift 3-0.** This field holds a 4-bit value that selects the number of pixels by which the image is shifted horizontally to the left. This function is available in both text and graphics modes and allows for pixel panning. In text modes with a 9-pixel wide character box, the image can be shifted up to 9 pixels to the left. In text modes with an 8-pixel wide character box, and in graphics modes other than those with 256 colors, the image can be shifted up to 8 pixels to the left. A pseudo 9-bit mode is when the 9-dot character is selected but overridden by the VGA control bit. In standard VGA mode 13h (where bit 6 of the Mode Control Register, AR10, is set to 1 to support 256 colors), bit 0 of this register must remain set to 0, and the image may be shifted up to only 4 pixels to the left. In this mode, the number of pixels by which the image is shifted can be further controlled using bits 6 and 5 of the Preset Row Scan Register (CR08).

Number of Pixels Shifted				
Bits [3:0]	**9-dot**	**Pseudo 9-dot**	**8-dot**	**256-Color**
0	1	1	0	0
1	2	2	1	Undefined
2	3	3	2	1
3	4	4	3	Undefined
4	5	5	4	2
5	6	6	5	Undefined
6	7	7	6	3

Bit	Description				
	7	8	7	7	Undefined
	8	0	0	Undefined	Undefined

AR14 - Color Select Register

Address: Read at 3C1h and Write at 3C0h; (index=14h)

Default: 0Uh (U=Undefined)

Attributes:
Read/Write

Bit	Description
7:4	**Reserved.**
3:2	**Palette Bits P[7:6].** These are the 2 upper-most of the 8 bits that are used to map either text attributes or pixel color input values (for modes that use 16 colors) to the 256 possible colors contained in the palette. These 2 bits are common to all 16 sets of bits P5 through P0 that are individually supplied by Palette Registers 0-F (AR[00:0F]).
1:0	**Alternate Palette Bits P[5:4].** These 2 bits can be used as an alternate version of palette bits P5 and P4. Unlike the P5 and P4 bits that are individually supplied by Palette Registers 0-F (AR[00:0F]), these 2 alternate palette bits are common to all 16 of Palette Registers. Bit 7 of the Mode Control Register (AR10) is used to select between the use of either the P5 and P4 bits that are individually supplied by the 16 Palette Registers or these 2 alternate palette bits.

VGA Color Palette Registers

In devices that have multiple display pipes, there is one palette for each display pipe. These palettes are the same for VGA modes and non-VGA modes. Accesses through VGA register methods will read or write from the palette of the pipe selected through MMIO VGA control register.

For each palette, the color data stored in these 256 color data positions can be accessed only through a complex sub-addressing scheme, using a data register and two index registers. The Palette Data Register at address 3C9h is the data port. The Palette Read Index Register at address 3C7h and the Palette Write Index Register at address 3C8h are the two index registers. The Palette Read Index Register is the index register that is used to choose the color data position that is to be read from via the data port, while the Palette Write Index Register is the index register that is used to choose the color data position that is to be written to through the same data port. This arrangement allows the same data port to be used for reading from and writing to two different color data positions. Reading and writing the color data at a color data position involves three successive reads or writes since the color data stored at each color data position consists of three bytes.

To read a palette color data position, the index of the desired color data position must first be written to the Palette Read Index Register. Then all three bytes of data in a given color data position may be read at the Palette Data Register. The first byte read from the Palette Data Register retrieves the 8-bit value specifying the intensity of the red color component. The second and third bytes read are the corresponding 8-bit values for the green and blue color components respectively. After completing the third read operation, the Palette Read Index Register is automatically incremented so that the data of the next color data position becomes accessible for being read. This allows the contents of all of the 256 color data positions of the palette to be read in sequence. This is done by specifying only the index of the 0th color data position in the Palette Read Index Register, and then simply performing 768 successive reads from the Palette Data Register.

Writing a color data position, entails a very similar procedure. The index of the desired color data position must first be written to the Palette Write Index Register. Then all three bytes of data to specify a given color may be written to the Palette Data Register. The first byte written to the Palette Data Register specifies the intensity of the red color component, the second byte specifies the intensity for the green color component, and the third byte specifies the same for the blue color component. One important detail is that all three of these bytes must be written before the hardware will actually update these three values in the given color data position. When all three bytes have been written, the Palette Write Index Register is automatically incremented so that the data of the next color data position becomes accessible for being written. This allows the contents of all of the 256 color data positions of the palette to be written in sequence. This is done by specifying only the index of the 0th color data position in the Palette Write Index Register, and then simply performing 768 successive writes to the Palette Data Register.

DACMASK - Pixel Data Mask Register

Address: 3C6h

Default:
Undefined
Attributes:
Read/Write

Bit	Description
7:0	**Pixel Data Mask.** In indexed-color mode, the 8 bits of this register are logically ANDed with the 8 bits of pixel data received from the frame buffer for each pixel. The result of this ANDing process becomes the actual index used to select color data positions within the palette. This has the effect of limiting the choice of color data positions that may be specified by the incoming 8-bit data. 0 = Corresponding bit in the resulting 8-bit index being forced to 0. 1 = Allows the corresponding bit in the resulting index to reflect the actual value of the corresponding bit in the incoming 8-bit pixel data.

DACSTATE - DAC State Register

Address: 3C7h

Default:
00h
Attributes:
Read Only

Bit	Description
7:2	**Reserved.** Read as 0.
1:0	**DACState.** This field indicates which of the two index registers was most recently written. **Bits [1:0]** Index Register Indicated 00 = Palette Write Index Register at Address 3C7h (default) 01 = Reserved 10 = Reserved 11 = Palette Read Index Register at Address 3C8h

DACRX - Palette Read Index Register

Address: 3C7h

Default:
00h
Attributes:
Write Only

Bit	Description
7:0	**Palette Read Index.** The 8-bit index value programmed into this register chooses which of 256 standard color data positions within the palette are to be made accessible for being read from via the Palette Data Register (DACDATA). The index value held in this register is automatically incremented when all three bytes of the color data position selected by the current index have been read. A write to this register will abort an uncompleted palette write sequence. This register allows access to the palette even when running non-VGA display modes.

DACWX - Palette Write Index Register

Address: 3C8h

Default:
00h
Attributes:
Write Only

Bit	Description
7:0	**Palette Write Index.** The 8-bit index value programmed into this register chooses which of 256 standard color data positions within the palette are to be made accessible for being written via the Palette Data Register (DACDATA). The index value held in this register is automatically incremented when all three bytes of the color data position selected by the current index have been written. This register allows access to the palette even when running non-VGA display modes.

DACDATA - Palette Data Register

Address: 3C9h

Default:
Undefined
Attributes:
Read/Write

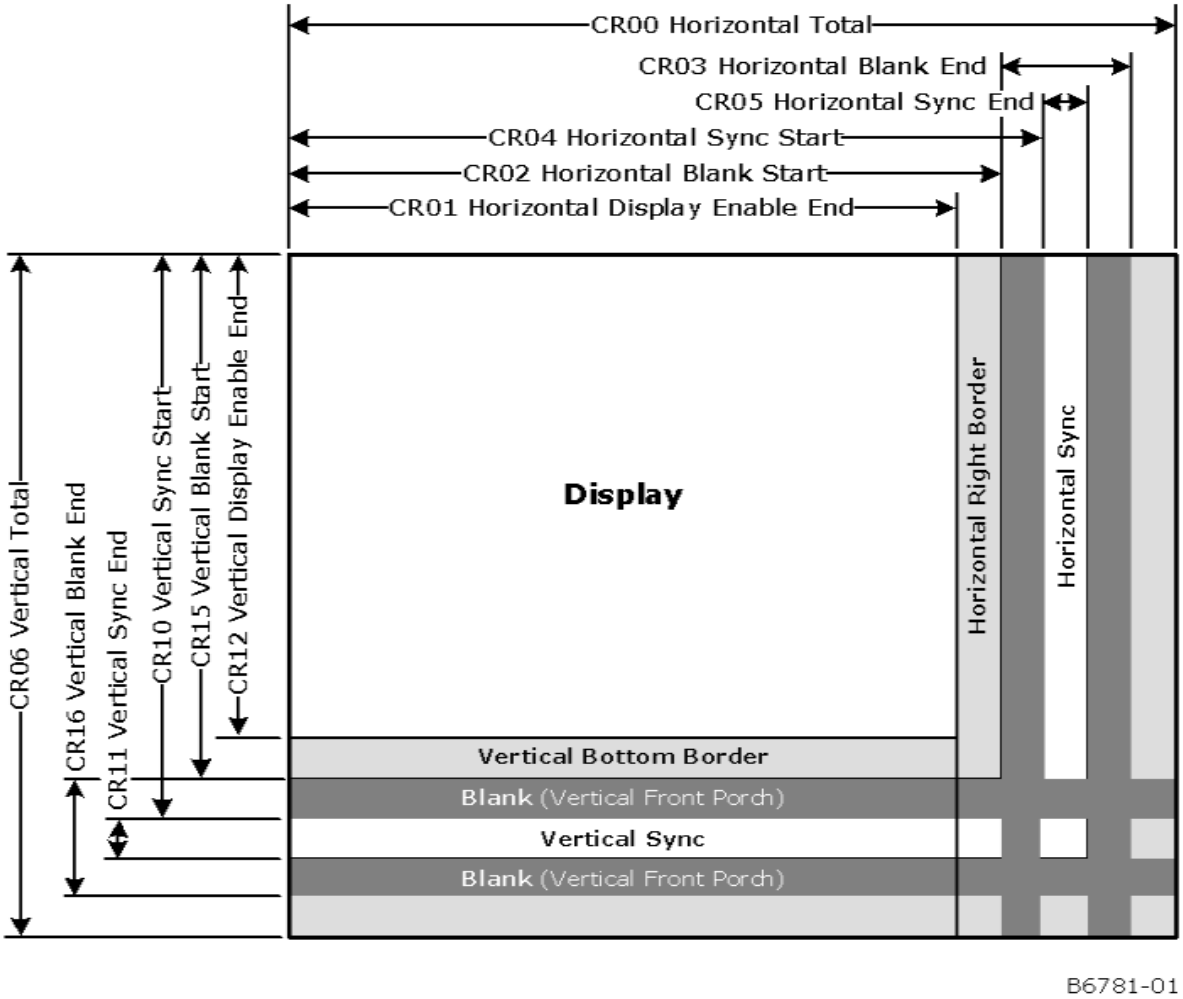
Bit	Description
7:0	**Palette Data.** This byte-wide data port provides read or write access to the three bytes of data of each color data position selected using the Palette Read Index Register (DACRX) or the Palette Write Index Register (DACWX). The three bytes in each color data position are read or written in three successive read or write operations. The first byte read or written specifies the intensity of the red component of the color specified in the selected color data position. The second byte is for the green component, and the third byte is for the blue component. When writing data to a color data position, all three bytes must be written before the hardware will actually update the three bytes of the selected color data position. When reading or writing to a color data position, ensure that neither the Palette Read Index Register (DACRX) or the Palette Write Index Register (DACWX) are written to before all three bytes are read or written. A write to either of these two registers causes the circuitry that automatically cycles through providing access to the bytes for red, green and blue components to be reset such that the byte for the red component is the one that will be accessed by the next read or write operation via this register. This register allows access to the palette even when running non-VGA display modes. Writes to the palette can cause sparkle if not done during inactive video periods. This sparkle is caused by an attempt to write and read the same address on the same cycle. Some devices contain anti-sparkle circuits which will substitute the previous pixel value for the read output.

CRT Controller Register

For native VGA modes, the CRTC registers determine the display timing that is to be used. In centered VGA modes, these registers determine the size of the VGA image that is to be centered in the larger timing generator defined rectangle.

The CRT controller registers are accessed by writing the index of the desired register into the CRT Controller Index Register at address 3B4h or 3D4h, depending on whether the graphics system is configured for MDA or CGA emulation. The desired register is then accessed through the data port for the CRT controller registers located at address 3B5h or 3D5h, again depending upon the choice of MDA or CGA emulation as per MSR[0].

The following figure shows display fields and dimensions and the particular CRxx register that provides the control.



Group 0 Protection: In the original VGA, CR[0:7] could be made write-protected by CR11[7]. In BIOS code, this write protection is set following each mode change. Other protection groups have no current use, and would not be used going forward by the BIOS or by drivers. They are the result of an industry fad some years ago to attempt to write protect other groups of registers; however, all such schemes were chip specific. Only the write protection (Group 0 Protection) is supported.

CRX - CRT Controller Index Register

Address: 3B4h/3D4h

Default:
0Uh (U=Undefined)
Attributes:
Read/Write

Bit	Description
7	**Reserved.** Read as 0.
6:0	**CRT Controller Index.** These 7 bits are used to select any one of the CRT controller registers to be accessed via the data port at location 3B5h or 3D5h, depending upon whether the graphics system is configured for MDA or CGA emulation.

CR00 - Horizontal Total Register

Address: 3B5h/3D5h (index=00h)

Default: 00h

Attributes:
Read/Write (Group 0 Protection)

Bit	Description
7:0	**Horizontal Total.** This register is used to specify the total length of each scan line. This encompasses both the part of the scan line that is within the active display area and the part that is outside of it. Programming this register to a zero has the effect of stopping the fetching of display data. This field should be programmed with a value equal to the total number of character clocks within the entire length of a scan line, minus 5.

CR01 - Horizontal Display Enable End Register

Address: 3B5h/3D5h (index=01h)

Default: Undefined

Attributes:
Read/Write (Group 0 Protection)

Bit	Description
7:0	**Horizontal Display Enable End.** This register is used to specify the end of the part of the scan line that is within the active display area relative to its beginning. In other words, this is the horizontal width of the active display area. This field should be programmed with a value equal to the number of character clocks that occur within the horizontal active display area, minus 1. Horizontal display

Bit	Description
	enable will go active at the beginning of each line during vertical active area, it will go inactive based on the programming of this register or the programming of the horizontal total (CR00) register. When this register value is programmed to a number that is larger than the total number of characters on a line, display enable will be active for all but the last character of the horizontal display line.

CR02 - Horizontal Blanking Start Register

Address: 3B5h/3D5h (index=02h)

Default: Undefined

Attributes:
Read/Write (Group 0 Protection)

Bit	Description
7:0	**Horizontal Blanking Start.** This register is used to specify the beginning of the horizontal blanking period relative to the beginning of the active display area of a scan line. Horizontal blanking should always be set to start no sooner than after the end of horizontal active. This field should be programmed with a value equal to the number of character clocks that occur on a scan line from the beginning of the active display area to the beginning of the horizontal blanking.

CR03 - Horizontal Blanking End Register

Address: 3B5h/3D5h (index=03h)

Default: 1UUU UUUUb (U=Undefined)

Attributes:
Read/Write (Group 0 Protection)

Bit	Description
7	**Reserved.** Values written to this bit are ignored, and to maintain consistency with the VGA standard, a value of 1 is returned when this bit is read. At one time, this bit was used to enable access to certain light pen registers. At that time, setting this bit to 0 provided this access, but setting this bit to 1 was necessary for normal operation.
6:5	**Display Enable Skew Control.** Defines the degree to which the start and end of the active display area are delayed along the length of a scan line to compensate for internal pipeline delays. These 2 bits describe the delay in terms of a number character clocks. **Bit [6:5]** Amount of Delay 00 = no delay

Bit	Description
	01 = delayed by 1 character clock 10 = delayed by 2 character clocks 11 = delayed by 3 character clocks
4:0	**Horizontal Blanking End Bits [4:0].** This field provides the 5 least significant bits of a 6-bit value that specifies the end of the blanking period relative to its beginning on a single scan line. Bit 7 of the Horizontal Sync End Register (CR05) supplies the most significant bit. This 6-bit value should be programmed to be equal to the least significant 6 bits of the result of adding the length of the blanking period in terms of character clocks to the value specified in the Horizontal Blanking Start Register (CR02). End of blanking should occur before horizontal total.

CR04 - Horizontal Sync Start Register

Address: 3B5h/3D5h (index=04h)

Default: Undefined

Attributes:
Read/Write (Group 0 Protection)

Bit	Description
7:0	**Horizontal Sync Start** This register is used to specify the position of the beginning of the horizontal sync pulse relative to the start of the active display area on a scan line. This field should be set equal to the number of character clocks that occur from beginning of the active display area to the beginning of the horizontal sync pulse on a single scan line. Horizontal sync should always occur at least 2 clocks after the start of horizontal blank and 2 clocks before the end of horizontal blank. The actual start of sync will also be affected by both the horizontal sync skew register field and whether it is a text or graphics mode.

CR05 - Horizontal Sync End Register

Address: 3B5h/3D5h (index=05h)

Default: 00h

Attributes:
Read/Write (Group 0 Protection)

Bit	Description
7	**Horizontal Blanking End Bit 5.** This bit provides the most significant bit of a 6-bit value that specifies the end of the horizontal blanking period relative to its beginning. Bits [4:0] of Horizontal Blanking End Register (CR03) supplies the 5 least significant bits.

Bit	Description
	See CR03[4:0] for further details. This 6-bit value should be set to the least significant 6 bits of the result of adding the length of the blanking period in terms of character clocks to the value specified in the Horizontal Blanking Start Register (CR02).
6:5	**Horizontal Sync Delay.** This field defines the degree to which the start and end of the horizontal sync pulse are delayed to compensate for internal pipeline delays. This capability is supplied to implement VGA compatibility. These field describes the delay in terms of a number character clocks. **Bit [6:5] — Amount of Delay** 00 — no delay 01 — delayed by 1 character clock 10 — delayed by 2 character clocks 11 — delayed by 3 character clocks
4:0	**Horizontal Sync End.** This field provides the 5 least significant bits of a 5-bit value that specifies the end of the horizontal sync pulse relative to its beginning. A value equal to the 5 least significant bits of the horizontal character counter value at which time the horizontal retrace signal becomes inactive (logical 0). Thus, this 5-bit value specifies the width of the horizontal sync pulse. To obtain a retrace signal of W, the following algorithm is used: Value of Horizontal Sync start Register (CR04) + width of horizontal retrace signal in character clock units = 5 bit result to be programmed in this field

CR06 - Vertical Total Register

Address: 3B5h/3D5h (index=06h)

Default: 00h

Attributes:
Read/Write (Group 0 Protection)

Bit	Description
7:0	**Vertical Total Bits [7:0].** This field provides the 8 least significant bits of either a 10-bit or 12-bit value that specifies the total number of scan lines. This includes the scan lines both inside and outside of the active display area. In standard VGA modes, the vertical total is specified with a 10-bit value. The 8 least significant bits of this value are supplied by these 8 bits of this register, and the 2 most significant bits are supplied by bits 5 and 0 of the Overflow Register (CR07).

CR07 - Overflow Register (Vertical)

Address: 3B5h/3D5h (index=07h)

Default: UU0U UUU0b (U=Undefined)

Attributes:

Read/Write (Group 0 Protection on bits [7:5, 3:0])

Bit	Description
7	**Vertical Sync Start Bit 9.** The vertical sync start is a 10-bit that specifies the beginning of the vertical sync pulse relative to the beginning of the active display area. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Sync Start Register (CR10), and the most and second-most significant bits are supplied by this bit and bit 2, respectively, of this register. This 10-bit value should be programmed to be equal to the number of scan lines from the beginning of the active display area to the start of the vertical sync pulse. Since the active display area always starts on the 0th scan line, this number should be equal to the number of the scan line on which the vertical sync pulse begins.
6	**Vertical Display Enable End Bit 9.** The vertical display enable end is a 10-bit that specifies the number of the last scan line within the active display area. In standard VGA modes, the vertical display enable end is specified with a 10-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Display Enable End Register (CR12), and the most and second-most significant bits are supplied by this bit and bit 1, respectively, of this register. This 10-bit value should be programmed to be equal to the number of the last scan line within in the active display area. Since the active display area always starts on the 0th scan line, this number should be equal to the total number of scan lines within the active display area, minus 1.
5	**Vertical Total Bit 9.** The vertical total is a 10-bit value that specifies the total number of scan lines. This includes the scan lines both inside and outside of the active display area. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Total Register (CR06), and the most and second-most significant bits are supplied by this bit and bit 0, respectively, of this register. This 10-bit value should be programmed equal to the total number of scan lines, minus 2.
4	**Line Compare Bit 8.** This bit provides the second most significant bit of a 10-bit value that specifies the scan line at which the memory address counter restarts at the value of 0. Bit 6 of the Maximum Scan Line Register (CR09) supplies the most significant bit, and bits 7-0 of the Line Compare Register (CR18) supply the 8 least significant bits. Normally, this 10-bit value is set to specify a scan line after the last scan line of the active display area. When this 10-bit value is set to specify a scan line within the active display area, it causes that scan line and all subsequent scan lines in the active display area to display video data starting at the very first byte of the frame buffer. The result is what appears to be a screen split into a top and

Bit	Description
	bottom part, with the image in the top part being repeated in the bottom part. When used in cooperation with the Start Address High Register (CR0C) and the Start Address Low Register (CR0D), it is possible to create a split display, as described earlier, but with the top and bottom parts displaying different data. The top part will display what data exists in the frame buffer starting at the address specified in the two aforementioned start address registers, while the bottom part will display what data exists in the frame buffer starting at the first byte of the frame buffer.
3	**Vertical Blanking Start Bit 8.** The vertical blanking start is a 10-bit that specifies the beginning of the vertical blanking period relative to the beginning of the active display area. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Blanking Start Register (CR15), and the most and second-most significant bits are supplied by bit 5 of the Maximum Scan Line Register (CR09) and this bit of this register, respectively. This 10-bit value should be programmed to be equal to the number of scan lines from the beginning of the active display area to the beginning of the blanking period. Since the active display area always starts on the 0th scan line, this number should be equal to the number of the scan line on which the vertical blanking period begins.
2	**Vertical Sync Start Bit 8.** The vertical sync start is a 10-bit value that specifies the beginning of the vertical sync pulse relative to the beginning of the active display area. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Sync Start Register (CR10), and the most and second-most significant bits are supplied by bit 7 and this bit, respectively, of this register. This 10-bit value should be programmed to be equal to the number of scan lines from the beginning of the active display area to the start of the vertical sync pulse. Since the active display area always starts on the 0th scan line, this number should be equal to the number of the scan line on which the vertical sync pulse begins.
1	**Vertical Display Enable End Bit 8.** The vertical display enable end is a 10-bit value that specifies the number of the last scan line within the active display area. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Display Enable End Register (CR12), and the two most significant bits are supplied by bit 6 and this bit, respectively, of this register. This 10-bit or value should be programmed to be equal to the number of the last scan line within in the active display area. Since the active display area always starts on the 0th scan line, this number should be equal to the total number of scan lines within the active display area, minus 1.
0	**Vertical Total Bit 8.** The vertical total is a 10-bit value that specifies the total number of scan lines. This includes the scan lines both inside and outside of the active display area. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Total Register (CR06), and the most and second-most significant bits

Bit	Description
	are supplied by bit 5 and this bit, respectively, of this register. This 10-bit value should be programmed to be equal to the total number of scan lines, minus 2.

CR08 - Preset Row Scan Register

Address: 3B5h/3D5h (index=08h)

Default: 0UUU UUUUb (U=Undefined)

Attributes:
Read/Write

Bit	Description
7	**Reserved.** Read as 0s.
6:5	**Byte Panning.** This field holds a 2-bit value that selects number of bytes (up to 3) by which the image is shifted horizontally to the left on the screen. This function is available in both text and graphics modes. In text modes with a 9-pixel wide character box, the image can be shifted up to 27 pixels to the left, in increments of 9 pixels. In text modes with an 8-pixel wide character box, and in all standard VGA graphics modes, the image can be shifted up to 24 pixels to the left, in increments of 8 pixels. When the Nine dot disable bit of the VGA control register is set, the pixel shift will be equivalent to the 8-dot mode. The image can be shifted still further, in increments of individual pixels, through the use of bits [3:0] of the Horizontal Pixel Panning Register (AR13).
4:0	**Starting Row Scan Count.** This field specifies which horizontal line of pixels within the character boxes of the characters used on the top-most row of text on the display will be used as the top-most scan line. The horizontal lines of pixels of a character box are numbered from top to bottom, with the top-most line of pixels being number 0. If a horizontal line of the these character boxes other than the top-most line is specified, then the horizontal lines of the character box above the specified line of the character box will not be displayed as part of the top-most row of text characters on the display. Normally, the value specified by these 5 bits should be 0, so that all of the horizontal lines of

Number of Pixels Shifted		
Bit [6:5]	**9-Pixel Text**	**8-Pixel Text & Graphics**
00	0	0
01	9	8
10	18	16
11	27	24

Bit	Description
	pixels within these character boxes will be displayed in the top-most row of text, ensuring that the characters in the top-most row of text do not look as though they have been cut off at the top.

CR09 - Maximum Scan Line Register

Address: 3B5h/3D5h (index=09h)

Default: 00h

Attributes:
Read/Write

Bit	Description
7	**Double Scanning Enable.** 0 = Disable. When disabled, the clock to the row scan counter is equal to the horizontal scan rate. This is the normal setting for many of the standard VGA modes. 1 = Enable. When enabled, the clock to the row scan counter is divided by 2. This is normally used to allow CGA-compatible modes that have only 200 scan lines of active video data to be displayed as 400 scan lines (each scan line is displayed twice).
6	**Line Compare Bit 9.** This bit provides the most significant bit of a 10-bit value that specifies the scan line at which the memory address counter restarts at the value of 0. Bit 4 of the Overflow Register (CR07) supplies the second most significant bit, and bits 7-0 of the Line Compare Register (CR18) supply the 8 least significant bits. Normally, this 10-bit value is set to specify a scan line after the last scan line of the active display area. When this 10-bit value is set to specify a scan line within the active display area, it causes that scan line and all subsequent scan lines in the active display area to display video data starting at the very first byte of the frame buffer. The result is what appears to be a screen split into a top and bottom part, with the image in the top part being repeated in the bottom part. When used in cooperation with the Start Address High Register (CR0C) and the Start Address Low Register (CR0D), it is possible to create a split display, as described earlier, but with the top and bottom parts displaying different data. The top part will display whatever data exists in the frame buffer starting at the address specified in the two aforementioned start address registers, while the bottom part will display whatever data exists in the frame buffer starting at the first byte of the frame buffer.
5	**Vertical Blanking Start Bit 9.** The vertical blanking start is a 10-bit value that specifies the beginning of the vertical blanking period relative to the beginning of the active display area. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Blanking Start Register (CR15), and the most and second-most significant bits are supplied by this bit and bit 3 of the Overflow Register (CR07), respectively. This 10-bit value should be programmed to be equal to the number of scan line from

Bit	Description
	the beginning of the active display area to the beginning of the blanking period. Since the active display area always starts on the 0th scan line, this number should be equal to the number of the scan line on which the vertical blanking period begins.
4:0	**Starting Row Scan Count.** This field provides all 5 bits of a 5-bit value that specifies the number of scan lines in a horizontal row of text. This value should be programmed to be equal to the number of scan lines in a horizontal row of text, minus 1.

CR0A - Text Cursor Start Register

Address: 3B5h/3D5h (index=0Ah)

Default: 00UU UUUUb (U=Undefined)

Attributes:
Read/Write

Bit	Description
7:6	**Reserved.** Read as 0.
5	**Text Cursor Off.** This text cursor exists only in text modes, so this register is entirely ignored in graphics modes. 0 = Enables the text cursor. 1 = Disables the text cursor.
4:0	**Text Cursor Start.** This field specifies which horizontal line of pixels in a character box is to be used to display the first horizontal line of the cursor in text mode. The horizontal lines of pixels in a character box are numbered from top to bottom, with the top-most line being number 0. The value specified by these 5 bits should be the number of the first horizontal line of pixels on which the cursor is to be shown.

CR0B - Text Cursor End Register

Address: 3B5h/3D5h (index=0Bh)

Default: 0UUU UUUUb (U=Undefined)

Attributes:
Read/Write

Bit	Description
7	**Reserved.** Read as 0.
6:5	**Text Cursor Skew.** This field specifies the degree to which the start and end of each horizontal line of pixels making up the cursor is delayed to compensate for internal

Bit	Description
	pipeline delays. These 2 bits describe the delay in terms of a number character clocks. **Bit [6:5]** Amount of Delay 00 = No delay 01 = Delayed by 1 character clock 10 = Delayed by 2 character clocks 11 = Delayed by 3 character clocks
4:0	**Text Cursor End.** This field specifies which horizontal line of pixels in a character box is to be used to display the last horizontal line of the cursor in text mode. The horizontal lines of pixels in a character box are numbered from top to bottom, with the top-most line being number 0. The value specified by these 5 bits should be the number of the last horizontal line of pixels on which the cursor is to be shown.

CR0C - Start Address High Register

Address: 3B5h/3D5h (index=0Ch)

Default: Undefined

Attributes:
Read/Write

Bit	Description
7:0	**Start Address Bits [15:8].**This register provides either bits 15 through 8 of a 16-bit value that specifies the memory address offset from the beginning of the frame buffer at which the data to be shown in the active display area begins. (default is 0) In standard VGA modes, the start address is specified with a 16-bit value. The eight bits of this register provide the eight most significant bits of this value, while the eight bits of the Start Address Low Register (CR0D) provide the eight least significant bits.

CR0D - Start Address Low Register

Address: 3B5h/3D5h (index=0Dh)

Default: Undefined

Attributes:
Read/Write

Bit	Description
7:0	**Start Address Bits [7:0]** This register provides either bits 7 through 0 of a 16 bit value that specifies the memory address offset from the beginning of the frame buffer at

Bit	Description
	which the data to be shown in the active display area begins. (default is 0) In standard VGA modes the start address is specified with a 16-bit value. The eight bits of the Start Address High Register (CR0C) provide the eight most significant bits of this value, while the eight bits of this register provide the eight least significant bits.

CR0E - Text Cursor Location High Register

Address: 3B5h/3D5h (index=0Eh)

Default: Undefined

Attributes:
Read/Write

Bit	Description
7:0	**Text Cursor Location Bits [15:8].** This field provides the 8 most significant bits of a 16-bit value that specifies the address offset from the beginning of the frame buffer at which the text cursor is located. Bit 7:0 of the Text Cursor Location Low Register (CR0F) provide the 8 least significant bits.

CR0F - Text Cursor Location Low Register

Address: 3B5h/3D5h (index=0Fh)

Default: Undefined

Attributes:
Read/Write

Bit	Description
7:0	**Text Cursor Location Bits [7:0].** This field provides the 8 least significant bits of a 16-bit value that specifies the address offset from the beginning of the frame buffer at which the text cursor is located. Bits 7:0 of the Text Cursor Location High Register (CR0E) provide the 8 most significant bits.

CR10 - Vertical Sync Start Register

Address: 3B5h/3D5h (index=10h)

Default: Undefined

Attributes:
Read/Write

Bit	Description
7:0	**Vertical Sync Start Bits [7:0].** This register provides the 8 least significant bits of a 10-bit that specifies the beginning of the vertical sync pulse relative to the

Bit	Description
	beginning of the active display area of a screen. In standard VGA modes, this value is described in 10 bits with bits [7,2] of the Overflow Register (CR07) supplying the 2 most significant bits. This 10-bit value should equal the vertical sync start in terms of the number of scan lines from the beginning of the active display area to the beginning of the vertical sync pulse. Since the active display area always starts on the 0th scan line, this number should be equal to the number of the scan line on which the vertical sync pulse begins.

CR11 - Vertical Sync End Register

Address: 3B5h/3D5h (index=11h)

Default: 0U00 UUUUb (U=Undefined)

Attributes:
Read/Write

Bit	Description
7	**Protect Registers [0:7].** The ability to write to Bit 4 of the Overflow Register (CR07) is not affected by this bit (i.e., bit 4 of the Overflow Register is always writeable). 0 = Enable writes to registers CR[00:07]. (default) 1 = Disable writes to registers CR[00:07].
6	**Reserved.** In the VGA standard, this bit was used to switch between 3 and 5 frame buffer refresh cycles during the time required to draw each horizontal line.
5	**Vertical Interrupt Enable.** This bit is reserved for compatibility only. While this bit may be written or read, it's value will have no effect. VGA does not provide an interrupt signal which would be connected to an input of the system's interrupt controller. Bit 7 of Input Status Register 0 (ST00) originally indicated the status of the vertical retrace interrupt. 0 = Enable the generation of an interrupt at the beginning of each vertical retrace period. 1 = Disable the generation of an interrupt at the beginning of each vertical retrace period.
4	**Vertical Interrupt Clear.** This is reserved for compatibility only. VGA does not provide an interrupt signal which would be connected to an input of the system's interrupt controller. 0 = Setting this bit to 0 clears a pending vertical retrace interrupt. This bit must be set back to 1 to enable the generation of another vertical retrace interrupt.

Bit	Description
3:0	**Vertical Sync End.** This 4-bit field provides a 4-bit value that specifies the end of the vertical sync pulse relative to its beginning. This 4-bit value should be set to the least significant 4 bits of the result of adding the length of the vertical sync pulse in terms of the number of scan lines that occur within the length of the vertical sync pulse to the value that specifies the beginning of the vertical sync pulse (see the description of the Vertical Sync Start Register for more details).

CR12 - Vertical Display Enable End Register

Address: 3B5h/3D5h (index=12h)

Default: Undefined

Attributes:
Read/Write

Bit	Description
7:0	**Vertical Display Enable End Bits [7:0].** This register provides the 8 least significant bits of a 10-bit value that specifies the number of the last scan line within the active display area. In standard VGA modes, this value is described in 10 bits with bits [6,1] of the Overflow Register (CR07) supplying the two most significant bits. This 10-bit value should be programmed to be equal to the number of the last scan line within in the active display area. Since the active display area always starts on the 0th scan line, this number should be equal to the total number of scan lines within the active display area,minus 1.

CR13 - Offset Register

Address: 3B5h/3D5h (index=13h)

Default: Undefined

Attributes:
Read/Write

Bit	Description
7:0	**Offset Bits [7:0].** This register provides either all 8 bits of an 8-bit value that specifies the number of words or DWords of frame buffer memory occupied by each horizontal row of characters. Whether this value is interpreted as the number of words or DWords is determined by the settings of the bits in the Clocking Mode Register (SR01). In standard VGA modes, the offset is described with an 8-bit value, all the bits of which are provided by this register. This 8-bit value should be programmed to be equal to either the number of words or DWords (depending on the setting of the bits in the Clocking Mode Register, SR01) of frame buffer memory that is occupied by each horizontal row of characters.

CR14 - Underline Location Register

Address: 3B5h/3D5h (index=14h)

Default: 0UUU UUUUb (U=Undefined)

Attributes:
Read/Write

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7</td><td>Reserved. Read as 0.</td></tr>
<tr><td>6</td><td>DWord Mode.
0 = Frame buffer addresses are interpreted by the frame buffer address decoder as being either byte addresses or word addresses, depending on the setting of bit 6 of the CRT Mode Control Register (CR17).
1 = Frame buffer addresses are interpreted by the frame buffer address decoder as being DWord addresses, regardless of the setting of bit 6 of the CRT Mode Control Register (CR17).
This bit is used in conjunction with bits 6 and 5 of the CRT Mode Control Register (CR17) to select how frame buffer addresses from the CPU are interpreted by the frame buffer address decoder as shown below:
<table>
<tr><th>CR14[6]</th><th>CR17[6]</th><th>Addressing Mode</th></tr>
<tr><td>0</td><td>0</td><td>Word Mode</td></tr>
<tr><td>0</td><td>1</td><td>Byte Mode</td></tr>
<tr><td>1</td><td>0</td><td>DWord Mode</td></tr>
<tr><td>1</td><td>1</td><td>DWord Mode</td></tr>
</table></td></tr>
<tr><td>5</td><td>Count By 4.
0 = The memory address counter is incremented either every character clock or every other character clock, depending upon the setting of bit 3 of the CRT Mode Control Register.
1 = The memory address counter is incremented either every 4 character clocks or every 2 character clocks, depending upon the setting of bit 3 of the CRT Mode Control Register. . This is used in mode x13 to allow for using all four planes.
This bit is used in conjunction with bit 3 of the CRT Mode Control Register (CR17) to select the number of character clocks are required to cause the memory address counter to be incremented as shown, below:
<table>
<tr><th>CR14[5]</th><th>CR17[3]</th><th>Addressing Incrementing Interval</th></tr>
<tr><td>0</td><td>0</td><td>every character clock</td></tr>
<tr><td>0</td><td>1</td><td>every 2 character clocks</td></tr>
<tr><td>1</td><td>0</td><td>every 4 character clocks</td></tr>
<tr><td>1</td><td>1</td><td>every 2 character clocks</td></tr>
</table></td></tr>
<tr><td>4:0</td><td>Underline Location. This field specifies which horizontal line of pixels in a character box is to be used to display a character underline in text mode. The horizontal lines of pixels in a character box are numbered</td></tr>
</table>

Bit	Description
	from top to bottom, with the top-most line being number 0. The value specified by these 5 bits should be the number of the horizontal line on which the character underline mark is to be shown.

CR15 - Vertical Blanking Start Register

Address: 3B5h/3D5h (index=15h)

Default: Undefined

Attributes:
Read/Write

Bit	Description
7:0	**Vertical Blanking Start Bits [7:0].** This register provides the 8 least significant bits of a 10-bit value that specifies the beginning of the vertical blanking period relative to the beginning of the active display area of the screen. In standard VGA modes, the vertical blanking start is specified with a 10-bit value. The most and second-most significant bits of this value are supplied by bit 5 of the Maximum Scan Line Register (CR09) and bit 3 of the Overflow Register (CR07), respectively. This 10-bit value should be programmed to be equal the number of scan lines from the beginning of the active display area to the beginning of the vertical blanking period. Since the active display area always starts on the 0th scan line, this number should be equal to the number of the scan line on which vertical blanking begins.

CR16 - Vertical Blanking End Register

Address: 3B5h/3D5h (index=16h)

Default: Undefined

Attributes:
Read/Write

This register provides a 8-bit value that specifies the end of the vertical blanking period relative to its beginning.

Bit	Description
7:0	**Vertical Blanking End Bits [7:0].** This 8-bit value should be set equal to the least significant 8 bits of the result of adding the length of the vertical blanking period in terms of the number of scan lines that occur within the length of the vertical blanking period to the value that specifies the beginning of the vertical blanking period (see the description of the Vertical Blanking Start Register for details).

CR17 - CRT Mode Control

Address: 3B5h/3D5h (index=17h)

Default: 0UU0 UUUUb (U=Undefined)

Attributes:
Read/Write

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7</td><td>CRT Controller Reset. This bit has no effect except in native VGA modes (non-centered).
0 = Forces horizontal and vertical sync signals to be inactive. No other registers or outputs are affected.
1 = Permits normal operation.</td></tr>
<tr><td>6</td><td>Word Mode or Byte Mode.
0 = The memory address counter's output bits are shifted by 1 bit position before being passed on to the frame buffer address decoder such that they are made into word-aligned addresses when bit 6 of the Underline Location Register (CR17) is set to 0.
1 = The memory address counter's output bits remain unshifted before being passed on to the frame buffer address decoder such that they remain byte-aligned addresses when bit 6 of the Underline Location Register (CR17) is set to 0.
This bit is used in conjunction with bits 6 and 5 of the CRT Mode Control Register (CR17) to control how frame buffer addresses from the memory address counter are interpreted by the frame buffer address decoder as shown below:
<table>
<tr><th>CR14[6]</th><th>CR17[6]</th><th>Address Mode</th></tr>
<tr><td>0</td><td>0</td><td>Word Mode - Addresses from the memory address counter are shifted once to become word-aligned</td></tr>
<tr><td>0</td><td>1</td><td>Byte Mode - Addresses from the memory address counter are not shifted</td></tr>
<tr><td>1</td><td>0</td><td>DWord Mode - Addresses from the memory address counter are shifted twice to become DWord-aligned</td></tr>
<tr><td>1</td><td>1</td><td>DWord Mode - Addresses from the memory address counter are shifted twice to become DWord-aligned</td></tr>
</table></td></tr>
<tr><td>5</td><td>Address Wrap. This bit is only effective when word mode is made active by setting bit 6 in both the Underline Location Register and this register to 0.
0 = Wrap frame buffer address at 16 KB. This is used in CGA-compatible modes.
1 = No wrapping of frame buffer addresses.</td></tr>
</table>

Bit	Description
4	**Reserved.** Read as 0.
3	**Count By 2.** This bit is used in conjunction with bit 5 of the Underline Location Register (CR14) to select the number of character clocks are required to cause the memory address counter to be incremented. 0 = The memory address counter is incremented either every character clock or every 4 character clocks, depending upon the setting of bit 5 of the Underline Location Register. 1 = The memory address counter is incremented either every other clock. CR14[5] CR17[3] Address Incrementing interval 0 0 every character clock 0 1 every 2 character clocks 1 0 every 4 character clocks 1 1 every 2 character clocks
2	**Horizontal Retrace Select.** This bit provides a way of effectively doubling the vertical resolution by allowing the vertical timing counter to be clocked by the horizontal retrace clock divided by 2 (usually, it would be undivided). 0 = The vertical timing counter is clocked by the horizontal retrace clock. 1 = The vertical timing counter is clocked by the horizontal retrace clock divided by 2.
1	**Select Row Scan Counter.** 0 = A substitution takes place, where bit 14 of the 16-bit memory address generated of the memory address counter (after the stage at which these 16 bits may have already been shifted to accommodate word or DWord addressing) is replaced with bit 1 of the row scan counter at a stage just before this address is presented to the frame buffer address decoder. 1 = No substitution takes place. See following tables.
0	**Compatibility Mode Support.** 0 = A substitution takes place, where bit 13 of the 16-bit memory address generated of the memory address counter (after the stage at which these 16 bits may have already been shifted to accommodate word or DWord addressing) is replaced with bit 0 of the row scan counter at a stage just before this address is presented to the frame buffer address decoder. 1 = No substitution takes place. See following tables.

The following tables show the possible ways in which the address bits from the memory address counter can be shifted and/or reorganized before being presented to the frame buffer address decoder. First, the address bits generated by the memory address counter are reorganized, if need be, to accommodate

byte, word or DWord modes. The resulting reorganized outputs (MAOut15-MAOut0) from the memory address counter may also be further manipulated with the substitution of bits from the row scan counter (RSOut1 and RSOut0) before finally being presented to the input bits of the frame buffer address decoder (FBIn15-FBIn0).

Memory Address Counter Address Bits [15:0]

	Byte Mode CR14 bit 6=0 CR17 bit 6=1 CR17 bit 5=X	Word Mode CR14 bit 6=0 CR17 bit 6=0 CR17 bit 5=1	Word Mode CR14 bit 6=0 CR17 bit 6=0 CR17 bit 5=0	DWord Mode CR14 bit 6=1 CR17 bit 6=X CR17 bit 5=X
MAOut0	0	15	13	12
MAOut1	1	0	0	13
MAOut2	2	1	1	0
MAOut3	3	2	2	1
MAOut4	4	3	3	2
MAOut5	5	4	4	3
MAOut6	6	5	5	4
MAOut7	7	6	6	5
MAOut8	8	7	7	6
MAOut9	9	8	8	7
MAOut10	10	9	9	8
MAOut11	11	10	10	9
MAOut12	12	11	11	10
MAOut13	13	12	12	11
MAOut14	14	13	13	12
MAOut15	15	14	14	13

X = Don't Care

Frame Buffer Address Decoder

	CR17 bit 1=1	CR17 bit 1=1	CR17 bit 1=0	CR17 bit 1=0
	CR17 bit 0=1	CR17 bit 0=0	CR17 bit 0=1	CR17 bit 0=0
FBIn0	MAOut0	MAOut0	MAOut0	MAOut0
FBIn1	MAOut1	MAOut1	MAOut1	MAOut1
FBIn2	MAOut2	MAOut2	MAOut2	MAOut2

	CR17 bit 1=1 CR17 bit 0=1	CR17 bit 1=1 CR17 bit 0=0	CR17 bit 1=0 CR17 bit 0=1	CR17 bit 1=0 CR17 bit 0=0
FBIn3	MAOut3	MAOut3	MAOut3	MAOut3
FBIn4	MAOut4	MAOut4	MAOut4	MAOut4
FBIn5	MAOut5	MAOut5	MAOut5	MAOut5
FBIn6	MAOut6	MAOut6	MAOut6	MAOut6
FBIn7	MAOut7	MAOut7	MAOut7	MAOut7
FBIn8	MAOut8	MAOut8	MAOut8	MAOut8
FBIn9	MAOut9	MAOut9	MAOut9	MAOut9
FBIn10	MAOut10	MAOut10	MAOut10	MAOut10
FBIn11	MAOut11	MAOut11	MAOut11	MAOut11
FBIn12	MAOut12	MAOut12	MAOut12	MAOut12
FBIn13	MAOut13	MAOut13	RSOut0	RSOut0
FBIn14	MAOut14	RSOut1	MAOut14	RSOut1
FBIn15	MAOut15	MAOut15	MAOut15	MAOut15

CR18 - Line Compare Register

Address: 3B5h/3D5h (index=18h)

Default: Undefined

Attributes:
Read/Write

Bit	Description
7:0	**Line Compare Bits [7:0].** This register provides the 8 least significant bits of a 10-bit value that specifies the scan line at which the memory address counter restarts at the value of 0. Bit 6 of the Maximum Scan Line Register (CR09) supplies the most significant bit, and bit 4 of the Overflow Register (CR07) supplies the second most significant bit. Normally, this 10-bit value is set to specify a scan line after the last scan line of the active display area. When this 10-bit value is set to specify a scan line within the active display area, it causes that scan line and all subsequent scan lines in the active display area to display video data starting at the very first byte of the frame buffer. The result is what appears to be a screen split into a top and bottom part, with the image in the top part being repeated in the bottom part. (This register is only used in split screening modes, and this is not a problem because split screening is not actually used for extended modes. As a result, there is no benefit to extending the existing overflow bits for higher resolutions.) When used in cooperation with the Start Address High Register (CR0C) and the Start Address Low Register (CR0D), it is possible to create a split display, as described

Bit	Description
	earlier, but with the top and bottom parts displaying different data. The top part will display whatever data exists in the frame buffer starting at the address specified in the two aforementioned start address registers, while the bottom part will display whatever data exists in the frame buffer starting at the first byte of the frame buffer.

CR22 - Memory Read Latch Data Register

Address: 3B5h/3D5h (index=22h)

Default: 00h

Attributes:
Read Only

Bit	Description
7:0	**Memory Read Latch Data.** This field provides the value currently stored in 1 of the four memory read latches. Bits 1 and 0 of the Read Map Select Register (GR04) select which of the four memory read latches may be read via this register.

CR24 - Toggle State of Attribute Controller Register

Address: 3B5h/3D5h (index=24h)

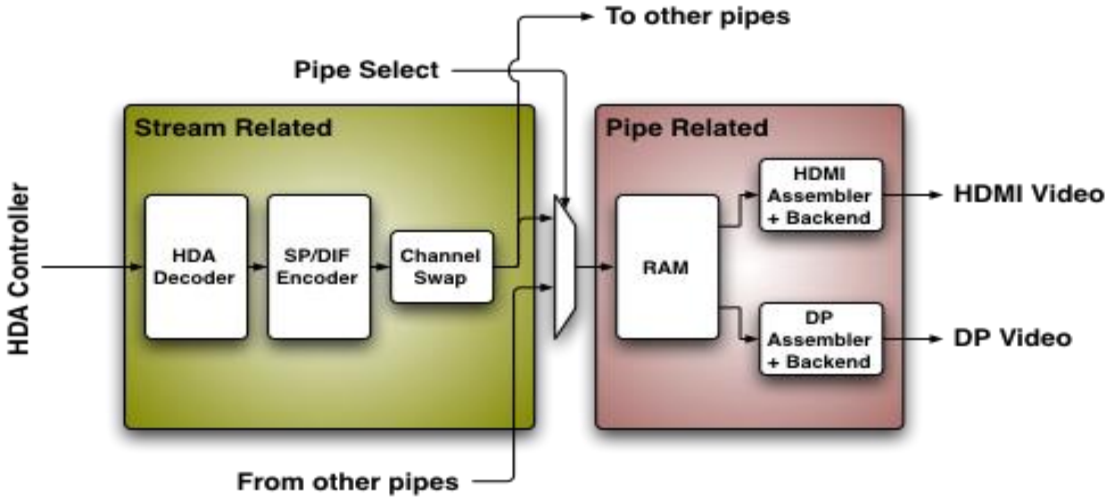
Default: 00h

Attributes:
Read Only

Bit	Description
7	**Toggle Status.** Indicates where the last write to attribute register was to: 0 = index port 1 = data port
6:0	**Reserved.** Read as 0.

Display Audio Codec Verbs

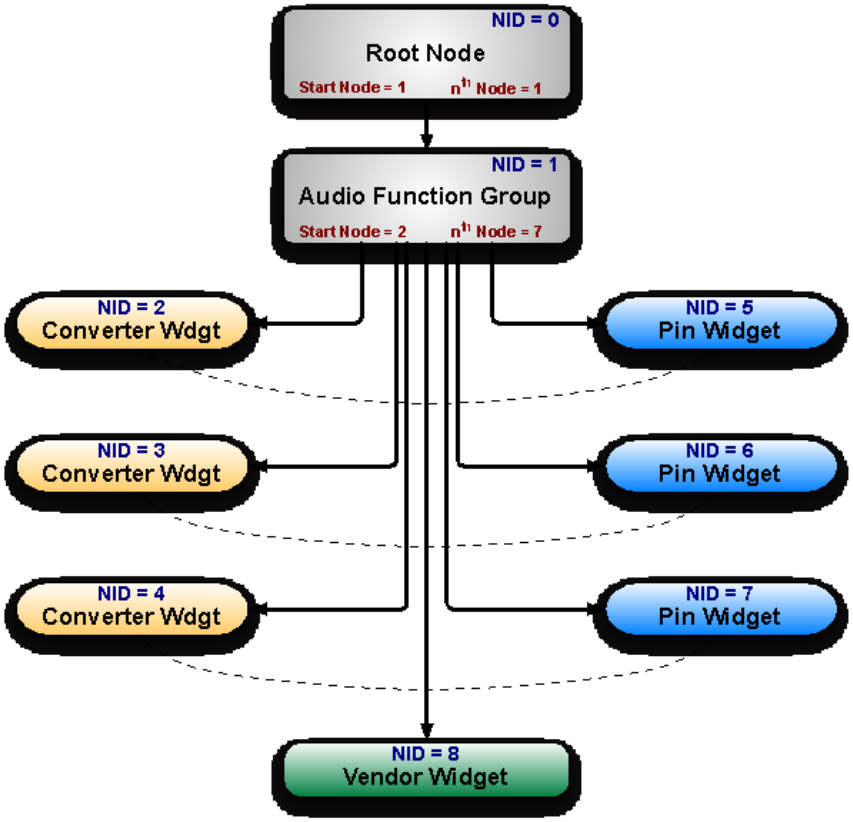
Block Diagram



Codec Node Hierarchy

The diagram below shows the hierarchy of the internal codec. The codec is presented as a single codec with multiple endpoints. By operating as a single codec, only one driver needs to be loaded on the system.

Inside the codec are three "converter widgets" and three "pin widgets", responsible for taking data from HD Audio DMA engines and placing into an HDMI/DP stream. Each pin widget has a 1-1 connection to a converter widget (as indicated by the dotted lines in the diagram).


NID = 0
Root Node
Start Node = 1
nth Node = 1
NID = 1
Audio Function Group
Start Node = 2
nth Node = 7
NID = 2
Converter Wdgt
NID = 3
Converter Wdgt
NID = 4
Converter Wdgt
NID = 5
Pin Widget
NID = 6
Pin Widget
NID = 7
Pin Widget
NID = 8
Vendor Widget

Programming

Programming of the codec is performed by "verbs" as described in the HD Audio specification. These verbs travel over the internal HD Audio link at a rate of 1 verb per frame. A verb can either come from the CORB, with responses using the RIRB, or using an immediate command and response mechanism (ICR). Device 2 contains its own copy of an ICR mechanism as a back-door into the audio codec.

Verb Support

Verb ID			Node ID							
Set	Get	Verb Name/Description	01h	02h	03h	04h	05h	06h	07h	08h
2h	Ah	Stream Descriptor Format		Y	Y	Y				
3h	Bh	Set Amplifier Mute					Y	Y	Y	
-	F00h	Get Parameters	Y	Y	Y	Y	Y	Y	Y	Y
701h	F01h	Connection Select Control					Y	Y	Y	
-	F02h	Connection List Entry					Y	Y	Y	
705h	F05h	Power State		Y	Y	Y	Y	Y	Y	
706h	F06h	Channel and Stream ID		Y	Y	Y				
707h	F07h	Pin Widget Control					Y	Y	Y	
708h	F08h	Unsolicited Response Enable					Y	Y	Y	Y
-	F09h	Pin Sense					Y	Y	Y	
-	F0Dh	Digital Converter		Y	Y	Y				
70Dh	-	Digital Converter 1		Y	Y	Y				
70Eh	-	Digital Converter 2		Y	Y	Y				
-	F1Ch	Configuration Default					Y	Y	Y	
71Ch	-	Configuration Default Byte 0					Y	Y	Y	
71Dh	-	Configuration Default Byte 1					Y	Y	Y	
71Eh	-	Configuration Default Byte 2					Y	Y	Y	
71Fh	-	Configuration Default Byte 3					Y	Y	Y	
-	F20h	Subsystem ID	Y							
-	F21h	Subsystem ID	Y							
-	F22h	Subsystem ID	Y							
-	F23h	Subsystem ID	Y							
720h	-	Subsystem ID[7: 0]	Y							
721h	-	Subsystem ID[15: 8]	Y							
722h	-	Subsystem ID[23:16]	Y							
723h	-	Subsystem ID[31:24]	Y							
72Dh	F2Dh	Converter Channel Count		Y	Y	Y				
-	F2Eh	HDMI/DP Info Size					Y	Y	Y	

Verb ID		Verb Name/Description	Node ID							
730h	F30h	HDMI Info Index					Y	Y	Y	
731h	F31h	HDMI Info Data					Y	Y	Y	
732h	F32h	HDMI Info Transmit Control					Y	Y	Y	
734h	F34h	Converter Channel Map					Y	Y	Y	
735h	F35h	Device Select					Y	Y	Y	
-	F36h	Display Device List Entry					Y	Y	Y	
73Ch	73Ch	DisplayPort Stream ID					Y	Y	Y	
73Eh	-	Digital Converter 3		Y	Y	Y				
73Fh	-	Digital Converter 4		Y	Y	Y				
-	F80h	HDMI / DP Status								Y
781h	F81h	HDMI Vendor Verb								Y
782h	-	GTC Capture Trigger								Y
-	F83h	Captured Wall Clock Value								Y
-	F84h	Captured GTC Value								Y
-	F85h	Get GTC Offset Value								Y
785h	-	Set GTC Offset Value[7: 0]								Y
786h	-	Set GTC Offset Value[15: 8]								Y
787h	-	Set GTC Offset Value[23:16]								Y
788h	-	Set GTC Offset Value[31:24]								Y
789h	F89h	Converter Channel Count								Y

Parameter Support

Param ID	Parameter Name	Node ID								
		00h	01h	02h	03h	04h	05h	06h	07h	08h
00h	Vendor ID	Y								
02h	Revision ID	Y								
04h	Subordinate Node Count	Y	Y							
05h	Function Group Type		Y							
08h	Audio Function Group Capabilities									
09h	Audio Widget Capabilities			Y	Y	Y	Y	Y	Y	Y
0Ah	Sample Size, Rate CAPs			Y	Y	Y				
0Bh	Stream Formats			Y	Y	Y				
0Ch	Pin Capabilities						Y	Y	Y	
0Dh	Input Amp Capabilities									
0Eh	Connection List Length						Y	Y	Y	
0Fh	Supported Power States		Y							

Param ID	Parameter Name	Node ID								
		00h	01h	02h	03h	04h	05h	06h	07h	08h
10h	Processing Capabilities									
11h	GPIO Count									
12h	Output Amp Capabilities						Y	Y	Y	
13h	Volume Knob Capabilities									
15h	Device List Length						Y	Y	Y	

Node ID Descriptions

Below is the description of the valid settings of the Vendor Verb 781h at node ID 02h for enabling the features of the Display Audio Codec. The bits 7:4 are not applicable when bit 0 is set to 1.

Vendor Verb 781h Bits					Node ID															
Port Select [7:4]	Reserved [3:2]	Enable DP1.2 [1]	Enable all pins and all Converters [0]	Description of the control bits	00h	01h	02h	03h	04h	05h	06h	07h	08h	09h	0Ah	0Bh	0Ch	0Dh	0Eh	0Fh
0000	0	0	0	Single Pin is exposed with DDIA on Node ID 4	Root	Function	Vendor	Converter1	DDIA	X	X	X	X	X	X	X	X	X	X	X
0001	0	0	0	Single Pin is exposed with DDIB on Node ID 4	Root	Function	Vendor	Converter1	DDIB	X	X	X	X	X	X	X	X	X	X	X
0010	0	0	0	Single Pin is exposed with DDIC on Node ID 4	Root	Function	Vendor	Converter1	DDIC	X	X	X	X	X	X	X	X	X	X	X
0011	0	0	0	Single Pin is exposed with USBC1 on Node ID 4	Root	Function	Vendor	Converter1	USBC1	X	X	X	X	X	X	X	X	X	X	X
010	0	0	0	Single	Ro	Functi	Vend	Convert	USB	X	X	X	X	X	X	X	X	X	X	X

Vendor Verb 781h Bits					Node ID														
Port Select [7:4]	Reserved [3:2]	Enable DP1.2 [1]	Enable all pins and all Converters [0]	Description of the control bits	00h	01h	02h	03h	04h	05h	06h	07h	08h	09h	0Ah	0Bh	0Ch	0Dh	0Eh	0Fh
0				Pin is exposed with USBC2 on Node ID 4	ot	on	or	er1	C2											
0101	0	0	0	Single Pin is exposed with USBC3 on Node ID 4	Root	Function	Vendor	Converter1	USBC3	X	X	X	X	X	X	X	X	X	X	X
0110	0	0	0	Single Pin is exposed with USBC4 on Node ID 4	Root	Function	Vendor	Converter1	USBC4	X	X	X	X	X	X	X	X	X	X	X
0111	0	0	0	Single Pin is exposed with USBC5 on Node ID 4	Root	Function	Vendor	Converter1	USBC5	X	X	X	X	X	X	X	X	X	X	X
1000	0	0	0	Single Pin is exposed with USBC6 on Node ID 4	Root	Function	Vendor	Converter1	USBC6	X	X	X	X	X	X	X	X	X	X	X

When Bit 0 is set to 1 of verb 781h then mapping as described below. Any converter can be mapped to any pin widget.

Node ID 0h = Root Node

Node ID 1h = Function Node

Node ID 2h = Vendor Widget

Node ID 3h = Convertor 1

Node ID 4h = DDI A

Node ID 5h = Convertor 2

Node ID 6h = DDI B

Node ID 7h = Convertor 3

Node ID 8h = DDI C

Node ID 9h = Convertor 4

Node ID Ah = USBC 1

Node ID Bh = USBC 2

Node ID Ch = USBC 3

Node ID Dh = USBC 4

Node ID Eh = USBC 5

Node ID Fh = USBC 6

Node ID 00h Root Node Verbs

The root node only contains a single verb - the "Get Parameters" verb at F00h.

F00h - Get Parameters

Parameters

Parameter	Symbol	Register Name
00h	PARAM_VID	Vendor ID
02h	PARAM_RID	Revision ID
04h	PARAM_SNC	Subordinate Node Count

Parameter 00h: VID - Vendor ID

Bit	Reset	Description
31:16	8086h	**Vendor ID (VID):** Indicates the 16-bit Vendor ID values used to identify the codec to the PnP subsystem.
15:00	<value in table below>	**Device ID (DID):** Indicates the 16-bit Device ID values used to identify the codec to the PnP subsystem.

Device ID
2812h

Parameter 02h: RID - Revision ID

Bit	Reset	Description
31:24	*0*	*Reserved*
23:20	1h	**Major Revision (MJR):** Indicates the major revision number (left of the decimal) of the High Definition Audio Specification to which the codec is fully compliant.
19:16	0h	**Minor Revision (MNR):** Indicates the minor revision number (right of the decimal) or "dot number" of the High Definition Audio Specification to which the codec is fully compliant.
15:08	00h	**Revision ID (RID):** Indicates the vendor's revision number for this given Device ID.
07:00	00h	**Stepping ID (SID):** Indicates optional vendor stepping number within the revision.

Parameter 04h: PARAM_SNC - Subordinate Node Count

Bit	Reset	Description
31:24	*0*	*Reserved*
23:16	0h	**Starting Node Number (SNN):** Indicates the first sub-node's ID is 01h.
15:08	00h	Reserved
07:00	01h	**Total Number of Nodes (TNN):** Indicates one sub-node

F37h GET CCF - Get Current Clock Frequency

Bits	Default	Description
31:6	0	*Reserved*
05	0	**Current Clock 192 MHz (C192)**: Indicates the current clock is 192 MHz. Reserved for Display Codec
04	1	**Current Clock 96 MHz (C96)**: Indicates the current clock is 96 MHz.
03	0	**Current Clock 48 MHz (C48)**: Indicates the current clock is 48 MHz.
02	0	**Current Clock 24 MHz (C24)**: Indicates the current clock is 24 MHz. Reserved for Display Codec
01	0	**Current Clock 12 MHz (C12)**: Indicates the current clock is 12 MHz. Reserved for Display Codec
00	0	**Current Clock 6 MHz (C6)**: Indicates the current clock is 6 MHz. Reserved for Display Codec

Node ID 01h Audio Function Group Verbs

Set Verb	Get Verb	Symbol	Name
-	F00h	GET PARAM	Get Parameters
705h	F05h	SET_PS / GET_PS	Set Power State
-	F20h	GET_SSID	Get Subsystem ID
720h	F20h	SET_SSID0	Set/Get Subsystem ID
721h	F21h	SET_SSID1	Set/Get Subsystem ID
722h	F22h	SET_SSID2	Set//Get Subsystem ID
723h	F23h	SET_SSID3	Set/Get Subsystem ID

Set Verb	Get Verb	Symbol	Name
724h	F24h	SET CCF	Set/Get Current Clock Frequency

F00h Get Parameters

Parameter	Symbol	Register Name
04h	PARAM_SNC	Subordinate Node Count
05h	PARAM_FGT	Function Group Type
08h	PARAM_FGC	Function Group Capabilities
0Fh	PARAM SPS	Supported Power States

Parameter 04h: PARAM_SNC - Subordinate Node Count

Bit	Reset	Description
31:24	*0*	*Reserved*
23:16	03h	**Start Node Number (SNN):** Indicates the start node number of widget or functional nodes in the Functional Group.
15:08	*0*	*Reserved*
07:00	07h	**Total Number of Nodes (TNN):** Indicates 7 widgets in the Functional Group. (HDMI/DP converters (3) + HDMI/DP pins (3) + Vendor Defined Widget (1)). If Wigig is enabled in the vendor defined widget then the total number of nodes should be reported as 08h. Default for D11 is 9 [HDMI/DP converters (4) + HDMI/DP pins (5)]. If wireless widgets are enabled then total is 10 or 11. Default for D11p5 is 13 [HDMI/DP converters (4) + HDMI/DP pins (9)]. If wireless widgets are enabled then total is 14 or 15.

Parameter 05h: PARAM_FGT - Function Group Type

Bit	Reset	Description
31:09	*0*	*Reserved*
08	0	**Unsolicited Capable (UC):** Not capable of generating an unsolicited response.
07:00	01h	**Node Type (NT):** Indicates Audio Function Group.

Parameter 08h: PARAM_FGC - Function Group Capability

Bit	Reset	Description
31:04	*0*	*Reserved*
03:00	00h	**Output Delay (OD)**Output Delay.

Parameter 0Fh: PARAM_SPS - Supported Power States

Bit	Reset	Description
31	1	**Extended Power State Supported (EPSS):** Indicates support for low power states
30	1	**Clock Stop (CS):** Indicates support for D3 when clock is stopped.
29:04	*0*	*Reserved*
03	1	**D3 Supported (D3S):** Indicates support for D3.
02	0	**D2 Supported (D2S):** Indicates no support for D2.
01	0	**D1 Supported (D1S):** Indicates no support for D1.
00	1	**D0 Supported (D0S):** Indicates support for D0.

Parameter 16h: PARAM_A2CAP - Azalia 2 Capabilities

Bits	Reset	Description
31:17	*0*	*Reserved*
16	0	**Independent Codec Clock (ICC):** When set, this indicates that the codec generates its own clock, which may drift from the link clock. When cleared, the codec's clock is locked to the link clock.
15:06	*0*	*Reserved*
05	*0*	*Reserved for 192 MHz support.*
04	1	**96MHz Supported (S96):** Indicates 96 MHz clock is supported.
03	1	**48MHz Supported (S48):** Indicates 48 MHz clock is supported.This bit must always be set.
02	0	**24MHz Supported (S24):** Indicates 24 MHz clock is supported. Reserved for Display Codec
01	0	**12MHz Supported (S12):** Indicates 12 MHz clock is supported. Reserved for Display Codec
00	0	**6MHz Supported (S6)**Indicates 6 MHz clock is supported. Reserved for Display Codec

705h SET_PS - Set Power State

Bits	Description
07:02	Reserved
01:00	**Requested Power State (RPS):** Only D0 (00) and D3 (11) may be requested

F05h GET_PS - Get Power State

Bits	Reset	Description
31:11	0	Reserved
10	0	**Settings Reset (SR):** Default values will not be changed. This bit will report 0 in all cases.
09	1	**Clock Stop OK (CSOK):** Clock stopping in D3 is OK
08	0	**Error (ERR):** No error will ever be reported.
07:06	0	Reserved
05:04	11	**Actual Power State (APS):** Indicates the current power state of the node.
03:02	0	Reserved

Bits	Reset	Description
01:00	11	**Requested Power State (CPS):** Reflects value written with SET_PS verb.

F20h GET SSID - Get Subsystem ID0

Bits	Reset	Description
31:00	80860101h	**Subsystem ID (SSID):** Reports the sub-system ID set via SET_SSIDx verbs.

720h SET SSID0 - Set Subsystem ID0

Bits	Description
07:00	Subsystem ID Bits [7:0]

721h SET SSID1 - Set Subsystem ID1

Bits	Description
07:00	Subsystem ID Bits [15:8]

722h SET SSID2 - Set Subsystem ID2

Bits	Description
07:00	Subsystem ID Bits [23:16]

723h SET SSID3 - Set Subsystem ID3

Bits	Description
07:00	Subsystem ID Bits [31:24]

724h SET CCF - Set Current Clock Frequency

Bits	Description
07:06	Reserved
05	**Set Clock 192 MHz (S192)**: Set clock to 192 MHz.
04	**Set Clock 96 MHz (S96)**: Set clock to 96 MHz
03	**Set Clock 48 MHz (S48)**: Set clock to 48 MHz
02	**Set Clock 24 MHz (S24)**: Set clock to 24 MHz
01	**Set Clock 12 MHz (S12)**: Set clock to 12 MHz
00	**Set Clock 6 MHz (S6)**: Set clock to 6 MHz

F24h GET CCF - Get Current Clock Frequency

Bits	Bits	Description
31:06	0	*Reserved*
05	0	**Current Clock 192 MHz (C192)**: Indicates the current clock is 192 MHz.
04	0	**Current Clock 96 MHz (C96)**: Indicates the current clock is 96 MHz.
03	0	**Current Clock 48 MHz (C48)**: Indicates the current clock is 48 MHz.
02	0	**Current Clock 24 MHz (C24)**: Indicates the current clock is 24 MHz.
01	0	**Current Clock 12 MHz (C12)**: Indicates the current clock is 12 MHz.
00	0	**Current Clock 6 MHz (C6)**: Indicates the current clock is 6 MHz.

7FFh SET Function Group Reset

Bits	Reset	Description
07:00	00h	The Function Reset command causes the functional unit, and all widgets associated with the functional unit, to return to their power-on reset values. Note that some controls such as the Configuration Default controls should not be reset with this command. It is also possible that certain other controls, such as Caller-ID, should not be reset. This command does not affect the Link interface logic, which must be reset with the link RST# signal. Therefore, a codec must not initiate a Status Change request on the link.

Audio Output Convertor Widget Verbs

Verb	Symbol	Verb Name
2h	SET_SDF	Set Stream Descriptor Format
Ah	GET_SDF	Get Stream Descriptor Format
F00h	GET_PARAM	Get Parameters
705h	SET_PS	Set Power State
F05h	GET_PS	Get Power State
706h	SET_CSID	Set Channel and Stream ID
F06h	GET_CSID	Get Channel and Stream ID
F0Dh	SET_DC1	Get Digital Converter
70Dh	SET_DC1	Set Digital Converter 1
70Eh	SET_DC2	Set Digital Converter 2
73Eh	SET_DC3	Set Digital Converter 3

Verb	Symbol	Verb Name
73Fh	SET_DC4	Set Digital Converter 4
72Dh	SET_CCC	Set Converter Channel Count
F2Dh	GET_CCC	Get Converter Channel Count

2hAh SETGET_SDF - SetGET Stream Descriptor Format

Bits	Reset	Description
31:15	0	Reserved
14	0	**Sample Base Rate (SBR):**
13:11	000	**Sample Base Rate Multiplier (SBRM):**
10:08	000	**Sample Base Rate Divisor (SBRD):**
07	0	Reserved
06:04	011	**Bits / Sample (BPS):** • 001b: Data is packed in memory in 16 bit containers on 16 bit boundaries • 010b: Data is packed in memory in 20 bit containers on 32 bit boundaries • 011b: Data is packed in memory in 24 bit containers on 24 bit boundaries • 100b: Data is packed in memory in 32 bit containers on 32 bit boundaries All other bit combinations reserved
03:00	1h	**# Channels in Stream (NCS):** 2 channels in each frame

F00h Get Parameters

Parameter	Symbol	Register Name
09h	PARAM_AWC	Audio Widget Capabilities
0Ah	PARAM_PSB	Parameter Sizes and Bit Rates
0Bh	PARAM_SF	Stream Formats
0Fh	PARAM_SPS	Power Supported States

Parameter 09h: AWC - Audio Widget Capabilities

Bits	Reset	Description
31:24	0	Reserved
23:20	0h	**Widget Type (TYPE):** Indicates this is an audio output widget
19:16		Sample Delay in Widget (DELAY):
15:13	011	**Channel Count Extension (CCE):** These three bits, combined with STRO, indicate that there are 8 channels supported.
11	0	**L-R Swap (LRS):** Indicates no left/right channel swap.
10	1	**Power Control (PC):** Indicates power state control
09	1	**Digital (DIG):** Indicates support for digital streams.
08	0	**Connection List (CL):** Indicates no connection list
07	0	**Unsolicited Capable (UC):** Indicates support for unsolicited responses.
06	0	**Processing Widget (PW):** Indicates no support for processing
05	0	**Stripe (STRP):** Indicates striping not supported.
04	1	**Format Override (FO):** Indicates support for formatting
03	1	**Amp Parameter Override (APO):** Indicates no amplifier support.
02	0	**Out Amp Present (OAP):** Indicates no output amplifier present.
01	0	**In Amp Present (IAP):** Indicates no input amplifier present.
00	1	**Stereo (STRO):** Indicates a stereo widget

Parameter 0Ah: PSB - PCM Sizes and Bit Rates

Bits	Reset	Description
31:21	0	*Reserved*
20	1	**32-bit Support (B32):** Indicates 32-bit samples supported
19	1	**24-bit Support (B24):** Indicates 24-bit samples supported
18	0	**20-bit Support (B20):** Indicates 20-bit samples supported
17	1	**16-bit Support (B16):** Indicates 16-bit samples supported
16	0	**8-bit Support (B8):** Indicates 8-bit samples not supported
15:12	0	*Reserved*
11	0	**384 kHz Support (R12):** Indicates 384 kHz not supported
10	1	**192 kHz Support (R11):** Indicates 192 kHz supported
09	1	**176.4 kHz Support (R10):** Indicates 176.4 kHz supported
08	1	**96 kHz Support (R9):** Indicates 96 kHz supported
07	1	**88.2 kHz Support (R8):** Indicates 88.2 kHz supported
06	1	**48 kHz Support (R7):** Indicates 48 kHz supported
05	1	**44.1 kHz Support (R6):** Indicates 44.1 kHz supported
04	1	**32 kHz Support (R5):** Indicates 32 kHz supported
03	0	**22.05 kHz Support (R4):** Indicates 22.05 kHz not supported
02	0	**16 kHz Support (R3):** Indicates 16 kHz not supported
01	0	**11.025 kHz Support (R2):** Indicates 11.025 kHz not supported
00	0	**8 kHz Support (R1):** Indicates 8 kHz not supported

Parameter 0Bh: SF - Stream Formats

Bits	Reset	Description
31:03	0	*Reserved*
02	1	**AC3 Support (AC3):** Indicates AC3 stream format is supported
01	0	**Float32 Support (F32):** Indicates float32 stream format not supported
00	1	**PCM Support (PCM):** Indicates PCM format is supported.

Parameter 0Fh: PARAM_SPS - Supported Power States

Bit	Reset	Description
31	1	**Extended Power State Supported (EPSS):** Indicates support for low power states
30:04	0	*Reserved*
03	1	**D3 Supported (D3S):** Indicates support for D3.
02	0	**D2 Supported (D2S):** Indicates no support for D2.
01	0	**D1 Supported (D1S):** Indicates no support for D1.
00	1	**D0 Supported (D0S):** Indicates support for D0.

705h SET_PS - Set Power State

Bits	Description
07:02	Reserved
01:00	**Requested Power State (RPS):** Only D0 (00) and D3 (11) may be requested

F05h GET_PS - Get Power State

Bits	Reset	Description
31:11	0	Reserved
10	0	**Settings Reset (SR):** Default values will not be changed. This bit will report 0 in all cases.
09	0	**Clock Stop OK (CSOK):** Clock stopping in D3 is not OK

Bits	Reset	Description
08	0	**Error (ERR):** No error will ever be reported.
07:06	0	Reserved
05:04	11	**Actual Power State (APS):** Indicates the current power state of the node.
03:02	0	Reserved
01:00	11	**Requested Power State (CPS):** Reflects value written with SET_PS verb.

706hF06h GETSET_CSID - GetSet Channel and Stream ID

Bits	Reset	Description
07:04	0h	**Stream ID (SID):** Link stream used by the converter for data output.
03:00	0h	**Lowest Channel Number (LCN):** Lowest channel used by the converter.

Digital Converter Verbs

F0Dh: GET_DC - Get Digital Converter

Bits	Reset	Description
31:24	0	Reserved
23	1	**Keep Alive (KA):** See SET_DC3.KA
22:20	0	Reserved
19:16	0h	**IEC Coding Type (ICT):** See SET_DC3.ICT
15	0	Reserved
14:08	00h	**Category Code (CC):** See SET_DC1.CC
07	0	**Level (LVL):** See SET_DC1.LVL
06	0	**Professional (PRO):** See SET_DC1.PRO
05	0	**Audio is not PCM (AUDIO):** See SET_DC1.AUDIO
04	0	**Copyright (COPY):** See SET_DC1.COPY
03	0	**Pre-emphasis (PRE):** See SET_DC1.PRE
02	0	**Validity Configuration (VCFG):** See SET_DC1.VCFG

Bits	Reset	Description
01	0	**Validity (V):** See SET_DC1.V
00	1	**Digital Enable (DIGEN):** See SET_DC1.DIGEN

70Dh: SET_DC1 - Set Digital Converter 1

Bits	Description
07	**Level (LVL):** S/PDIF IEC Generation Level.
06	**Professional (PRO):** When set, indicates professional use of channel.
05	**Audio is not PCM (AUDIO):** When set, data is non-PCM format.
04	**Copyright (COPY):** When set, copyright asserted.
03	**Pre-emphasis (PRE):** When set, enables filter pre-emphasis.
02	**Validity Configuration (VCFG):** Determines S/PDIF transmitter behavior when data is not being transmitted.
01	**Validity (V):** Affects the validity flag transmitted in each sub-frame, and enables S/PDIF transmitter to maintain connection during error or mute conditions.
00	**Digital Enable (DIGEN):** When set, enables digital content

70Eh: Digital Converter 2

Bits	Description
07	Reserved
06:00	**Category Code (CC):** S/PDIF IEC Category Code.

73Eh: Digital Converter 3

Bits	Description
07	Keep Alive
06:04	Reserved
03:00	IEC Coding Type

73Fh: Digital Converter 4

Bits	Description
07:00	Reserved

72DhF2Dh GETSET_CCC - GetSet Converter Channel Count

Bits	Reset	Description
07:04	0	Reserved
03:00	0000	Converter Channel Count 1 (0th order)

Pin Widget Verbs

Set Verb	Get Verb	Symbol	Verb Name
3h	-	SET_AM	Set Amplifier Mute
-	Bh	GET_AM	Get Amplifier Mute
-	F00h	-	Get Parameters
701h	F01h	SET_CSC / GET_CSC	Set/Get Connection Select Control
-	F02h	-	Get Connection List Entry
705h	F05h	SET_PS / GET_PS	Set/Get Power State
707h	F07h	SET_PWC / GET_PWC	Set/Get Pin Widget Control
708h	F08h	SET_UE / GET_UE	Set/Get Unsolicited Response Enable
-	F09h	-	Get Pin Sense
71Ch	-	SET_CD0	Set Configuration Default Byte 0
71Dh	-	SET_CD1	Set Configuration Default Byte 1
71Eh	-	SET_CD2	Set Configuration Default Byte 2
71Fh	-	SET_CD3	Set Configuration Default Byte 3
-	F1Ch	GET_CD	Get Configuration Default
-	F2Eh	GET_HDIS	Get HDMI/DP Info Size
730h	F30h	SET_HII / GET_HII	Set/Get HDMI Info Index
731h	F31h	SET_HID / GET_HID	Set/Get HDMI Info Data
732h	F32h	SET_HITC / GET_HITC	Set/Get HDMI Info Transmit Control
733h	F33h	SET_PC / GET_PC	Set/Get Protection Control
734h	F34h	SET_CCM / GET_CCM	Set/Get Converter Channel Map
735h	F35h	SET_DS / GET_DS	Set/Get Device Select
-	F36h	GET_DDLE	Get Display Device List Entry
73Ch	F3Ch	SET_DPID / GET_DPID	Set/Get DisplayPort Stream ID

3h SET_AM - Set Amplifier Mute

Bits	Bits	Description
15	0	**Set Output Amp (SOA):**.
14	0	**Set Input Amp (SIA):**.
13	0	**Set Left Amp (SLA):**.
12	0	**Set Right Amp (SRA):**.
11:08	0h	**Index (IDX):**
07	1	**Mute (MUTE):** When set, amp muted.
06:00	0	*Reserved*

B8h GET_AM - Get Amplifier Mute

Bits	Bits	Description
31:08	0	Reserved
07	1	**Mute (MUTE):** When set, amp is muted.
06:00	0	Reserved

F00h Get Parameters

Parameter	Symbol	Register Name
09h	PARAM_AWC	Audio Widget Capabilities
0Ch	PARAM_PC	Pin Capabilities
0Eh	PARAM_CLL	Connection List Length
12h	PARAM OAC	Output Amplifier Capabilities
15h	PARAM_DLL	Device List Length
0Fh	PARAM_SPS	Supported Power States

Parameter 09h: AWC - Audio Widget Capabilities

Bits	Reset	Description
31:24	0	Reserved
23:20	4h	**Widget Type (TYPE):** Indicates this is a pin complex widget
19:16	0	**Sample Delay in Widget (DELAY):** No delay through the pin widget.
15:13	011	**Channel Count Extension (CCE):** This field, combined with STRO, indicate 8 channels supported.
11	0	**L-R Swap (LRS):** Indicates no left/right channel swap.
10	1	**Power Control (PC):** Indicates power state control
09	1	**Digital (DIG):** Indicates support for digital streams.
08	1	**Connection List (CL):** Indicates a connection list
07	1	**Unsolicited Capable (UC):** Indicates support for unsolicited responses.
06	0	**Processing Widget (PW):** Indicates no support for processing
05	0	**Stripe (STRP):** Indicates striping not supported.
04	0	**Format Override (FO):** Indicates no support for formatting
03	1	**Amp Parameter Override (APO):** Indicates no amplifier override support.
02	1	**Out Amp Present (OAP):** Indicates no output amplifier present.
01	0	**In Amp Present (IAP):** Indicates no input amplifier present.
00	1	**Stereo (STRO):** Indicates a stereo widget

Parameter 0Ch: PC - Pin Capabilities

Bits	Reset	Description
31:28	0	*Reserved*
27	1	**High Bit Rate (HBR):** Indicates support for high bit-rate audio
26	0	*Reserved*
25	1	**Multi Stream Capable (MSC):** Indicates support for DisplayPort Multistream. Will be zero in vanilla mode.
24	1	**DisplayPort (DP):** Indicates support for DisplayPort
23:08	0	*Reserved*
07	1	**HDMI (HDMI):** Indicates support for HDMI
06:05	0	*Reserved*
04	1	**Output Capable (OC):** Pin is output capable
03	0	*Reserved*
02	1	**Presence Detect Capable (PDC):** Indicates capability for presence detection
01:00	0	*Reserved*

Parameter 0Eh: CLL - Connection List Length

Bits	Reset	Description
31:08	0	*Reserved*
07	0	**Long Form (LF):** Indicates connection list is short form
06:00	03h	**Length (LEN):** Indicates there is one item in the connection list.

Parameter 12h: OAC - Output Amplifier Capabilities

Bits	Reset	Description
31	1	**Mute Capable (MC):** Muting is capable on this pin
30:00	0	Reserved

Parameter 15h: DLL - Device List Length

Bits	Reset	Description
31:06	0	*Reserved*
05:00	00h	**Length (LEN):** Indicates no devices

Parameter 0Fh: PARAM_SPS - Supported Power States

Bit	Reset	Description
31	1	**Extended Power State Supported (EPSS):** Indicates support for low power states
30:04	0	*Reserved*
03	1	**D3 Supported (D3S):** Indicates support for D3.
02	0	**D2 Supported (D2S):** Indicates no support for D2.
01	0	**D1 Supported (D1S):** Indicates no support for D1.
00	1	**D0 Supported (D0S):** Indicates support for D0.

701hF01h SETGET_CSC - SetGet Connection Select Control

Bits	Reset	Description
07:00	00h	**Connection Select Control (CSC):**

F02h GET_CLE - Get Connection List Entry

Bits	Reset	Description
31:08	0	Reserved
07:00	Varies	**Connection List Entry (CLE):** 02h for NodeID 05h, 03h for NodeID 06h, and 04h for NodeID 07h.

705h SET_PS - Set Power State

Bits	Description
07:02	Reserved
01:00	**Requested Power State (RPS):** Only D0 (00) and D3 (11) may be requested

F05h GET_PS - Get Power State

Bits	Reset	Description
31:11	0	Reserved
10	0	**Settings Reset (SR):** Default values will not be changed. This bit will report 0 in all cases.
09	0	**Clock Stop OK (CSOK):** Clock stopping in D3 is not OK
08	0	**Error (ERR):** No error will ever be reported.
07:06	0	Reserved
05:04	11	**Actual Power State (APS):** Indicates the current power state of the node.
03:02	0	Reserved
01:00	11	**Requested Power State (CPS):** Reflects value written with SET_PS verb.

707hF07h SETGET_PWC - SetGet Pin Widget Control

Bits	Reset	Description
07	0	Reserved
06	1	**Out Enable (OE):** When set, the audio is enabled
05:02	0	Reserved
01:00	00	**Encoded Packet Type (EPT):**

708hF08h SETGET_UE - SetGet Unsolicited Enable

Bits	Description
07	**Unsolicited Enable (UE):** When set, unsolicited responses are allowed
06	Reserved
05:00	**Tag (TAG):**

F09h GET_PS - Get Pin Sense

Determining SST / MST Mode

The audio codec will use multi-stream based indexing (MST mode) as described in the following sections if the following conditions are met, else it will use Non-MST mode.

Conditions for MST Mode
• In register **TRANS_DDI_FUNC_CTL**, the bit-field "TRANS DDI Mode Select" is set to "DP MST" **AND** • In verb GetSet iDisp Codec Vendor Verb (781h), the bit-field "Enable DP1.2 Features" (bit 1) is set to 1b.

Verb Parameters (Non-MST Mode)

For non-MST mode, the F09h verb response will automatically provide the status of the single pipe associated with the given pin node.

Nothing needs to be set in the data byte of the F09h Get Pin Sense verb.

Bits	Reset	Description
07:00	0	Reserved

Verb Parameters (MST Mode)

For MST mode, the F09h verb response will provide the status of the pipe selected in the Pipe Select bit-field, among the pipe(s) associated with the given pin node.

The data byte of the F09h Get Pin Sense verb should be programmed to select the desired pipe as shown below.

Bits	Reset	Description
07:06	0	Reserved
05:00	0	**Pipe Select:** Set to 000000b for pipe A, 000001b for pipe B, 000010b for pipe C, 000011b for pipe D

Verb Response (All Modes)

The verb response for the F09h Get Pin Sence verb command contains the following data regardless of which mode (Non-MST or MST) is used.

Bits	Reset	Description
31	0	**Presence Detect (PD):** When set presence is detected on this pin.
30	0	**ELD Value (ELDV):**

Bits	Reset	Description
29	0	**Inactive (INA):**
28:00	28:00	Reserved

71Ch SET_CD0 - Set Configuration Default Byte 0

Bits	Description
07:04	Default Association (DA):
03:00	Sequence (SEQ):

71Dh SET_CD1 - Set Configuration Default Byte 1

Bits	Description
07:04	Color (COL):
03:00	Miscellaneous (MISC):

71Eh SET_CD2 - Set Configuration Default Byte 2

Bits	Description
07:04	Default Device (DD):
03:00	Connection Type (CT):

71Fh SET_CD3 - Set Configuration Default Byte 3

Bits	Description
07:06	**Port Connectivity (PC):** External connectivity of the pin complex. • 00 = Connected to jack • 01 = No physical connection • 10 = Fixed function device (integrated speaker, mic, etc.) • 11 = Both a jack and internal connection

Bits	Description
05:00	**Location (LOC):** Bits 3:0 Bits 5:4 / 0h: N/A / 1h: Rear / 2h: Front / 3h: Left / 4h: Right / 5h: Top / 6h: Bottom / 7h: Special / 8h: Special / 9h: Special / Ah-Fh Reserved 00: External — Y Y Y Y Y Y Y Y Y 01: Internal — Y Y Y Y 10: Separate Chassis — Y Y Y Y Y Y Y 11: Other — Y Y Y Y Y

F1Ch GET_CD - Get Configuration Default

Bits	Description
31:30	**Port Connectivity (PC):** See SET_CD3.PC
29:24	**Location (L):** See SET_CD3.L
23:20	**Default Device (DD):** See Set_CD2.DD
19:16	**Connection Type (CT):** See Set_CD2.CT
15:12	**Color (COL):** See SET_CD1.COL
11:08	**Miscellaneous (MISC):** See SET_CD1.MISC
07:04	**Default Association (DA):** See SET_CD0.DA
03:00	**Sequence (SEQ):** See SET_CD0.SEQ

F2Eh HDMIDP Info Size

Bits	Reset	Description
31:08	0	Reset
07:00	Varies	**Size (SZ):** Indexes 0 - 3 return 1Eh, index 1000 returns 53h, others reserved.

F2Fh Get ELD Data

Parameter	Symbol	Register Name
07:0h	PARAM_INDX	ELD DATA Index

Parameter nn: ELD Data

Bits	Reset	Description
31:00	0	ELD Data

730hF30h SETGET_HII - SetGet HDMI Info Index

Bits	Reset	Description
07:05	000	**Infoframe Packet Index (IPI):** Value / Name / Value / Name 000 / Audio / 011 / GP3 001 / GP / 100 / GP4 010 / GP2 / Others / Reserved
04:00	00h	**Byte Offset Index Pointer (BOI):**

731hF31h SETGET_HID - SetGet HDMI Info Data

Bits	Reset	Description
07:00	00h	**Data (DATA):** Data at current index pointed to from SET_HII verb.

732hF32h SETGET_HITC - SetGet HDMI Info Transmit Control

Bits	Reset	Description
07:06	00	**InfoFrame Control Current Indexed Frame (IFCCIF):** • 00 = Disable Transmit • 01 = Reserved • 10 = Transmit Once • 11 = Best Effort
05:00	0	*Reserved*

733h SET_PC - Set Protection Control

Bits	Description
07:03	**Unsolicited Response Sub Tag (URST):** Subtag to use for unsolicited responsed.
02	Reserved

734hF34h SETGET_CCM - GetSet Converter Channel Map

Bits	Reset	Description
07:04	0h	Converter Channel (CC):
03:00	0h	Slot (SN):

735h SET_DS - Set Device Select

Bits	Reset	Description
07:06	0	*Reserved*
05:00	00h	**Device (D):** 000000, 000001, 000010, 000011 (based upon number of devices present)

F35h: GET_DS - Get Device Select

Bits	Description
31:12	**Reserved:** Set to 0
11:06	**SinK Device ID:** Sink Device ID in the multi stream topology of the DP hierarchy. Device attached to Pipe A will have ID of "000000", PipeB will have "000001", Pipe C will have "000010" and Pipe D will have "000011".
05:00	**Device (D):**Device Entry index currently set

F36h GET_DDLE - Get Display Device List Entry

Bits	Bits	Description
31:16	0	*Reserved*
15	0	*Reserved*
14	0	**IA of Entry 3**
13	0	**ELDV of Entry 3**
12	0	**PD of Entry 3**
11	0	*Reserved*
10	0	**IA of Entry 2**

Bits	Bits	Description
09	0	**ELDV of Entry 2**
08	0	**PD of Entry 2**
07	0	*Reserved*
06	0	**IA of Entry 1**
05	0	**ELDV of Entry 1**
04	0	**PD of Entry 1**
03	0	*Reserved*
02	0	**IA of Entry 0**
01	0	**ELDV of Entry 0**
00	0	**PD of Entry 0**

73ChF3Ch SETGET_DPID - SetGet DisplayPort Stream ID

Bits	Reset	Description
07:03	00h	**Tag (TAG):** Represents the SSID that will go in the lower 5 bits of the SSID
02:00	000	**Index (IDX):** Pointer to program multiple SSID

Intel Vendor Widget Verbs

Set Verb	Get Verb	Symbol	Verb Name
-	F00h	GET_PARAM	Get Parameters
-	F80h	GET_HDPS	Get HDMI/DP Status
781h	F81h	SET_HVV / GET_HVV	Set/Get iDisp Codec Vendor Verb
782h	-	SET_GTCT	Set GTC Trigger
-	F83h	GET_CWC	Get Captured Wall Clock
	F84h	GET_CGTC	Get Captured GTC Value
	F85h	GET_GOF	Get GTC Offset Value
785h	-	SET_GOF0	Set GTC Offset Value Byte 0
786h	-	SET_GOF1	Set GTC Offset Value Byte 1
787h	-	SET_GOF2	Set GTC Offset Value Byte 2

Set Verb	Get Verb	Symbol	Verb Name
788h	-	SET_GOF3	Set GTC Offset Value Byte 3
789h	F89h	SET_GDI / GET_GDI	Set/Get GTC Offset Device Index

F00h Get Parameters

Parameter	Symbol	Register Name
09h	PARAM_AWC	Audio Widget Capabilities

Parameter 09h: AWC - Audio Widget Capabilities

Bits	Reset	Description
31:24	0	Reserved
23:20	Fh	**Widget Type (TYPE):** Indicates this is a vendor defined widget
19:00	0	Reserved

71Eh SET_GET_GFXMAILBOX - Set Get GFX MAILBOX Byte 2

71Eh: SET GFX MAILBOXM

Bits	Default	Description
07:00	00	Contents to be defined by GFX driver and audio driver

F1Eh: GET GFX MAILBOX

Bits	Default	Description
31:24	00h	Other values of 71F, verbs
23:16	00h	Contents to be defined by GFX driver and audio driver
15:00	0000h	Other values of 71D, 71C verb

728h SET CLOCK OFF - Set Clock Off Command

Bits	Description
07:00	Data is Irrelevant

708hF08h SETGET_UE - SetGet Unsolicited Enable

Bits	Description
07	**Unsolicited Enable (UE):** When set, unsolicited responses from GFX MAIL BOX register writes are allowed.
06	Reserved
05:00	**Tag (TAG):** Tag for GFX Mail box register unsol responses.

781hF81h GETSET_VV - GetSet iDisp Codec Vendor Verb

Bits	Default	Description
07:04	0	**Port Select:** This filed is programmed to select the port to be exposed to the inbox driver in the vanilla mode. Will not reset on double function group reset. 0000 -> DDI A 0001 -> DDI B 0010 -> DDI C 0011 -> USBC1 0100 -> USBC2 0101 -> USBC3 0110 -> USBC4 0111 -> USBC5 1000 -> USBC6
03:02	0	**Reserved**
01	0	**Enable DP1.2 Features (EDP12):** When set, DP1.2 features are enabled. Will not reset on double function group reset.
00	0	**Enable all Pins and all Converter Widget (E3P):** When set, all the pins and converter widgets are enabled and can respond to HD Audio Verbs. When cleared, only one Pin and one converter widgets are exposed. Pin selection is done programming bits 7:5. Will not reset on double function group reset.

782h SET_GTCT - Set GTC Trigger

Bits	Bits	Description
07:00	0	**Any data:** The value of this field is irrelevant. The access of the SET causes a capture to occur.

F83h GET_CGTC - Get Captured GTC Value

Bits	Bits	Description
07	0	**GTC Value:** 32-bit GTC value captured on the SET_GTCT verb

F84h GET_CWC - Get Captured Wall Clock Value

Bits	Bits	Description
31:00	0	**Wall Clock Value:** 32-bit wall clock value captured on the SET_GTCT verb

F85h GET GOF - Get GTC Offset Value

Bits	Reset	Description
31:00	0h	**Value (VAL):** Reports the GTC Offset Value.

785h SET GOF0 - Set GTC Offset Value Byte 0

Bits	Description
07:00	GTC Offset Value Bits [7:0]

786h SET GOF1 - Set GTC Offset Value Byte 1

Bits	Description
07:00	GTC Offset Value Bits [15:8]

787h SET GOF2 - Set GTC Offset Value Byte 2

Bits	Description
07:00	GTC Offset Value Bits [23:16]

788h SET GOF3 - GTC Offset Value Byte 3

Bits	Description
07:00	GTC Offset Value Bits [31:24]

789hF89h SETGET_GDI - SetGet GTC Device Index

Bits	Reset	Description
07:06	0	*Reserved*
05:00	00h	**Device (D):** 000001, 000010 (based upon number of devices present) This is the pipe based. 000000 i s pipeA and so on and so forth.

Intel Wireless Widget Verbs

Set Verb	Get Verb	Symbol	Verb Name
3h	-	SET_AM	Set Amplifier Mute
-	Bh	GET_AM	Get Amplifier Mute
-	F00h	-	Get Parameters
701h	F01h	SET_CSC / GET_CSC	Set/Get Connection Select Control
-	F02h	-	Get Connection List Entry
705h	F05h	SET_PS / GET_PS	Set/Get Power State
707h	F07h	SET_PWC / GET_PWC	Set/Get Pin Widget Control
708h	F08h	SET_UE / GET_UE	Set/Get Unsolicited Response Enable
-	F09h	-	Get Pin Sense
71Ch	-	SET_CD0	Set Configuration Default Byte 0
71Dh	-	SET_CD1	Set Configuration Default Byte 1
71Eh	-	SET_CD2	Set Configuration Default Byte 2
71Fh	-	SET_CD3	Set Configuration Default Byte 3
-	F1Ch	GET_CD	Get Configuration Default
-	F2Eh	GET_HDIS	Get HDMI/DP Info Size
733h	F33h	SET_PC / GET_PC	Set/Get Protection Control
734h	F34h	SET_CCM / GET_CCM	Set/Get Converter Channel Map
73Ch	F3Ch	SET_STRID / GET_STRID	Set/Get Widi Stream ID
740h-743h	F40h-F43h	SET / GET_PTSOFFSET	Set/Get PTS Offset Byte 0, 1, 2, 3
750h-753h	F50h-F53h	SET / GET Widi Buf LowBase Addr	Set/Get Widi Lower base address Byte 0, 1, 2, 3
754h-757h	F54h-F57h	SET / GET Widi Buf Upper Base Addr	Set/Get Widi Upper base address Byte 0, 1, 2, 3
760h-763h	F60h-F63h	SET / GET Widi Buf size	Set/Get Widi Buf Size Byte 0, 1, 2, 3
764h	F64h	SET / GET WDE TFD Size	Set/Get WDE TFD Size
770h	F70h	SET/GET Widi PID	Set/Get Widi PID
780h-783h	F80h-F83h	SET/GET WDE/WFA Header	Set/Get WDE/WFA Header
784h	F84h	SET/GET WDE Program Number	Set/Get WDE Program number

Set Verb	Get Verb	Symbol	Verb Name
785h-786h	F85h-F86h	SET/GET WDE STD Bit Rate	Set/Get WDE STD bit rate
787h	F87h	SET/GET WDE PKT PER FRM	Set/Get Packets per frame
788h	F88h	SET/GET Wireless Widget Mode	Set/Get Wireless Widget Mode (Wigig, Widi, WFA)

3h SET_AM - Set Amplifier Mute

Bits	Bits	Description
15	0	**Set Output Amp (SOA):**.
14	0	**Set Input Amp (SIA):**.
13	0	**Set Left Amp (SLA):**.
12	0	**Set Right Amp (SRA):**.
11:08	0h	**Index (IDX):**
07	1	**Mute (MUTE):** When set, amp muted.
06:00	0	*Reserved*

B8h GET_AM - Get Amplifier Mute

Bits	Bits	Description
31:08	0	*Reserved*
07	1	**Mute (MUTE):** When set, amp muted.
06:00	0	*Reserved*

F00h Get Parameters

Parameter	Symbol	Register Name
09h	PARAM_AWC	Audio Widget Capabilities
0Ch	PARAM_PC	Pin Capabilities
0Eh	PARAM_CLL	Connection List Length
12h	PARAM OAC	Output Amplifier Capabilities
0Fh	PARAM_PS	Supported Power States

Parameter 09h: AWC - Audio Widget Capabilities

Bits	Reset	Description
31:24	0	Reserved
23:20	4h	**Widget Type (TYPE):** Indicates this is a pin complex widget
19:16	0	**Sample Delay in Widget (DELAY):** No delay through the pin widget.
15:13	011	**Channel Count Extension (CCE):** This field, combined with STRO, indicate 8 channels supported.
12	0	**Content Protection Capable (CPC):** Indicates pin capable of content protection.
11	0	**L-R Swap (LRS):** Indicates no left/right channel swap.
10	1	**Power Control (PC):** Indicates power state control
09	1	**Digital (DIG):** Indicates support for digital streams.
08	1	**Connection List (CL):** Indicates a connection list
07	1	**Unsolicited Capable (UC):** Indicates support for unsolicited responses.
06	0	**Processing Widget (PW):** Indicates no support for processing
05	0	**Stripe (STRP):** Indicates striping not supported.
04	0	**Format Override (FO):** Indicates no support for formatting
03	1	**Amp Parameter Override (APO):** Indicates no amplifier override support.
02	1	**Out Amp Present (OAP):** Indicates no output amplifier present.
01	0	**In Amp Present (IAP):** Indicates no input amplifier present.
00	1	**Stereo (STRO):** Indicates a stereo widget

Parameter 0Ch: PC - Pin Capabilities

Bits	Reset	Description
31:28	0	*Reserved*
27	1	**High Bit Rate (HBR):** Indicates support for high bit-rate audio
26	0	*Reserved*
25	0	*MST capable (DP1.2)*
24	1	**DisplayPort (DP):** Indicates support for DisplayPort
23:08	0	*Reserved*
07	1	**HDMI (HDMI):** Indicates support for HDMI
06:05	0	*Reserved*
04	1	**Output Capable (OC):** Pin is output capable
03	0	*Reserved*
02	1	**Presence Detect Capable (PDC):** Indicates capability for presence detection
01:00	0	*Reserved*

Parameter 0Eh: CLL - Connection List Length

Bits	Reset	Description
31:08	0	*Reserved*
07	0	**Long Form (LF):** Indicates connection list is short form
06:00	03h	**Length (LEN):** Indicates there is one item in the connection list.

Parameter 12h: OAC - Output Amplifier Capabilities

Bits	Reset	Description
31	1	**Mute Capable (MC):** Muting is capable on this pin
30:00	0	Reserved

Parameter 0Fh: PARAM_SPS - Supported Power States

Bit	Reset	Description
31	1	**Extended Power State Supported (EPSS):** Indicates support for low power states
30:04	0	*Reserved*
03	1	**D3 Supported (D3S):** Indicates support for D3.
02	0	**D2 Supported (D2S):** Indicates no support for D2.
01	0	**D1 Supported (D1S):** Indicates no support for D1.
00	1	**D0 Supported (D0S):** Indicates support for D0.

701hF01h SETGET_CSC - SetGet Connection Select Control

Bits	Reset	Description
07:00	00h	**Connection Select Control (CSC):**

F02h GET_CLE - Get Connection List Entry

Bits	Reset	Description
31:24	0	Reserved
23:16	100	**Connection List Entry (CLE):** Hardwired to Converter ID 4
15:08	011	**Connection List Entry (CLE)**: Hardwired to Converter ID 3
07:00	010	**Connection List Entry (CLE)**: Hardwired to Converter ID 2

705h SET_PS - Set Power State

Bits	Description
07:02	Reserved
01:00	**Requested Power State (RPS):** Only D0 (00) and D3 (11) may be requested

F05h GET_PS - Get Power State

Bits	Reset	Description
31:11	0	Reserved
10	0	**Settings Reset (SR):** Default values will not be changed. This bit will report 0 in all cases.
09	0	**Clock Stop OK (CSOK):** Clock stopping in D3 is not OK
08	0	**Error (ERR):** No error will ever be reported.
07:06	0	Reserved
05:04	11	**Actual Power State (APS):** Indicates the current power state of the node.
03:02	0	Reserved
01:00	11	**Requested Power State (CPS):** Reflects value written with SET_PS verb.

707hF07h SETGET_PWC - SetGet Pin Widget Control

Bits	Reset	Description
07	0	Reserved
06	1	**Out Enable (OE):** When set, the audio is enabled
05:02	0	Reserved
01:00	00	**Encoded Packet Type (EPT):**

708hF08h SETGET_UE - SetGet Unsolicited Enable

Bits	Description
07	**Unsolicited Enable (UE):** When set, unsolicited responses are allowed
06	Reserved
05:00	**Tag (TAG):**

F09h GET_PS - Get Pin Sense

Bits	Reset	Description
31	0	**Presence Detect (PD):** When set presence is detected on this pin.
30	0	**ELD Value (ELDV):**
29	0	**Inactive (INA):**
28:00	28:00	Reserved

71Ch SET_CD0 - Set Configuration Default Byte 0

Bits	Description
07:04	Default Association (DA):
03:00	Sequence (SEQ):

71Dh SET_CD1 - Set Configuration Default Byte 1

Bits	Description
07:04	Color (COL):
03:00	Miscellaneous (MISC):

71Eh SET_CD2 - Set Configuration Default Byte 2

Bits	Description
07:04	Default Device (DD):
03:00	Connection Type (CT):

71Fh SET_CD3 - Set Configuration Default Byte 3

Bits	Description
07:06	**Port Connectivity (PC):** External connectivity of the pin complex. • 00 = Connected to jack • 01 = No physical connection • 10 = Fixed function device (integrated speaker, mic, etc.) • 11 = Both a jack and internal connection
05:00	**Location (LOC):**

Bits	Description

	Bits 3:0										
Bits 5:4	0h: N/A	1h: Rear	2h: Front	3h: Left	4h: Right	5h: Top	6h: Bottom	7h: Special	8h: Special	9h: Special	Ah-Fh Reserved
00: External	Y	Y	Y	Y	Y	Y	Y	Y	Y		
01: Internal	Y							Y	Y	Y	
10: Separate Chassis	Y	Y	Y	Y	Y	Y	Y				
11: Other	Y						Y	Y	Y		

F1Ch GET_CD - Get Configuration Default

Bits	Description
31:30	**Port Connectivity (PC):** See SET_CD3.PC
29:24	**Location (L):** See SET_CD3.L
23:20	**Default Device (DD):** See Set_CD2.DD
19:16	**Connection Type (CT):** See Set_CD2.CT
15:12	**Color (COL):** See SET_CD1.COL
11:08	**Miscellaneous (MISC):** See SET_CD1.MISC
07:04	**Default Association (DA):** See SET_CD0.DA
03:00	**Sequence (SEQ):** See SET_CD0.SEQ

F2Eh WIDI ELD Size

Bits	Reset	Description
31:08	0	Reset
07:00	Varies	**Size (SZ)**Reserved as we do not have Info frame support for WIdi widget. Will return the value 53h as the response.

F2Fh Get ELD Data

Parameter	Symbol	Register Name
07:0h	PARAM_INDX	ELD DATA Index

Parameter nn: ELD Data

Bits	Reset	Description
31:00	0	ELD Data

733h SET_PC - Set Protection Control

Bits	Description
07:03	**Unsolicited Response Sub Tag (URST):** Subtag to use for unsolicited responsed.
02	Reserved
01:00	**Content Protection Requested (CPR):** • 00 = Don't Care • 01 = Reserved • 10 = protection OFF • 11 = Protection ON

734hF34h SETGET_CCM - GetSet Converter Channel Map

Bits	Reset	Description
07:04	0h	Converter Channel (CC):
03:00	0h	Slot (SN):

73ChF3Ch SETGET_STRID - SetGet WIDI Stream ID

Bits	Reset	Description
07:00	00h	**WDSTRMID (WDSTRMID):** Represents the SSID that will go in the TS HEADER of PES and DATA packets

740hF40h SETGET PTS Offset Byte0

Bits	Reset	Description
07:00	00h	**PTS B0(PTSB0):** Bits 7:0 of the 32 bit 2's complement offset value. Should be added to the PTS value before sending to the circular buffer in the PES packet. Get verb will give the 32 bits value

741hF41h SETGET PTS Offset Byte1

Bits	Reset	Description
07:00	00h	**PTS B1(PTSB1):** Bits 15:8 of the 32 bit 2's complement offset value. Should be added to the PTS value before sending to the circular buffer in the PES packet. Get verb will give the 32 bits value

742hF42h SETGET PTS Offset Byte2

Bits	Reset	Description
07:00	00h	**PTS B2(PTSB2):** Bits 23:16 of the 32 bit 2's complement offset value. Should be added to the PTS value before sending to the circular buffer in the PES packet. Get verb will give the 32 bits value

743hF43h SETGET PTS Offset Byte3

Bits	Reset	Description
07:00	00h	**PTS B3(PTSB3):** Bits 31:24 of the 32 bit 2's complement offset value. Should be added to the PTS value before sending to the circular buffer in the PES packet. Get verb will give the 32 bits value

750hF50h SETGET Widi Buffer Base Address Byte0

Bits	Reset	Description
07:00	00h	**BLADDR Byte0(BLADDR0):** Bits 7: 0 of the 32 bits of the lower base address of the widi circular buffer. This is a physical address of the Stolen memory. Address is cacheline aligned.

751hF51h SETGET Widi Buffer Base Address Byte1

Bits	Reset	Description
07:00	00h	**BLADDR Byte1(BLADDR1):** Bits 15: 8 of the 32 bits of the lower base address of the widi circular buffer. This is a physical address of the Stolen memory. Address is cacheline aligned.

752hF52h SETGET Widi Buffer Base Address Byte2

Bits	Reset	Description
07:00	00h	**BLADDR Byte2(BLADDR2):** Bits 23:16 of the 32 bits of the lower base address of the widi circular buffer. This is a physical address of the Stolen memory. Address is cacheline aligned.

753hF53h SETGET Widi Buffer Base Address Byte3

Bits	Reset	Description
07:00	00h	**BLADDR Byte3(BLADDR3):** Bits 31:24 of the 32 bits of the lower base address of the widi circular buffer. This is a physical address of the Stolen memory. Address is cacheline aligned.

754hF54h SETGET Widi Buffer Upper Base Address Byte0

Bits	Reset	Description
07:00	00h	**BUADDR Byte0(BUADDR0):** Bits 7: 0 of the 32 bits of the upper base address of the widi circular buffer. This is a physical address of the Stolen memory. Address is cacheline aligned.

755hF55h SETGET Widi Buffer Upper Base Address Byte1

Bits	Reset	Description
07:00	00h	**BUADDR Byte1(BUADDR1):** Bits 15:8 of the 32 bits of the upper base address of the widi circular buffer. This is a physical address of the Stolen memory. Address is cacheline aligned.

756hF56h SETGET Widi Buffer Upper Base Address Byte2

Bits	Reset	Description
07:00	00h	**BUADDR Byte2(BUADDR2):** Bits 23:16 of the 32 bits of the Upper base address of the widi circular buffer. This is a physical address of the Stolen memory. Address is cacheline aligned.

757hF57h SETGET Widi Buffer Upper Base Address Byte3

Bits	Reset	Description
07:00	00h	**BUADDR Byte3(BUADDR3):** Bits 31:24 of the 32 bits of the Upper base address of the widi circular buffer. This is a physical address of the Stolen memory. Address is cacheline aligned.

760hF60h SETGET Widi Buffer Size Byte0

Bits	Reset	Description
07:00	00h	**BSIZE Byte0(BSIZE0):** Bits 7: 0 of the 32 bits of the size of the widi circular buffer. Size is in number of cachelines. It should be multiple of 3 Cachelines. Total buffer size cannot exceed more than 4MBytes For Wigig mode this represents the number of TFDs in the buffer.

761hF61h SETGET Widi Buffer Size Byte1

Bits	Reset	Description
07:00	00h	**BSIZE Byte1(BSIZE1):** Bits 15:8 of the 32 bits of the size of the widi circular buffer. Size is in number of cachelines. It should be multiple of 3 Cachelines. Total buffer size cannot exceed more than 4MBytes. For Wigig mode this represents the number of TFDs in the buffer.

762hF62h SETGET Widi Buffer Size Byte2

Bits	Reset	Description
07:00	00h	**BSIZE Byte2(BSIZE2):** Bits 23:16 of the 32 bits of the size of the widi circular buffer. Size is in number of cachelines. It should be multiple of 3 Cachelines. Total buffer size cannot exceed more than 4MBytes For Wigig mode this represents the number of TFDs in the buffer.

763hF63h SETGET Widi Buffer Size Byte3

Bits	Reset	Description
07:00	00h	**BSIZE Byte3(BSIZE3):** Bits 31:24 of the 32 bits of the size of the widi circular buffer. Size is in number of cachelines. It should be multiple of 3 Cachelines. Total buffer size cannot exceed more than 4MBytes. For Wigig mode this represents the number of TFDs in the buffer.

764hF64h SETGET WDE TFD size

764h: SET WDE TFD Size

Bits	Default	Description
07:04	00000	Reserved
03	0	This is an enable. If set, the tfd size is value is taken from here. Otherwise the tfd size is taken from WD_CNTRL register of MMIO.
02:00	000	This is TFD size. Valid only for WiGig mode. The total circular buffer should be a multiple of this number. 000b 512B [Default] default size of TFD is 512B 001b 1024B TFD Size is 1024B 010b 2048B TFD Size is 2048B 011b 4096B TFD Size is 4096B 100b 8192B TFD Size is 8192B Others: Reserved

F64h: GET WDE TFD Size

Bits	Default	Description
31:04	0	Reserved
03	0	This is an enable. If set, the tfd size is value is taken from here. Otherwise the tfd size is taken from WD_CNTRL register of MMIO.
02:00	000	This is TFD size. Valid only for WiGig mode. The total circular buffer should be a multiple of this number. 000b 512B [Default] default size of TFD is 512B 001b 1024B TFD Size is 1024B 010b 2048B TFD Size is 2048B 011b 4096B TFD Size is 4096B 100b 8192B TFD Size is 8192B Others: Reserved

770hF70h SETGET Widi Packet ID

Bits	Reset	Description
07:00	00h	**PID byte0 (PID0):**This goes in the bits 7:0 of the PID.

771hF71h SETGET Widi Packet ID

Bits	Reset	Description
07:05	00h	**Reserved:** MBZ
04:00	00h	**PID byte1**These bits are bits 12:8 of the PID

780h SET WFAWDE Header

780h: SET WFA Header (WFA mode)

Bits	Reset	Description
07:00	00h	**WFA HB0(WFAHB0):** WFA Header Byte zero. Valid only in WFA mode. Otherwise behavior is unexpected

F80h: GET WFA Header (WFA mode)

Bits	Description
31:24	**WFAHB3:** Header Byte 3. Valid only when WFA mode is enabled
23:16	**WFAHB2:** Header Byte 2. Valid only when WFA mode is enabled
15:08	**WFAHB1:** Header Byte 1. Valid only when WFA mode is enabled
07:00	**WFAHB0:** Header Byte 0. Valid only when WFA mode is enabled

780h: SET WDE Header (Wigig Mode)

Bits	Reset	Description
07:04	00h	Format Type (FT): Indicates the format to use. These are the CT bits as defined in Display Port 1.2. The most common value will be 0001 => PCM.
03	0	High Bit Rate (HBR): Indicates whether the stream is HBR or not
02:00	000	Channel Count (CC): Indicates how many channels there are. The encoding is as follows: 000 Mono 001 2 Channel Stereo

Bits	Reset	Description
		010 3 Channels 011 4 Channels 100 5 Channels 101 6 Channels 110 7 Channels 111 8 Channels

F80h: GET WDE Header (Wigig Mode)

Bits	Description
31:24	Reserved
23:16	Channel Allocation (CA): How are the channels allocated (L/R/etc.)
15	Reserved
14:12	Sampling Frequency (SF): Indicates the sampling frequency. The encoding is as follows: 000 - Reserved 001 - 32KHz 010 - 44.1KHz 011 - 48KHz 100 - 88.2KHz 101 - 96KHz 110 - 176.4KHz 111 - 192KHz
11:10	Reserved
09:08	Sampling length (SL): Indicates the sampling length. The encoding is as follows: 00 - Reserved (or 32 bits) 01 - 16bit 10 - 20bit 11 - 24bit
07:04	Format Type (FT): Indicates the format to use. These are the CT bits as defined in Display Port 1.2. The most common value will be 0001 => PCM.
03	High Bit Rate (HBR): Indicates whether the stream is HBR or not
02:00	Channel Count (CC): Indicates how many channels there are. The encoding is as

Bits	Description
	follows: 000 - Mono 001- 2 Channel Stereo 010 - 3 Channels 011 - 4 Channels 100 - 5 Channels 101 - 6 Channels 110 - 7 Channels 111 - 8 Channels

781h SET WFAWDE Header

781h: SET WFA Header (WFA Mode)

Bits	Reset	Description
07:00	00h	**WFA HB1(WFAHB1):** WFA Header Byte One. Valid only in WFA mode. Otherwise behavior is unexpected

F81h: GET WFA Header (WFA Mode)

Bits	Description
31:24	**WFAHB3:** Header Byte 3. Valid only when WFA mode is enabled
23:16	**WFAHB2:** Header Byte 2. Valid only when WFA mode is enabled
15:08	**WFAHB1:** Header Byte 1. Valid only when WFA mode is enabled
07:00	**WFAHB0:** Header Byte 0. Valid only when WFA mode is enabled

781h: SET WDE Header (Wigig Mode)

Bits	Reset	Description
07	00h	Reserved
06:04	000	Sampling Frequency (SF): Indicates the sampling frequency. The encoding is as follows: 000 - Reserved 001 - 32KHz 010 - 44.1KHz

Bits	Reset	Description
		011 - 48KHz 100 - 88.2KHz 101 - 96KHz 110 - 176.4KHz 111 - 192KHz
03:02	00	Reserved
01:00	00	Sampling length (SL): Indicates the sampling length. The encoding is as follows: 00 - Reserved (32 bits) 01 - 16bit 10 - 20bit 11 - 24bit

F81h: GET WDE Header (Wigig Mode)

Bits	Description
31:24	Reserved
23:16	Channel Allocation (CA): How are the channels allocated (L/R/etc.)
15	Reserved
14:12	Sampling Frequency (SF): Indicates the sampling frequency. The encoding is as follows: 000 - Reserved 001 - 32KHz 010 - 44.1KHz 011 - 48KHz 100 - 88.2KHz 101 - 96KHz 110 - 176.4KHz 111 - 192KHz
11:10	Reserved
09:08	Sampling length (SL): Indicates the sampling length. The encoding is as follows: 00 - Reserved (32 bits)

Bits	Description
	01 - 16bit 10 - 20bit 11 - 24bit
07:04	Format Type (FT): Indicates the format to use. These are the CT bits as defined in Display Port 1.2. The most common value will be 0001 => PCM.
03	High Bit Rate (HBR): Indicates whether the stream is HBR or not
02:00	Channel Count (CC): Indicates how many channels there are. The encoding is as follows: 000 - Mono 001- 2 Channel Stereo 010 - 3 Channels 011 - 4 Channels 100 - 5 Channels 101 - 6 Channels 110 - 7 Channels 111 - 8 Channels

782h SET WFAWDE Header

782h: GET WFA Header (WFA Mode)

Bits	Reset	Description
07:00	00h	**WFA HB2(WFAHB2):** WFA Header Byte two. Valid only in WFA mode. Otherwise behavior is unexpected

F82h: GET WFA Header (WFA Mode)

Bits	Description
31:24	**WFAHB3:** Header Byte 3. Valid only when WFA mode is enabled
23:16	**WFAHB2:** Header Byte 2. Valid only when WFA mode is enabled
15:08	**WFAHB1:** Header Byte 1. Valid only when WFA mode is enabled
07:00	**WFAHB0:** Header Byte 0. Valid only when WFA mode is enabled

782h: GET WDE Header (WiGig Mode)

Bits	Reset	Description
07:00	00h	Channel Allocation (CA): How are the channels allocated (L/R/etc.)

F82h: GET WDE Header (Wigig Mode)

Bits	Description
31:24	Reserved
23:16	Channel Allocation (CA): How are the channels allocated (L/R/etc.)
15	Reserved
14:12	Sampling Frequency (SF): Indicates the sampling frequency. The encoding is as follows: 000 - Reserved 001 - 32KHz 010 - 44.1KHz 011 - 48KHz 100 - 88.2KHz 101 - 96KHz 110 - 176.4KHz 111 - 192KHz
11:10	Reserved
09:08	Sampling length (SL): Indicates the sampling length. The encoding is as follows: 00 - Reserved (or 32 bits) 01 - 16bit 10 - 20bit 11 - 24bit
07:04	Format Type (FT): Indicates the format to use. These are the CT bits as defined in Display Port 1.2. The most common value will be 0001 => PCM.
03	High Bit Rate (HBR): Indicates whether the stream is HBR or not
02:00	Channel Count (CC): Indicates how many channels there are. The encoding is as follows: 000 - Mono 001- 2 Channel Stereo

Bits	Description
	010 - 3 Channels 011 - 4 Channels 100 - 5 Channels 101 - 6 Channels 110 - 7 Channels 111 - 8 Channels

783h SET WFA Header

Bits	Reset	Description
07:00	00h	**WFA HB3(WFAHB3):** WFA Header Byte Three. Valid only in WFA mode. Otherwise behavior is unexpected

F83h: GET WFA Header

Bits	Description
31:24	**WFAHB3:** Header Byte 3. Valid only when WFA mode is enabled
23:16	**WFAHB2:** Header Byte 2. Valid only when WFA mode is enabled
15:08	**WFAHB1:** Header Byte 1. Valid only when WFA mode is enabled
07:00	**WFAHB0:** Header Byte 0. Valid only when WFA mode is enabled

784hF84h GETSET WDE Program Number Verb

Bits	Bits	Description
07:00	0	Program Number

785hF85h GETSET WDE STD Bit Rate Verb

785h: SET WDE STD Bit Rate

Bits	Bits	Description
07:00	0	Lower byte of WDE STD Bit Rate

F85h: GET WDE STD Bit Rate

Bits	Bits	Description
31:16	0	Reserved
15:08	0	Upper byte of WDE STD Bit Rate
07:00	0	Lower byte of WDE STD Bit Rate

786hF86h GETSET WDE STD Bit Rate Verb

786h: SET WDE STD Bit Rate

Bits	Bits	Description
07:00	0	Upper byte of WDE STD Bit Rate

F86h: GET WDE STD Bit Rate

Bits	Bits	Description
31:16	0	Reserved
15:08	0	Upper byte of WDE STD Bit Rate
07:00	0	Lower byte of WDE STD Bit Rate

787hF87h GETSET WDE PKT PER FRM Verb

787h: SET WDE PKT PER FRM

Bits	Default	Description
07:05	000	Reserved
04:00	00111	Valid only for WiGig mode. Packets per frame. Number of samples per packet are shown below. Legal values are: 00000: 1 packet = 192 samples per packet 00001: 2 packets = 96 samples per packet 00011: 4 packets = 48 samples per packet 00101: 6 packets = 32 samples per packet 00111: 8 packets = 24 samples per packet 01011: 12 packets = 16 samples per packet

Bits	Default	Description
		01111: 16 packets = 12 samples per packet 10011: 24 packets = 8 samples per packet 11111: 32 packets = 6 samples per packet

F87h: GET WDE PKT PER FRM

Bits	Default	Description
31:05	0	Reserved
04:00	00111	Packets per frame. Valid for Wigig mode only.

788hF88h GETSET Wireless Widget Mode

788h: SET Wireless Widget Mode

Bits	Default	Description
07:02	000	Reserved
01:00	00	Mode Selection 00: WiGig Mode 01: Widi Mode 10: WFA Mode 11: Reserved

F88h: GET Wireless Widget Mode Response data format

Bits	Default	Description
31:02	0	Reserved
01:00	00	Mode Selection 00: WiGig Mode 01: Widi Mode 10: WFA Mode 11: Reserved

North Display Engine Registers

This chapter contains the register descriptions for the display engine portion of a family of graphics devices.

These registers vary by devices within the family of devices, so special attention needs to be paid to which devices use which registers and register fields.

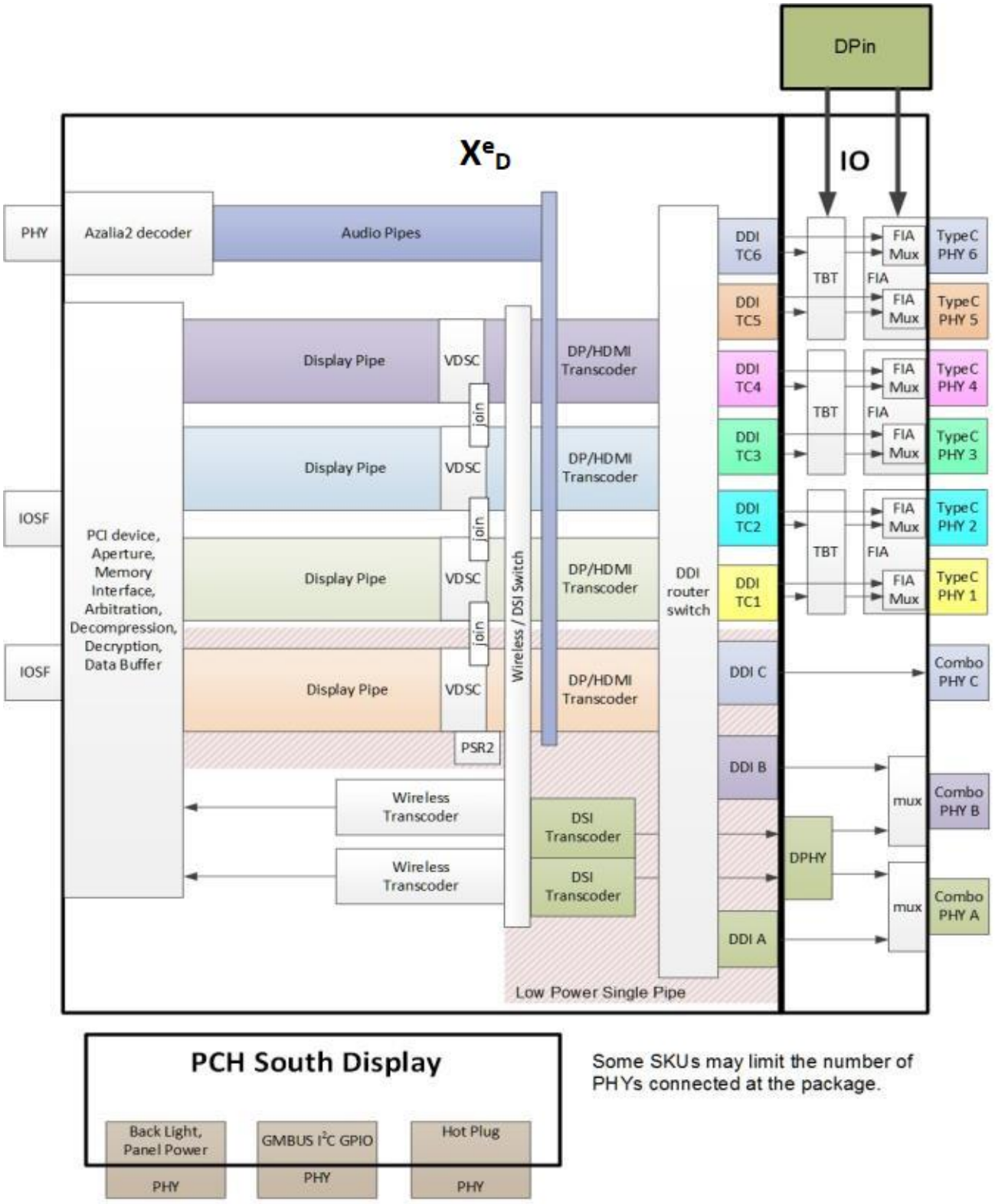
Different devices within the family may add, modify, or delete registers or register fields relative to another device in the same family based on the supported functions of that device.

Xe_D Overview

Genesis

This project uses Xe_D.

Block Diagram



The front end of the display contains the pipes. The pipes connect to the transcoders. The transcoders, except for wireless, connect to the DDIs to drive the IO/PHY. Wireless writes back to memory. DPin allows external graphics DisplayPort output to be routed to typeC and thunderbolt.

General Capabilities

Four simultaneous displays (pipes A, B, C, D)

- 7 planes and 1 cursor per pipe
- Audio streams per pipe to go to external ports
- HDR support for 3 planes per pipe
- VESA DSC compression support for all pipes
- Post-DSC joining to connect 2 adjacent pipes for resolutions that require more bandwidth than one pipe can support
- Pipe A optimized for low power

External display connections

- 2 wireless
- 3 combo ports (DisplayPort, HDMI, or eDP)
- 6 USB Type C ports (DisplayPort alternate mode, DisplayPort over thunderbolt, native DisplayPort on legacy connector, native HDMI on legacy connector)
- Hotplug for type C
- AUX channels for DisplayPorts
- Multi-stream support for DisplayPorts
- Genlock support on all combo ports at display IP level but not at SoC level. Feature cannot be used.

Embedded/local display connections

- 3 combo ports that can support eDP, DSI, and external ports, but only one low power optimized port at a time
- 1 x8 MIPI DSI or 2 x4 MIPI DSI (shared pins on DDIA and DDIB)
- Combo IO shares pins between DSI0 and DDIA and between DSI1 and DDIB
- AUX channel for eDP
- PSR1, PSR2, and MSO (multi-segmented operation, chip on glass) for eDP

PCH

- Backlight modulation PWM, panel power sequencing, GMBUS I2C, non-typeC hot plug detection

Port Configurations

Port Type	# of ports[1]	Protocol	Maximum Speed	Maximum Resolution[2]
Combo	1 port 8 lanes or 2 ports 4 lanes	MIPI DSI 1.3	2.5 Gbps	5k 60Hz compressed
"	3[3]	eDP 1.4b	HBR3 8.1 Gbps	5k 60Hz 24bpp
"	3	DP 1.4a	HBR2 5.4 Gbps	4k 60Hz 24bpp
"	3	HDMI 1.4	2.97 Gbps	4k 30Hz 24bpp
"	3	HDMI 2.0b	5.94 Gbps	4k 60Hz 24bpp
USB Type-C[4]	6	DP 1.4a	HBR3 8.1 Gbps	8k 60Hz compressed, 5k 120Hz compressed
"	6	HDMI 1.4	2.97 Gbps	4k 30Hz 24bpp
"	6	HDMI 2.0b	5.94 Gbps	4k 60Hz 24bpp
Capture	2	N/A	N/A	Single display 4k 60Hz 30bpp Dual display 2560x1600 60Hz 30bpp

[1] Many SKUs will limit the number of ports connected at the package. Only one power optimized port for eDP.

[2] Resolution restrictions

- Resolutions supported, but not a guarantee of user experience.
- Thermal constraints may apply.
- Multi-display resolution support limited.
- Subject to memory bandwidth availability. Software configuration constrained to fit available bandwidth.
- These are single cable panel (non-tiled panel) resolutions. Panels with two cables may support higher resolution by using two ports.
- Higher bits per color may be possible at lower resolutions.
- Some resolutions require compression (VDSC). VDSC input and output bits per color are limited.
- Supports other resolutions and bits per color that fit the same bandwidth and size constraints.
- Software configuration constrained to fit available bandwidth with current memory configuration.

[3] Only one power optimized port for eDP.

[4] USB type-C PHY can support DP alternate mode, DP tunneled over thunderbolt, DP with legacy connector, or HDMI with legacy connector.

Multiple Display Resolution Support

This table gives the required resolution support. Actual results may have higher resolution, depending on exact details of the configuration, such as the number of planes enabled and pixel formats.

Number of simultaneous displays	Maximum Common Resolution
1	8k 60Hz HDR or 5k 120Hz HDR
2	5k 60Hz SDR or 4k 60Hz HDR
3 or 4	4k 60Hz HDR

All subject to resolution restrictions noted under the Port Configurations table.

HDR (High Dynamic Range) increases memory bandwidth compared to standard dynamic range (SDR).

Software does not have explicit restrictions for multiple display cases, just constraints to fit available bandwidth with current memory configuration.

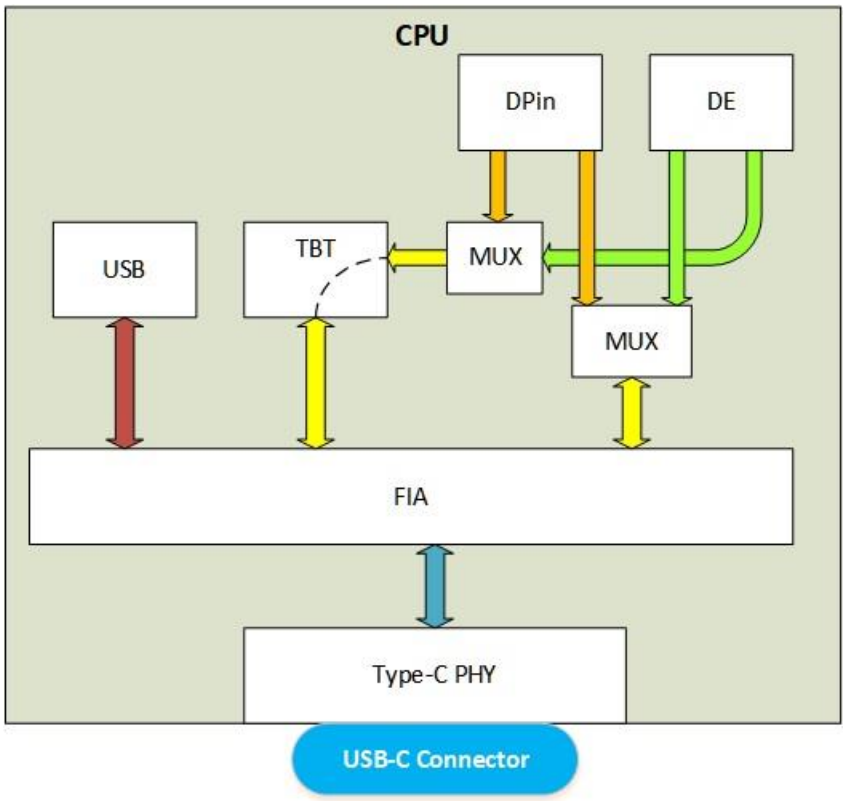
See the Resolution Support page for detailed restrictions and calculations.

Port Availability

Port	Usage	Capability	Alternate Names
DDI A	Combo port A	eDP, DP, HDMI, MIPI DSI 0	DDIA, Port A, Combo Port A, MIPIA, MIPI0
DDI B	Combo port B	eDP, DP, HDMI, MIPI DSI 1	DDIB, Port B, Combo Port B, MIPIB, MIPI1
DDI C	Combo port C	eDP, DP, HDMI	DDIC, Port C, Combo Port C
TC 1	Type C port 1	DP alternate mode, DP over Thunderbolt, native/legacy DP, native/legacy HDMI	Type C Port 1, TC1, USBC1
TC 2	Type C port 2	DP alternate mode, DP over Thunderbolt, native/legacy DP, native/legacy HDMI	Type C Port 2, TC2, USBC2
TC 3	Type C port 3	DP alternate mode, DP over Thunderbolt, native/legacy DP, native/legacy HDMI	Type C Port 3, TC3, USBC3
TC 4	Type C port 4	DP alternate mode, DP over Thunderbolt, native/legacy DP, native/legacy HDMI	Type C Port 4, TC4, USBC4
TC 5	Type C port 5	DP alternate mode, DP over Thunderbolt, native/legacy DP, native/legacy HDMI	Type C Port 5, TC5, USBC5
TC 6	Type C port 6	DP alternate mode, DP over Thunderbolt, native/legacy DP, native/legacy HDMI	Type C Port 6, TC6, USBC6

Some SKUs will limit which ports are connected in the die and package. Software should rely on hotplug to determine which ports are actually available.

USB TypeC



The USB type C ports are integrated into the CPU. Each type C port is shared between USB, thunderbolt (TBT), DPin, and display engine.

FIA selects between the controllers and handles cable orientation swapping. Multiple controllers can share the connector simultaneously, but with reduced number of lanes for each.

Display engine has 4 output types for type C

- DisplayPort alternate mode: Display engine sends DisplayPort data to FIA to be muxed to the correct PHY lanes. DP can use 2 or 4 of the data lanes, depending on whether the connector is being shared with another controller.
- DisplayPort over thunderbolt: Display engine sends DisplayPort data to the thunderbolt controller, which combines up to 2 DisplayPort streams with other data streams and sends it to FIA to be muxed to the correct PHY lanes.
- Native HDMI: This is used for a legacy HDMI connector on a type C port. Display engine sends data to FIA to be passed through to the correct PHY lanes. There is no muxing with other controllers.
- Native DisplayPort: This is used for a legacy DisplayPort connector on a type C port. Display engine sends data to FIA to be passed through to the correct PHY lanes. There is no muxing with other controllers.

The type C sub-system IOM and power delivery controller determine the ownership of the type C connector. They interpret type C hotplug events and decide whether the hotplug should be routed to display engine or to one of the other controllers, and which lanes are available for DP alternate mode.

If the hotplug is directed to display engine it can take 3 paths

- Legacy hotplug interrupt: This interrupt comes through the south display hotplug detection and interrupt. This is used for the native HDMI or DP case with a legacy connector on a type C port.
- Thunderbolt hotplug interrupt: This interrupt comes through the north display thunderbolt hotplug control and interrupt. This is used for DP over thunderbolt.
- DP alternate hotplug interrupt: This interrupt comes through the north display typeC hotplug control and interrupt. This is used for DP alternate mode.

Depending on the type of interrupt, software needs to configure the display output to go to the correct destination.

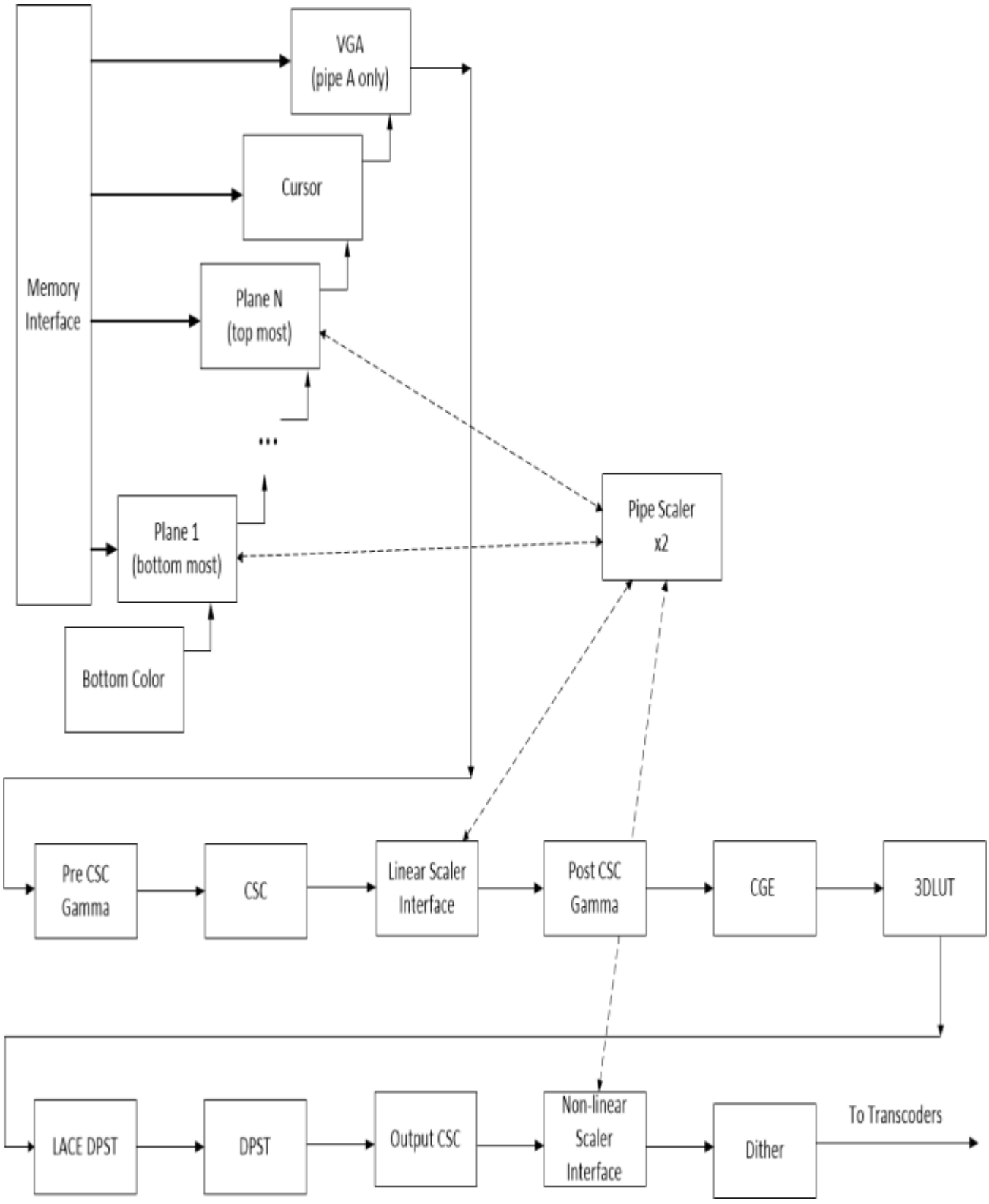
Power Wells

Display engine functions are spread across several different power gated (PG) wells that can be shut down to optimize power for different configurations.

Hardware can dynamically control some PGs for display power states. That is enabled through the DC State programming.

Refer to Sequences for Power Wells for more details on the functions in each power well and the sequences for enabling and disabling power.

Pipes


VGA
(pipe A only)
Cursor
Memory
Interface
Plane N
(top most)
...
Plane 1
(bottom most)
Bottom Color
Pipe Scaler
x2
Pre CSC
Gamma
CSC
Linear Scaler
Interface
Post CSC
Gamma
CGE
3DLUT
LACE DPST
DPST
Output CSC
Non-linear
Scaler
Interface
Dither
To Transcoders

The display pipes contain the planes, blending, color measurement and adjustment, scaling, and dithering.

FBC and LDPST are supported only on pipe A.

3D LUT is supported only on pipes A and B.

The planes read data from memory, format it into pixels, and can apply color correction and scaling.

Each display pipe has 7 planes and a cursor.

The plane blending combines the output from all the planes following a fixed Z-order. Plane 1 is the bottom most plane and higher numbered planes stack on top of it. Cursor goes ontop of all the planes.

Planes 1-3 support HDR. Planes 4-7 support SDR. Plane details are in the Universal Plane and Plane Capability and Interoperability sections.

The background color that is seen under the bottom most plane is programmable.

The blended pixels pass through several color correction functions and then output to the transcoders.

The pipes each have two pipe scalers. Each pipe scaler can be assigned to scale a plane or scale the blended output.

Transcoders

The display transcoders contain the timing generators, port encoders, DisplayPort transport control, Audio/Video mixers, Video Data Island Packet mixers, and Panel Self Refresh controllers.

Except for the WD (Wireless Display, Capture) transcoders, the transcoders convert pixel data to the appropriate format for the port, mix in data islands and audio, and output the data to the DDIs.

The WD transcoders convert pixel data to the appropriate format, then write it to system memory.

Transcoder A-D can support DP or HDMI, with audio.

Transcoders DSI* only support MIPI DSI.

Only transcoder A supports PSR2.

Only transcoder A supports eDP CoG.

The transcoders, except for wireless transcoders, can make use of VDSC compression.

Audio

The Azalia2 interface provides data to the audio codec.

The audio codec connects to the Audio/Video mixers in the transcoders.

The audio codec can also write data back to system memory for wireless audio.

DDIs (Digital Display Interfaces)

The DDIs contain port logic to interface to the DDI physical (PHY) layer.

The DDIs going to the combo PHY support lane reversal where the internal lane to package lane mapping is swapped. The DDIs going to the type C PHY rely on FIA to do lane reversal.

A Combo PHY capable of supporting DDI or MIPI DSI is attached to DDI A and DDI B. It can select between MIPI DSI output or DDI output. Dynamic switching between the two configurations is not supported.

Some of the DDI GPIO and HPD pins are reused for MIPI DSI panel controls.

DDI Lane Mapping:

Package Pin	DisplayPort Non-Reversed	DisplayPort Reversed	HDMI/DVI Non-Reversed	HDMI/DVI Reversed
DDI data 3	Main Link Lane 3	Main Link Lane 0	TMDS Clock	TMDS Data2
DDI data 2	Main Link Lane 2	Main Link Lane 1	TMDS Data0	TMDS Data1
DDI data 1	Main Link Lane 1	Main Link Lane 2	TMDS Data1	TMDS Data0
DDI data 0	Main Link Lane 0	Main Link Lane 3	TMDS Data2	TMDS Clock

MIPI DSI Lane Mapping:

Package Pin	MIPI DSI Connection
DDIA data 3	DSI 0 Data 3
DDIA data 2	DSI 0 Clock
DDIA data 1	DSI 0 Data 2
DDIA data 0	DSI 0 Data 1
DDIA AUX	DSI 0 Data 0
DDIB data 3	DSI 1 Data 3
DDIB data 2	DSI 1 Clock
DDIB data 1	DSI 1 Data 2
DDIB data 0	DSI 1 Data 1
DDIB AUX	DSI 1 Data 0

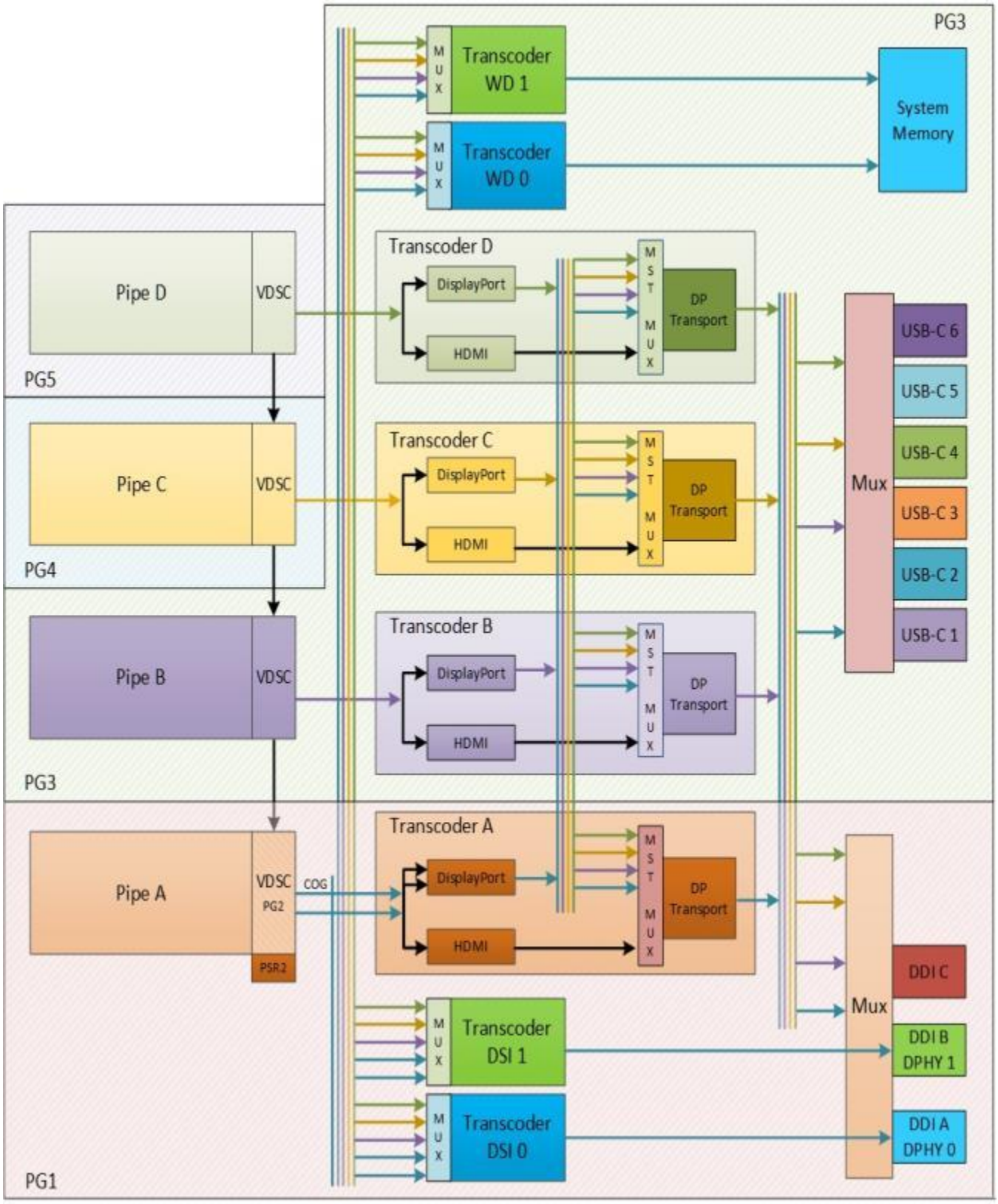
DDI and MIPI DSI Pin Sharing:

DDI pins	Non-MIPI DSI board	MIPI DSI 8 lane board	MIPI DSI 4 lane board
DDIA data and AUX	DDIA embedded DisplayPort	MIPI DSI 0	MIPI DSI 0
DDIB data and AUX	DDIB DisplayPort, HDMI, or DVI	MIPI DSI 1	DDIB DisplayPort, HDMI, or DVI
DDIA HPD	EDP HPD	MIPI panel RESET	MIPI panel RESET
DDIA DDC clock	Unused	MIPI panel AVEE	MIPI panel AVEE
DDIA DDC data	Unused	MIPI panel VIO	MIPI panel VIO

MIPI DSI Panel Pins Mapping to Display GPIO Pins

MIPI Panel Pin	GPIO Pin	Controls
RESET	DDSP_HPD_A	South display SHOTPLUG_CTL_DDI DDIA HPD Output Data
AVEE	DDPA_CTRLCLK	South display GPIO_CTL_1 (DDIA) clock
VIO	DDPA_CTRLDATA	South display GPIO_CTL_1 (DDIA) data
AVDD	L_VDDEN	South display panel power sequencing power enable
BKLTEN	L_BKLTEN	South display panel power sequencing backlight enable
PWM	L_BKLTCTL	South display backlight PWM output
DSI 0 TE	Utility Pin	UTIL_PIN_CTL
DSI 1 TE	DSI_DE_TE_2	Enabled automatically by DSI transcoder 1

Pipe to Transcoder to DDI Mappings



Twin modes are not supported. A pipe cannot drive more than one display.

A pipe cannot connect to more than one transcoder simultaneously.

With DisplayPort multistream it is possible to have multiple pipes/transcoders driving a single DDI. Multistream requires at transcoder to be enabled to drive the DP transport, even if the associated pipe streams are disabled.

Transcoders WD* and DSI* do not support multistream.

Two pipes can drive a single display through two transcoders and two DDIs that are joined in the panel (tiled display) or by using DSC compression and joining the pipe outputs to go to a single transcoder and DDI.

Transcoders A-D are tied to the respective Pipes A-D. Each pipe output can go to either the respective transcoder or to transcoders WD* or DSI*.

Pipe A VDSC outputs 2 streams to split the display for eDP Chip on Glass or MSO.

Transcoders A-D can connect to any DDI. Transcoders WD* can only go to system memory. Transcoders DSI* can connect only to DDIA and DDIB.

Mode Set

A mode set sequence is the programming sequence that must be followed when enabling or disabling output to a display. There are several different mode set sequences documented in the following sections. The sequence to use depends on which type of port is being enabled or disabled.

Sequences for Intel MIPI DSI IP

This folder contains the sequences for the Intel MIPI DSI IP.

Sequences for MIPI DSI

These sequences are used to enable and/or disable the new MIPI DSI ports and associated functions. Unless the sequence specifically talks about multi-port requirements (i.e. Dual Link mode) these sequences are for a given DSI port (i.e. either MIPI DSI port can be programmed with the below sequences).

Note that some of the registers outside of the DSI register space still refer to the old naming convention of MIPIA and MIPIC. The DSI ports are no longer associated with specific pipes and, so are referred to as DSI0 and DSI1. The mapping between the old naming convention and the new is: DSI0 = MIPIA; DSI1 = MIPIC

DSI Transcoder Enable Sequence

1. Initialize display (Sequences to Initialize Display)
2. Enable Power Wells
    a. If Pipe A is not being used to drive MIPI DSI, then the power wells for those pipes will also need to be enabled (see Sequences for Power Wells)
3. Enable IO Power
    a. Configure the mode of operation of the IO for MIPI DSI (via **DSI_IO_MODECTL**)
        i. Note that setting the IO mode will assert the HS Word clock (a.k.a. the 1X clock) request to the IO. This will enable the IO to forward the HS Word clock back to the Display Engine
    b. Enable **PWR_WELL_CTL_DDI** Power Requests for the DDI that will be used (DSI0 uses DDI A; DSI1 uses DDI B)
        i. Note that the DPHY automatically powers up the AUX lane (DSI Data Lane 0) when it receives the power request with the IO mode set to MIPI DSI. Do **not** try to power on the AUX channel using the PWR_WELL_CTL_AUX register
    c. Wait for PWR_WELL_CTL_DDI Power Status = Enabled
        i. Timeout after 20us
4. Enable PLL
    a. If PLL is not already enabled, follow Port Clock Programming enable sequence
    b. Configure PLL to port mapping to direct the PLL output to the DDI
    c. If not already done, gate the DDI clock associated with port being enabled (DPCLKA_CFGCR0)
        i. If DSI 0 is being used, then set the DDIA Clock Off bit of DPCLKA_CFGCR0
        ii. If DSI 1 is being used, then set the DDIB Clock Off bit of DPCLKA_CFGCR0
5. Configure and Enable DPHY
    a. Configure PORT_CL_DW10 Static Power Down to power up all lanes of the DDI that will be used
        i. See table below for the mapping of the DSI lanes to the Static Power Down bits
    b. Configure the Lane sequencing of the Combo-PHY transmitters for MIPI DSI HS operation

i. Set the Loadgen Select (**PORT_TX_DW4**) for Transmit lanes 0, 1, and 3 to be secondary to the Aux lane (see Lane Sequence Programming table below)

1. Note that Transmit lane 2 is the DSI Clock lane and it must be a primary lane (i.e. it cannot be secondary to any other lane) for proper operation

ii. Set the Latency Optimization (**PORT_TX_DW2** and **PORT_PCS_DW1**) for the Aux lane and Transmit lanes (see Lane Sequence Programming table below for values)

1. The DSI Clock lane operates independent of the Data lanes, so there is no latency optimization needed

c. Configure voltage swing and skew. Refer to DDI Buffer section for the programming not already specified above (i.e. the Loadgen Select and Latency Optimization should be used from here, all other DDI buffer programming can be taken from the DDI Buffer section).
    i. Note that the voltage swing and skew programming is for the HS data only. The HW automatically handles the swing and skew for the LP data
d. Setup D-PHY timings
    i. If SW is going to override any of the D-PHY timing parameters, then configure DPHY_CLK_TIMING_PARAM, DPHY_DATA_TIMING_PARAM, or DPHY_TA_TIMING_PARAM with the desired D-PHY timings. SW must also program the equivalent DSI_*_TIMING_PARAM registers with the appropriate overrides (see following step)
e. Program Tx Equalization for the port (**PORT_TX_DW0**), if necessary (determined by VBT)
    i. Note that the programming should be applied across all lanes of the PHY, so Software can use the TXGRP offset to program all lanes at once.
f. If the link is operating at 1.5Gbps or below, then program the TX slew rate to 11b in **PORT_TX_DW1**, else program it to 0b0.
g. If the link is operating at 2.5Gbps, then program clock lane (lane 2) o_ldo_ref_sel_cri = 000011b in **PORT_TX_DW6_<lane 2 instance>**, else program it to 000000b.
h. Enable DDI Buffer and wait for DDI_BUF_CTL DDI Idle Status = 0b (Not Idle)
    i. Timeout after 500us

6. Enable DSI Transcoder
    a. Setup link initialization/wakeup timings
        i. Configure the DSI_T_INIT_PRIMARY with the Initialization time if different than the default 100us
        ii. Configure the DSI_T_WAKEUP with the T_{WAKEUP} timing parameter if different than the default 1ms
        iii. If Software overrode the transitional timing parameters for the clock or data lanes within the DPHY (i.e. DPHY_CLK_TIMING_PARAM/DPHY_DATA_TIMING_PARAM registers), then it must reflect those overrides within the DSI controller's timing parameter registers (**DSI_CLK_TIMING_PARAM**, or **DSI_DATA_TIMING_PARAM**)
    b. Enable and configure the **UTIL_PIN_CTL** register as an input, if enabling DSI port 0 to be operating in Command Mode and the TE events will be received out-of-band
        i. This IO is shared with Audio
    c. Enable and configure UTIL2_PIN_CTL as an input, if enabling DSI port 1 to be operating in Command Mode and the TE events will be received out-of-band
        i. This IO is shared with Genlock.
    d. Configure DSI timeouts - **DSI_HTX_TO**, **DSI_CALIB_TO**, **DSI_LRX_H_TO**, **DSI_PWAIT_TO**, and **DSI_TA_TO**.
        i. Note that the DSI_HTX_TO and DSI_CALIB_TO registers don't have to be programmed if Software is satisfied with the defaults.

ii. The "Programming Reference" section below gives some guidelines on how the DSI_TA_TO and DSI_LRX_H_TO registers can be programmed.

e. Configure **TRANS_DSI_FUNC_CONF**

f. Configure **TRANS_DDI_FUNC_CTL2** Port Sync Mode Enable, if both DSI transcoders are going to operate in Dual Link mode

i. See "Dual Link Sequence" below for details

g. Configure and enable **TRANS_DDI_FUNC_CTL**

h. Wait for DSI Link Ready (TRANS_DSI_FUNC_CONF)

i. Timeout after 2X the value programmed in the DSI_T_WAKEUP register + 500us

ii. Note that the timeout duration is to account for the possibility that the DSI Link is in ULPS. When exiting ULPS the controller will have to serialize the wakeup of the Clock Lane and Data Lanes.

7. Program Panel

a. Send a "*Set Maximum Return Packet Size*" DSI packet to the Panel. Refer to "Sending Commands to the Panel" section. This is a Short Packet format, only a Header credit will be needed to form the packet within the **DSI_CMD_TXHDR**register (with the Payload bit cleared). The packet is composed of:

i. Data Type: 0x37

ii. Virtual Channel: Panel specific

iii. Data: The Maximum Return Packet Size can be programmed for up to 32 bytes

b. Software will handle any additional panel setup here.

c. Software needs to ensure that all of the Panel programming has been dispatched before the Transcoder is enabled. To do this, the following sequence will need to be done:

i. Software will wait for all Header and Payload Credits to be released (DSI_CMD_TXCTL)

1. Timeout after 100us

ii. Build a NOP DCS command using the DSI_CMD_TXHDR register

1. The packet will use a Data Type of 0x05 (DCS Short Write - 0 Parameters)
2. The Word Count/Parameter bytes will be set to all zeroes (NOP DCS command)
3. Set the LPDT bit

iii. After loading the NOP with the DSI_CMD_TXHDR write, Software will wait for the following to be true:

1. Wait for all Header Credits to be released
    i. Timeout after 20us
2. Wait for LP Tx in Progress bit of the **DSI_LP_MSG** register to be cleared
    i. Timeout after 20us

iv. Software can now continue the enabling sequence

8. Enable Planes, Pipe, and Transcoder

a. Configure and enable planes (VGA or hires)

i. If VGA, clear VGA I/O register SR01 bit 5
ii. This can be done later, if desired

b. Enable Scaler, if needed (must be enabled for VGA)
c. Configure Transcoder Timings
    i. Make sure to program TRANS_VBLANK appropriately
        1. The Vertical Blank End should always be programmed to be equal to the Vertical Total programming of TRANS_VTOTAL
        2. If the Display Stream Buffer (DSB) is not enabled, then the Vertical Blank Start should be programmed to be equal to the Vertical Active programming of TRANS_VTOTAL
        3. If the DSB is enabled, then the Vertical Blank Start may need to be pushed out
    ii. If operating in Command Mode (a.k.a. DBI), then the Vertical and Horizontal totals should still be programmed even though the DSI transcoder is not truly maintaining these timings.
        1. Program the Horizontal Total to be Horizontal Active + 160
        2. Program the Vertical Total to include a Vertical blanking time that satisfies the following: ceiling(400us / Line Time). See the "**Determining Vertical Blank in DBI**" section within the "Transcoder DSI Function" page for programming details.
    iii. Software does not need to program the TRANS_*SYNC registers, if the DSI transcoder is operating in Command Mode
d. Configure and enable **TRANS_CONF**
    i. Wait for the Transcoder State to be enabled, if additional Panel Commands need to be sent at this time

9. Turn on the backlight, if necessary

The following table shows the mapping of the Static Power Down bits within the PORT_CL_DW10 register and the PWR_WELL_CTL_DDI register to the physical I/O lanes used for a DSI transcoder's link.

Phys Uni-Dir Lane 3 (Bit 3)	Phys Uni-Dir Lane 2 (Bit 2)	Phys Uni-Dir Lane 1 (Bit 1)	Phys Uni-Dir Lane 0 (Bit 0)	Phys Bi-Dir Lane (PWR_WELL_CTL_DDI)	Notes
MIPI DSI Usage					
Data Lane 3	**Clock Lane**	**Data Lane 2**	**Data Lane 1**	**Data Lane 0**	**Lane mapping**
OFF	**ON**	**OFF**	**OFF**	**ON**	**x1 DSI**
OFF	**ON**	**OFF**	**ON**	**ON**	**x2 DSI**
OFF	**ON**	**ON**	**ON**	**ON**	**x3 DSI**
ON	**ON**	**ON**	**ON**	**ON**	**x4 DSI**

Lane Sequence Programming

AFE Lane	DSI Lane	Latency Optimization				Load Generation Select	
		PORT_TX_DW2		PORT_PCS_DW1		PORT_TX_DW4	
		Bits	Value[1]	Bits	Value[2]	Bits	Value
Aux	Data Lane 0	[10:8]	'h5	[3:2]	'h0	[31]	0
0	Data Lane 1		'h5		'h1		1
1	Data Lane 2		'h5		'h1		1
2	**Clock Lane**		**'h0**		**'h1**		**0**
3	Data Lane 3		'h5		'h1		1

1. The Latency Optimization of the Clock Lane can be either left at it's default value ('h0) or programmed to the same value as the other lanes. If programmed with the same value as the other lanes, then the Group access can be used for PORT_TX_DW2 programming
2. The Group access register for PORT_PCS_DW1 can be used to program the non-Aux AFE lanes.

DSI Transcoder Disable Sequence

Note that this sequence is agnostic to whether the transcoder is operating in the Video (i.e. DPI) or Command (i.e. DBI) mode of operation.

1. Turn off Backlight, if desired
2. Disable Planes, Pipe, and Transcoder
    a. If VGA
        i. Set VGA I/O register SR01 bit 5 for screen off
        ii. Wait for 100us
    b. Disable planes (VGA or hires)
    c. Disable **TRANS_CONF**
    d. Wait for off status in TRANS_CONF
        i. Poll TRANS_CONF register or timeout after two frame times
    e. Send any necessary commands to the Panel now
    f. Make sure all commands have been delivered to the Panel
        i. Software will wait for all Header and Payload Credits to be released (**DSI_CMD_TXCTL**)
            1. Poll DSI_CMD_TXCTL or Timeout after 100us
        ii. Build a NOP DCS command using the DSI_CMD_TXHDR register
            1. The packet will use a Data Type of 0x05 (DCS Short Write - 0 Parameters)
            2. The Word Count/Parameter bytes will be set to all zeroes (NOP DCS command)
            3. Set the LPDT bit
        iii. After loading the NOP with the DSI_CMD_TXHDR write, Software will wait for the following to be true:
            1. Wait for all Header Credits to be released
                1. Poll DSI_CMD_TXCTL or Timeout after 20us
            2. Wait for LP Tx in Progress bit of the DSI_LP_MSG register to be cleared
                1. Poll DSI_LP_MSG or Timeout after 20us
    g. Put the Link in ULPS
        i. Configure ULPS Type and Entry in **DSI_LP_MSG**
        ii. Wait for the Link to enter ULPS (logged within the DSI_LP_MSG register) or an interrupt indicating ULPS entry is done
            1. Option to timeout after 10us
    h. Disable **TRANS_DDI_FUNC_CTL**
    i. Wait for 1us to allow the PPI lane enables to propagate to the DPHY
    j. Disable TRANS_DDI_FUNC_CTL2 Port Sync Mode Enable, if both DSI transcoders are operating in Dual Link mode
    k. Disable panel fitter, if enabled

3. Disable Port. Note that if the Port is going to be disabled, then the IO power will have to be disabled as well
    a. Disable **DDI_BUF_CTL**
    b. Wait for 8us or poll on DDI_BUF_CTL Idle Status for buffers to return to idle
4. Disable IO Power
    a. Disable **PWR_WELL_CTL_DDI** IO Power Request for the DDI that was used
    b. Wait for PWR_WELL_CTL_DDI Power Status = Disabled
    c. Configure **DSI_IO_MODECTL**for DDI operation
5. Disable PLL
    a. Configure PLL to port mapping to direct no clock to the DDI
    b. If this PLL is no longer needed, disable
6. Disable Power Wells being used by the transcoder if there are no required resources within the power well

There are two levels of port disablement allowed by the IO when it is operating in the MIPI mode.

Level	ULPS	Disable Buffer	Disable IO Power	Notes
1	x			This will gate the clocks within the IO AFE. This is equivalent to disabling the buffer without disabling the IO power when the IO is operating as a DDI port.
2	x	x	x	This will power gate the IO

Note that it is invalid to put the Link into ULPS and then disable the Buffer without also disabling the IO power. E.g. the following flow is not allowed:

- Enter ULPS
- Disable Buffer
- Re-enable Buffer
- Exit ULPS

Dual Link Enable Sequence

When setting up two DSI transcoder's to be synchronized, there are two different configurations possible:

1. Single Pipe to two ports (Dual Link - Single Pipe)
2. Two Pipes to two ports (Dual Link - Double Pipes)

Notes:

- Currently, only Pipe A can support a Dual Link - Single Pipe configuration.
- The expectation from the DSI transcoders when they are synchronized is:
    - **Only a single PLL is being used to generate the 1X clock**.

- TE events will need to be supplied to both transcoders

The following table illustrates the programming of the TRANS_DDI_FUNC_CTL2 and the configuration of the DSI ports

Dual Pipe Sync	Port Sync Mode	DSI Port configuration
0	0	Port is independent
0	1	Dual Link - Single Pipe
1	0	Not valid
1	1	Dual Link - Dual Pipe

Regardless of the type of synchronization, the following enable sequence will be common:

1. Enable the Power Well. This step is only done once since both Primary and Secondary transcoders are in the same well.
2. Enable IO power. This step will have to be done for **each** IO port
    a. Note that if enabling for Dual Link - Single Pipe (DLSP), then Software can send the Power Requests (PWR_WELL_CTL_DDI) to both IO ports at the same time and then wait for the Power Status = Enabled for both ports
3. Enable the PLL. This step is only done once since both ports should run off of the same PLL.
4. Configure and Enable DPHY will have to be done for **each** IO port
    a. Again for DLSP, Software can enable both buffers at the same time via the DDI_BUF_CTL register and wait for both buffers to go to the Ready state

Beyond this point the enabling sequences will differ slightly.

Dual Link - Single Pipe (DLSP)

When setting up the DSI transcoders for dual link - single pipe mode, DSI Transcoder 0 will always be the primary and DSI Transcoder 1 will always be the secondary. Each transcoder will transmit 1/2 of the image to be displayed by the panel plus any overlap. It is beyond the scope of the DSI transcoder on which portion of the panel image it is transmitting.

The majority of the primary transcoder registers can be mirrored to the secondary transcoder when the Port Sync Mode is enabled within the TRANS_DDI_FUNC_CTL2 register of DSI Transcoder 0 (i.e. the **primary**). This allows Software to perform the enabling sequence for a single DSI transcoder (the primary). **However, Software must:**

1. Power up and enable **both** port IO's before moving to step 5 of the enabling sequence
2. Program the DSI0 transcoder (the primary) first. If Software is not taking advantage of the mirroring done in hardware, then it can program DSI1 (the secondary), but only after programming DSI0.

After the IO's for both ports have been powered up and enabled, Software can move to step 5 of the enabling sequence.

- Again, Software only needs to program the primary registers

- When programming the **TRANS_DDI_FUNC_CTL2** register set the Port Sync Mode Enable, but keep the Dual Pipe Sync disabled
- Configure the **PIPE_DSS_CTL1** register for Pipe A
    - Enable the Splitter
    - Note that Software **cannot** enable the small Joiner in DLSP
    - Select the Dual Link Mode of operation (Front-Back or Interleave)
    - Program the Overlap, if the overlap is non-zero
- Configure the **PIPE_DSS_CTL2** register for Pipe A
    - If compression is being used, then both Left and Right branch VDSC engines will need to be enabled
    - Configure the Right dual link buffer target depth, if required
- Program the DSI Input Select of the **TRANS_DDI_FUNC_CTL** register to Pipe A when enabling the function
- Wait for the DSI Link Ready (**TRANS_DSI_FUNC_CONF**) for **both** ports

Once both transcoder functions are enabled and ready, Software can move to programming the Panels.

- Software will have to do this for each Panel independently
- Software should finish initializing both Panels before moving on

After Software is finished initializing the Panels it can enable the Planes, Pipe A, and Transcoder. When programming the Horizontal Active size, Software will program it for **half** of the total horizontal active size across both Panels. **Software should not include any overlap within this programming** (HW will handle this)

Dual Link - Dual Pipe (DLDP)

When operating in the dual link - dual pipe configuration, each transcoder is being supplied by a separate and independent Pipe (any Pipe can drive a given transcoder as long as it's not the same Pipe).

Since both transcoders are being supplied by separate Pipes there are a few major differences from DLSP

1. The mirroring that occurs with DLSP does not happen (the Dual Pipe Sync bit of DSI 0 will disable it). This means that Software is responsible for programming both ports with identical timings
2. Either port can be disabled and/or enabled on the fly. When a port is enabled in the presence of an already running port Software can have Hardware automatically synchronize the newly enabled port to the already running port.
3. In Command Mode Software will have to initiate Frame Updates to each transcoder independently. The transcoders will ensure both ports start transmitting the frame after both ports have received their TE events

Because both transcoders are essentially independent yet synchronized, the concept of a primary and secondary port changes. A given transcoder that has its Port Sync enabled will draft off of the other port (i.e. the Primary). Not only will the Primary not know that it is being drafted off of by the other port, but it's possible for both ports to be subordinate to each other (i.e. both ports have the Port Sync bit set, so

there is no primary). The latter scenario will be the normal mode of operation when both ports are in Command Mode.

Synchronizing in Video Mode

There are a couple of ways to get the ports synchronized to each other in Video Mode

1. Bring both ports up at the same time where both ports are sync'ed to each other (i.e. both ports have their Port Sync Enable set)
2. Bring the ports up independently where the second port is subordinate to the first port

   When bringing both ports up synchronized at the same time in Video Mode:

   - Power up and enable both port IO's before moving to step 5
   - Software will have to do the following for each transcoder
     - When programming the **TRANS_DDI_FUNC_CTL2** register set the Port Sync Mode Enable and the Dual Pipe Sync Enable
     - Configure the **PIPE_DSS_CTL1** register for the Pipe that will be feeding the transcoder, if needed
       - Software should not enable Splitting
       - Software can enable Joining
     - Configure the **PIPE_DSS_CTL2** register for the Pipe that will be feeding the transcoder, if compression is needed
     - Configure and enable **TRANS_DDI_FUNC_CTL**
       - The DSI Input Select should be different for each transcoder
     - Wait for the DSI Link Ready (**TRANS_DSI_FUNC_CONF**)
     - Program the Panel
     - Enable the Planes and Pipe, but stop short of enabling the Transcoder
       - Note that the Horizontal Active should be programed for the active size of the given Panel
   - When above programming for both transcoders has been complete, then enable both Transcoders (**TRANS_CONF**).
     - Wait for both Transcoder States to be enabled
   - If allowing DC*co, then do that now

   When bringing up a single port to be synchronized to an already running port in Video Mode:

   - The port already running is enabled normally without Port Sync Mode being enabled
   - The new port is enabled normally but it will enable the Port Sync Mode and the Dual Pipe Sync within the **TRANS_DDI_FUNC_CTL2** register

Synchronizing in Command Mode

When in Command Mode, if Software wishes to synchronize the frames between the two ports then it must enable the Port Sync for **both** ports (i.e. both ports have to have the Port Sync Mode and Dual Pipe Sync bits set) before sending the Frame Update Request to either port. The enabling sequence, otherwise, is not unique.

When the transcoders are sync'ed to each other in Command Mode, each transcoder will satisfy the following conditions before starting the transmission of the frame data:

1. This transcoder has met its minimum V. Blank requirements as programmed within the **TRANS_VTOTAL** register
2. This transcoder has received the TE event from its Panel (if TE Gating is enabled)
3. The remote transcoder has met the above two requirements

The transcoders do supply some hooks within the **DSI_CMD_FRMCTL** register for Software to initiate frame updates to the Panels.

- **Single Panel Update**. This attribute will allow a transcoder that is synchronized to another port to send out a frame update without requiring the other port to send a frame
- **Forward Frame Update Request**. This attribute will allow SW to send a single Frame Update Request to a single transcoder and have that transcoder forward the request to the other transcoder
- **Accumulate Frame Update Requests**. This attribute will enable the synchronized transcoders to synchronize the Frame Start to each Pipe

Initializing Panel in LP Escape Mode

Some Panels may require initialization (i.e. Panel programming) to be performed by the Host in the LP Escape mode without the Clock Lane running. After initialization is complete in the LP Escape mode and before the transcoder's timing generator is enabled (TRANS_CONF), the Panel may need the Clock Lane to be running continuously. The following sequence describes the flow needed to reliably perform these Panel requirements.

1. Software will enable the Power Wells, PLL, and IO Power per the above enabling sequence
2. When Enabling the DSI Port and D-PHY, Software will leave the Continuous Clock setting of the TRANS_DSI_FUNC_CONF register to its default value of "Always enter LP after Data Lanes"
    a. Software will otherwise continue the enabling flow within this step normally where the DSI Function will be enabled at the end (TRANS_DDI_FUNC_CTL / TRANS_DSI_FUNC_CONF)
3. Software will perform the Program Panel step normally except it will set the "LPDT" attribute when writing to the DSI_CMD_TXHDR register.
4. When Software has finished sending all of the Panel initialization commands, it will ensure they have all been dispatched by doing the following serialization:
    a. Software will wait for all Header and Payload Credits to be released (**DSI_CMD_TXCTL**)

i. Timeout after 100us

b. Build a NOP DCS command using the DSI_CMD_TXHDR register
   i. The packet will use a Data Type of 0x05 (DCS Short Write - 0 Parameters)
   ii. The Word Count/Parameter bytes will be set to all zeroes (NOP DCS command)
   iii. Set the LPDT bit

c. After loading the NOP with the DSI_CMD_TXHDR write, Software will wait for the following to be true:
   i. Wait for all Header Credits to be released
      1. Timeout after 20us
   ii. Wait for LP Tx in Progress bit of the **DSI_LP_MSG** register to be cleared
      1. Timeout after 20us

5. When Software has finished the above serialization it will program the Continuous Clock setting of the TRANS_DSI_FUNC_CONF register to the desired setting
6. When finished with the above sequence, Software can continue with the enabling sequence and go to the step to enable Planes, Pipe, and Transcoder

Note that this sequence is only for Panels that require initialization in the LP Escape mode. If the Panel does not have this requirement, then Software should perform the normal enabling sequence described above and send the Panel initialization programming in the HS mode.

Sending Commands to the Panel

Software has the ability to send command's to the Panel (e.g. Display Set Command packets) using some specific registers within the DSI transcoder. The DSI transcoder supplies Software with a pool of credits (both header and payload) that Software can use to inject commands onto the DSI Link to the panel.

Command Injection Sequence

1. SW will read the DSI Command Transmission Control register (**DSI_CMD_TXCTL**) to determine how many credits are available.
   a. SW must only use the available credits advertised within the register at the time of the read (i.e. it cannot combine the credits from a previous read)
2. If there are enough credits available for the packet it wishes to build then it will begin issuing writes to the Payload and/or the Header registers
   a. If the packet contains a payload:
      i. SW will use the Payload credits it read in step 1 to write to the Command Payload register (**DSI_CMD_TXPYLD**)
      ii. Every write to DSI_CMD_TXPYLD will load the data written to the register into a payload queue within the transcoder which will consume a Payload resource (i.e. credit)
         1. HW will maintain this credit pool and reflect the current number of available credits within the DSI_CMD_TXCTL register

2. SW can also maintain the credits that it read in step 1 to pipeline the writes to this register, if it has not consumed all of the credits read from that step

iii. SW will write the first DW of the payload first followed by the subsequent DW's of payload data

b. If the packet does not contain a payload, or it has finished writing the payload data to the DSI_CMD_TXPYLD register, it will then write to the Command Header register (**DSI_CMD_TXHDR**) using a Header credit it read in step 1

i. If the packet contains a payload, then SW will:

1. Program the Word Count (WC) field of the packet header to the number of **bytes** within the packet payload that was written to the Payload queue
2. Set the Payload Present attribute

ii. Otherwise, SW will clear the Payload Present attribute when writing the Command Header register

iii. SW will also set the other transmission attributes it wishes to be applied to the packet

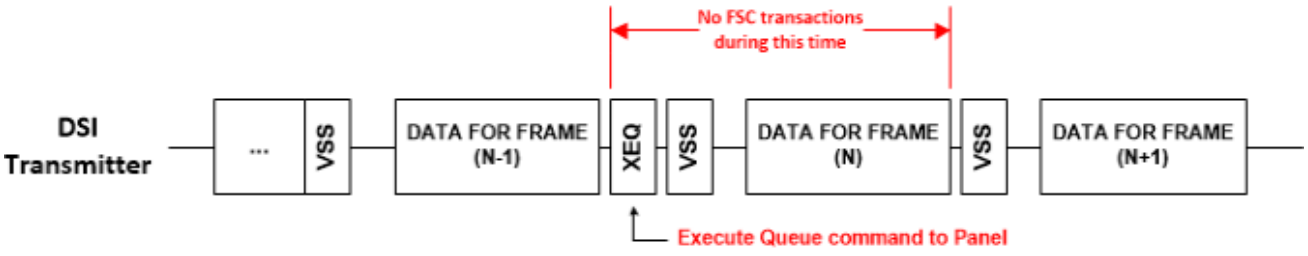
1. The LPDT bit will indicate whether the packet should be sent to the Periphery in the High Speed (HS) or the Low Power (LP) mode
2. The Vertical Blank Fence bit will determine whether the packet will be gated until the next Vertical Blank entry (for Video Mode) or the end of the current frame update is seen

iv. A write to DSI_CMD_TXHDR will load the data written to the register into a header queue within the transcoder which will consume a Header resource (i.e. credit)

1. HW will maintain this credit pool and reflect the current number of available credits within the DSI_CMD_TXCTL register
2. SW can also maintain the credits that it read in step 1 to continue issuing writes to this register, if it has not consumed all of the credits read from that step

Frame Synchronized Command Sequences

The MIPI DSI specification defines a mechanism for the Peripheral to synchronize commands to vertical blanking events internal to the panel. The types of commands that the Peripheral considers to be Frame Synchronized Commands (FSC) is panel specific, but the specification does have restrictions on how and when FSC transactions can be sent to the panel from the Host.



The Vertical Blank Fence attribute within the DSI_CMD_TXHDR is the mechanism within the DSI transcoder that Software can use to ensure these restrictions are met.

Command Mode

1. Load Execute Queue command (Data Type = 0x16) into the **DSI_CMD_TXHDR** register
2. Set the Frame Update Request bit of the **DSI_CMD_FRMCTL** (this can happen at any time after loading the Execute Queue command)
3. If there is another FSC transaction to send, then load it into the DSI_CMD_TXHDR register with the Vertical Blank Fence bit set

Video Mode

1. Load Execute Queue command (Data Type = 0x16) into the DSI_CMD_TXHDR register with the Vertical Blank Fence bit set
2. If there is another FSC transaction to send, then load it into the DSI_CMD_TXHDR register with the Vertical Blank Fence bit set

Low Power Messaging Sequence

1. SW will read the DSI Command Transmission Control register (**DSI_CMD_TXCTL**) to determine if there are any Header credits available.
    a. Each LP message consumes a single Header credit
2. If there is a Header credit available, then Software will program the **DSI_LP_MSG** register with the LP message it wishes to send
    a. If the LP message is either a Trigger or a ULPS Entry, then the respective message type will also need to be programmed
    b. If the LP message is for a ULPS Entry, then Software will need to follow the "DSI Transcoder Disable Sequence" up until step 1.f
3. When HW is servicing the LP message it will advertise the following through the DSI_LP_MSG register:
    a. For all LP messages the transcoder will set the "LP Tx in Progress" bit
    b. The Direction of the Link is always advertised within the register. When doing a Bus Turn-Around, Software will be able to see when the Link changes direction
    c. When doing a ULPS Entry, the register will advertise when the Link has entered the ULP state.
4. When HW is done servicing the LP message it will do the following (Software can key off of any of these to determine when the LP message is complete):
    a. It will clear out the LP message request bit that Software set (i.e. ULPS Entry, Bus Turnaround, or Trigger Message)
    b. Except for ULPS Entry messages, it will clear the "LP Tx in Progress" status bit
        i. For ULPS Entry messages, this bit will not be cleared out until the DSI Function is disabled
    c. For ULPS Entry messages:
        i. The "In ULPS" bit will be set. This bit will be cleared when the DSI Function is disabled

ii. If unmasked, the "ULPS Entry Done" interrupt will be set

5. When done with the ULPS entry, Software can continue with the DSI Transcoder Disable Sequence

Data Receipt Sequence

1. If SW is sending a read to the Panel, then it will follow the "Command Injection Sequence" above to place a read request within the header queue via a write to the **DSI_CMD_TXHDR**. Some notes on this step:
    a. If Software is just looking for an error status from the Panel, then it can go directly to step 2
    b. Before sending a read to the Panel, Software must have sent a "*Set Maximum Packet Size*" DSI packet (Data Type = 0x37) to the Panel to control the amount of data the Panel returns to the Host
    c. The DSI transcoder will send the read command within the next Vertical blank line time that it sees, if operating in Video Mode. If Software wishes the command to be sent at the beginning of the next Vertical blank region, then it can set the Vertical Blank Fence bit when writing to the DSI_CMD_TXHDR
2. SW will write to **DSI_LP_MSG** to place a Bus Turn-Around (BTA) request within the header queue
    a. Prior to this SW should have set up the following (see **Performing LP Transactions in Video Mode** below for more details):
        i. The Turn-Around Timeout (**DSI_TA_TO**) to ensure the BTA request to the Panel proceeds in a timely manner
            1. SW will need to program the equivalent register within the Panel to a time that is less than what is programmed within the DSI_TA_TO register. The command needed to update any registers within the Panel it is beyond the scope of this flow.
        ii. The maximum allowed time the Peripheral will be given to own the Link (**DSI_LRX_H_TO**)
            1. Again, SW will need to program the equivalent register within the Panel with a time that is less than what is programmed within the DSI_LRX_H_TO register
            2. If in Video Mode, the value should be programmed such that the time given to the panel is less than the time between Sync events
        iii. The Peripheral Response Timeout (**DSI_PWAIT_TO**) to give the Panel some time before it has to send back a response.
        iv. If the Panel does not support ECC, then both of the Single and Multi ECC Mask bits of the Interrupt Mask Register (**IMR**) should be set
        v. If the Panel does not support CRC checksums, then the Checksum Error bit of the IMR should be set
    b. Notes:
        i. SW will need a Header Credit to place the Bus Turn-Around request within the Header queue

ii. The DSI transcoder dispatches the commands it receives from Software via the DSI_CMD_TXHDR and DSI_LP_MSG registers in the order they are received

iii. The DSI transcoder will dispatch the BTA at the beginning of the next Vertical blank line time that it sees, if it is in Video Mode. In other words, it will not dispatch the read and the BTA within the same Vertical blank line time.

3. If SW wishes to have HW announce the receipt of the response from the Panel through an interrupt, then it has the following interrupts that it can use. To enable these interrupts, SW should clear the mask bit within the Interrupt Mask Register (IMR)
    a. Rx Trigger: This is for non-TE triggers received from the Panel. If the Panel does not have any return data to send back to the Host, and it has not detected any errors since the last BTA, then it will only send a trigger message (most likely an Acknowledge trigger)
    b. Rx Data/BTA Terminated: HW will generate this type of event when it either receives a response that SW can read from the **DSI_CMD_RXHDR** and/or **DSI_CMD_RXPYLD**, or the BTA was terminated
    c. Error Report: If the Panel detected an error after the last BTA, then it will send an Acknowledge and Error Report packet back to the Host. The error report carried within this packet is sent directly to the interrupt register (the lower 16 bits of the interrupt registers). Depending on the severity of the error, the Peripheral may not send any other responses (i.e. a READ return packet).
4. When the response is received from the Panel, SW will be able to read the header of the response through the DSI_CMD_RXHDR register and the payload, if there is a payload, through the DSI_CMD_RXPYLD register
    a. The payload of the response will be contained within a queue that will be unloaded for every read to the DSI_CMD_RXPYLD register. SW can control whether the DW of data is unloaded from the queue or not through the **DSI_CMD_RXCTL** register
    b. The DSI_CMD_RXCTL register will also advertise the number of DW's available within the payload queue, so SW can use this to determine how many reads it needs to make to DSI_CMD_RXPYLD
5. If the Panel detected any errors since the last BTA, then it will send an Acknowledge and Error Report packet to the Host that will contain the errors that it detected. If the masks for these errors are cleared within the IMR, then SW will have these errors logged within the Interrupt Identity Register (IIR)

Send a Frame in Command Mode

1. During the enabling sequence, Software will:
    1. Setup the source of the Tear Effect (TE) events within the TRANS_DSI_FUNC_CTL register - in-band over the DSI Link or out-of-band through a GPIO.
    2. Choose how the start of the incoming frame pixels are to be handled by the DSI transcoder through the Mode of Operation of the TRANS_DSI_FUNC_CTL register - will it wait for a TE event to be received or not

2. To start a frame, Software will set the Frame Update Request bit of the DSI_CMD_FRMCTL. Hardware will clear this bit once it has begun processing the request.
    1. Note that if configured in Dual Link mode, then only the DSI_CMD_FRMCTL register of the primary port (DSI port 0) will initiate a frame update. However, if Software wishes to observe the Frame in Progress bit of the register, it should look at the secondary port's DSI_CMD_FRMCTL register (DSI port 1) since the secondary port will always be slightly behind the primary port.

Tear Effect (TE) Signaling Sequences

There are three different methods for the Host to get TE information from the Panel:

1. Polling the Panel to detect the current scan line (i.e. using DCS *get_scan_line* commands)
2. Receiving TE events out-of-band (over GPIO)
3. Receiving TE events in-band (over the DSI Link)

For the polling option, Software will simply send out DCS *get_scan_line* requests to the Panel (through DSI_CMD_TXHDR followed by a BTA request through DSI_LP_MSG) until the Display has reached a certain scan line. To receive this information through TE events, Software will need to program the Panel to send these events at specified times (through the DCS *set_tear_on* and *set_tear_scanline* commands). The following is the basic sequences needed to handle receiving TE events.

Receiving Out-of-Band TE Events

1. Software will configure the transcoder to take the TE events from the GPIO when configuring the TRANS_DSI_FUNC_CONF
    a. Software will need to enable and configure the **UTIL_PIN_CTL** register as the TE input for DSI 0
2. During the Panel Programming section of the enabling sequence, Software will configure the Panel to send TE events at the desired times
3. Software will continue the enabling sequence and finally enable the transcoder
4. If TE events are gating:
    a. Software will initiate the Frame Request (DSI_CMD_FRMCTL)
    b. Transcoder hardware will automatically send the frame data to the Panel when it receives the TE event
    c. Software will repeat with a new Frame Request
5. If TE events are not gating:
    a. Software will initiate a Frame Request (DSI_CMD_FRMCTL)
    b. Transcoder hardware will automatically send the frame data to the Panel
    c. Software will repeat with a new Frame Request

Receiving In-Band TE Events

1. Software will configure the transcoder to take the TE events from the DSI receiver when configuring the TRANS_DSI_FUNC_CONF
2. During the Panel Programming section of the enabling sequence, Software will configure the Panel to send TE events (set_tear_on, set_tear_scanline DCS commands)
    a. When operating in Dual Link, both ports will need to be programmed to send the TE triggers
    b. After enabling the tearing on the Panel, a BTA should be sent to the Panel (DSI_LP_MSG) so that the Panel can Acknowledge that the set_tear_* DCS commands were received without any errors
3. Software will continue the enabling sequence and finally enable the transcoder
4. If TE events are gating:
    a. Software will initiate the Frame Request (DSI_CMD_FRMCTL). If in Dual Link Mode, this is done on the Primary port (DSI 0).
    b. Software will send a BTA request to the Panel (DSI_LP_MSG). If in Dual Link Mode, this is done on both ports.
    c. Transcoder hardware will automatically send the frame data to the Panel when it receives the TE event and the Panel turns control back over to the Host
    d. Repeat when the Frame is done (Frame In Progress de-asserts)
5. If TE events are not gating:
    a. Software will initiate the Frame Request (DSI_CMD_FRMCTL). Again, if in Dual Link Mode, then this is performed on the Primary port.
    b. Transcoder hardware will automatically send the frame data to the Panel
    c. When the Frame is done (Frame In Progress de-asserts), Software will send a BTA request to the Panel (DSI_LP_MSG)
        i. If in Dual Link Mode, then Software will monitor the Primary port's DSI_CMD_FRMCTL.Frame In Progress bit, and will send the BTA request on both ports
    d. Repeat when the TE event is received (through either interrupt or DSI_CMD_RXCTL)

Periodic Command Mode

As described above, frame updates sent to the Panel in Command Mode are typically initiated by Software (i.e. Intel's graphics driver). When the Intel graphics driver is not present (e.g. the driver is uninstalled, the OS basic driver is present, or pre-boot time) there will be no frame update requests to the DSI transcoder which will lead to a stale screen. In this situation, the DSI transcoder will have the ability to automatically issue frame updates to the Panel by trapping off of GPIO TE events received from the Panel.

The requirements to enable this feature are:

1. The DSI transcoder will be configured to send Periodic Frame Updates (DSI_CMD_FRMCTL)
    a. By default this feature will be disabled

2. The Panel will need to be configured to send TE events to the Host (i.e. set_tear_on, etc.)
3. The DSI controller will override the programming of the TE Source and TE Gating/Non-Gating when this mode of operation is enabled. The HW overrides will be:
    a. The TE source will be taken from the GPIO. The DSI transcoder HW does not have a mechanism to send Bus Turn-Around's to the Panel
    b. Has to be configured for non-gating Command mode

Switching between Video Mode (DPI) and Command Mode (DBI)

1. Disable the **TRANS_CONF** Transcoder Enable
2. Wait for off status in TRANS_CONF
    i. Timeout if transcoder hasn't moved to the off status after two frame times
3. Change the TRANS_DSI_FUNC_CTL Mode of Operation to the new mode
4. Change any of the other transcoder settings in preparation for Command Mode
5. Re-enable the TRANS_CONF Transcoder Enable
6. If the new mode of operation is Command Mode, then Software can start transmitting frame updates to the Panel using the **DSI_CMD_FRMCTL** register (see "Send a Frame in Command Mode"). Otherwise, the transcoder will automatically start sending frames to the panel

Programming References

Programming references has been moved to the "Transcoder DSI Function" page

Sequences to Initialize Display

These sequences are used to initialize the display engine before any display engine functions can be enabled. To save power, do not initialize display if it will not be used, such as on "headless" systems.

Most display engine functions will not operate while display is not initialized. Only basic PCI, I/O, and MMIO register read/write operations are supported when display is not initialized.

Initialize Sequence

1. Enable PCH Reset Handshake
    a. Set **NDE_RSTWRN_OPT** RST PCH Handshake En to 1b.
    b. Configure south display Raw Clock before first enabling GMBUS, south display hotplug detection, or panel power sequencing.
2. Initialize all combo PHYs with Combo PHY DDI Buffer Combo PHY Initialization Sequence.
3. Enable Power Well 1 (PG1)
    a. Poll for **FUSE_STATUS** Fuse PG0 Distribution Status = 1b.
        - Timeout and fail after 20 us.
    b. Set **PWR_WELL_CTL** Power Well 1 Request to 1b.

- There are two sets of PWR_WELL_CTL registers for software use. It is expected that BIOS uses PWR_WELL_CTL1 and driver uses PWR_WELL_CTL2.

c. Poll for **PWR_WELL_CTL** Power Well 1 State = 1b.
   - Timeout and fail after 30 us (38.4 MHz reference frequency) or 45us (24 MHz reference frequency).
   - Register DSSM Reference Frequency indicates the reference frequency.

d. Poll for **FUSE_STATUS** Fuse PG1 Distribution Status = 1b.
   - Timeout and fail after 20 us.

- Combo PHY Aux IO can be optionally enabled in parallel with PG1 following DDI AUX Channel Aux IO Power Enabling, or it can be enabled later when the Aux will be used.

4. Enable CD clock following the Sequences for Changing CD Clock Frequency
5. Enable first DBUF
   a. Set **DBUF_CTL_<first DBUF>** DBUF Power Request to 1b.
   b. Poll for **DBUF_CTL_<first DBUF>** DBUF Power State = 1b.
      - Timeout and fail after 10 us.
   - Additional DBUFs may be enabled later as needed for the display bandwidth and number of pipes enabled.
6. Setup MBUS. Refer to MBus page for the MBus credits programming.
7. Program BW_BUDDY registers. Refer to Arbiter page.

Un-initialize Sequence

Software should only run the un-initialize sequence as part of DC9. It can conflict with other DC states.

1. Disable all display engine functions using the full mode set disable sequence on all pipes, transcoders, ports, planes, and power wells above PG1.
2. Disable DBUFs
   a. Clear **DBUF_CTL_*** DBUF Power Request to 0b.
   b. Poll for **DBUF_CTL_*** DBUF Power State = 0b.
      - Timeout and fail after 10 us.
3. Disable CD clock following the Sequences for Changing CD Clock Frequency
4. Disable Power Well 1 (PG1) and Aux IO Power.
   a. Follow AUX Channel Aux IO Power Disabling for every Aux that is powered up.
   b. Clear **PWR_WELL_CTL**_* Power Well 1 Request to 0b.
   c. Wait for 10us. Do not poll for the power well to disable. Other clients may be keeping it enabled.
5. Follow Combo PHY DDI Buffer Combo PHY Un-Initialization Sequence for all combo PHYs.

Sequences for DisplayPort

This topic describes how to enable and disable DisplayPort.

The DP transport is within the transcoder. Multi-stream requires a transcoder to be enabled to act as the primary transcoder, hosting the DP transport that consolidates the multiple streams. The primary transcoder's clocking, TRANS_DDI_FUNC_CTL, and DP_TP_CTL must be kept enabled while the multistream DP link is enabled, and not enabled and disabled with the video stream.

Enable Sequence

Display must already be initialized

DDIA Lane Capability Control must be configured prior to enabling any ports or port clocks

1. **Enable Power Wells**
    a. Based on the resources to be used, enable the appropriate power wells following the Sequences for Power Wells
    b. Type-C: Note that AUX power is required for running main link.
2. **If panel power sequencing is required - Enable Panel Power**
    a. Enable panel power sequencing
    b. Wait for panel power sequencing to reach the enabled state
3. **Type-C ports**
    a. Skip this step for TBT.
    b. Program DFLEXDPMLE.DPMLETC* to maximum number of lanes allowed as determined by FIA and panel lane count.
        i. Fixed/legacy/static - Program the number of lanes as per DDI_BUF_CTL DP Port Width Selection
        ii. Dynamic - Program the number of lanes as per DFLEXDPSP.DPX4TXLATC*
4. **Enable Port PLL**
    a. If PLL is not already enabled, follow port clock programming sequence from Clocks section
    b. If PLL to port mapping is flexible, configure PLL to port mapping to direct the PLL output to the DDI
5. **If IO power is controlled through PWR_WELL_CTL - Enable IO Power**
    a. Skip this step if TBT.
    b. If the DDI is going to combo PHY, the PHY must be initialized with the Combo PHY DDI Buffer Combo PHY Initialization Sequence
    c. Enable PWR_WELL_CTL DDI IO Power Request for the DDI that will be used
    d. Wait for PWR_WELL_CTL DDI IO Power Request = Enabled, timeout after 20 us (30us for type-C PHYs with fixed/static/native connectors)
6. **Program DP_MODE**
    a. Skip this step for display attached to Combo PHY and TBT.

b. Program PHY lane0 and lane1 DP_MODE registers according to TypeC PHY programming page.

7. **Enable and Train DisplayPort**
   a. If single stream or multi-stream primary transcoder - Configure Transcoder Clock Select to direct the Port clock to the Transcoder
   b. If single stream or multi-stream primary transcoder - Configure TRANS_DDI_FUNC_CTL DDI Select, DDI Mode Select & MST Transport Select
   c. Configure and enable DP_TP_CTL with link training pattern 1 selected
   d. Configure voltage swing and related IO settings.
      i. Refer to Combo DDI Buffer section for display attached to Combo PHY.
      ii. Refer to Type-C DDI buffer page for all other cases.
   e. Combo PHY: Configure PORT_CL_DW10 Static Power Down to power up the used lanes of the DDI.
   f. Configure and enable DDI_BUF_CTL
   g. Wait for DDI_BUF_CTL DDI Idle Status = 0b (Not Idle), timeout after 1 ms.
   h. Follow DisplayPort specification training sequence (see notes for failure handling)
   i. If DisplayPort multi-stream - Set DP_TP_CTL link training to Idle Pattern, wait for 5 idle patterns (DP_TP_STATUS Min_Idles_Sent) (timeout after 800 us)
   j. Set DP_TP_CTL link training to Normal.
   k. Configure and enable FEC if needed. NOTE: Refer to Bspec "DDI FEC" page for enabling protocol with FEC capable sink.
8. **If not in compliance mode: Enable Planes, Pipe, and Transcoder (repeat to add multiple pipes on a single port for multi-streaming)**
   a. If DisplayPort multi-stream - use AUX to program receiver VC Payload ID table to add stream
   b. Configure Transcoder Clock Select to direct the Port clock to the Transcoder. Skip this step if clock select already programmed above.
   c. Configure and enable planes (VGA or hires). This can be done later if desired.
   d. If VGA - Clear VGA I/O register SR01 bit 5
   e. Configure and enable VDSC if needed.
   f. Enable panel fitter if needed (must be enabled for VGA)
   g. Configure transcoder timings, M/N/TU/VC payload size, and other pipe and transcoder settings
   h. Configure TRANS_DDI_FUNC_CTL2 if port sync mode needs to be configured. Then configure and enable TRANS_DDI_FUNC_CTL.
      i. Please refer to the note below for big joiner mode operation.
   i. Configure VRR if needed. (note: VRR needs to be programmed after TRANS_DDI_FUNC_CTL and before TRANS_CONF).
      i. program TRANS_VRR_VMIN

ii. program TRANS_VRR_VMAX
iii. Enable push bit if required
iv. enable TRANS_VRR_CTL

j. If DisplayPort multistream - Clear ACT sent status in primary
k. If DisplayPort multistream - Enable pipe VC payload allocation in TRANS_DDI_FUNC_CTL
l. If DisplayPort multistream - Wait for ACT sent status in primary transcoder DP_TP_STATUS and receiver DPCD (timeout after >410us)
m. For DP 1.4 MST with FEC, set bit [23] at applicable offset 0x420C0/0x420C4/0x420C8/0x420D8 to '1'.
n. Configure and enable TRANS_CONF
o. If panel power sequencing is required - Enable panel backlight

SRD and/or Audio can be enabled after everything is complete. Follow the audio enable sequence in the audio registers section.

Notes

Changing voltage swing during link training:

- Change the swing setting following the DDI Buffer section. The port does not need to be disabled.

Changing port width (lane count) or frequency during link training:

1. Follow Disable Sequence for DisplayPort to Disable Port.
2. For TBT follow the steps to turn off the clock for the port. Otherwise, if PLL frequency needs to change, follow the Disable Sequence for DisplayPort to Disable PLL.
3. For TBT, follow the steps to map the new frequency to the port and turn on the clock for the port. For all other port types, follow the Enable Sequence for DisplayPort to Enable PLL, using the new frequency settings.
4. Type-C ports: skip this step for TBT.
    a. Program DFLEXDPMLE.DPMLETC* for the new number of lanes (down training)
    b. Program PHY lane0 and lane1 DP_MODE registers for the new number of lanes (down training)
5. Follow the Enable Sequence for DisplayPort to Enable and Train DisplayPort, using the new port width settings.

If the mode set fails, follow the disable sequence to disable everything that had been enabled up to the failing point.

Enabling DisplayPort Sync Mode (non-MST)

See TRANS_DDI_FUNC_CTL2 Port Sync Mode Enable for restrictions.

1. Follow the enable sequence for the DisplayPort secondary, but skip the step that sets DP_TP_CTL link training to Normal (stay in Idle Pattern).
    - Set secondary TRANS_DDI_FUNC_CTL2 Port Sync Mode Primary Select and Port Sync Mode Enable before configuring and enabling secondary TRANS_DDI_FUNC_CTL.
2. Follow the enable sequence for the DisplayPort primary, but skip the step that sets DP_TP_CTL link training to Normal (stay in Idle Pattern).
3. Set DisplayPort secondary DP_TP_CTL link training to Normal.
4. Wait 200 uS.
5. Set DisplayPort primary DP_TP_CTL link training to Normal.

Software may need to use DOUBLE_BUFFER_CTL to ensure updates to plane and pipe registers will take place in the same frame.

For example: If pipe A and pipe B are synchronized together and software needs the surface addresses for two planes to update at the same time, software should use DOUBLE_BUFFER_CTL when writing the surface address registers for both planes, otherwise there is a possibility that the updates could be split across a vertical blank such that one plane would update on the current vertical blank and the other plane would update on the next vertical blank.

Enabling DisplayPort with Big Joiner

Big joiner (such as for 8K on a single port) uses two pipes to drive a single transcoder.

1. Follow the enable sequence through the steps to Enable and Train DisplayPort
2. Follow the steps to Enable Planes, Pipe, and Transcoder for pipe C, but do not configure or enable transcoder C.
3. Follow the steps to Enable Planes, Pipe, and Transcoder for pipe B and transcoder B.

Enabling DisplayPort Sync Mode (MST with 1 DDI)

General constraint is that secondary transcoder has to be enabled first before primary transcoder.

See TRANS_DDI_FUNC_CTL2 Port Sync Mode Enable for restrictions.

1. Follow MST enable sequence for the DisplayPort steps 1 to 7
2. Enable the secondary pipe following the step 8 of MST enable sequence
    1. Skip the steps to enable TRANS_DDI_FUNC_CTL2, TRANS_DPT_PAT, TRANS_DDI_FUNC_CTL and TRANS_CONF.
3. Repeat step 8 of MST enable sequence to enable the primary pipe.
    1. Skip the steps to enable TRANS_DPT_PAT, TRANS_DDI_FUNC_CTL and TRANS_CONF.
4. Set secondary TRANS_DDI_FUNC_CTL2 Port Sync Mode Primary Select and Port Sync Mode Enable before configuring and enabling secondary TRANS_DDI_FUNC_CTL. Configure and enable secondary TRANS_CONF and secondary TRANS_DPT_PAT.
5. Configure and enable primary TRANS_DDI_FUNC_CTL. Configure and enable primary TRANS_CONF and primary TRANS_DPT_PAT.

Enabling DisplayPort Sync Mode (MST with 2 DDIs)

General constraint is that secondary transcoder has to be enabled first before primary transcoder.

See TRANS_DDI_FUNC_CTL2 Port Sync Mode Enable for restrictions.

1. secondary DDI: Follow MST enable sequence for the DisplayPort steps 1 to 7
2. Enable the secondary pipe following the step 8 of MST enable sequence
    1. Skip the steps to enable TRANS_DDI_FUNC_CTL2, TRANS_DPT_PAT, TRANS_DDI_FUNC_CTL and TRANS_CONF.
3. primary DDI: Follow MST enable sequence for the DisplayPort steps 1 to 7
4. Repeat step 8 of MST enable sequence to enable the primary pipe.
    1. Skip the steps to enable TRANS_DPT_PAT, TRANS_DDI_FUNC_CTL and TRANS_CONF.
5. Set secondary TRANS_DDI_FUNC_CTL2 Port Sync Mode Primary Select and Port Sync Mode Enable before configuring and enabling secondary TRANS_DDI_FUNC_CTL. Configure and enable secondary TRANS_CONF and secondary TRANS_DPT_PAT.
6. Configure and enable primary TRANS_DDI_FUNC_CTL. Configure and enable primary TRANS_CONF and primary TRANS_DPT_PAT.

Disable Sequence

SRD and Audio must be disabled first. Follow the audio disable sequence in the audio registers section.

1. **If panel power sequencing is required - Disable panel backlight**
2. **If not in compliance mode: Disable Planes, Pipe, and Transcoder (repeat to remove multiple pipes from a single port for multi-streaming)**
    a. If VGA
        i. Set VGA I/O register SR01 bit 5 for screen off
        ii. Wait for 100 us
    b. Disable planes (VGA or hires)
    c. Disable TRANS_CONF
    d. Clear bit [23] to '0' at applicable offset 0x420C0/0x420C4/0x420C8/0x420D8 if set to '1'.
    e. Wait for off status in TRANS_CONF, timeout after two frame times
    f. If DisplayPort multistream - use AUX to program receiver VC Payload ID table to delete stream
    g. If done with this VC payload
        i. Clear ACT sent status in primary
        ii. Disable VC payload allocation in TRANS_DDI_FUNC_CTL
        iii. Wait for ACT sent status in DP_TP_STATUS (multi-stream use the primary transcoder DP_TP_STATUS) and receiver DPCD
    h. Disable VRR, if enabled.
    i. Disable push bit if enabled.

j. If MST primary transcoder: Disable TRANS_DDI_FUNC_CTL and do not change DDI_Select
    1. All other transcoders: Disable TRANS_DDI_FUNC_CTL with DDI_Select set to None
k. Disable DSS_CTL1 Joiner enable (bit 30) if enabled
l. Disable DSS_CTL2 left VDSC (bit 31) and/or right VDSC (bit 30) if enabled
m. Disable panel fitter
n. If multi-stream secondary transcoder: Configure Transcoder Clock Select to direct no clock to the transcoder

3. **Disable Port (all pipes and VC payloads on this port must already be disabled)**
    a. If multi-stream primary transcoder: Set DDI_Select set to None
    b. Disable DDI_BUF_CTL
    c. Disable DP_TP_CTL (do not set port to idle when disabling)
    d. Disable FEC if it is enabled.
    e. Wait 8 us or poll on DDI_BUF_CTL Idle Status for buffers to return to idle
    f. If single stream or multi-stream primary transcoder: Configure Transcoder Clock select to direct no clock to the transcoder
4. **If panel power sequencing is required - Disable Panel Power**
    a. Disable panel power sequencing
5. **If IO power is controlled through PWR_WELL_CTL - Disable IO Power**
    a. Skip this step for TBT. For all other port types, disable PWR_WELL_CTL DDI IO Power Request for the DDI that was used.
6. **Disable Port PLL**
    a. If PLL to port mapping is flexible, configure PLL to port mapping to direct no clock to the DDI
    b. If this PLL is no longer needed, follow PLL disable sequence from Clocks section
7. **Disable Power Wells**
    a. Disable power wells that are no longer required, following the Sequences for Power Wells

Disabling DisplayPort Sync Mode

1. Follow the disable sequence for the DisplayPort secondary.
2. Follow the disable sequence for the DisplayPort primary.

Disabling DisplayPort Sync Mode (MST with 1 DDI)

1. Follow the disable sequence for the MST DisplayPort secondary pipe but keep secondary VC enabled. Step 2 of Disable Sequence.
2. Follow the disable sequence for the MST DisplayPort primary stream. Step 2 of Disable Sequence.
3. Disable secondary VC.

Disabling DisplayPort Sync Mode (MST with 2 DDIs)

1. Follow the disable sequence for the MST DisplayPort secondary stream. Step 2 of Disable Sequence.
2. Follow the disable sequence for the MST DisplayPort primary stream. Step 2 of Disable Sequence.

Disable Displayport with Big Joiner

1. Follow the steps to disable Planes, Pipe, and Transcoder for pipe B and transcoder B
2. Follow the steps to Disable Planes and Pipe for pipe C
3. Disable port associated with Transcoder B

Sequences for HDMI and DVI

This topic describes how to enable and disable HDMI and DVI.

Enable Sequence

Display must already be initialized

1. **Enable Power Wells**
    1. Based on the resources to be used, enable the appropriate power wells following the Sequences for Power Wells
    2. Type-C static connection: Enable AUX I/O Power. It is required for running the main link, even with HDMI.
2. **Type-C static connection: Program DFLEXDPMLE.DPMLETC* to maximum number of lanes of the port.**
3. **Enable Port PLL**
    a. If PLL is not already enabled, follow port clock programming sequence from Clocks section
    b. If PLL to port mapping is flexible, configure PLL to port mapping to direct the PLL output to the DDI
4. **If IO power is controlled through PWR_WELL_CTL - Enable IO Power**
    a. Enable PWR_WELL_CTL DDI IO Power Request for the DDI that will be used
    b. Wait for PWR_WELL_CTL DDI IO Power Request = Enabled, timeout after 20 us (30us for type-C PHYs with fixed/static/native connectors)
5. **Type-C static connection: Program Type-C PHY DP_MODE register for static x4**
6. **Enable Planes, Pipe, and Transcoder**
    a. Configure Transcoder Clock Select to direct the Port clock to the Transcoder
    b. Configure and enable planes (VGA or hires). This can be done later if desired.
    c. If VGA - Clear VGA I/O register SR01 bit 5
    d. Enable panel fitter if needed (must be enabled for VGA)
    e. Configure transcoder timings and other pipe and transcoder settings

f. Configure and enable TRANS_DDI_FUNC_CTL
g. Configure and enable TRANS_CONF

7. **Enable Port**
    1. Configure voltage swing and related IO settings. Refer to the DDI Buffer section.
    2. If not type-C static connection, configure PORT_CL_DW10 Static Power Down to power up all lanes of the DDI.
    3. Configure and enable DDI_BUF_CTL
    4. Wait for DDI_BUF_CTL DDI Idle Status = 0b (Not Idle), timeout after 500 us.

Audio can be enabled after everything is complete. Follow the audio enable sequence in the audio registers section.

Notes

If the mode set fails, follow the disable sequence to disable everything that had been enabled up to the failing point.

Disable Sequence

Audio must be disabled first. Follow the audio disable sequence in the audio registers section.

1. **Disable Planes, Pipe, and Transcoder**
    a. If VGA
        i. Set VGA I/O register SR01 bit 5 for screen off
        ii. Wait for 100 us
    b. Disable planes (VGA or hires)
    c. Disable TRANS_CONF
    d. Wait for off status in TRANS_CONF, timeout after two frame times
    e. Disable TRANS_DDI_FUNC_CTL with DDI_Select set to None
    f. Disable panel fitter
2. **Disable Port**
    1. Disable DDI_BUF_CTL
    2. Wait 8 us or poll on DDI_BUF_CTL Idle Status for buffers to return to idle
    3. Configure Transcoder Clock Select to direct no clock to the transcoder
3. **If IO power is controlled through PWR_WELL_CTL - Disable IO Power**
    a. Disable PWR_WELL_CTL DDI IO Power Request for the DDI that was used used
4. **Disable Port PLL**
    a. If PLL to port mapping is flexible, configure PLL to port mapping to direct no clock to the DDI
    b. If this PLL is no longer needed, disable it
5. **Type-C static connection: Disable AUX I/O Power for the port**

6. **Type-C static connection: Clear DFLEXDPCSSS.DPPMSTC to '0' for the port**
7. **Disable Power Wells**
    1. Disable power wells that are no longer required, following the Sequences for Power Well

Sequences for WD

Enable Sequence

Display must already be initialized

1. **Enable Power Wells**
    1. Based on the resources to be used, enable the appropriate power wells following the Sequences for Power Wells
2. **Enable Planes, Pipe, and Transcoder**
    a. Configure WD_STRIDE, WD_SURF, and WD_TAIL_CFG
    b. Configure and enable planes (VGA or hires). This can be done later if desired.
    c. If VGA - Clear VGA I/O register SR01 bit 5
    d. Enable panel fitter if needed (must be enabled for VGA)
    e. Configure transcoder timings and other pipe and transcoder settings
    f. Configure and enable TRANS_WD_FUNC_CTL
    g. Configure and enable TRANS_CONF

If the mode set fails, follow the disable sequence to disable everything that had been enabled up to the failing point.

Disable Sequence

If using Triggered Capture Mode (TRANS_WD_FUNC_CTL) wait for the frame to complete (polling WD_FRAME_STATUS Frame Complete or using the WD Frame Complete interrupt) before starting the disable sequence.

1. **Disable Planes, Pipe, and Transcoder**
    a. If VGA
        i. Set VGA I/O register SR01 bit 5 for screen off
        ii. Wait for 100 us
    b. Disable planes (VGA or hires)
    c. If using triggered capture mode, wait for current capture to complete, and do not trigger any more captures
    d. Disable TRANS_CONF
    e. Disable TRANS_WD_FUNC_CTL
    f. Disable panel fitter
2. **Disable Power Wells**

1. Disable power wells that are no longer required, following the Sequences for Power Wells

Sequences for Display C5 and C6

Display C5 (DC5) is a power saving state where hardware dynamically disables power wells and PLLs and saves the associated registers.

DC5 can be entered when software allows it, only certain power wells are enabled, and hardware detects that all pipes are disabled or low power pipe(s) are enabled with PSR active.

Display C6 (DC6) is a deeper power saving state where hardware dynamically disables power and saves the associated registers.

DC6 can be entered when software allows it, the conditions for DC5 are met, and the PCU allows DC6.

DC6 is required for S0ix.

The context save and restore program is reset on cold boot, warm reset, PCI function level reset, and hibernate/suspend.

Pipe 3D LUT contents are lost during DC5 and DC6.

Ports A IO/PHY programming is preserved when DC6 is enabled. The programming for other combo PHY ports may not be. After disabling DC6, before enabling the other ports, follow the combo phy initialization sequence and enable Aux IO Power for those ports.

Sequence to Allow DC5 or DC6

1. Load the correct stepping specific Display Context Save and Restore (CSR) program from the binary package.
    a. Read the package header and extract the correct individual firmware. Binary package format details can be found in sections below.
    b. Skip the header section at the start of the program binary.
    c. Copy the payload into Display CSR Program Storage.
    d. Perform the MMIO writes specified in the header section.
2. Configure display engine to disable power wells from table below, following the appropriate mode set disable sequences for any ports using those power wells. This can be done earlier if desired. DMC will prevent entry if these wells are not disabled.
    - Disable power wells PG3 and greater
3. Set display register 0x45520 bits [1:0] to 2'b11. The bits do not need to be cleared at any time.
4. Set DC_STATE_EN Dynamic DC State Enable = "Enable up to DC5" for DC5 or "Enable up to DC6" for DC6.

Programming Note	
Context:	Display C5 and C6
Do not switch between "Enable up to DC5" and "Enable up to DC6" without moving to "Disable" and reloading the CSR program in between. Disable DC5/DC6 during mode set and re-enable after the mode set programming is completed. Disable DC5 and DC6 before a DDI AUX channel transaction is sent. MMIO accesses have more latency when DC5/DC6 is enabled. For optimal performance, disable DC5/DC6 when programming a set of registers and re-enable them after the programming is completed.	

Sequence to Disallow DC5 and DC6

1. Set DC_STATE_EN Dynamic DC State Enable = "Disable".

MIPI DSI

Hardware does not support DC5 or DC6 with MIPI DSI enabled.

DMC Firmware Package

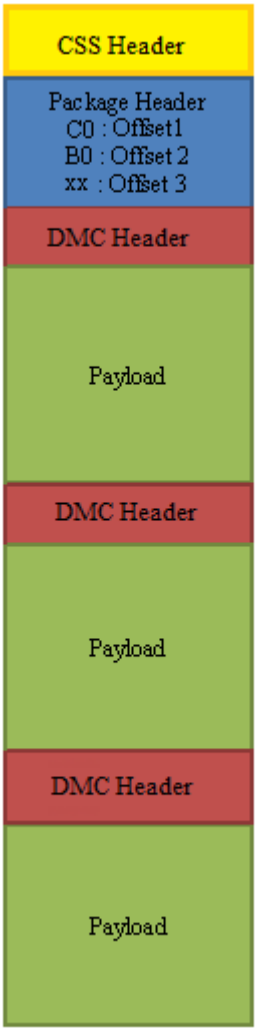
Display Micro-Controller firmware package includes all the firmwares that are required for different steppings of the product. The stepping dependent firmwares are all packaged and released as a single binary package. The package contains the CSS header, followed by the package header and the actual DMC firmwares.

Packaged firmware uses the following naming convention - <project>_dmc_ver<major>_<minor>.bin. The major version will get incremented whenever there is a change in the header layout and would require an update to the driver firmware loading module.

Major version 2

CSS Header	1.0
Package Header	2
DMC Header	3

Package Layout



CSS Header

```
typedef struct _CssHeader {
uint32_t moduleType; // 0x09 for DMC
uint32_t headerLen; // CSS header length in dwords
uint32_t headerVer; // 0x10000
uint32_t moduleID; // Not used
uint32_t moduleVendor; // Not used
uint32_t date; // YYYYMMDD(YYYY « 16 + MM « 8 + DD)
uint32_t size; // Total dmc fw binary size in dwords - (CSS_Headerlen +
PackageHeaderLen + dmc FWsLen)/4
uint32_t keySize; // Not used
uint32_t modulusSize; // Not used
```

```
uint32_t exponentSize; // Not used
uint32_t reserved1[12]; // Not used
uint32_t version; // Major Minor
uint32_t reserved2[8]; // Not used
uint32_t uKernelHeaderInfo; // Not used
} CssHeader;
```

Package Header

Package header contains the firmware/stepping mapping table and the corresponding firmware offsets to the individual binaries, within the package. Mapping table will list the exceptions first, followed by the default entries. An Offset value of "0xFFFFFFFF" in the mapping table indicates that there is no firmware available/supported for that stepping. The offsets to the individual binary are DWord aligned. The first individual binary starts at an offset value of "0x00000000" after the CSS Header and the Package Header.

Stepping/Version mapping example

Stepping	FW Version
A1	1.1
B*	1.6
**	2.3

DMC IDs

DMC	DMC ID
Main	0
Pipe A	1
PipeB	2
PipeC	3
PipeD	4

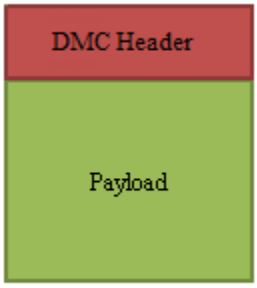
```
typedef struct _PackageHeader {
uint8_t headerLen; // DMC package header length in dwords
uint8_t headerVer; // 0x02
uint8_t reserved[10]; // Reserved
uint32_t numEntries; // Number of valid entries in the FWInfo array below
struct _FWInfo_ {
uint8_t reserved1; // Reserved
uint8_t dmc_id; // DMC ID (Main-0, pipeA-1, pipeB-2, pipeC-3, pipeD-4)
```

```
char stepping; // Stepping (A, B, C, ..., *). * is a wildcard
char substepping; // Sub-stepping (0, 1, ..., *). * is a wildcard
uint32_t offset; // FW offset within the package in dwords
uint32_t reserved2; // Reserved
} FWInfo[32];
} PackageHeader;
```

DMC firmware binary

Each individual DMC firmware binary has a header followed by a payload whose size is specified in the header section. Along with the version, length, firmware size etc. the header section also specifies a list of MMIO addresses and data. These MMIO write cycles, in the 0x80000 - 0x8FFFF address range, should be executed as part of the initial CSR program setup.

DMC firmware Layout




```
for i = 1 to <mmioCount>
Perform MMIO write to address <mmioaddr[i]> with data <mmiodata[i]>
```

DMC Header - Version 3

```
 typedef struct _DMCHeader {
       uint32_t     signature;                        // 0x40403E3E
       uint8_t             headerLen;                        // DMC specific header length in
dwords       uint8_t             headerVer;                        // 0x03       uint16_t
       dmccVer;                        // dmcc compiler version
       uint32_t     project;                          // Major, Minor: 0x00090000, 0x00090001,
0x000A0000
       uint32_t     fwSize;                                 // Firmware program size in dw
       uint32_t     fwVersion;                        // Major Minor
       uint32_t     startMMIOAddr;                           // DMC RAM start MMIO address
       uint32_t     reserved[9];
       uint8_t             dfile [32];                           // .d file name (encoded using
ascii - 40)

  uint32_t mmioCount; // number of mmio uint32_t mmioaddr[20]; // MMIO address uint32_t
mmiodata[20]; // MMIO data

} DMCHeader;
```

Header field	Value
project	0x000C0000

eDP Sequences for Display C3 Clock Off

eDP Display C3 Clock Off (DC3co) is a power saving state where hardware dynamically disables the CDCLK PLL, DBUF power, and Panel Fitter power when PSR2 is idle between frame updates.

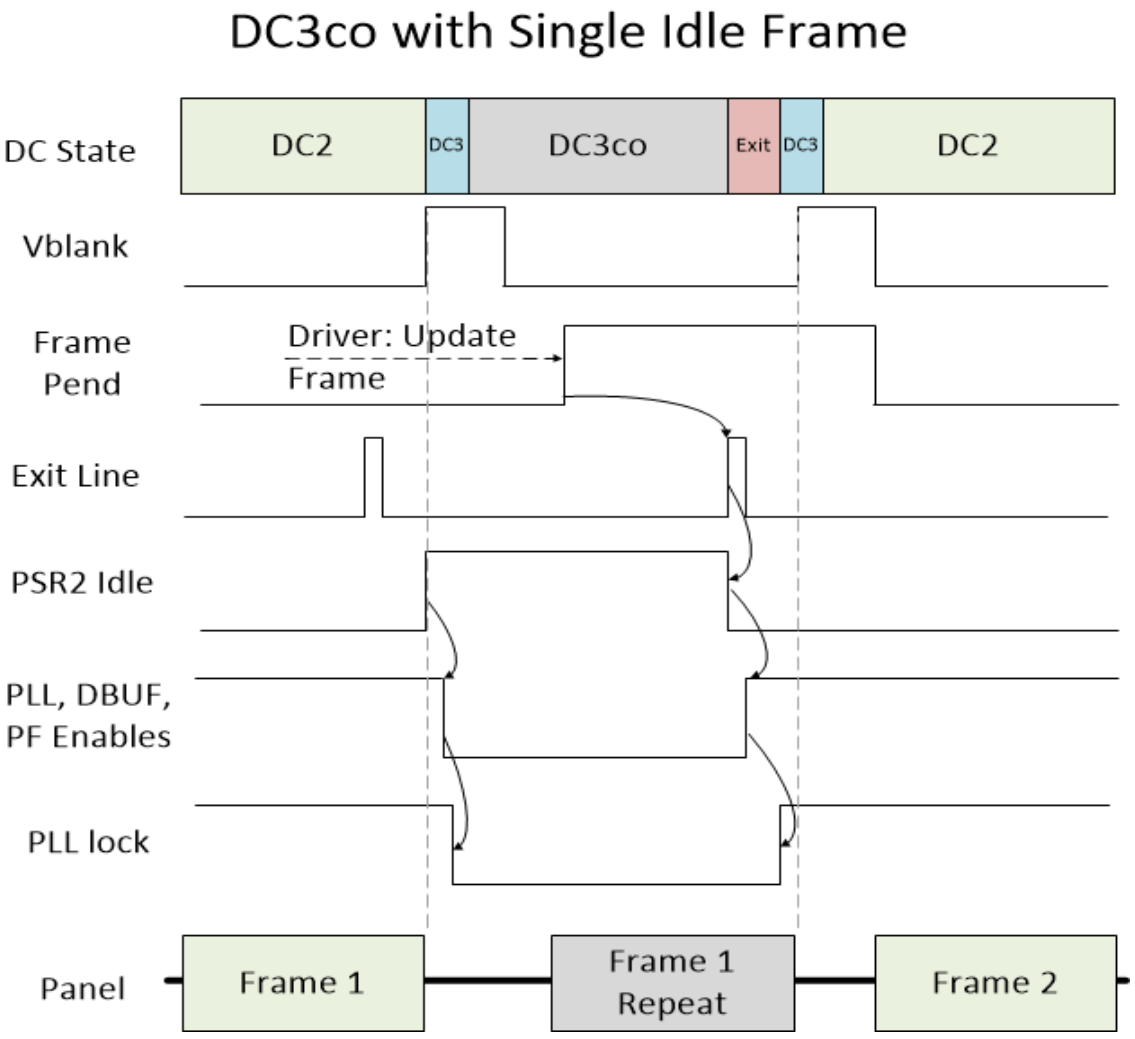
DC3co is entered when software allows it and hardware detects that PSR2 is idle.

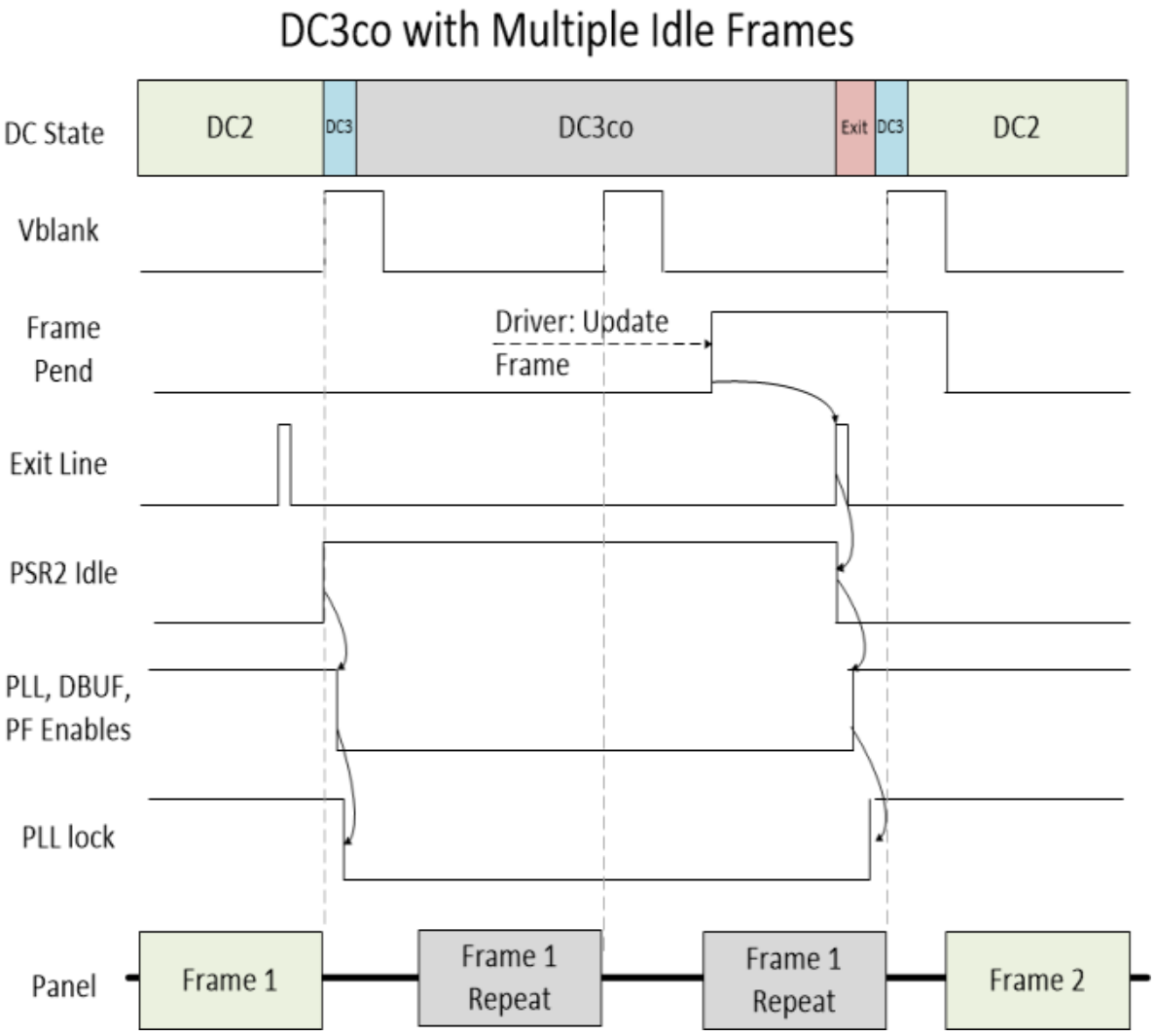
When the CDCLK PLL is disabled, display engine runs CDCLK from a slower SoC clock to maintain accessibility.

DC3co is exited when software disallows it or PSR2 becomes active to send an updated frame.

DC3co is controlled by a context save and restore program which has to be loaded before allowing DC3co and is reset on DC9, cold boot, warm reset, PCI function level reset, and hibernate/suspend.

DC3CO will not work with pipeA and DDIB combination. It only works with pipeA and DDIA.





Requirements

Audio codec idle and disabled.

External displays disabled. WD transcoders and DP/HDMI transcoders must be disabled.

Backlight cannot be driven from the display utility pin. It can be driven from the south display.

This feature should be enabled only on eDP.

DC5 and DC6 not allowed when this feature is enabled.

- DC5 and DC6 saves more power, but cannot be entered during usages like video playback because there are not enough idle frames in a row to meet most panel PSR2 deep sleep entry requirements (typically 4 frames). Software must select between DC5 and DC6 versus DC3co based on flip rate or other hints.

PSR2 deep sleep disabled (PSR2_CTL Idle Frames = 0000b).

Disable DC3co before mode set, or other Aux, PLL, and DBUF programming, and do not re-enable until after that programming is completed.

DC3co is only compatible with PSR2. Follow the sequences to enable and disable PSR2 and DC3co at the correct times.

Sequence to Allow DC3co

1. If not already loaded, load the correct stepping specific Display Context Save and Restore (CSR) program from the binary package.
    a. Read the package header and extract the correct individual firmware. Binary package format details explained in the Sequences for Display C5 and C6 section.
    b. Skip the header section at the start of the program binary.
    c. Copy the payload into Display CSR Program Storage.
    d. Perform the MMIO writes specified in the header section.
2. Configure display engine to meet the requirements for DC3co, following the appropriate disable sequences. This can be done earlier if desired.
    1. Disable DC states by programming DC_STATE_EN Dynamic DC State enable = 00b
    2. Disable PSR2 following the Panel Self Refresh section PSR2 disable sequence.
    3. Disable PSR2 deep sleep by programming PSR2_CTL Idle Frames = 0000b
    4. Enable PSR2 following the Panel Self Refresh section PSR2 enable sequence.
3. Configure and enable the PSR2 transcoder's Early Exit line (TRANS_EXITLINE)
    a. DC3co exit time = Polling rate + instruction execution time + delays from driver MMIO access + DBUF enable time + PLL lock time + guardband = 200us
    b. ROUNDUP(DC3co exit time / line time) + 1
4. Set DC_STATE_EN Display Clock Off Enable = 1b.

Sequence to Disallow DC3co

1. Clear DC_STATE_EN Display DC*CO State Status by writing a 0 to it.
2. Set DC_STATE_EN Display Clock Off Enable = 0b.
3. Wait DC3co exit time for disable to complete. See above for exit time latency calculation above.
    - This ensures exit has completed and it is safe to do things such as program PLL or DBUF registers.
4. Disable PSR2 following the Panel Self Refresh section PSR2 disable sequence.
5. Restore PSR2 deep sleep by programming correct allowed Idle Frames in PSR2_CTL.
6. If PSR2 is still needed, re-enable PSR2 following the Panel Self Refresh section PSR2 enable sequence.

Sequences for Display C9

Display C9 (DC9) is a power saving state where the display engine is powered off.

DC9 supports S0ix with more power savings than DC6.

Display software must follow certain programming sequences to allow or dis-allow DC9.

Hardware will dynamically enter and exit DC9 when allowed, saving and restoring some of the display state.

Sequence to Allow DC9

1. Follow Sequence to Disallow DC5.
2. Disable all display engine functions using the full mode set disable sequence on all pipes, transcoders, ports, planes, and power wells above PG1.
3. Disable and mask all graphics interrupts in north and south display.
4. Save state of MMIO display registers.
    - The exact registers depend on software policy.
    - Hardware will save and restore the PCI Config and DGunit registers
5. Follow Sequences to Initialize Display - Un-initialize Sequence.
6. Set DC_STATE_EN DC9 Allow to 1b.

Sequence to Disallow DC9

1. Clear DC_STATE_EN DC9 Allow to 0b.
2. Follow Sequences to Initialize Display - Initialize Sequence.
3. Restore state of MMIO display registers:
    - The exact registers depend on software policy.
    - The context save and restore program is reset on DC9 and has to be restored.
4. Enable and unmask graphics interrupts as needed.

Xe_{D+} Resolution Support

A display resolution is only supported if it meets all the restrictions below for Maximum Pipe Pixel Rate, Maximum Port Link Rate, Maximum Size, Maximum Bandwidth, and Maximum Watermark.

Core Display Clock (CDCLK)

Refer to the Clocks section for details of the frequencies.

Maximum Pipe Pixel Rate

The display resolution must fit within the maximum pixel rate output from the pipe.

For each enabled plane on the pipe {

    If plane scaling enabled {

        Plane Ratio = 1 / Plane down scale amount

Refer to the Watermark Calculations section, Scaling sub-section, for down scale calculation.

}

Else {

Plane Ratio = 1

}

}

Pipe Ratio = Minimum Plane Ratio of all enabled planes on the pipe

If pipe scaling is enabled {

Pipe down scale amount = Horizontal down scale amount * Vertical down scale amount

Refer to the Watermark Calculations section, Scaling sub-section, for down scale calculation.

Pipe Ratio = Pipe Ratio / Pipe down scale amount

}

Pipe maximum pixel rate = 2 * CDCLK frequency * Pipe Ratio

YUV420 Full Blend Mode

Pipe Maximum Y channel pixel rate = 2 * CDCLK frequency * Y channel Pipe ratio

Resolutions Requiring Joined Pipes

For resolutions requiring multiple pipes to be combined together (pipe joining), the pixel rate seen by each pipe is 1/<# of joined pipes> of the pixel rate of the full resolution.

The overlapping excess horizontal pixels added for scaling smoothly across the seam between pipes do not impact the pixel rate.

For example: 7680x4320 CVT1.2 RB1 pixel rate is 2089.75 MHz. That is split across 2 pipes, so each pipe is 3840x4320 with a pixel rate of 1044.875 MHz.

Maximum Port Link Rate

The display resolution must fit within the maximum link rate for each port type.

Refer to the project overview sections for the maximum rates.

YUV420 Full Blend Mode

Maximum Port Link Rate = CDCLK frequency * Y channel Pipe Ratio * (bits per color / 8)

Maximum Size

There are limits on the maximum horizontal and vertical size.

Plane stride maximum listed in PLANE_STRIDE register.

Multiple pipes can be joined together side by side in the display engine to drive a single port or a tiled panel.

Unjoined pipe source size, plane size, and pipe active size maximum vertical 4096.
Unjoined pipe source size, plane size, and pipe active size maximum horizontal 5120.
Joined pipes active size maximum is 7680x4320, with each pipe and maximum horizontal plane size at 7680/<# of joined pipes> and vertical size at 4320, plus extra overlapping horizontal pixels for scaling excess.
PSR2 maximum pipe horizontal active size 5120 pixels. PSR2 has additional restrictions on minimum vblank size and horizontal total line time. See the PSR section for those.
LACE DPST maximum size 5120x3200 pixels.
Pipe scalers maximum horizontal source size 5120 pixels.

Maximum Bandwidth

Refer to the Bandwidth Restrictions section

Maximum Watermark

The display resolution must not exceed the level 0 maximum watermark value. See the section on Watermark Programming.

Example Display Resolution Capabilities

Refer to the project overview sections for the common resolutions that meet all the resolution restrictions.

Examples

Example pipe pixel rate

Plane 1 enabled at 64bpp and plane down scale amount 1.25, plane 2 enabled at 32bpp, no pipe scaling enabled, and CDCLK 312 MHz:

Plane 1 ratio = 1/1.25 = 0.8

Plane 2 ratio = 1

Pipe ratio = Minimum[1, 0.8] = 0.8

Pipe maximum pixel rate = 2 * 312 MHz * 0.8 = 499.2 MHz

Bandwidth Restrictions

There are bandwidth restrictions that limit the display resolution and configuration. A display resolution is only supported if it meets all the restrictions below for Maximum Memory Read Bandwidth found from the Available Memory Bandwidth Calculation and Required Memory Bandwidth Calculation, Maximum Data Buffer Bandwidth, and Maximum Pipe Read Bandwidth.

Maximum Memory Read Bandwidth

The display required memory bandwidth for a resolution configuration must not exceed the available memory bandwidth, considering efficiency and GV points.

Calculate the available memory bandwidth and the required memory bandwidth across all GV points. Refer to the SAGV chapter for the flow to find GV points and restrict those that do not provide enough available memory bandwidth. If no points provide enable available memory bandwidth, then the display configuration can not be supported.

Note: These calculations are valid even if GV points are disabled. They then use the single available GV point.

Available Memory Bandwidth Calculation

This calculates the memory bandwidth available for display. It considers the efficiency, impacted by memory type and configuration details and number of enabled planes, and differences between GV points.

The InitializeBandwidthAndPlanes function populates arrays with the bandwidth and number of planes that can be enabled in several groupings for each GV point. See the SAGV chapter for information on retrieving the memory configuration to populate the structs at the start of the function.

The CalculateMaxBW function returns the bandwidth available for a specified number of enabled planes and GV point.

```
#define MIN(a, b) (a < b ? a : b)
 #define MAX(a, b) (a > b ? a : b)

 inline int divide_roundup(int n, int d) {
   return (n + d - 1) / d;
 }

 inline int isYtile() {
   return TRUE if any plane is using tileY surface format, and do not count tile4 as tileY
 }

 struct PCODE_GLOBAL_INFO
 {
   uint32_t DdrType : 4;
   uint32_t NumOfPolulatedChannels : 4;
   uint32_t NumOfEnabledQgvPoints : 4;
   uint32_t NumOfEnabledPsf0Points : 4;
 };
 PCODE_GLOBAL_INFO PcodeGlobalInfo();

 struct PCODE_QGV_POINT
 {
   uint32_t Dclk : 16; // in 16.66666MHz units
   uint32_t tRP : 8; // In DCLKs
   uint32_t tRCD : 8; // In DCLKs
   uint32_t tRDPRE : 8; // In DCLKs
   uint32_t tRAS : 9; // In DCLKs
```

```
};
PCODE_QGV_POINT PcodeQgvPoint(int point);

// Number of plane groups.
#define NUM_GROUPS 6

// Max number of Qclk GV points
#define NUM_SAGV_POINTS 4

// These state variables should be stored in a device context. The initialize function will
generate the records
int g22_deratedbw[NUM_SAGV_POINTS][NUM_GROUPS];
int h24_maxplanes[NUM_GROUPS];

// Find out the memory timings for each of the SAGV points and initialize the state variables
above.
void InitializeBandwidthAndPlanes()
{
  // Static config for all gens
  int c3_derating = 10;
  int c2_ipqdepthpch = 16;
  int c25_deprogbwpclimit = 60; // %
  int c7_mempagesize = 16;
  int g4_numchannels;
  int g4_maxnumchannels;

  // The following require the appropriate SAGV timings, however they can be precalculated
for each of the SAGV points
  int g3_clk[NUM_SAGV_POINTS];
  int g7_tRP[NUM_SAGV_POINTS];
  int g8_tRCD[NUM_SAGV_POINTS];
  int g10_tRDPRE[NUM_SAGV_POINTS];
  int g11_tRC[NUM_SAGV_POINTS];

  PCODE_GLOBAL_INFO mem_global_info = PcodeGlobalInfo();
  g4_numchannels = mem_global_info.NumOfPolulatedChannels;
  MEM_TYPE mem_type = (MEM_TYPE)mem_global_info.DdrType;

  for (uint32_t sagv = 0; sagv < NUM_SAGV_POINTS; sagv++)
  {
    if (sagv < mem_global_info.NumOfEnabledQgvPoints)
    {
      PCODE_QGV_POINT gv_point_info = PcodeQgvPoint(sagv);
      g3_clk[sagv] = (16667 * gv_point_info.Dclk + 500) / 1000;
      g7_tRP[sagv] = gv_point_info.tRP;
      g10_tRDPRE[sagv] = gv_point_info.tRDPRE;
      g11_tRC[sagv] = gv_point_info.tRP + gv_point_info.tRAS;
      g8_tRCD[sagv] = gv_point_info.tRCD;
    }
    else
    {
      // Initialize any undefined GV points to zero
      g3_clk[sagv] = 0;
      g7_tRP[sagv] = 0;
      g10_tRDPRE[sagv] = 0;
      g11_tRC[sagv] = 0;
      g8_tRCD[sagv] = 0;
    }
  }

  // Detect the maximum clock from the SAGV points
  int clk_max = 0;
  for (uint32_t sagv = 0; sagv < NUM_SAGV_POINTS; sagv++)
    clk_max = MAX(clk_max, g3_clk[sagv]);
```

```
// Project specific config
int c4_deBurst;
int c14_displayrtids;
int c24_deprogbwlimit;
int g9_tBL;
int g6_channelwidth;
int g5_deinterleave;


  c4_deBurst = 16;
  c14_displayrtids = 256;
  switch (mem_type)
  {
  case DDR4:
    g4_maxnumchannels = 2;
    g5_deinterleave = isYtile() ? 1 : 2;
    g9_tBL = isYtile() ? 8 : 4;
    g6_channelwidth = 64;
    break;
  case DDR5:
    g4_maxnumchannels = 4;
    g5_deinterleave = isYtile() ? 1 : 2;
    g9_tBL = isYtile() ? 16 : 8;
    g6_channelwidth = 32;
    break;
  case LPDDR4:
  case LPDDR5:
    g4_numchannels *= 2; // convert to number of subchannels
    g4_maxnumchannels = 8;
    g5_deinterleave = isYtile() ? 2 : 4;
    g9_tBL = 16;
    g6_channelwidth = 16;
    break;
  default:
    assert(0); // not supported
    break;
  }

  c24_deprogbwlimit = 25;
if (g4_numchannels < g4_maxnumchannels)
{
  // Halve the DE interleave if channels are not present
  g5_deinterleave = MAX(g5_deinterleave / 2, 1);
}


int g13_peakbw = g4_numchannels * (g6_channelwidth / 8) * clk_max;
int g14_maxdebw = MIN(c24_deprogbwlimit * 1000, (g13_peakbw * c25_deprogbwpclimit) / 100);
int g17_ipqdepth = MIN(c2_ipqdepthpch, c14_displayrtids / g4_numchannels);

h24_maxplanes[0] = 0;
int c8_4kpagespermempage = c7_mempagesize / 4;
int c9_clperchperblock = (8 / g4_numchannels) * g5_deinterleave;
int g16_clperchgroup = c8_4kpagespermempage * c9_clperchperblock;

// Figure out the max plane groups
for (int i = 0; i < NUM_GROUPS; i++)
{
  int g18_clpchgroup = (c4_deBurst * g5_deinterleave / g4_numchannels) << i;
  if (i < NUM_GROUPS - 1)
  {
    if (g18_clpchgroup < g16_clperchgroup)
```

```
        {
          h24_maxplanes[i + 1] = (g17_ipqdepth - g18_clpchgroup) / g18_clpchgroup;
        }
        else
        {
          h24_maxplanes[i + 1] = 0;
        }
      }
      for (uint32_t sagv = 0; sagv < NUM_SAGV_POINTS; sagv++)
      {
        int g20_CT = MAX(g11_tRC[sagv], g7_tRP[sagv] + g8_tRCD[sagv] + (g18_clpchgroup - 1) *
g9_tBL + g10_tRDPRE[sagv]);
        int g21_BW = g18_clpchgroup * 32 * g4_numchannels * g3_clk[sagv] / g20_CT;
        g22_deratedbw[sagv][i] = MIN(g14_maxdebw, ((100 - c3_derating) * g21_BW) / 100);
      }
    }
  }

  // This function will return the maximum bandwidth that can be consumed by display when the
specified number of planes are enabled
  // at the specified SAGV point. It should be used at run time to limit the modes and the
planes allowed.
  int CalculateMaxBW(int numPlanes, int point)
  {
    for (int i = NUM_GROUPS - 1; i >= 0; i--)
    {
      if (numPlanes <= h24_maxplanes[i])
      {
        return g22_deratedbw[point][i];
      }
    }
    return g22_deratedbw[point][0];
  }
```

Required Memory Bandwidth Calculation

This calculates the display required memory bandwidth.

Note on planar formats: The NV12 format counts as 2 planes of 2 Bpp each. The P01x formats count as 2 planes of 4 Bpp each. Vertical sub-sampling does not reduce this calculation because the bandwidth has to be counted over a few microsecond period.

DBUF maximum data buffer bandwidth MB/s = CDCLK frequency MHz * 64 Bytes

For each DBUF {

DBUF maximum pipe bandwidth MB/s = DBUF maximum data buffer bandwidth / number of enabled pipes using this buffer

For each pipe using this buffer {

Pipe cumulative bytes per pixel = 0

DBUF maximum plane bandwidth MB/s = DBUF maximum pipe bandwidth / number of enabled planes

For each plane enabled on the pipe { // cursor can be ignored because it only impacts bandwidth for a very short time

Plane required bandwidth MB/s = pixel rate MHz * source pixel format in bytes * plane down scale amount * pipe down scale amount

Display required memory bandwidth MB/s += Plane required bandwidth

Pipe cumulative bytes per pixel += plane source pixel format in bytes

If plane required bandwidth > DBUF maximum plane bandwidth {Return: Failure maximum data buffer bandwidth is exceeded}

}

If pipe cumulative bytes per pixel> (CDCLK frequency MHz /(pixel rate MHz * plane down scale amount * pipe down scale amount)) * 51.2 {Return: Failure maximum pipe read bandwidth is exceeded}

}

}

Return: Display required memory bandwidth MB/s

Maximum Data Buffer Bandwidth

The display resolution must not exceed the maximum bandwidth from each display data buffer (DBUF). There are multiple buffers that can be enabled, and planes can be allocated data blocks from multiple. See the Display Buffer Programming section for requirements on how and when to use multiple buffers.

This is calculated as part of the Required Memory Bandwidth Calculation where it returns a failure if maximum data buffer bandwidth is exceeded. The calculation for this check can be separated out if required.

Maximum Pipe Read Bandwidth

The display resolution must not exceed the maximum memory read bandwidth from each display pipe. This restriction is met by limiting the cumulative bytes per pixel for the pipe, which accounts for the limits of memory reads, translation reads, and data buffer reads.

This is calculated as part of the Required Memory Bandwidth Calculation where it returns a failure if maximum pipe read bandwidth is exceeded. The calculation for this check can be separated out if required.

Clocks

Xe_D Clocks

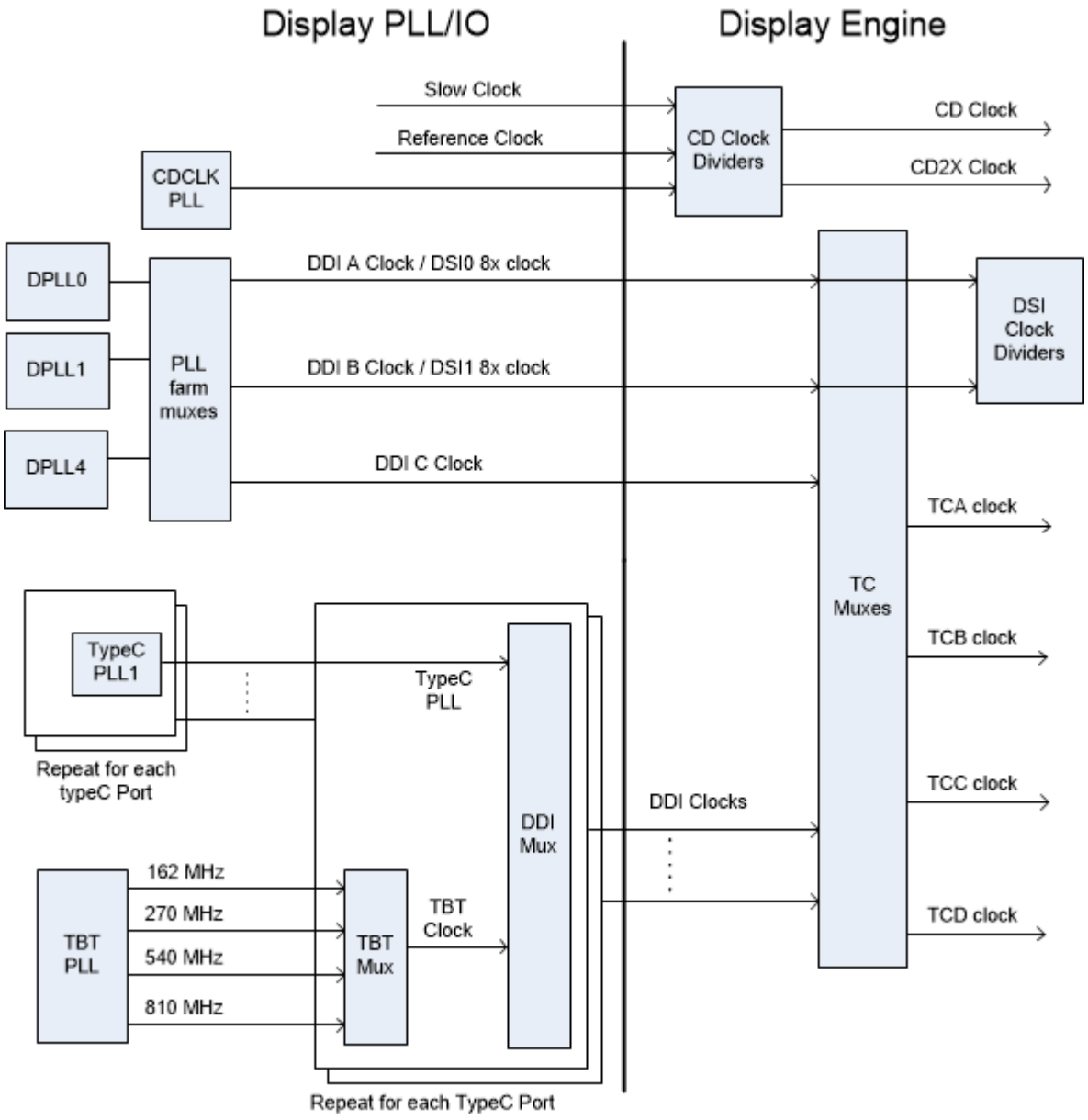
Registers

Registers
CDCLK_CTL
CDCLK_PLL_ENABLE
DPLL_CFGCR0
DPLL_CFGCR1
DPLL_SSC
DPCLKA_CFGCR0
DPLL_ENABLE
TRANS_CLK_SEL
DDI_CLK_SEL
TIMESTAMP_CTR
DKL_PLL_DIV0
DKL_PLL_DIV1
DKL_PLL_LF
DKL_PLL_FRAC_LOCK
DKL_REFCLKIN_CTL
DKL_CLKTOP2_HSCLKCTL
DKL_CLKTOP2_CORECLKCTL1

Overview of Display Clock Paths

The display engine clocking structure has multiple PLLs and clocks. The flow is from PLL to DDI (port) clock to transcoder clock.

PLL Arrangement



Display Engine Clock Reference

There is one display engine clock reference.

	Reference
Usage	Reference for the PLLs and for miscellaneous timers in display engine
Frequency	19.2, 24, 38.4 MHz Non-SSC (Register DSSM Reference Frequency indicates the frequency)
Default after reset	Enabled
Programming	Not programmable by display software.

Combo PHY PLLs

DPLLs are used for the combo PHY ports.

The PLL output frequencies are 5x the symbol/TMDS rate, which is 1/2 the bit rate.

The PLL output is divided by 5 to become the symbol/TMDS clock frequency used in the display engine.

Both edges of the PLL output are used in the I/Os to double the frequency to bit clock rate.

A single PLL output may be used by multiple ports simultaneously if those ports all require the same frequency and spread characteristics.

	Combo PHY PLLs (DPLLs)
Usage	Sources for DSI0, DSI1 clocks, and DDI clocks for the combo PHY ports (non-typeC)
Input	Reference clock
Frequency	Programmable eDP/DP link bit rates: 1.62, 2.16, 2.7, 3.24, 4.32, 5.4, 6.48*, 8.1* GHz, SSC and Non-SSC HDMI/DVI symbol rates: 20 to 600 MHz, Non-SSC** DSI 8x clock rates: 0.340 - 2.5 GHz, SSC and Non-SSC *OEM must use VBT to specify a miximum that is tolerated by the board design. *Frequencies between 500 and 533.2 MHz (exclusive) not supported
Default after reset	Disabled
Programming	Must be programmed by software when enabling and disabling a display output. See the section on Port Clock Programming. PLLs are automatically disabled and re-enabled by hardware for some power states.

Thunderbolt (TBT) PLL

DPLL2 is used for thunderbolt ports.

The PLL output is the symbol clock frequencies used in the display engine.

This PLL is used to deliver symbols from the display engine to the thunderbolt controller. The thunderbolt controller will re-clock and convert the symbols to the thunderbolt format.

The PLL outputs all 4 thunderbolt frequencies simultaneously.

	TBT PLL
Usage	Source for DDI clocks on typeC ports when using thunderbolt
Input	Reference clock
Frequency	PLL fixed frequency divided to give simultaneous DP thunderbolt link rates: 162, 270, 540, and 810 MHz, Non-SSC* *SSC, if needed, will be applied on the link by the thunderbolt controller
Default after reset	Disabled
Programming	Must be programmed by software when enabling and disabling a display output. See the section on Port Clock Programming.

TypeC PLLs

The TypeC PLLs are used for Type C ports. There are 2 TypeC PLLs per port, but display engine can only use one PLL, which the register offsets select.

The PLL output frequencies are 5x the symbol/TMDS rate, which is 1/2 the bit rate.

The PLL output is divided by 5 to become the symbol/TMDS clock frequency used in the display engine.

Both edges of the PLL output are used in the DDI I/Os to double the frequency to bit clock rate.

The TypeC PLL is also called a Dekel (DKL) PLL.

	TypeC PLL per port
Usage	Sources for DDI clocks on type C ports when using DP alternate mode or legacy DP or HDMI connectors
Input	Reference clock
Frequency	Programmable eDP/DP/DP-altmode link bit rates: 1.62, 2.7, 5.4, 8.1 GHz, SSC and Non-SSC HDMI/DVI symbol rates: 20 to 600 MHz, Non-SSC
Default after reset	Disabled

Programming	Must be programmed by software when enabling and disabling a display output. See the section on Port Clock Programming.

DDI clocks

There is one DDI clock tied to each DDI port.

A single DDI clock output may be used by multiple transcoders simultaneously for DisplayPort Multi-streaming.

	Combo PHY (non-typeC) DDI clocks	**TypeC DDI clocks**
Usage	DDI ports I/O bit clock and symbol/TMDS clock, source for transcoder clocks	
Input	Programmable selection between the DPLLs	Programmable selection between TypeC PLLs and TBT PLL outputs
Frequency	DPLL frequency	TypeC PLLs or TBT frequency
Default after reset	Disabled	
Programming	Must be programmed by software when enabling and disabling a display. Programming is done through the DPLL, PORT_CLK_SEL, TypeC, and TBT registers. See the section on Port Clock Programming.	

Transcoder Clocks

There is one Transcoder clock tied to each display transcoder.

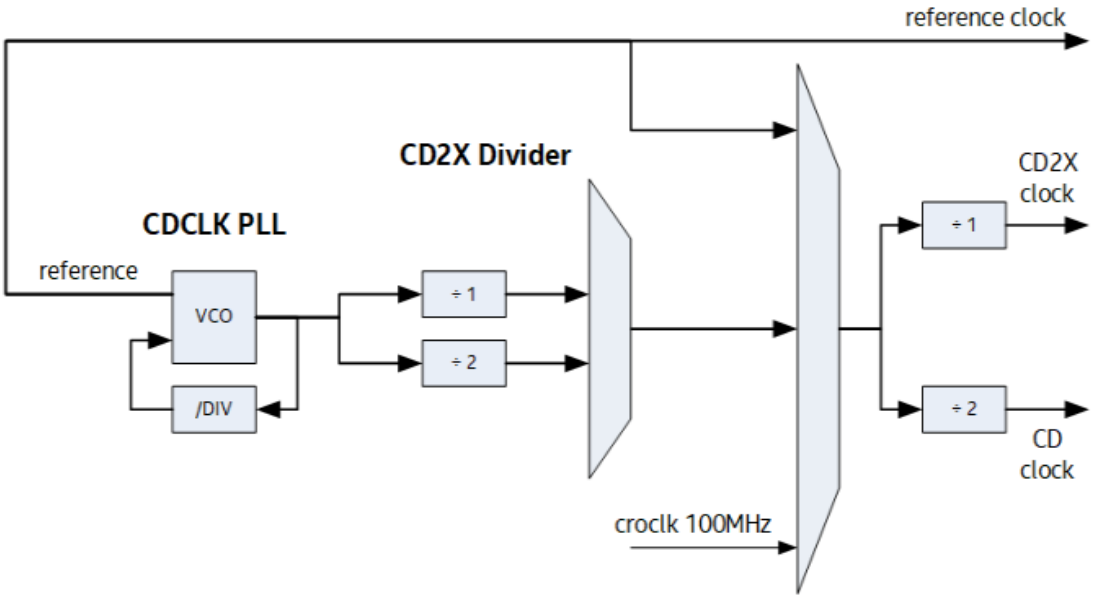
	Transcoder clocks
Usage	Transcoder symbol/TMDS clocks
Input	Programmable selection between DDI clocks
Frequency	PLL output frequency divided by 5
Default after reset	Disabled
Programming	Must be programmed by software when enabling and disabling a display. Programming is done through the TRANS_CLK_SEL registers. See the section on Port Clock Programming.

CD Clock

CD clock refers to the Core Display clock which includes the Core Display 1X Clock (CD clock, CDclk, cdclk, CDCLK) and the Core Display 2X Clock (CD2X clock, cd2xclk, CD2XCLK).

	CD clock
Usage	Clocking for most display engine functions
Input	CDCLK PLL output or reference clock.
Frequency	CDCLK PLL disabled - reference or slow clock divided by 2 CDCLK PLL enabled - Frequencies in table above
Default after reset	Running on the reference clock
Programming	Must be programmed by software when enabling and disabling a display. Programming is done through the CDCLK_CTL and CDCLK_PLL_ENABLE registers.

CD Clock Generation



CD Clock PLL

The CDCLK PLL is the main source for CD clock. A programmable divider inside the PLL controls the PLL frequency. The CD2X Divider after the PLL provides additional dividing to create the CD2X clock used by the display engine. The programmable divider value must be set to account for the reference frequency for in the **DSSM** Reference Frequency field.

When the CDCLK PLL is disabled, the CD clock runs at reference clock divided by 2 to maintain register accessibility. The slow clock (croclk) is used by hardware for some power state transitions.

	CDCLK PLL
Alternate Name	ADPLL or DEPLL
Usage	Source for CD clock
Input	Reference clock
Frequency	Programmable - Frequencies in table below
Default after reset	Disabled
Programming	Must be programmed by software when enabling and disabling a display. May be automatically enabled and disabled by hardware for some power states. Programming is done through the CDCLK_PLL_ENABLE register.

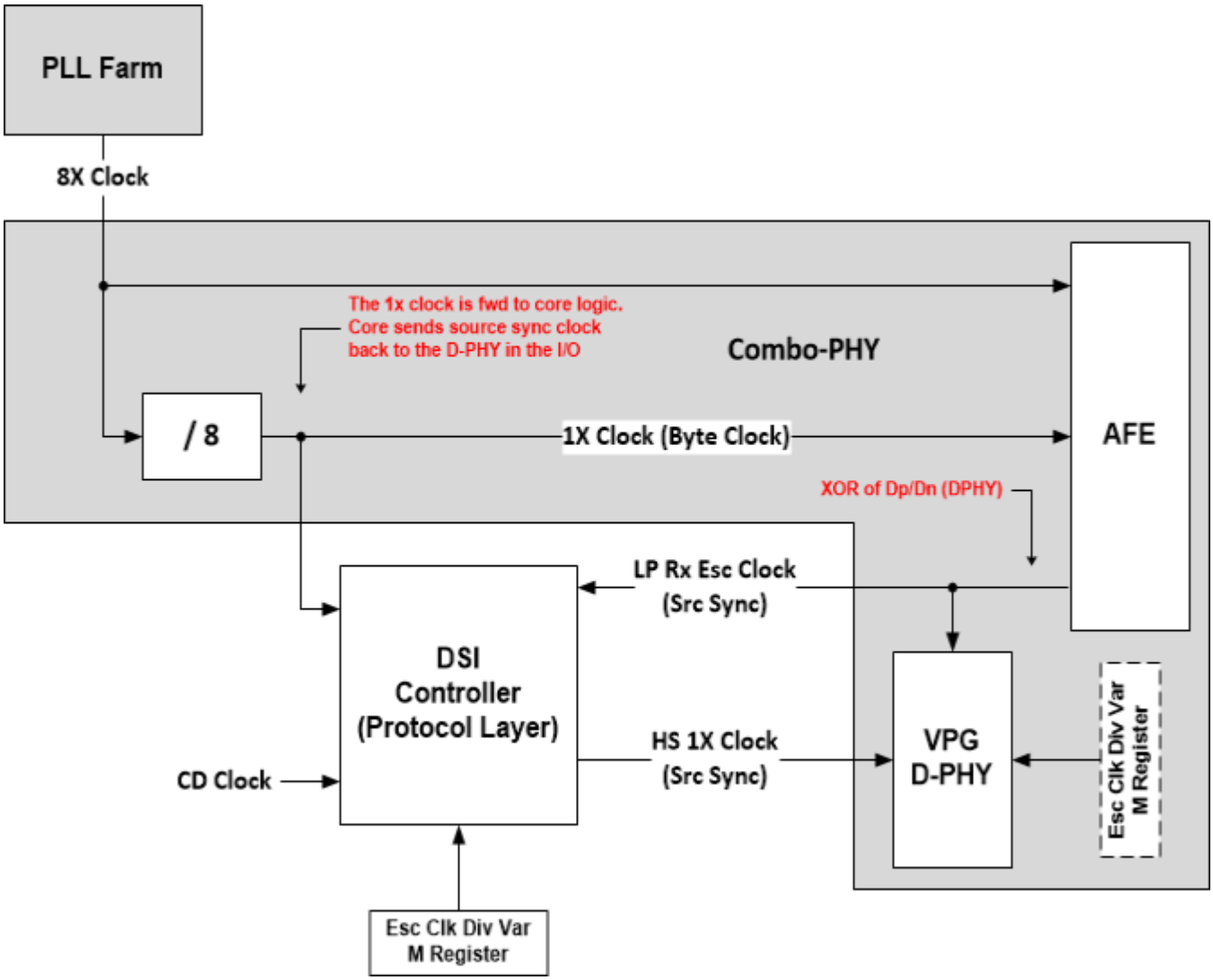
CDCLK PLL Ratio and Divider Programming and Resulting Frequencies

PLL Ratio for 19.2 MHz Reference	PLL Ratio for 38.4 MHz Reference	CD2X Divider	CDCLK MHz
18	9	1	172.8
20	10	1	192
32	16	1	307.2
68	34	2	326.4
58	29	1	556.8
68	34	1	652.8

PLL Ratio for 24 MHz Reference	CD2X Divider	CDCLK MHz
15	1	180
16	1	192
26	1	312
54	2	324
46	1	552
54	1	648

MIPI DSI Clocks

The MIPI DSI clocks are used for the DSI 0 and DSI 1 ports and transcoders.



The DSI port operates within several different clock domains.

- **CD Clock** - The DSI controller receives the Pixel and Software control within this clock domain. The DSI controller clock crosses this information over to the 1X clock domain.
- **8X Clock -** This clock domain is received from the PLL farm and used to transfer the serialized data across the Link (i.e. it is the per-Lane frequency of the DSI Link). Only the I/O block will receive this clock. The IO contains a programmable divider that will divide this clock down to the 1X clock needed by the DSI controller.
- **1X Clock (a.k.a. Byte clock, Word clock)** - This clock domain is used to transfer the HS parallel data (a byte) per Data Lane. Both the DSI Controller and the D-PHY will receive this clock
- **Escape Clock** - This clock is used for LP communication across the DSI Link. Per all MIPI physical layer specifications, this clock has a fixed frequency that should be as close, but not greater than, 20 MHz.
    - **Tx Escape clock** - The Display Engine does not use this clock. Instead it emulates it based off of the DSI_ESC_CLK_DIV register programming and the 1X clock that the Display receives

from the IO. All of the LP Tx signaling from the controller to the DPHY will be driven off of the 1X clock, but will transition at an effective Tx Escape clock frequency.

- **Rx Escape clock** - The IO XOR's together the LP signaling received from the Panel to generate this clock. The IO forwards this clock to the DSI controller

Notes:

1. The MCD block (shaded in grey) delivers a 1X clock to the DSI Controller within the Display Core and to the AFE within the MCD block. The DSI Controller forwards this 1X clock back to the D-PHY within the MCD block along with its data (i.e. the data is source synchronous). The D-PHY will work entirely within the forwarded 1X clock domain. The inlet to the AFE contains a FIFO to handle the phase crossing between the forwarded clock that the D-PHY is operating within and the 1X clock the AFE receives directly from the PLL.
2. The LP Tx Escape clock is derived from a variable M that Software programs within the DSI_ESC_CLK_DIV register. The divider used to live within the IO, so the variable M represents the divisor for generating a clock based off of the 8X clock. Since the divider is within the Display and the clock it is operating on is a 1X clock the divisor becomes "N" which is equal to ceiling(M/8).

	DSI Clocks
Usage	DSI I/O and Transcoder
Input	Programmable selection between the DPLLs
Frequency	PLL clock is the 8X clock. 8X clocks have variable divider to create ~20 MHz escape clocks. Fixed divides for 4X clock and 1X (byte) clocks.
Default after reset	Disabled
Programming	Must be programmed by software when enabling and disabling a display using MIPI DSI.

X^e_D Port Clock Programming

PLL and Clock Usage

This is for mapping the DPCLKA_CFGCR0 bit fields to the PLLs and port names used in display engine.

Note: SKUs with only 2 combo PHY ports should only use DPLL0 and DPLL1.

DPLL0 = PLL available for display combo PHY usage

DPLL1 = PLL available for display combo PHY usage

DPLL2 = TBTPLL, the PLL for thunderbolt

DPLL3 = Not available to display

DPLL4 = PLL available for display combo PHY usage (not available on Y/U SKUs)

DDIA Clock = DDIA and DSI0 (combo port A) symbol clock

DDIB Clock = DDIB and DSI1 (combo port B) symbol clock

DDIC Clock = DDIC (combo port C) symbol clock

TC1 Clock = DDI USBC1 (type C port 1) symbol clock

TC2 Clock = DDI USBC2 (type C port 2) symbol clock

TC3 Clock = DDI USBC3 (type C port 3) symbol clock

TC4 Clock = DDI USBC4 (type C port 4) symbol clock

TC5 Clock = DDI USBC5 (type C port 5) symbol clock

TC6 Clock = DDI USBC6 (type C port 6) symbol clock

TypeC PLL = A PLL in the typeC port, more info in the TypeC PLL Programming section

PLL Frequency Changes

PLL frequency should not be changed while the PLL is enabled.

1. Follow PLL Disable Sequence
2. Follow PLL Enable Sequence using the new frequency

DisplayPort Mode Combo PHY Programming

This programming is for DPLLs when using eDP or DP on combo PHY.

DisplayPort Mode PLL Enable Sequence

1. Enable DPLL power in DPLL_ENABLE.
2. Wait for DPLL power state enabled in DPLL_ENABLE.
    - Should complete immediately.
3. Configure DPLL_SSC to set SSC enable/disable.
4. Configure DPLL_CFGCR0 to set DCO frequency.
5. Configure DPLL_CFGCR1 to set the dividers.
6. Read back DPLL_CFGCR0 or DPLL_CFGCR1, ignoring the data value, this is to ensure writes completed before the next step.
7. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence Before Frequency Change.
8. Enable DPLL in DPLL_ENABLE.
9. Wait for PLL lock status in DPLL_ENABLE.
    - Should complete within 50us.
10. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence After Frequency Change.
11. For each DDI that will use this DPLL
    a. Configure DPCLKA_CFGCR0 to map the DPLL to the DDI.

b. Configure DPCLKA_CFGCR0 to turn on the clock for the DDI. This step and the step before must be done with separate register writes.

DisplayPort Mode PLL Disable Sequence

1. For each DDI that was using this DPLL, configure DPCLKA_CFGCR0 to turn off the clock for the DDI.
2. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence Before Frequency Change.
3. Disable DPLL through DPLL_ENABLE.
4. Wait for PLL not locked status in DPLL_ENABLE.
   - Should complete within 1us.
5. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence After Frequency Change.
6. Disable DPLL power in DPLL_ENABLE.
7. Wait for DPLL power state disabled in DPLL_ENABLE.
   - Should complete immediately.

DisplayPort Mode PLL Values

Link Bit Rate GHz	Reference Clock MHz	DCO Integer	DCO Fraction	Pdiv	Kdiv	Qdiv Mode	Qdiv Ratio
5.4	24	151h	4000h	0010b (3)	001b (1)	0b	00h (1)
2.7	24	151h	4000h	0010b (3)	010b (2)	0b	00h (1)
1.62	24	151h	4000h	0100b (5)	010b (2)	0b	00h (1)
3.24	24	151h	4000h	0100b (5)	001b (1)	0b	00h (1)
2.16	24	168h	0000h	0001b (2)	010b (2)	1b	02h (2)
4.32	24	168h	0000h	0001b (2)	010b (2)	0b	00h (1)
6.48	24	195h	0000h	0010b (3)	001b (1)	0b	00h (1)
8.1	24	151h	4000h	0001b (2)	001b (1)	0b	00h (1)
5.4	19.2 or 38.4	1A5h	7000h	0010b (3)	001b (1)	0b	00h (1)
2.7	19.2 or 38.4	1A5h	7000h	0010b (3)	010b (2)	0b	00h (1)
1.62	19.2 or 38.4	1A5h	7000h	0100b	010b	0b	00h (1)

Link Bit Rate GHz	Reference Clock MHz	DCO Integer	DCO Fraction	Pdiv	Kdiv	Qdiv Mode	Qdiv Ratio
				(5)	(2)		
3.24	19.2 or 38.4	1A5h	7000h	0100b (5)	001b (1)	0b	00h (1)
2.16	19.2 or 38.4	1C2h	0000h	0001b (2)	010b (2)	1b	02h (2)
4.32	19.2 or 38.4	1C2h	0000h	0001b (2)	010b (2)	0b	00h (1)
6.48	19.2 or 38.4	1FAh	2000h	0010b (3)	001b (1)	0b	00h (1)
8.1	19.2 or 38.4	1A5h	7000h	0001b (2)	001b (1)	0b	00h (1)

Example of DisplayPort on DDIA using HBR 2.7 GHz link rate with SSC and reference 24 MHz

This example assumes DPLL0 is available.

1. Enable DPLL0 power in the DPLL0_ENABLE register
    - Power Enable = 1b (Enable)
2. Wait for DPLL0 power state enabled in DPLL0_ENABLE.
3. Configure DPLL0_SSC register
    - sscen = 1b (Enable)
4. Configure DPLL0 in the DPLL0_CFGCR0 register
    - DCO Fraction = 4000h
    - DCO Integer = 151h
5. Configure DPLL0 in the DPLL0_CFGCR1 register
    - Qdiv Ratio = 00h (1)
    - Qdiv Mode = 0b (Disable)
    - Kdiv = 010b (2)
    - Pdiv = 0010b (3)
6. Read back DPLL0_CFGCR0 or DPLL0_CFGCR1 to ensure writes completed
7. Display Voltage Frequency Switching - Sequence Before Frequency Change
    - Skipped if 270 MHz is low enough to not trigger an increase in the voltage requirement
8. Enable DPLL0 in the DPLL0_ENABLE register
    - PLL Enable = 1b (Enable)
9. Wait for PLL lock status in DPLL0_ENABLE
10. Display Voltage Frequency Switching - Sequence After Frequency Change
    - Skipped if 270 MHz is low enough to not trigger an increase in the voltage requirement

11. Configure DPLL0 mapping to DDIA in the DPCLKA_CFGCR0 register
    - DDIA Clock Select = 00b (DPLL0)
12. Turn on DDIA clock in the DPCLKA_CFGCR0 register
    - DDIA Clock Off = 0b (On)

HDMI Mode Combo PHY Programming

This programming is for DPLLs when using HDMI or DVI on combo PHY.

HDMI Mode PLL Enable Sequence

Sequence is the same as for DisplayPort, except SSC must be disabled.

HDMI Mode PLL Disable Sequence

Sequence is the same as for DisplayPort

Formula for HDMI Mode DPLL Programming

Reference frequency = 38.4, 24, or 19.2 MHz (Register DSSM Reference Frequency indicates the frequency). If reference frequency is 38.4, use 19.2 because the DPLL automatically divides that by 2.

Symbol clock frequency MHz = DCO Frequency / (Pdiv * Qdiv * Kdiv) / 5.

AFE clock = PLL output frequency = Symbol clock frequency MHz * 5 = DCO Frequency / (Pdiv * Qdiv * Kdiv).

Pdiv can be 2, 3, 5, or 7.

Kdiv can be 1, 2, or 3.

If Kdiv != 2, then Qdiv must be 1. Else Qdiv can be 1 to 255.

Minimum DCO frequency = 7,998 MHz

Maximum DCO frequency = 10,000 MHz

Midpoint DCO frequency = 8,999 MHz

Note: If the desired symbol clock frequency cannot be achieved with the valid values of P, Q, K, and DCO frequencies, use a different screen resolution with a different symbol clock frequency.

Note: HDMI symbol clock frequency can be different from the pixel rate.

HDMI Color Format	Bits per color	HDMI Symbol Clock	HDMI Pixel Rate
RGB or YUV444	8	1 * pixel rate	1 * symbol clock
RGB or YUV444	10	5/4 * pixel rate	4/5 * symbol clock
RGB or YUV444	12	3/2 * pixel rate	2/3 * symbol clock
YUV420	8	1/2 * pixel rate	2 * symbol clock
YUV420	10	5/8 * pixel rate	8/5 * symbol clock
YUV420	12	3/4 * pixel rate	4/3 * symbol clock

Algorithm to Find HDMI Mode DPLL Programming

1. Calculate AFE clock
2. For each legal divider (P*Q*K), find the DCO.
3. Select the divider that gives the DCO closest to the midpoint and fits within the minimum or maximum.
4. Find the P, Q, K values to create that divider.

Pseudo-code for HDMI Mode DPLL Programming

```
 $freq = Desired symbol clock frequency in MHz

 $div=0; $bestdiv=0; $dco=0; $bestdco=0; $dcocentrality=0; $Pdiv = 0; $Qdiv = 0; $Kdiv = 0;
$afeclk = 0;
 $bestdcocentrality = 999999;
 @dividerlist = (2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 24, 28, 30, 32, 36, 40, 42, 44, 48, 50,
52, 54, 56, 60, 64, 66, 68, 70, 72, 76, 78, 80, 84, 88, 90, 92, 96, 98, 100, 102, 3, 5, 7, 9,
15, 21);
 $dcomin = 7998;
 $dcomax = 10000;
 $dcomid = ($dcomin + $dcomax) / 2;

 $afeclk = 5 * $freq;

 foreach $div (@dividerlist) {
     $dco = $afeclk * $div;
     if (($dco <= $dcomax) & ($dco >= $dcomin))
     {
         $dcocentrality = abs($dco-$dcomid);
         if ($dcocentrality < $bestdcocentrality)
         {
             $bestdcocentrality = $dcocentrality;
             $bestdiv = $div;
             $bestdco = $dco
         }
     }
 }

 if ($bestdiv != 0) # Good divider found
 {
     if ($bestdiv % 2 == 0) # Even
     {
         if ($bestdiv == 2)
         {
             $Pdiv = 2;
```

```
            $Qdiv = 1;
            $Kdiv = 1;
        }
        elsif ($bestdiv % 4 == 0)
        {
            $Pdiv = 2;
            $Qdiv = $bestdiv / 4;
            $Kdiv = 2;
        }
        elsif ($bestdiv % 6 == 0)
        {
            $Pdiv = 3;
            $Qdiv = $bestdiv / 6;
            $Kdiv = 2;
        }
        elsif ($bestdiv % 5 == 0)
        {
            $Pdiv = 5;
            $Qdiv = $bestdiv / 10;
            $Kdiv = 2;
        }
        elsif ($bestdiv % 14 == 0)
        {
            $Pdiv = 7;
            $Qdiv = $bestdiv / 14;
            $Kdiv = 2;
    }
    else # odd
    {
        if ($bestdiv == 3 || $bestdiv == 5 || $bestdiv == 7)
        {
            $Pdiv = $bestdiv;
            $Qdiv = 1;
            $Kdiv = 1;
        }
        else # 9, 15, 21
        {
            $Pdiv = $bestdiv / 3;
            $Qdiv = 1;
            $Kdiv = 3;
        }
    }
    SUCCESS
    Program DPLL_CFGCR0 with Best DCO / Reference Frequency
    Program DPLL_CFGCR1 with Pdiv, Qdiv, and Kdiv
}
else # No good divider found
{
    FAIL, try a different frequency
}
```

Calculating Frequency from Divider Values

Symbol clock frequency in MHz = DCO divider * Reference Frequency in MHz / (5 * Pdiv * Qdiv * Kdiv)

DCO divider from DPLL_CFGCR0 DCO Integer + (DPLL_CFGCR0 DCO Fraction / 2^15)

Pdiv from DPLL_CFGCR1 Pdiv

Qdiv from DPLL_CFGCR1 Qdiv Mode ? DPLL_CFGCR1 Qdiv Ratio : 1

Kdiv from DPLL_CFGCR1 Kdiv

Example of DVI on DDIB using 113.309 MHz symbol clock and reference 24 MHz

This example assumes DPLL1 is available.

Frequency programming algorithm finds Pdiv=2, Qdiv=4, Kdiv=2, DCO Frequency = 9064.72

DCO Integer = INT(9064.72/24) = 377

DCO Fraction = INT(((9064.72/24) - DCO Integer) * 2^15) = 22,828

1. Enable DPLL1 power in the DPLL1_ENABLE register
    - Power Enable = 1b (Enable)
2. Wait for PLL power state enabled in DPLL1_ENABLE.
3. Configure DPLL1_SSC register
    - sscen = 0b (Disable)
4. Configure DPLL1 in the DPLL1_CFGCR0 register
    - DCO Fraction = 592Ch (22828)
    - DCO Integer = 179h (377)
5. Configure DPLL1 in the DPLL1_CFGCR1 register
    - Qdiv Ratio = 04h (4)
    - Qdiv Mode = 1b (Enable)
    - Kdiv = 010b (2)
    - Pdiv = 0001b (2)
6. Read back DPLL1_CFGCR0 or DPLL1_CFGCR1 to ensure writes completed
7. Display Voltage Frequency Switching - Sequence Before Frequency Change
    - Skipped if 113.309 MHz is low enough to not trigger an increase in the voltage requirement
8. Enable DPLL1 in the DPLL1_ENABLE register
    - PLL Enable = 1b (Enable)
9. Wait for PLL lock in DPLL1_ENABLE
10. Display Voltage Frequency Switching - Sequence After Frequency Change
    - Skipped if 113.309 MHz is low enough to not trigger an increase in the voltage requirement
11. Configure DPLL1 mapping to DDIB in the DPCLKA_CFGCR0 register
    - DDIB Clock Select = 01b (DPLL1)
12. Turn on DDIB clock in the DPCLKA_CFGCR0 register
    - DDIB Clock Off = 0b (On)

MIPI DSI DPLL Programming

This programming is for DPLLs when using MIPI DSI.

Note that if the Driver wishes to maintain a single enabling sequence between DP/eDP and MIPI DSI, then the same enabling sequence defined for DP/eDP on combo PHY can be used for MIPI DSI. The only difference between the two sequences is:

1. The order of the DPLL mapping (DPCLKA_CFGCR0) with respect to the DPLL_ENABLE. The order of these two steps is unimportant
2. The programming of the *_ESC_CLK_DIV registers. The placement of the *_ESC_CLK_DIV registers can be placed anywhere within the sequence (i.e. at the beginning or end of the sequence) to facilitate a single function.

MIPI DSI DPLL Enable Sequence

1. Enable DPLL power in DPLL_ENABLE.
2. Wait for DPLL power state enabled in DPLL_ENABLE.
    - Should complete immediately.
3. Configure DPLL_SSC to set SSC enable/disable.
4. Configure DPLL_CFGCR0 to set DCO frequency.
5. Configure DPLL_CFGCR1 to set the dividers.
6. Read back DPLL_CFGCR0 or DPLL_CFGCR1, ignoring the data value, this is to ensure writes completed before the next step.
7. Configure DPCLKA_CFGCR0 to map the DPLL to the DDI/DSI.
    - Do not turn on the clock for the DDI. DSI has independent clock gating controlled by MIPI DSI mode programming.
8. Read back DPCLKA_CFGCR0, ignoring the data value, this is to ensure write completed before the next step.
9. Configure both **DSI_ESC_CLK_DIV** and **DPHY_ESC_CLK_DIV** registers. Both registers must be programmed with the same value.
10.
11. Read back both DPHY_ESC_CLK_DIV, ignoring the data value, this is to ensure write completed before the next step.
12. Enable DPLL in DPLL_ENABLE.
13. Wait for PLL lock status in DPLL_ENABLE.
    - Should complete within 50us.

MIPI DSI DPLL Disable Sequence

Sequence is the same as for DisplayPort, except there is not need to configure DPCLKA_CFGCR0 to turn off the clock for the DDI.

Formula for MIPI DSI DPLL Programming

Follow the same formula as for HDMI Mode. The AFE clock is the MIPI 8X clock.

Formula for MIPI DSI Escape Clock Programming

The MIPI DSI port operates with a constant frequency clock referred to as the Escape clock. The Escape clock should be programmed to be as close to, but not greater than 20 MHz. The Escape clock frequency will be determined by dividing the 8X frequency by a variable "M".

$f_{esc} = f_{8x} / M <= 20$ MHz

Solving for M supplies the programming necessary for the DSI_ESC_CLK_DIV and DPHY_ESC_CLK_DIV registers.

Escape clock divider M = ceiling(f_{8x} (MHz) / 20 MHz)

Example for DSI0 8X 566.545 and reference 24 MHz

This example assumes DPLL0 is available.

8x clock = AFE Clock = 566.545 MHz

Frequency programming algorithm finds Pdiv=3, Qdiv=1, Kdiv=1, DCO Frequency = 8498.175

Escape clock divider M = ceiling(566.545 / 20 MHz) = 29

DCO Integer = INT(8498.175/24) = 354

DCO Fraction = INT(((8498.175/24) - DCO Integer) * 2^15) = 2969

1. Enable DPLL0 power in the DPLL0_ENABLE register
    - Power Enable = 1b (Enable)
2. Wait for PLL power state enabled in DPLL0_ENABLE.
3. Configure DPLL0_SSC register
    - sscen = 0b (Disable)
4. Configure DPLL0 in the DPLL0_CFGCR0 register
    - DCO Fraction = 0B99h (2969)
    - DCO Integer = 162h (354)
5. Configure DPLL0 in the DPLL0_CFGCR1 register
    - Qdiv Ratio = 00h (1)
    - Qdiv Mode = 0b (Disable)
    - Kdiv = 001b (1)
    - Pdiv = 0010b (3)
6. Read back DPLL0_CFGCR0 or DPLL0_CFGCR1 to ensure writes completed
7. Configure DPCLKA_CFGCR0 DPLL0 mapping to DSI0 in the DPCLKA_CFGCR0 register
    - DDIA (DSI0) Clock Select = 00b (DPLL0)
8. Turn on DSI0 clock in the DPCLKA_CFGCR0 register
    - DDIA (DSI0) Clock Off = 0b (On)
9. Read back DPCLKA_CFGCR0 to ensure write completed

10. Configure MIPI escape clock dividers in DSI_ESC_CLK_DIV_0 and DPHY_ESC_CLK_DIV_0
    - Escape Clock Divider M = 0x01D (Divide by 29)
11. Read back DPHY_ESC_CLK_DIV to ensure write completed
12. Enable DPLL0 in the DPLL0_ENABLE register
    - PLL Enable = 1b (Enable)
13. Wait for PLL lock in DPLL0_ENABLE

Thunderbolt PLL Programming

This programming is for ports going to Thunderbolt. Those ports use the TBT PLL (a version of DPLL) shared across all thunderbolt ports. Thunderbolt supports a few fixed frequencies which are all produced simultaneously from parallel dividers after the PLL.

Thunderbolt PLL Enable Sequence

1. Enable PLL power in TBT_PLL_ENABLE (an instance of the DPLL_ENABLE register).
2. Wait for PLL power state enabled in TBT_PLL_ENABLE.
    - Should complete immediately.
3. Configure TBTPLL_CFGCR0 to set DCO frequency.
4. Read back TBTPLL_CFGCR0, ignoring the data value, this is to ensure writes completed before the next step.
5. Enable PLL in TBT_PLL_ENABLE.
6. Wait for PLL lock status in TBT_PLL_ENABLE.
    - Should complete within 50us.
7. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence Before Frequency Change.
    - Since the PLL produces multiple frequencies simultaneously, use the frequency selected for the port when determining the voltage requirement.
8. For each port that will use this PLL, program DDI_CLK_SEL to map the TBT clock to the port.
9. For each port that will use this PLL, configure DPCLKA_CFGCR0 to turn on the clock for the port.
10. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence After Frequency Change.
    - Since the PLL produces multiple frequencies simultaneously, use the frequency selected for the port when determining the voltage requirement.

Thunderbolt PLL Disable Sequence

- Since multiple ports can share the same PLL, the PLL is not disabled until after all the ports using it have turned off the clock.

1. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence Before Frequency Change.

2. For each port that was using this PLL, configure DPCLKA_CFGCR0 to turn off the clock for the port.
3. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence After Frequency Change.
4. Disable PLL through TBT_PLL_ENABLE.
5. Wait for PLL not locked status in TBT_PLL_ENABLE.
    - Should complete within 1us.
6. Disable DPLL power in TBT_PLL_ENABLE.
7. Wait for DPLL power state disabled in TBT_PLL_ENABLE.
    - Should complete immediately.

Thunderbolt PLL Divider Values

The thunderbolt PLL must always be configured to give a 1620 MHz output, depending on reference clock, using the values below.

Reference Clock	Pdiv	Kdiv	Qdiv	DCO MHz	DCO divided by Reference Clock MHz	DCO Integer hex	DCO Fraction hex
19.2 or 38.4 MHz	Not used	Not used	Not used	1620	84.375	0x54	0x3000
24 MHz	Not used	Not used	Not used	1620	67.5	0x43	0x4000

Thunderbolt Port Frequency Selection

Hardcoded dividers on the PLL 1620 MHz output produce 162, 270, 540, and 810 MHz symbol clock frequencies. The DDI_CLK_SEL register allows each port to select between those 4 frequencies.

Example of Thunderbolt on TC1 using 270 MHz symbol clock

This example assumes the reference frequency is 24 MHz.

1. Enable TBT_PLL power in the TBT_PLL_ENABLE register
    - Power Enable = 1b (Enable)
2. Wait for PLL power state enabled in TBT_PLL_ENABLE.
3. Configure TBTPLL_CFGCR0 register
    - DCO Fraction = 4000h
    - DCO Integer = 43h
4. Read back TBTPLL_CFGCR0 to ensure writes completed
5. Enable PLL in the TBT_PLL_ENABLE register
    - PLL Enable = 1b (Enable)
6. Wait for PLL lock in TBT_PLL_ENABLE
7. Display Voltage Frequency Switching - Sequence Before Frequency Change
    - Skipped if 270 MHz is low enough to not trigger an increase in the voltage requirement
8. Configure TC1 to use TBT 270 MHz in DDI_CLK_SEL_C register

- Clock Select = 1101b (TBT 270)

9. Configure DPCLKA_CFGCR0 to turn on TC1 clock
    - TC1 Clock Off = 0b (On)
10. Display Voltage Frequency Switching - Sequence After Frequency Change
    - Skipped if 270 MHz is low enough to not trigger an increase in the voltage requirement

TypeC PHY PLL Programming for Displayport

NOTE: Display uses PLL1, not PLL0. The defaults of associated registers may be different.

TypeC PHY and PLL registers need to be accessed as per the instructions in Dekel PHY Access page.

Register programming for DP frequencies consists of 2 parts: rate independent programming and rate dependent programming.

DKL PLL Enable Sequence

1. Enable PLL power in MGPLL_ENABLE (an instance of the DPLL_ENABLE register).
2. Wait for PLL power state enabled in MGPLL_ENABLE.
    - Should complete within 10us.
3. Program PLL registers as given in tables below - rate independent first and then rate dependent.
4. Read back the last programmed PHY PLL register, ignoring the data value, to ensure writes completed before the next step.
5. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence Before Frequency Change.
6. Enable PLL in MGPLL_ENABLE.
7. Wait for PLL lock status in MGPLL_ENABLE.
    - Timeout and fail after **900**us.
8. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence After Frequency Change.
9. Program DDI_CLK_SEL to map the Type-C PLL clock to the port.
10. Configure DPCLKA_CFGCR0 to turn on the clock for the port.

DKL PLL Disable Sequence

1. Configure DPCLKA_CFGCR0 to turn off the clock for the port.
2. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence Before Frequency Change.
3. Disable PLL through MGPLL_ENABLE.
    - Should complete within 50us.
4. Wait for PLL not locked status in MGPLL_ENABLE.

5. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence After Frequency Change.
6. Disable PLL power in MGPLL_ENABLE.
7. Wait for PLL power state disabled in MGPLL_ENABLE.
    - Should complete immediately.

The following DP rate independent and rate dependent programming must check DFLEXDPSP[12] for IOM FW support.

MFD DP - Rate independent programming - 100 MHz reference clock

This table covers programming for cases where DisplayPort is sharing lanes with USB.

PHY Register Address	Controller Register Address	Block	Register Name	Register Value	
0x2200	Check DKL PHY register access page	PLL1	DKL_PLL_DIV0	0x50272228	0x50272228
0x2204	Check DKL PHY register access page	PLL1	DKL_PLL_DIV1	0x0CD8C40D	0x0CD8C40D
0x2208	Check DKL PHY register access page	PLL1	DKL_PLL_LF	0x00401300	0x00401300
0x220C	Check DKL PHY register access page	PLL1	DKL_PLL_FRAC_LOCK	0x8044B56A	0x8044B56A
0x2210	Check DKL PHY register access page	PLL1	DKL_SSC[2]	0x800020FF	0x800020FF
0x2214	Check DKL PHY register access page	PLL1	DKL_BIAS	0xE0000000	0xE0000000
0x2218	Check DKL PHY register access page	PLL1	DKL_TDC_COLDST_BIAS	0x00000023	0x00000023
0x2228	Check DKL PHY register access page	PLL1	DKL_XXX_TDC_CRO	0x80008150	0x80008150
0x2244	Check DKL PHY register access page	PLL1	DKL_PLL1_CNTR_XXXX_SETTINGS	0x0004003F	0x0004003F
0x203C	Check DKL PHY register access page	CMN_DIG	DKL_CMN_DIG_PLL_MISC	0x00020000	0x00020000
0x02D4/0x12D4	Check DKL PHY register access page	PCS_Glue_Lane0/1	RTT_CR_SPARE	Do not program[1]	Do not program[1]

[1]default applies

[2]SSC can be disabled as per requirement.

Native and ALT DP - Rate independent programming - 38.4 MHz reference clock

This table covers programming for cases where DisplayPort is not sharing lanes with USB (native and ALT modes).

PHY Register Address	Controller Register Address	Block	Register Name	Register Value		
0x2200	Check DKL PHY register access page	PLL1	DKL_PLL_DIV0	0x70272269	0x70272269	0x70272269
0x2204	Check DKL PHY register access page	PLL1	DKL_PLL_DIV1	0x0CDCC422	0x0CDCC422	0x0CDCC422
0x2208	Check DKL PHY register access page	PLL1	DKL_PLL_LF	0x00401300	0x00401300	0x00401300
0x220C	Check DKL PHY register access page	PLL1	DKL_PLL_FRAC_LOCK	0x8044B56A	0x8044B56A	0x8044B56A
0x2210	Check DKL PHY register access page	PLL1	DKL_SSC(2)	0x401322FF	0x401322FF	0x401322FF
0x2214	Check DKL PHY register access page	PLL1	DKL_BIAS	Do not program(1)	Do not program(1)	Do not program(1)
0x2218	Check DKL PHY register access page	PLL1	DKL_TDC_COLDST_BIAS	Do not program(1)	Do not program(1)	Do not program(1)
0x2228	Check DKL PHY register access page	PLL1	DKL_XXX_TDC_CRO	Do not program(1)	Do not program(1)	Do not program(1)
0x2244	Check DKL PHY register access page	PLL1	DKL_PLL1_CNTR_XXXX_SETTINGS	Do not program(1)	Do not program(1)	Do not program(1)
0x203C	Check DKL PHY register access page	CMN_DIG	DKL_CMN_DIG_PLL_MISC	0x00000000	0x00000000	0x00000000
0x212C	Check DKL	CMN_ANA	DKL_REFCLKIN_CTL	0x00000101	0x00000303	0x00000101

PHY Register Address	Controller Register Address	Block	Register Name	Register Value		
	PHY register access page					
0x2130	Check DKL PHY register access page	CMN_ANA	DKL_CMN_ANA_DWORD28	0x14158888	0x14158888	0x14158888
0x02D4/0x12D4	Check DKL PHY register access page	PCS_Glue_Lane0/1	RTT_CR_SPARE	Do not program(1)	Do not program(1)	Do not program(1)

(1)default applies

(2)SSC can be disabled as per requirement.

MFD/Native DP - Rate dependent programming

Register Name	8p1	5p4	2p7	1p62
CLKTOP2_HSCLKCTL	0000011D (default)	0x0000121D	0x0000521D	0x0000621D
CLKTOP2_CORECLKCTL1	10080510 (default)	10080510 (default)	0x10080A12	0x10080A12

TypeC PHY PLL Programming - algorthmic steps

NOTE: Display uses PLL1, not PLL0. The defaults of associated registers may be different.

If the desired symbol clock frequency cannot be achieved with the valid programming, use a different screen resolution with a different symbol clock frequency.

The following algorithmic steps apply for HDMI protocol and enumerated reference clocks. For DP protocol, refer to the tables above.

```
 $symbol_frequency = Desired HDMI symbol clock frequency in MHz(example: 594 MHz for HDMI 2.0
4K 60 Hz)
 $refclk_mhz = Reference frequency (19.2 MHz or 24.0 MHz or 38.4 MHz)
 $ssc_en = SSC enable  // DO NOT SELECT SSC $dp = 1 for DisplayPort, 0 for HDMI // DO NOT
SELECT DISPLAYPORT OPTION Set $dp = 0
 $dp_dco = 8.1;
 $dco_min_freq = $dp ? $dp_dco : $ssc_en ? 8 : 7.992;
 $dco_max_freq = $dp ? $dp_dco : 10;
 @div1_vals = ( 7, 5, 3, 2);
 @div2_vals = ( 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 );
 $success = 0;

 $frequency = $symbol_frequency/100;

 for $div1 (@div1_vals) {
     for $div2 (@div2_vals) {
         $dco = $div1 * $div2 * $frequency / 2;
         if ( $dco >= $dco_min_freq and $dco <= $dco_max_freq ) {
           Program DKL_REFCLKIN_CTL od_refclkin2_refclkmux_2_0 = 1
           Program DKL_CLKTOP2_HSCLKCTL od_clktop2_coreclk_inputsel = 1
```

```
            Program DKL_CLKTOP2_CORECLKCTL1 od_clktop2_coreclka_divratio = 5

            if ($div2 >= 2) {
              Program DKL_CLKTOP2_HSCLKCTL od_clktop_tlinedrv_clksel_1_0 = 1
            } else {
              Program DKL_CLKTOP2_HSCLKCTL od_clktop_tlinedrv_clksel_1_0 = 0
            }
            Program DKL_CLKTOP2_HSCLKCTL od_clktop_coreclk_inputsel = $dp ? 0b : 1b
            if ($div1 == 2) {Program DKL_CLKTOP2_HSCLKCTL od_clktop_hsdiv_divratio_1_0 = 0}
            if ($div1 == 3) {Program DKL_CLKTOP2_HSCLKCTL od_clktop_hsdiv_divratio_1_0 = 1}
            if ($div1 == 5) {Program DKL_CLKTOP2_HSCLKCTL od_clktop_hsdiv_divratio_1_0 = 2}
            if ($div1 == 7) {Program DKL_CLKTOP2_HSCLKCTL od_clktop_hsdiv_divratio_1_0 = 3}
            Program DKL_CLKTOP2_HSCLKCTL od_clktop_dsdiv_divratio = $div2 converted to 4 bit
binary;
            $target_dco_mhz = $dco * 1000;
            $success = 1;
            last;
          }
      }
      if ($success == 1) {last;}
 }


 if ($success == 0) {
      FAIL and try a different frequency
 }


 $m1div = 2;           # pre-divider ratio
 $ndiv = 1;            # reference clock divider
 $tdc_res = 0.000003;  # TDC resolution
 $ssc_freq_mhz = .032; # SSC frequency
 $ssc_stepnum = 32;    # SSC step num
 $ssc_amp = .0047;     # SSC amplitude

 # calculate the M2 ratio
 $m2div = ($target_dco_mhz / ($refclk_mhz / $ndiv)) / $m1div;

 # 8 bits for M2
 if ($m2div > 255) {
      FAIL and try a different frequency
 }

 # integer portion
 $m2div_int = floor($m2div);

 # fractional portion
 $m2div_frac = floor(2**22 * ($m2div - $m2div_int));

 # iref ndiv
 $iref_ndiv = ($refclk_mhz > 80) ? 4 : ($refclk_mhz > 38) ? 2 : 1;

 # TDC target count
 $tdc_targetcnt = int(2/($tdc_res*8*50*1.1)/$refclk_mhz + 0.5);

 # feed forward gain
 $feedfwdgain = ($ssc_en || $m2div_frac > 0) ? floor($m1div*(1/$target_dco_mhz)/$tdc_res) : 0;

 # iref trim value
 $iref_trim = (($refclk_mhz/$iref_ndiv) <= 19.2) ? 28 :
                 (($refclk_mhz/$iref_ndiv) > 19.2 && ($refclk_mhz/$iref_ndiv) <= 25) ? 25 :
24;

 # bias start-up pulse width
```

```
$iref_pulse_width = (($refclk_mhz/$iref_ndiv) > 20) ? 2 : 1;

# lf coeffs
$prop_coeff = ($target_dco_mhz >= 9000) ? 5  : 4;
$int_coeff  = ($target_dco_mhz >= 9000) ? 10 : 8;

# ssc params
$ssc_stepsize = $ssc_en ? (floor($m2div * $ssc_amp / $ssc_stepnum * 2**10))        : 0;
$ssc_steplen  = $ssc_en ? (ceil($refclk_mhz / $ssc_freq_mhz / (2 * $ssc_stepnum))) : 0;
$ssc_steplog  = log($ssc_stepnum)/log(2)-1;

# calculate register values (these are read-modify-write ops)
$div0 = ($int_coeff * 2**16) + ($prop_coeff * 2**12) + ($m1div * 2**8) + ($m2div_int);
$div1 = ($iref_trim * 2**16) + ($tdc_targetcnt);
$ssc = ($iref_ndiv * 2**29) + ($ssc_steplen * 2**16) + ($ssc_stepnum * 2**11) + ($ssc_en *
2**9);
$bias = ($m2div_frac > 0 ? 2**30 : 0) + ($m2div_frac * 2**8);
$tdc = ($ssc_stepsize * 2**8) + $feedfwdgain;

Program DKL_PLL_DIV0 = $div0 converted to 32 bit hex;
Program DKL_PLL_DIV1 = $div1 converted to 32 bit hex;
Program DKL_SSC = $ssc converted to 32 bit hex;
Program DKL_BIAS = $bias converted to 32 bit hex;
Program DKL_TDC_COLDST_BIAS = $tdc converted to 32 bit hex;
```

Calculating PLL Frequency from Divider Values

Symbol clock frequency in MHz = M1 * M2 * Reference Frequency in MHz / (5 * DIV1 * DIV2)

M1 from DKL_PLL_DIV0[i_fbprediv_3_0]

M2 from DKL_PLL_DIV0[i_fbdiv_intgr_7_0] + DKL_BIAS[i_fracnen_h] ? DKL_BIAS[i_fbdivfrac_21_0/2^22) : 0

DIV1 from DKL_CLKTOP2_HSCLKCTL[od_clktop_hsdiv_divratio]

DIV2 from DKL_CLKTOP2_HSCLKCTL[od_clktop_dsdiv_divratio]

HDMI Example

HDMI, 5.94 Gbps, SSC disabled, 38.4 MHz reference

Register	Default	Value to program
DKL_BIAS	0xde000000	0xc1000000
DKL_PLL_DIV0	0x50273269	0x50284274
DKL_PLL_DIV1	0x0cdcc522	0x0cdcc527
DKL_SSC	0x401322ff	0x400020ff
DKL_TDC_COLDST_BIAS	0x00000f23	0x0000004a
TAP_OVRD	0x50003100	0x50003100
DFX_TDC_CRO	0x80008130	0x80008130
CLKTOP2_HSCLKCTL	0x0000011d	0x0001111d
CLKTOP2_CORECLKCTL1	0x10080510	0x10080510

Display Voltage Frequency Switching

Display Voltage and Frequency Switching (DVFS) is used to adjust the display voltage to match the display clock frequencies. If voltage is set too low, it will break functionality. If voltage is set too high, it will waste power. The voltage rail for display is shared by the entire system agent, so it has a large power impact.

When changing clock frequencies, graphics software must inform the power controller of the display voltage requirement. The power controller tracks requests from all users of the voltage rail and sets voltage to accommodate all the requests. The voltage requirements are separated into discrete levels aligned to the frequencies of several clocks used by display.

Voltage Requirement Selection

The voltage requirement is specified by selecting from several discrete voltage levels. The power controller will map these levels to the actual voltage values, which are usually determined on a part by part basis during manufacturing.

Only the CD clock and DDI clocks (max of all DDI clocks) are used to select the voltage levels. Other display clocks are either supported at all frequencies with the minimum voltage, or have their voltage requirements accounted for by non-graphics software.

Voltage Level	CD clock MHz	Max DDI symbol clock MHz	Comment
0	307.2, 312, or lower	<=594	No 5k or 8k
1	324 or 326.4	<=594	These display clocks are mainly for resolutions under 5k with multiple displays enabled (cdclk PLL divided by 2 to allow for smooth increase to max cdclk)
2	556.8 or 552	>594	These display clocks are mainly for non-HDR 5K and 8k, or some cases with DP multistream or USB TypeC
3	652.8 or 648	>594	These display clocks are mainly for HDR 5k and 8k

If CD clock <= 312 MHz AND Max of DDI clocks <= 594 MHz, use level 0.

Else If CD clock <= 326.4 MHz AND Max of DDI clocks <= 594 MHz, use level 1.

Else If CD clock <= 556.8 AND Max of DDI clocks > 594 MHz, use level 2.

Else, use level 3.

Note: For these calculations, disabling a clock is the same as switching it to the lowest frequency.

The sequences below are used when changing CD clock frequency and when changing DDI clock frequency during a mode set.

The following sequences are not used on their own. They are called as part of other sequences which change the display clock frequencies.

Sequence Before Frequency Change

- This sequence requests the power controller to raise voltage to the maximum

1. Ensure any previous GT Driver Mailbox transaction is complete.
2. Write GT Driver Mailbox Data Low = 0x00000003.
3. Write GT Driver Mailbox Data High = 0x00000000.
4. Write GT Driver Mailbox Interface = 0x80000007.
5. Poll GT Driver Mailbox Interface for Run/Busy indication cleared (bit 31 = 0).
    - Timeout after 150 us. Do not change CD clock frequency if there is a timeout.
6. Read GT Driver Mailbox Data Low, if bit 0 is 0x1, continue, else go to step 2.
    - If the condition in step 6 is not satisfied after cycling through steps 2-6 for 3 ms (typically <200 us), timeout and fail and do not change clock frequency.

Sequence After Frequency Change

- This sequence requests the power controller to set the voltage to the selected level. The power controller may choose to keep the voltage higher to accommodate other users outside of display.

1. Write GT Driver Mailbox Data Low with the voltage level selection, following the table above.
    - For Level 0, write 0x00000000.
    - For Level 1, write 0x00000001.
    - For Level 2, write 0x00000002.
    - For Level 3, write 0x00000003.
2. Write GT Driver Mailbox Data High = 0x00000000.
3. Write GT Driver Mailbox Interface = 0x80000007.
    - There is no need for display software to wait for the voltage to adjust.

Sequences for Changing CD Clock Frequency

Restrictions
The CD clock frequency impacts the maximum supported pixel rate and display watermark programming.
The CD clock frequency must be at least twice the frequency of the Azalia BCLK.

Sequence for Changing CD Clock Frequency

1. Unless changing only the CD2X Divider, disable all display engine functions using the full mode set disable sequence on all pipes, ports, and planes.
    - Includes Global Time Code
    - Display power wells may be left enabled
2. Follow the Display Voltage Frequency Switching - Sequence Before Frequency Change

3. Enable or change the frequency of CD clock
    a. If enabling CDCLK PLL
        i. Write CDCLK_PLL_ENABLE with the PLL ratio, but not yet enabling it.
        ii. Set CDCLK_PLL_ENABLE PLL Enable
        iii. Poll CDCLK_PLL_ENABLE for PLL lock
        iv. Timeout and fail if not locked after 200 us
        v. Write CDCLK_CTL with the CD2X Divider selection and CD Frequency Decimal value to match the desired CD clock frequency
    b. If disabling CDCLK PLL
        i. Clear CDCLK_PLL_ENABLE PLL Enable
        ii. Poll CDCLK_PLL_ENABLE for PLL unlocked
        iii. Timeout and fail if not unlocked after 200 us
    c. If changing the CDCLK PLL frequency
        i. Follow steps above for disabling CDCLK PLL.
        ii. Follow steps above for enabling CDCLK PLL, using the new PLL ratio.
    d. If changing only the CD2X Divider
        i. Write CDCLK_CTL with the CD2X Pipe selection, CD2X Divider selection, and CD Frequency Decimal value to match the desired CD clock frequency
        ii. If pipe is enabled, wait for start of vertical blank for change to take effect
4. Follow the Display Voltage Frequency Switching - Sequence After Frequency Change
5. Update programming of functions that use the CD clock frequency

If these features are not currently enabled, the programming can be delayed to when they are enabled.

- Wireless Display 27 MHz frequency in the WD_27_M and WD_27_N registers.
    - For CD clock 168 MHz, program M=9 and N=56 (decimal).
    - For CD clock 336 MHz, program M=9 and N=112 (decimal).
    - For CD clock 528 MHz, program M=9 and N=176 (decimal).
    - For CD clock 172.8 MHz, program M=45 and N=288 (decimal).
    - For CD clock 307.2 MHz, program M=45 and N=512 (decimal).
    - For CD clock 556.8 MHz, program M=45 and N=928 (decimal).
    - For CD clock 652.8 MHz, program M=45 and N=1088 (decimal).
    - For CD clock 180 MHz, program M=45 and N=300 (decimal).
    - For CD clock 312 MHz, program M=45 and N=520 (decimal).
    - For CD clock 552 MHz, program M=45 and N=920 (decimal).
    - For CD clock 648 MHz, program M=45 and N=1080 (decimal).
- Utility pin backlight frequency and duty cycle in the BLC_PWM_DATA register.

Resets

The north and south display engines are reset by PCI Function Level Resets (FLR) and the chip level resets.

The south display engine runs panel power down sequencing (if configured to do so by NDE_RSTWRN_OPT) before resetting.

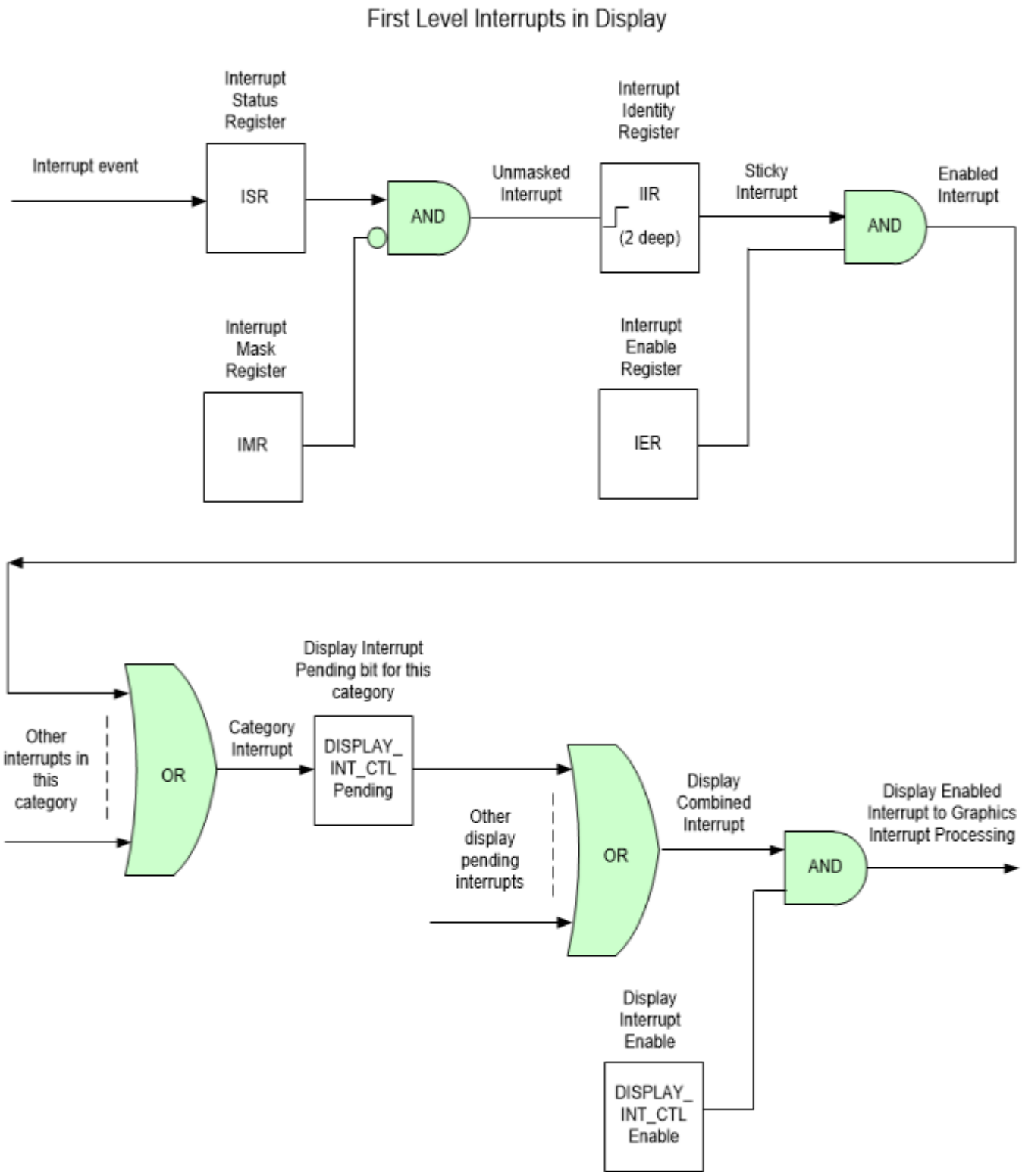
NDE_RSTWRN_OPT

Shared Functions

Display Interrupts

Registers
DISPLAY_INT_CTL **DE Pipe Interrupt Definition** **DE Port Interrupt Definition** **DE Misc Interrupt Definition** **Audio Codec Interrupt Definition** **INTERRUPT Structure**
DE HPD Interrupt Definition
Graphics Primary Interrupt
Graphics Interrupt Introduction

Interrupt Flow



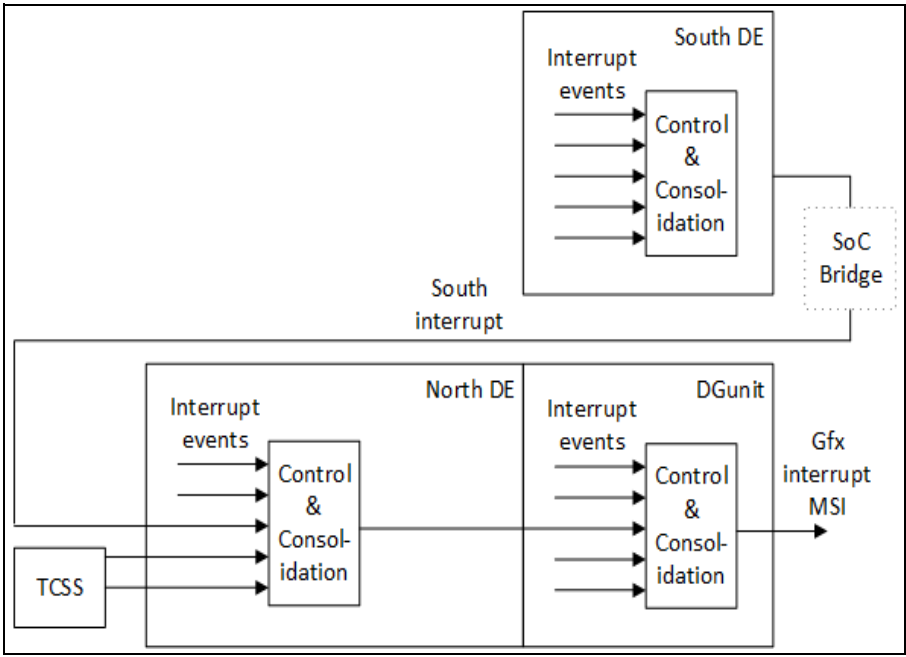
1. For every first level interrupt bit
    a. The interrupt event comes in to the interrupt handling logic.
        - There may be more levels of interrupt handling behind each event. For example, the PCH Display interrupt event is the result of the SDE interrupt registers.
    b. The interrupt event goes to the Interrupt Status Register (ISR) where live status can be read back.
        - The live status is mainly useful for hotplug interrupts where it gives the live state of the hotplug line.
        - The live status is not useful for pulse interrupt events due to the short period that the status will be present.
    c. The interrupt event is ANDed with the inverted Interrupt Mask Register (IMR) to create the unmasked interrupt.
    d. The unmasked interrupt rising edge sets the sticky bit in the Interrupt Identity Register (IIR).
        - The IIR can be cleared by writing a 1 to it.
        - The IIR can queue up to two interrupt events. When the IIR is cleared, it will set itself again if a second event was stored.
    e. The sticky interrupt is ANDed with the Interrupt Enable Register (IER) to create the enabled interrupt.
2. All enabled interrupts are then ORed by category (Pipe, Audio, etc.) to create a category interrupt which is then visible in one of the Display Interrupt Control Register (DISPLAY_INT_CTL) pending category bits.
3. All pending interrupts are then ORed to create the display combined interrupt.
4. The display combined interrupt is ANDed with the Display Interrupt Enable (DISPLAY_INT_CTL Bit 31) to create the display enabled interrupt.
5. The display enabled interrupt then goes to graphics interrupt processing before eventually creating an interrupt message which will reach the OS.

Interrupt Service Routine

1. Read graphics primary interrupt register to find a display interrupt pending.
2. Write graphics primary interrupt register to clear display interrupt.
3. Disable Display Interrupt Control (can be done here, or anywhere before the final step)
    - Clear bit 31 of DISPLAY_INT_CTL.
    - This de-asserts the display enabled interrupt and is required in order for the final step to cause a new assertion when all interrupts are not cleared during the service routine.
4. Find the category of interrupt that is pending
    - Read DISPLAY_INT_CTL and record which interrupt pending category bits are set.
5. Find the source(s) of the interrupt and clear the Interrupt Identity bits (IIR)
    - Read the IIR associated with each pending interrupt category, record which bits are set, then write back 1s to clear the bits that are set.

- There can be up to 2 interrupts recorded per source, requiring multiple writes to the IIR to fully clear.

6. Process the interrupt(s) that had bits set in the IIRs.
7. Optionally go to step 4 again to check for any new display interrupts that have happened during previous steps
    - This is optional since any new interrupts will also be found when they trigger the interrupt service routine again after the final step and then re-enter from the start of the routine.
8. Re-enable Display Interrupt Control
    - Set bit 31 of DISPLAY_INT_CTL.
    - If interrupts were not fully cleared before this point, then the display enabled interrupt will re-assert and there will be a new display interrupt in graphics primary interrupt register.



South display interrupt routed to north display.

North display connected to DGunit.

MBus

Mbus is an internal ring type bus. Display high priority clients reside on this bus. The introduction of a ring architecture allow for easy addition/deletion of clients as well as improves Bandwidth. The various "boxes" are different types of tap points where display clients are plugged in.

MBus
MBUS_ABOX_CTL
MBUS_BBOX_CTL
MBUS_DBOX_CTL
MBUS_UBOX_CTL

MBus programming during display Initialization

Program credits in MBUS_ABOX_CTL, MBUS_ABOX1_CTL, MBUS_ABOX2_CTL

The MBus credits should be setup once with the following default values during the display initialization.

MBUS_ABOX_CTL for this pipe -> BT Credits Pool1 = 16

MBUS_ABOX_CTL for this pipe -> BT Credits Pool2 = 16

MBUS_ABOX_CTL for this pipe -> B Credits = 1

MBUS_ABOX_CTL for this pipe -> BW Credits = 1

DBUF_CTL for this pipe -> Tracker State Service = 8

The following programming must be done when enabling each pipe as a part of *configure other pipe settings* in the *Enable Planes, Pipe, and Transcoders* sequence.

MBUS_DBOX_CTL for this pipe -> A Credits = 2

MBUS_DBOX_CTL for this pipe -> BW Credits = 2

MBUS_DBOX_CTL for this pipe -> B Credits = 12

Fuses and Straps

FUSE_STATUS

DFSM

DSSM

DFSDONE

Render Response

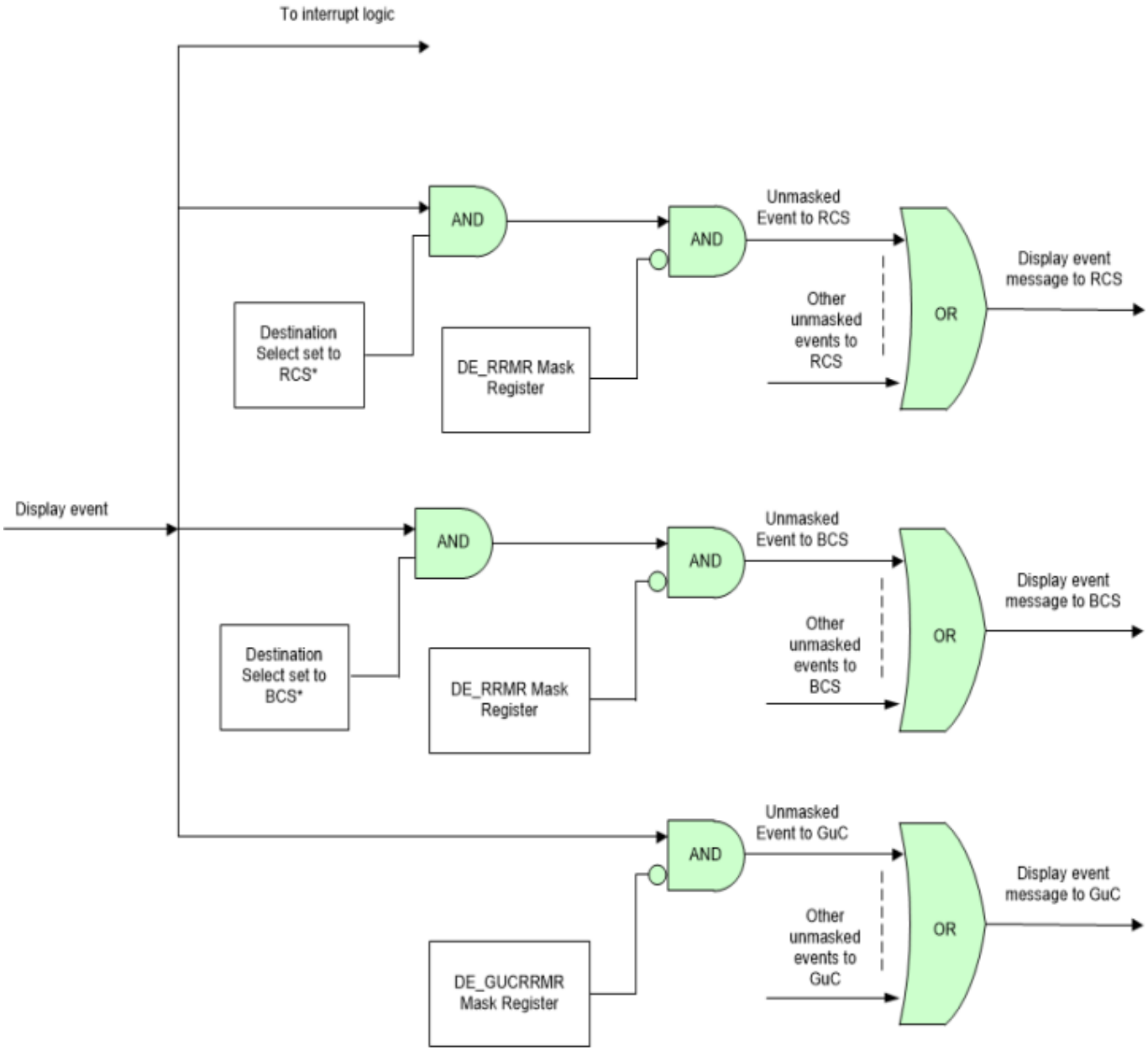
Display Engine Render Response Message Definition

DE_RRMR

DE_RRMR_DW1

DE_RRMR_DW2

DE_RR_DEST



*For Vblank events, destination select is configured in DE_RR_DEST to direct the event to RCS, BCS, or both at once.
For MMIO triggered flip done events, destination select is configured in PLANE_SURF Ring Flip Source to direct the event to either RCS or BCS.
For ring (AKA message) triggered flip done events, destination select is configured automatically by decoding whether the flip came from BCS or RCS.

GuC supports some additional events beyond what RCS and BCS support. In that case the event can feed into the GuC masking directly without branching to RCS and BCS.

Older documentation refers to RCS as CS.

Hot Plug Detection

This topic describes hot plug detection for type C ports with DP alternate mode or DP over thunderbolt. Hotplug detection for non-typeC is done through the south display.

Type C Port Hot Plug Detection

HOTPLUG_CTL

These registers are used for detecting hot plug on thunderbolt or type C (DP alternate). They will generate interrupts in DE_HPD_INTERRUPT. HOTPLUG_CTL has the status bits to indicate if long or short pulses are detected. Do not use the interrupt ISR for finding the live connect state. It can become out of sync with the typeC sub-system after some power states and resets. Instead, use DFLEXDPSP TC Live State fields.

The south display hot plug detection registers are used for detecting hot plug for combo PHY ports and type C ports with non-type C connectors (legacy or static configuration).

HPD Interrupt Sequence

- This sequence is for identifying each type of hotplug. Follow interrupt documentation for how interrupt service routine should enable, disable, and clear interrupts.

1. Display interrupt received and DISPLAY_INT_CTL shows source of the interrupt is DE HPD Interrupts pending (north display HPD)
2. Read DE_HPD_INTERRUPT IIR to find if source port type is thunderbolt or typeC (DP alternate)
3. Read <port type, TBT for thunderbolt, TC for DP alternate>_HOTPLUG_CTL to find the HPD Status
    - Long pulse, short pulse, both, or none
4. If long pulse or both
    1. Read PORT_TX_DFLEXDPSP TC Live State to find the live state
        - Long pulse with live state = 1 indicates a connect
        - Long pulse with live state = 0 indicates a disconnect
    2. Follow the TypeC Programming connect flow or disconnect flow
5. If short pulse and not HDMI
    1. Query the panel and then proceed as required by panel

Resets and Power States

Hotplug interrupts can be missed around resets and power states. When coming out of any state where HPD interrupts can be missed (cold boot, warm reset, S3/S4, DC9, Function Level Reset, handoff between VBIOS/GOP/EFI and driver, etc.) use the following sequence to enable HPD interrupts and find if any previous HPD was missed and needs to be serviced.

1. Enable HOTPLUG_CTL and configure interrupt related registers so that HPD interrupts will be received.

2. If HPD interrupt received after step 1, handle it as usual and abort the remaining steps for that port.
3. Process live state for each typeC port.
    - Read PORT_TX_DFLEXDPPMS DPPMSTC, PORT_TX_DFLEXDPCSSS DPPMSTC, and PORT_TX_DFLEXDPSP TC Live State.
    - Process connect or disconnect as per truth table, abort if HPD interrupt is received for this port and handle the disconnect/connect requested by the HPD.

The table below illustrates the next steps based on the live state and connection state.

DPSP TC Live State	DPPMS DPPMSTC	DPCSSS DPPMSTC	IOM State	Software Next Steps
00	0	0	Disconnect complete	Do nothing right now. IOM will send HPD if something is needed.
00	1	0	IOM action pending	Do nothing right now. IOM will send HPD if something is needed.
01	1	0	DP-alt connect pending	Follow connect flow for DP alternate.
01	1	1	DP-alt connect complete	Follow connect flow for DP alternate from the step after DPCSSS DPPMSTC is set to 1.
00	1	1	DP-alt disconnect pending	Follow disconnect flow for DP alternate.
10	x	x	TBT connect complete	Follow connect flow for thunderbolt.
00	0	1	Invalid	Follow disconnect flow for DP alternate.
01	0	0	Invalid	Do nothing right now
01	0	1	Invalid	Follow disconnect flow for DP alternate.
11	x	x	Invalid	Follow disconnect flow for DP alternate.

Refer to the TypeC Programming page for details on signaling between IOM and display driver.

Arbiter

ARB_CTL

ARB_CTL2

PIPE_ARB_CTL

BW_BUDDY0_CTL

BW_BUDDY1_CTL

BW_BUDDY2_CTL

BW_BUDDY1_PAGE_MASK

BW_BUDDY2_PAGE_MASK

Bandwidth Buddy Registers
These registers are used to improve bandwidth efficiency by pairing up memory requests with similar addresses. The mask register selects which address bits to compare when buddying up transactions. The mask must be programmed based on the memory type and memory controller attributes.
Abox1 and Abox2 are used for pixel data reads. Program the BW_BUDDY1 and BW_BUDDY2 registers.

Optimal programming of the BW_BUDDY_PAGE_MASK registers depends on the memory configuration used. Program the BW_BUDDY_PAGE_MASK according to the table below. For memory configurations not listed below, set the BW_BUDDY_CTL Disable bit.

MEMORY CHANNELS/Number of modules	MEMORY TYPE	BW_BUDDY_PAGE_MASK
1/1	DDR4	0x0000000F
1/1	DDR5	0x0000000F
2/1	LP4,LP5	0x0000001C
2/2	DDR4	0x0000001F
2/2	DDR5	0x0000001E
4/2	LP4,LP5	0x00000038

Data Buffer

DBUF_CTL

DBUF_STATUS

DBUF_ECC_STAT

DBUF_CTL2

See the Display Buffer Programming section for information on how to allocate the buffer to planes.

Backlight

This section refers to the CPU display backlight control over the utility pin. For PCH display backlight control, see South Display Engine Registers.

The backlight PWM output frequency is determined by the PWM clock frequency, increment, and frequency divider.

PWM output frequency = PWM clock frequency / PWM increment / PWM frequency divider

The frequency divider must be greater than or equal to the number of brightness levels required by software; typically 100 or 256.

Backlight Enabling Sequence

1. Enable utility pin, select PWM mode, and set polarity in UTIL_PIN_CTL Util Pin Enable, Util Pin Mode, and Util Pin Output Polarity.

2. Set frequency and duty cycle in BLC_PWM_DATA Backlight Frequency and Backlight Duty Cycle.

3. Enable PWM output and set granularity in BLC_PWM_CTL PWM Enable and PWM granularity.

...

4. Change duty cycle as needed in BLC_PWM_DATA Backlight Duty Cycle.

If needed, granularity and polarity can be programmed earlier in the enabling sequence.

Backlight Frequency Change Sequence

1. Disable PWM output in BLC_PWM_CTL PWM Enable.
2. Set new frequency and duty cycle in BLC_PWM_DATA Backlight Frequency and Backlight Duty Cycle.
3. Enable PWM output and set granularity in BLC_PWM_CTL PWM Enable and PWM granularity.

Backlight Registers

BLC_PWM_CTL

BLC_PWM_DATA

Miscellaneous Shared Functions

UTIL_PIN_CTL **UTIL_PIN_BUF_CTL**
UTIL2_PIN_CTL **UTIL2_PIN_BUF_CTL**
The utility pin(s) can be used for various functions, including MIPI DSI TE and genlock, on SoCs supporting those functions. The pins must be configured for those functions as part of the function enable and disable sequences.
AUDIO_PIN_BUF_CTL

Double Buffer Control

DOUBLE_BUFFER_CTL

Double Buffer Disabling/Stalling

Double buffer stalling (also called disabling) is used to synchronize register updates across multiple display resources for an atomic update. While stalled, double buffer registers can be written with new values, updating the 1st stage buffer, but the double buffering will not activate to transfer those values to the 2nd stage (live/working value) buffer to be used by the hardware resource. This allows for multiple register programming to cross the double buffer update point and keep the registers in sync.

Without double buffer stalling, updates to multiple registers can become un-synchronized if the double buffer update event (typically vblank rising edge) happens during the programming. Then some registers will have double buffer updates a frame before other registers and the screen contents will be incorrect.

The double buffer stalling is controlled by the global double buffer disable. Disabling double buffering means it is stalled.

- DOUBLE_BUFFER_CTL Global Double Buffer Disable register bit is used to stall double buffering in all pipes and transcoders.

Allow Double Buffer Stall

The double buffer stalling is also controlled by the Allow Double Buffer Update Disable register bit that each resource has in one of its control registers. The Allow Double Buffer Update Disable is useful if certain resources need to update freely and not be tied into the synchronizing of other resources, such as if cursor update needs to happen as soon as possible while some plane updates are being stalled to synchronize together.

The double buffering for a particular resource will only be stalled if that resource's Allow Double Buffer Update Disable is set and the global double buffer disable is set.

Details

This only stalls the double buffer update for periodic events, like the start of vertical blank. It does not change the behavior for constant events, like pipe not enabled.

Double buffer stalling does not stall asynchronous flips initiated by MMIO or command streamers.

Synchronous flips will not complete or give the flip done indication while double buffering is stalled for a plane. They will complete and give the flip done at the next double buffer update point after the double buffering is no longer stalled.

Sequence for synchronizing double buffer updates

1. If not already set, set the Allow Double Buffer Update Disable field for each resource to be synchronized together.
2. Set the Global Double Buffer Disable register bit. Double buffering is now stalled.
3. Program the registers that need to be synchronized together.
4. Clear the Global Double Buffer Disable register bit. Double buffering is now un-stalled. Any pending updates will take place at the next periodic update event.
5. Depending on configuration, variable refresh rate and PSR require extra programming to trigger a frame update after the registers have been updated. See the VRR and PSR programming pages.
6. If a resource no longer needs to be synchronized, clear the Allow Double Buffer Update Disable field for that resource. If the resource will continue to be synchronized, the field can remain set.

It is not necessary to set and clear the Allow Double Buffer Update Disable around every register update. Allow Double Buffer Update Disable only has to change if a resource's requirements for synchronization are changed.

Central Power

SAGV

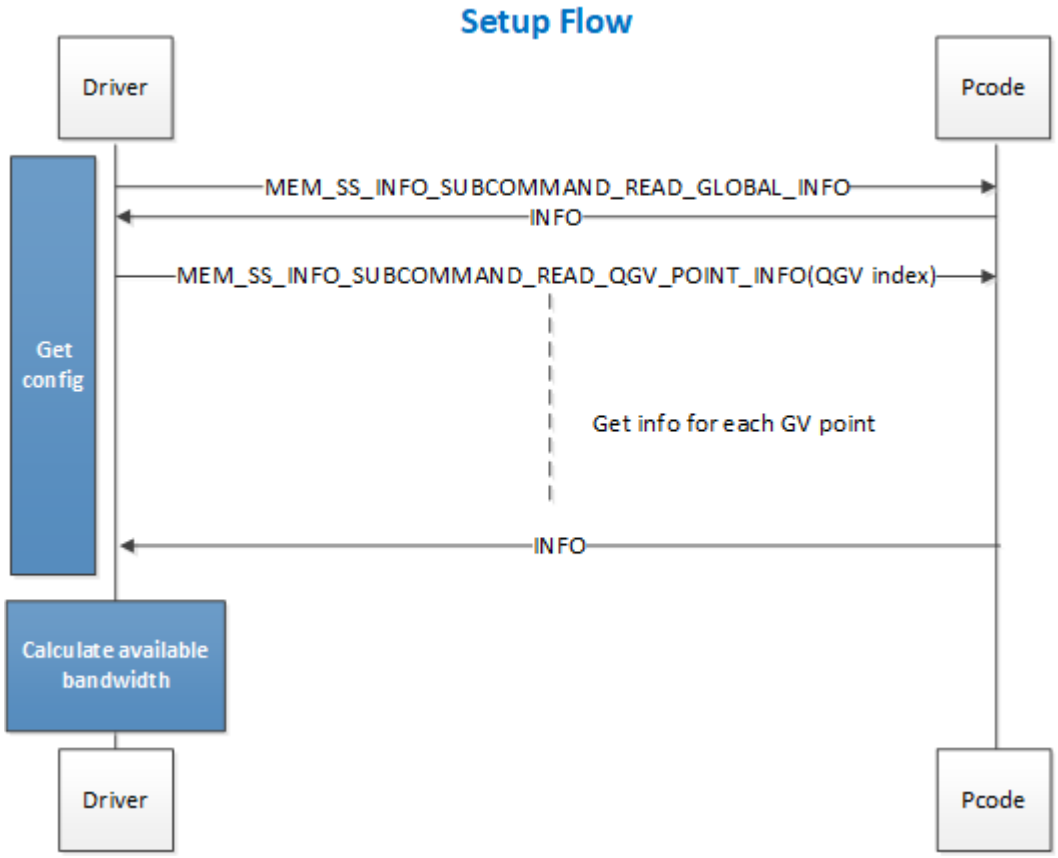
System Agent Geyserville (SAGV) dynamically adjusts the system agent voltage and clock frequencies depending on power and performance requirements. SAGV impacts display engine in two ways. SAGV point selections can limit the system memory bandwidth to display. SAGV transitions can temporarily block display engine access to system memory. Display software must restrict the SAGV point selection to control the bandwidth availability and setup watermarks to tolerate the temporary memory block.

SAGV Point Selection

Setup

Find the memory bandwidth availability for display. The results can be cached so that the flow does not have to be rerun.

The driver does not have to use the mailbox if it has other ways of finding the memory and GV point info.



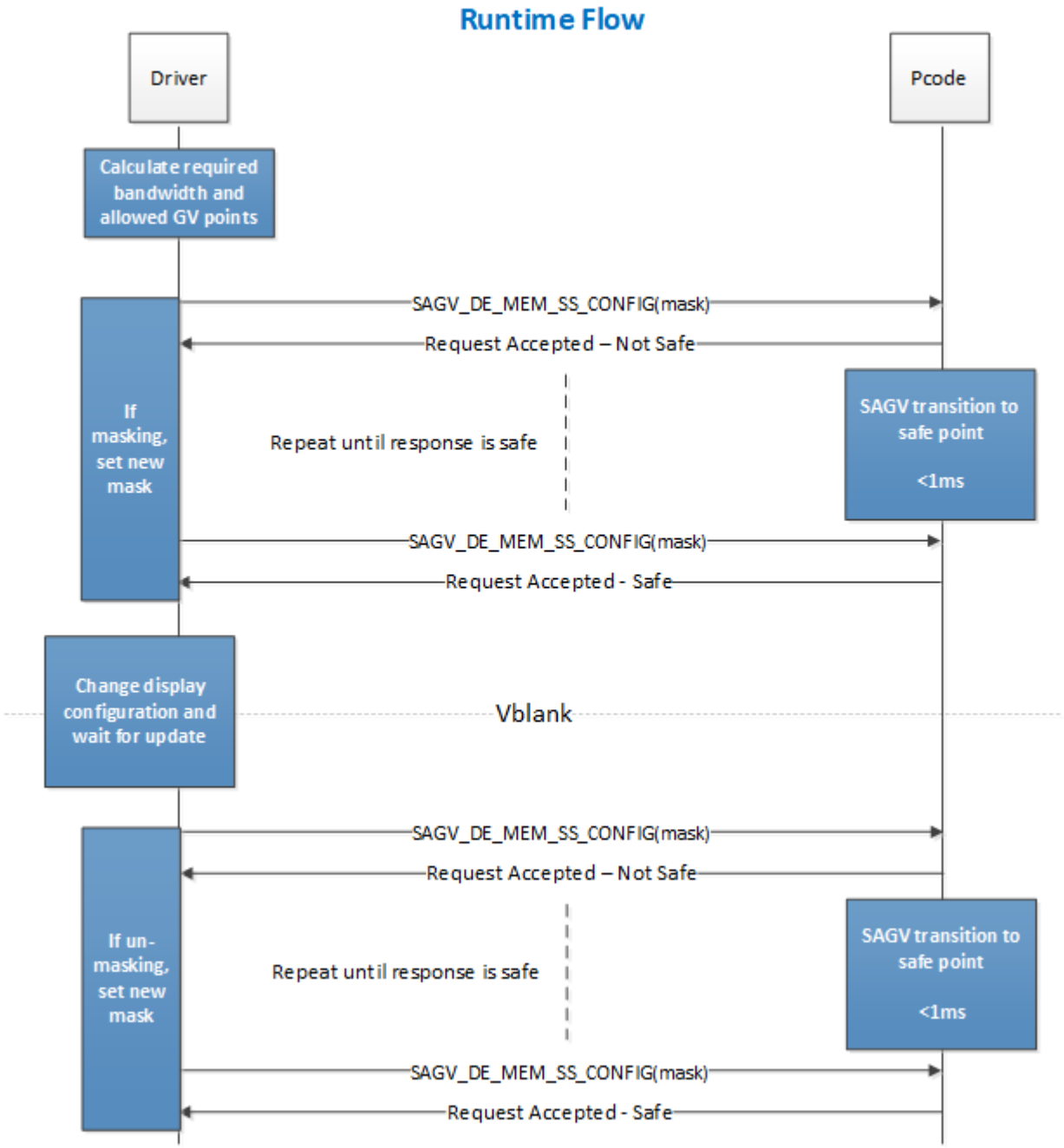
1. Driver reads memory subsystem (MEM SS, MemSS) configuration information from Pcode using the GT Pcode mailbox command MAILBOX_GTDRIVER_CMD_MEM_SS_INFO, first reading the global info and number of GV points, then reading GV info for each GV point.
    - Described in the Mailbox Commands section
2. Driver uses the mailbox info to calculate the available memory bandwidth for each GV point and number of enabled planes.
    - Described in the Bandwidth Restrictions chapter, Available Memory Bandwidth Calculation section.

There are cases where the highest numbered GV point allows less bandwidth for display than a lower point.

Overclocking reprogramming of memory parameters dynamically (system running without a reboot when changes are applied) can result in a display underrun since the display driver will not know that the configuration has changed.

Runtime

Adjust the allowed GV points to match display configuration changes.



1. Before changing display configuration in a way that impacts required bandwidth or number of planes, driver calculates the new bandwidth requirement.
    - Described in the Bandwidth Restrictions chapter, Required Memory Bandwidth Calculation section.
2. Driver finds which GV points supply enough available bandwidth to meet the required bandwidth and which GV points do not supply enough and must be restricted (masked off).
    - At least one GV point of each type must always remain unmasked. The GV point of each type providing the highest bandwidth for display must always remain unmasked.

- Each cycle through this sequence can only mask or unmask points of a GV type, not do both for a single GV type. It is allowed for one GV type to unmask while another masks, then one cycle through this sequence will mask one type of GV and the unmask another type of GV.
- If no GV point will provide enough bandwidth, then the display configuration must be constrained to fit within the available bandwidth.
- If driver has lost track of what was previously masked off, it can mask off all but the highest bandwidth point of each type to get to a known safe setting, then cycle through the full sequence again to set the correct mask.
- To disable SAGV when watermarks do not meet the SAGV block time requirement, mask off all the QGV points except for the point providing the highest bandwidth for display. To re-enable SAGV, unmask all of the QGV points that meet the bandwidth requirements for display. See the Watermark Calculations section for more info.

3. If masking GV points
    1. Driver uses GT Pcode mailbox command MAILBOX_GTDRIVER_CMD_SAGV_DE_MEM_SS_CONFIG to send Pcode the bitmask of restricted points and polls by repeatedly issuing the command with the same bitmask until receiving the acknowledge with point safe for each type, hitting a timeout after 1 millisecond, or receiving the error code.
        - Command described in the Mailbox Commands section
        - Pcode adjusts GV
            1. Pcode performs needed changes to ensure that all future transitions will be performed to the allowed states only.
            2. If pcode currently resides in a point which is restricted, it issues a GV transition to an allowed state.
            3. Pcode responds to the mailbox with point safe.
        - If only one type of GV is masking, keep the mask unchanged for other types.
4. Driver changes the display configuration
5. If unmasking GV points
    1. Driver waits for the configuration to be updated (typically at the next vertical blank). Driver can wait even longer, but should eventually complete unmasking in order to let GV move between points to give more optimal power and performance.
    2. Driver uses GT Pcode mailbox command MAILBOX_GTDRIVER_CMD_SAGV_DE_MEM_SS_CONFIG to send Pcode the bitmask of restricted points and polls by repeatedly issuing the command with the same bitmask until receiving the acknowledge with point safe for each type, hitting a timeout after 1 millisecond, or receiving the error code.
        - Command described in the Mailbox Commands section
        - Pcode adjusts GV
            1. Pcode performs needed changes to ensure that all future transitions will be performed to the allowed states only.

2. If pcode currently resides in a point which is restricted, it issues a GV transition to an allowed state.
3. Pcode responds to the mailbox with point safe.

- If only one type of GV is unmasking, keep the mask unchanged for other types.

All points are unrestricted by default until driver requests otherwise or BIOS disables SAGV.

Pcode will reset the bitmask from display on the graphics device function level reset (FLR).

The BIOS mailbox for SAGV gets precedence over Display Mailbox for GV request.

Display driver must allow at least one GV point of each type at all times. Driver must not mask off all the points of any type.

Display driver must wait for the previous GV point restriction to finish before starting a new restriction.

Legacy Behavior

The legacy command MAILBOX_GTDRIVER_CMD_DE_LTR_SETTING has the effect of masking off all but the highest numbered QGV point. the command continues to be supported and can be used as a limited alternative to MAILBOX_GTDRIVER_CMD_SAGV_DE_MEM_SS_CONFIG. The highest numbered QGV point is not always the highest bandwidth for display, and masking off more QGV points than necessary is not optimal for power and performance, so the legacy command is of limited use and should be phased out by newer software. Eventually a future project may de-feature the legacy command.

Mailbox Commands

Mailbox Access Routine

1. Ensure any previous GT Driver Mailbox transaction is complete
2. Write GT Driver Mailbox Data0=<request data 31:0> and GT Driver Mailbox Data1=<request data 63:32>
3. Write GT Driver Mailbox Interface RUN_BUSY=1, PARAM2=<parameter2>, PARAM1=<parameter1>, COMMAND=<command>
4. Poll GT Driver Mailbox Interface for RUN_BUSY==0
    - Timeout and fail after 150 us
    - Response COMMAND encoding listed below
5. Read GT Driver Mailbox Data0=<response data 31:0> and Data1=<response data 63:32>

MAILBOX_GTRDIVER_CMD_MEM_SS_INFO

This mailbox command provides the MemSS information requested by display. Pcode collates this information from MemSS registers, parses display relevant information and passes it to display on query. It contains subcommands selected through the field PARAM1[15:8] of the INTERFACE register.

MAILBOX_GTRDIVER_CMD_MEM_SS_INFO_SUBCOMMAND_READ_GLOBAL_INFO

Subcommand used to read MemSS global configuration.

Mailbox COMMAND=0xD and PARAM1=0x0

Response Data Bits	Description
3:0	DDR Type; 0:DDR4, 1:DDR5, 2:LPDDR5, 3:LPDDR4, 4:DDR3, 5:LPDDR3
7:4	Number of populated channels
11:8	Number of enabled QGV points

If GV is fused disabled, BIOS should configure GV such that only 1 point will be reported.

MAILBOX_GTRDIVER_CMD_MEM_SS_INFO_SUBCOMMAND_READ_QGV_POINT_INFO

Subcommand used to read QGV point related timing info. Input is the QGV index following the number of enabled QGV points from the global configuration. Out of bounds QGV index results in error code MAILBOX_GTDRIVER_CC_ILLEGAL_DATA error.

Mailbox COMMAND=0xD and PARAM1=0x1 and PARAM2=<QGV index, counting from 0>

Response Data Bits	Description
15:0	Dclk in multiples of 16.6666 MHz
23:16	TRP in DCLKs
31:24	tRCD in DCLKs
39:32	TRDPRE in DCLKs
48:40	TRAS in DCLKs

Pcode reports the maximums values from across multiple memory channels.

MAILBOX_GTDRIVER_CMD_SAGV_DE_MEM_SS_CONFIG

This mailbox command provides pcode with the bitmap containing the list of allowed GV operation points. Value of 1 in the mask is restricting the associated GV point.

Mailbox COMMAND=0xE

Request Data Bits	Description
0	Restricted QGV point 0
1	Restricted QGV point 1
2	Restricted QGV point 2
3	Restricted QGV point 3
4	Restricted QGV point 4
5	Restricted QGV point 5
6	Restricted QGV point 6
7	Restricted QGV point 7
Other bits	Reserved: Must program all 0s

The response data indicates when a safe GV point has been reached.

Response Data Encoding
Bit 1:0 0x0 = Request accepted. QGV point safe. 0x1 = Request accepted. QGV point not safe yet. Poll again. 0x2 = Request rejected error. QGV point cannot be be made safe. It may be blocked by a BIOS override or fuse. Retry with different points or fail.

The response command indicates if there were any errors.

Response COMMAND Encoding
00h = Success 03h = Illegal data or all GV points masked off 04h = Illegal subcommand 06h = Mailbox locked (BIOS loading not done or GV points not programmed or BIOS has only enabled a single GV point) 11h = Rejected (BIOS overriding the GV selection)

Pcode will quickly respond to accept the request, then take up to 1 millisecond (typically 100-200us) to move to a safe point. Driver has to poll Pcode with repeated MAILBOX_GTDRIVER_CMD_SAGV_DE_MEM_SS_CONFIG commands (with unchanged request data field) to find when the move to the safe point is complete.

MAILBOX_GTDRIVER_CMD_DE_LTR_SETTING (Legacy Command)

This mailbox command is the legacy method for disabling Qclk SAGV. It requests Pcode to move to the highest numbered enabled Qclk GV point and then hold there, equivalent to masking off all but one QGV point.

Mailbox COMMAND=0x21

Request Data Bits	Description
2:0	EL_THLD LTR Override: 0 = Disable Qclk GV 3 = Enable Qclk GV
3	Interlace Override: Unused must program with 0
4	VGA Override: Unused must program with 0
63:5	Reserved: Must program all 0s

The response data indicates when a safe Qclk point has been reached.

Response Data Encoding
0x0 = Request accepted. Qclk point not safe yet. Poll again.
0x1 = Request accepted. Qclk point safe.

Pcode will quickly respond to accept the request, then take up to 1 millisecond (typically 100-200us) to move to a safe point. Driver has to poll Pcode with repeated MAILBOX_GTDRIVER_CMD_DE_LTR_SETTING commands (with unchanged request data field) to find when the move to the safe point is complete.

Frame Buffer Compression

FBC Registers

FBC_CFB_BASE

FBC_CTL

DPFC_CONTROL_SA

DPFC_CPU_FENCE_OFFSET

FBC Overview

Frame Buffer Compression (FBC) gives a lossless compression of the display frame buffer to save power by reducing system memory read bandwidth and increasing the time between display engine reads to system memory.

FBC is only available on specific plane(s), depending on project. See FBC_CTL for details. FBC compresses pixels for the plane(s) it is attached to as they are displayed. The compressed data is written into the Compressed Frame Buffer (CFB) in graphics data stolen memory. The compressed data is then read the next time the same line needs to be displayed. Changes to the display front buffer (currently displayed memory surface) through host aperture (GMADR) tiling fences (on projects that support that), render (RCS), and blitter (BCS) are tracked and cause the compressed lines to be invalidated and recompressed. Flips or changes to plane size and panning cause the entire buffer to be recompressed (nuke).

FBC Compression Limit

The FBC compression limit reduces the size of the Compressed Frame Buffer (CFB) by limiting which lines will be compressed. This is used when the graphics stolen memory available for the FBC CFB is smaller than the size of the original uncompressed frame buffer. There is also a FBC compressed vertical limit, listed in FBC_CTL, which is the maximum number of lines FBC can compress. Lines beyond the vertical limit do not need to be accounted for in the CFB size.

When the compression limit is 1:1, every line is written to the CFB, so the CFB width is the same as the original uncompressed frame buffer.

When the compression limit is 2:1, only lines that compress to 1/2 their original size will be written to the CFB, so the CFB width can be 1/2 the original uncompressed frame buffer.

When the compression limit is 4:1, only lines that compress to 1/4 their original size will be written to the CFB, so the CFB width can be 1/4 the original uncompressed frame buffer.

DSM allocation in bytes = min (640, ceiling (plane height / 4))* Compressed buffer stride in cachelines * 64

The 16bpp plane pixel format requires compression limit to be 2:1 or 4:1.

Compressed buffer stride in cachelines = at_least (ceiling [plane_width_in_pixels / (32 * compression_limit_factor)] * 8 + 1)

FBC Programming Overview

1. Set up the compressed frame buffer.
    - The compressed buffer resides in graphics data stolen memory.
    - The stolen memory must be contiguous and un-cached.
    - The stolen memory needed for compressed frame buffer must be greater or equal to CFB size (calculation above).
    - Manage the compressed buffer size at run-time by balancing other graphics memory needs with the FBC allocation, and implement appropriate memory needs prioritization schemes.
2. Tracking for CPU host front buffer modifications
    - CPU Host Front Buffer Tracking sequence below
3. Tracking for display front buffer rendering
    - Render Tracking sequence below
4. Tracking from display front buffer BLTs
    - Blitter Tracking sequence below

LRI commands to MSG_FBC_REND_STATE are used as part of the render and blitter tracking. Those LRIs must be followed SRM commands to the same address.

LRI to MSG_FBC_REND_STATE with data 0x00000004 tells FBC to nuke and invalidate the entire compressed buffer.

Render Tracking With Nuke

- Software must send the nuke LRI after each render to the display front buffer

1. Render commands that touch the display front buffer
    a. Render submission
    b. PIPE_CONTROL
    c. LRI to MSG_FBC_REND_STATE with data 0x00000004 (nuke)
    d. SRM to read MSG_FBC_REND_STATE and store to a scratch page
2. More render commands that touch the display front buffer
    a. Render submission
    b. PIPE_CONTROL

c. LRI to MSG_FBC_REND_STATE with data 0x00000004 (nuke)
d. SRM to read MSG_FBC_REND_STATE and store to a scratch page

3. Render commands that do not touch the display front buffer
    a. Render submission
    b. PIPE_CONTROL

Blitter Tracking With Nuke

- Software must send the nuke LRI after each BLT to the display front buffer. Never set 221d0h bit 3 (Address Valid for FBC).

1. BLT commands that touch the display front buffer
    a. BLT submission
    b. MI_FLUSH_DW
    c. LRI to MSG_FBC_REND_STATE with data 0x00000004 (nuke)
    d. SRM to read MSG_FBC_REND_STATE and store to a scratch page
2. More BLT commands that touch the display front buffer
    a. BLT submission
    b. MI_FLUSH_DW
    c. LRI to MSG_FBC_REND_STATE with data 0x00000004 (nuke)
    d. SRM to read MSG_FBC_REND_STATE and store to a scratch page
3. BLT commands that do not touch the display front buffer
    a. BLT submission
    b. MI_FLUSH_DW

MSG_FBC_REND_STATE address for pipe A FBC is 0x50380

CPU Host Front Buffer Tracking

- Host tracking must be setup by software, then hardware will track FBC line invalidations to the specified fence in the graphics aperture.
    1. Program Y offset from the CPU fence to the Display Buffer base in DPFC_CPU_FENCE_OFFSET.
    2. Program CPU fence ID and enable CPU fence tracking in DPFC_CONTROL_SA.
    3. Disable CPU Fence tracking in DPFC_CONTROL_SA if no tiling fence is mapped to display buffer.

Display Plane Enabling with FBC

- FBC has to be enabled after the attached display plane is enabled.
- This is the general sequence. See FBC_CTL for any workarounds.
  1. Enable display plane
  2. Enable FBC as described in FBC_CTL register

Display Plane Disabling with FBC

- FBC has to be disabled at least one frame before FBC attached display plane disabling.
- This is the general sequence. See FBC_CTL for any workarounds.
  1. Disable FBC as described in FBC_CTL register.
  2. Wait for at least one start of vblank for the disable double-buffering.
  3. Disable display plane.

Watermarks

The watermark registers are used to control the display to memory request timing. The watermarks must be programmed according to the Display Watermark Programming section.

Plane and cursor watermark registers are in the planes section.

WM_MISC

WM_LINETIME

DE_POWER1

DE_POWER2

DC States

DC_STATE_EN

Sequences for Power Wells

Registers

PWR_WELL_CTL

PWR_WELL_CTL_AUX

PWR_WELL_CTL_DDI

FUSE_STATUS

Functions Within Each Well

Except where noted, the registers for a function reside within the same power well as that function.

Note: Some views of the PRM will show a power domain within each register definition. Those domains may be incorrect.

- PG0 contains the functions for the graphics PCI device and bringing up the display
  - PCI
  - Clocks
    - The port PLLs are in the PLL and IO/PHY/AFE power domains and require power enabling which is explained in the mode set sequences
  - Shared Functions
    - Interrupts are in PG0, except for pipe interrupts which reside in the power wells associated with the pipes.
    - MBus, except for PIPE_MBUS_DBOX_CTL which reside in the power wells associated with the pipes.
    - Data Buffer (DBUF) registers are in PG0, but the function is in PG1.
  - Central Power, except for FBC which resides in PG1.
  - Top Level GTC. DDI Level GTC is in the power well associated with each DDI.
- PG1 contains the functions for pipe A display on eDP or MIPI DSI, without VDSC.
  - Data Buffer (DBUF) function is in PG1, but the registers are in PG0.
  - Pipe A and associated Planes, except for VGA.
  - Transcoder A, including PSR
  - Transcoder DSI
  - DDI A-C, including Aux. See note below about IOs.
  - FBC
  - DSS
- PG2 contains the functions for VDSC on pipe A
  - VDSC/joining for pipe A.
- PG3 the functions for pipe B, external displays, and VGA.
  - Pipe B and associated Planes and VDSC/joining
  - Audio
  - Transcoder WD
  - VGA. The VGA_CONTROL register is in PG0, but requires PG3 to be enabled before VGA is enabled.
  - Transcoder B
  - DDI USBC1-6, including Aux. See note below about IOs.
  - KVMR. Hardware will automatically enable PG2 and PG3 for KVMR.

- PG4 contains pipe C.
    - Pipe C and associated Planes and VDSC/joining
    - Transcoder C (registers reside in PG3, but access path goes through associated pipe)
- PG5 contains pipe D.
    - Pipe D and associated Planes and VDSC/joining
    - Transcoder D (registers reside in PG3, but access path goes through associated pipe)
- The port PLLs and IOs and Aux IOs are in the IO/PHY/AFE power domains and require power enabling which is explained in the mode set sequences.
    - For typeC ports, there are separate Aux channel IO power domains for thunderbolt (TBT fields) and for non-thunderbolt (USBC fields).

Enable Sequence

PG0 -> PG1 -> PG2 -> PG3 -> PG4 -> PG5

Power wells must be enabled sequentially and no intermediate power well can be skipped.

1. **PG0 is controlled by the CPU power controller and will be automatically enabled any time software accesses display**
2. **PG1 is enabled using** Sequences to Initialize Display
3. **To turn on VDSC with pipeA**
    a. PG1 must be turned on first (display initialization sequence will turn PG1 on)
    b. Enable PWR_WELL_CTL Power Well 2 Request
    c. Wait for PWR_WELL_CTL Power Well 2 State == Enabled, timeout after 20 us
    d. Wait for FUSE_STATUS FUSE PG2 Distribution Status == Done, timeout after 20 us
4. **To turn on PipeB, PG1 and PG2 must be turned on and then do the following to turn on PG3**
    a. Enable PWR_WELL_CTL Power Well 3 Request
    b. Wait for PWR_WELL_CTL Power Well 3 State == Enabled, timeout after 20 us
    c. Wait for FUSE_STATUS FUSE PG3 Distribution Status == Done, timeout after 20 us
5. **To turn on PipeC, PG1, PG2, and PG3 must be turned on and then do the following to turn on PG4**
    a. Enable PWR_WELL_CTL Power Well 4 Request
    b. Wait for PWR_WELL_CTL Power Well 4 State == Enabled, timeout after 20 us
    c. Wait for FUSE_STATUS FUSE PG4 Distribution Status == Done, timeout after 20 us
6. **To turn on PipeD, PG1, PG2, PG3 and PG4 must be turned on and then do the following to turn on PG5**
    a. Enable PWR_WELL_CTL Power Well 5 Request
    b. Wait for PWR_WELL_CTL Power Well 5 State == Enabled, timeout after 20 us
    c. Wait for FUSE_STATUS FUSE PG5 Distribution Status == Done, timeout after 20 us

Disable Sequence

Power wells have to be turned off in the reverse order of the enabling. The following sequence should be followed to disable each power well

1. Disable PWR_WELL_CTL Power Well # Request
2. Wait for PWR_WELL_CTL Power Well # State == disabled, timeout after 20 us

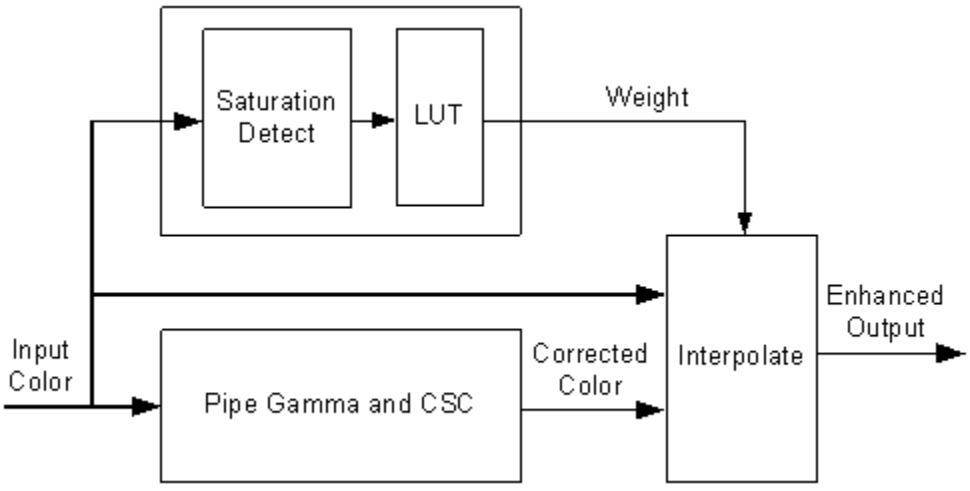
Pipe

Pipe Color Gamut Enhancement

Pipe color gamut enhancement is used to enhance display of standard gamut content on wide gamut displays. It processes the color value from before and after the pipe gamma and color space correction blocks to create the color gamut enhanced output. The typical usage is to output the pipe gamma and CSC corrected color for areas of low saturated content and the input (not gamma or CSC corrected) color for areas of high saturated content. It is not recommended to use color gamut enhancement with wide gamut inputs.

CGE_CTRL
CGE_WEIGHT

The pipe Gamma and CSC must be programmed to either the split gamma mode or gamma after CSC mode when using pipe color gamut enhancement.



The saturation level of the pipe gamma and CSC input color is detected and used to index into a look up table (LUT) containing programmable weights. The saturation values are linearly distributed across the LUT indexes from the lowest index for lowest saturation to the highest index for highest saturation.

The enhanced output color is created by using the weight value to interpolate between the input color and corrected color. See the following table of weights to amount of input or corrected color used to create the enhanced output color.

Weighting of input and corrected colors

Weight from LUT	Amount of Input Color in Enhanced Output	Amount of Corrected Color in Enhanced Output
00 0000b (minimum)	0%	100%
...	...	...
00 1000b	25%	75%
...	...	...
01 0000b	50%	50%
...	...	...
01 1000b	75%	25%
...	...	...
10 0000b (maximum)	100%	0%

Example weight programming

CGE LUT Index	CGE Weight Value Decimal	CGE Weight Value Binary	CGE Weight Percent Input Color	CGE Weight Percent Corrected Color
0 (lowest saturation)	0	00 0000b	0%	100%
1	0	00 0000b	0%	100%
2	0	00 0000b	0%	100%
3	0	00 0000b	0%	100%
4	0	00 0000b	0%	100%
5	0	00 0000b	0%	100%
6	1.6	00 0010b	5%	95%
7	3.2	00 0011b	10%	90%
8	4.8	00 0101b	15%	85%
9	6.4	00 0110b	20%	80%
10	8.64	00 1001b	27%	73%
11	12.8	00 1101b	40%	60%
12	19.2	01 0011b	60%	40%
13	25.6	01 1010b	80%	20%
14	28.8	01 1101b	90%	10%
15	32	10 0000b	100%	0%
16 (highest saturation)	32	10 0000b	100%	0%

HDR

High Dynamic Range (HDR)

Key HDR features

- HDR mode supports up to 3 planes in each pipe.
- Tone mapping support in planes.
- Linear blending of HDR planes.
- Linear scaling support in planes and pipes.
- Dedicated chroma upsampler to handle YUV420.
- Programmable color space convertors in planes/cursor.
- Enhanced gamma mode for PQ encoding.

Hardware Capabilities

FP16 Normalizer:

FP16 Normalizer normalize the pixels to -1.0 to 1.0 range. FP16 Normalizer is programed in the PLANE_PIXEL_NORMALIZE register. The programmed FP16 value gets multiplied with the pixel value for normalizing them. Out of band values get clamped to -1.0 to 1.0 value. The output of this block directly feeds into plane CSC block bypassing Chroma up-sampler, input CSC and de-gamma. The FP16 source content must be linear with no gamma encoding.

Chroma up-sampler:

HDR planes have a dedicated bi-linear Chroma up-sampler for converting P0xx/NV12 source pixel formats to YUV444. Supports up to 4k resolution. Chroma up-sampler is programmed in the PLANE_CUS_CTL register.

CUS supports 4 different chroma siting positions that can be programmed through the initial phase. For initial phase programming, refer PLANE_CUS_CTL.

Input CSC:

A programmable 3x3 color space converter generally used for converting YUV content to RGB. This operates in non-linear space. Plane input CSC is programmed in the PLANE_INPUT_CSC_* registers along with the enable bit in the PLANE_COLOR_CTL register.

Plane pre-CSC Gamma:

Plane de-gamma LUT is used for linearizing HDR or SDR source content with gamma. The 128 entries LUT is uniformly spaced with additional 3 entries for 1.0,3.0 and 7.0. Plane input CSC is programmed in the PLANE_PRE_CSC_GAMC_INDEX_ENH and PLANE_PRE_CSC_GAMC_DATA_ENH registers along with the enable bit in the PLANE_COLOR_CTL register.

Plane CSC:

The programmable plane CSC can be used for color space conversion of source content to BT.2020 or panel color gamut. Plane CSC is programmed in the PLANE_CSC_* registers along with the register bit in the PLANE_COLOR_CTL register.

Plane post-CSC Gamma:

In HDR mode, this block can be used for tone mapping/dynamic range adjustment of each source content to a common reference luminance range before blending. The LUT must be programmed to use "Multiply mode" in the PLANE_POST_CSC_GAMC_INDEX_ENH, PLANE_POST_CSC_GAMC_DATA_ENH and PLANE_COLOR_CTL registers.

Hardware computes the pseudo luminance of the incoming pixel using the following equation, uses it index the LUT and compute an adjustment factor 'F'. Toned mapped output R, G and B values are computed by scaling the input R, G and B channel by 'F'.

Lin = 0.25*Red input + 0.625*Green input + 0.125*Blue input.

Plane scaling:

Plane scaling is supports both linear and non-linear scaling modes. The scaling mode is programmed in the PS_CTRL. In HDR mode, scaling and blending operations are generally performed in linear mode.

Linear Blending:

With precision improvements, hardware supports blending in linear mode. Linear blending is enabled by programming the HDR mode bit in the PIPE_MISC register.

Cursor:

For HDR usages, the cursor plane supports a programmable De-gamma LUT, Color Space Convertor and a Luminance scaler. Luminance scaler scales each color component by a programmed 10 bit value fractional value.

Cursor de-gamma LUT is programmed in in CUR_PRE_CSC_GAMMA_INDEX and CUR_PRE_CSC_GAMMA_DATA registers along with the enable bit in CUR_CTL. Cursor CSC is programmed in the CUR_CSC_COEFF register along with the enable bit in CUR_CTL. Luminance scaling is enabled and programmed in CUR_COLOR_CTL.

Pipe Scaler:

Pipe scaler supports linear scaling.

Pipe Gamma:

Supports a 12 bit multi-segmented gamma mode that provides high quality HDR PQ encoding. Refer to the "Pipe Palette and Gamma" page for details

Pipe LDPST

DPLC_CTL

DPLC_HIST_INDEX

DPLC_HIST_DATA

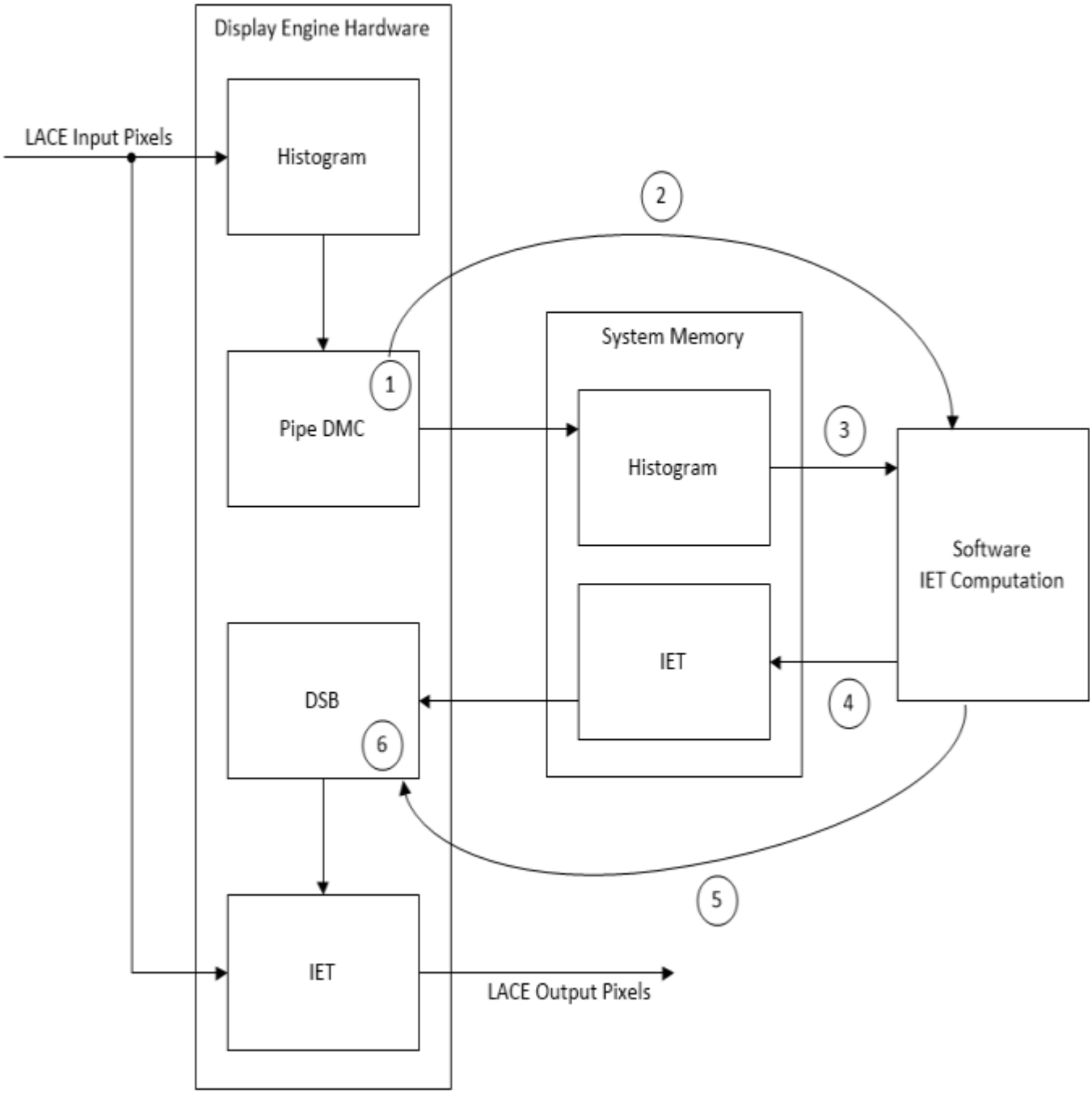
DPLC_IE_INDEX

DPLC_IE_DATA

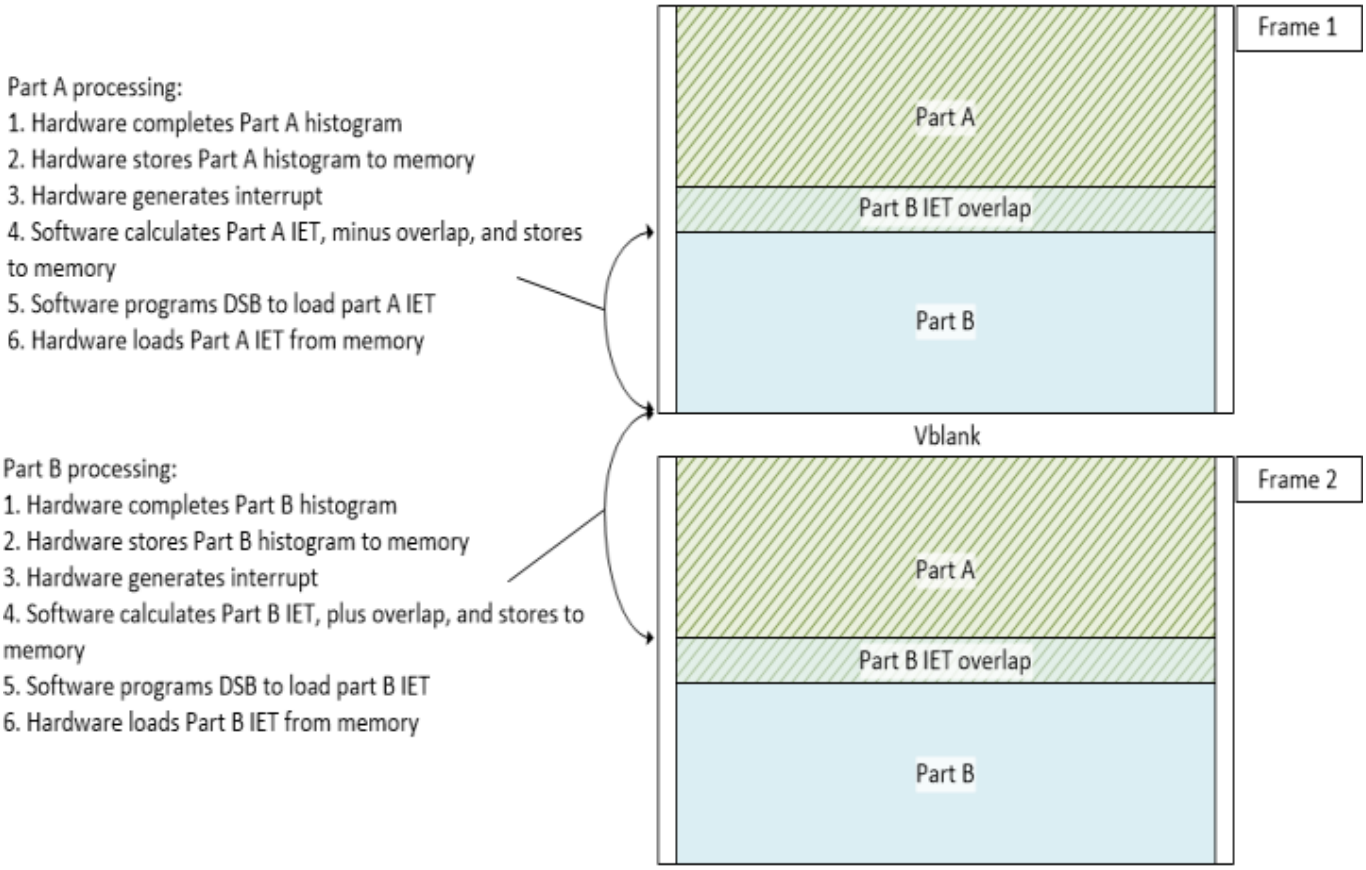
Fast LACE Access

The fast LACE access improves the latency for programming LACE by moving histogram and image enhancement table (IET) programming to system memory instead of display engine MMIO. This also allows software to write updates only to IET tiles that have changes, instead of writing them to all IET tiles when there is any change.

1. DMC reads LACE histogram data and writes it into a system memory location indicated by display driver.
2. DMC triggers interrupt to driver.
3. Display driver reads histogram data and computes IET.
4. Display driver writes IET into system memory in DSB data format. If the same memory surface is used for multiple frames, driver can skip writing any tiles that do not need update since memory will hold the old values of those tiles.
5. Driver programs DSB to start loading IET.
6. DSB reads the IET data from system memory and writes into LACE IET.



To reduce the latency between histogram computation and IET application to 1 frame, each active frame is broken roughly in half, into part A and part B. While hardware computes the histogram of one part, software works on computing IET for the other part. The size of part A and B is programmable in number of LACE tile rows. The part B IET overlaps part A by one tile row in order to provide a smooth transition across parts.



Registers

DPLC_PART_CTL

DPLC_PTRCFG

DPLC_PTRCFG_LIVE

DPLC_RDLENGTH

DPLC_RDLENGTH_LIVE

DPLC_FA_STATUS

DPLC_FA_IMR

DPLC_FA_IIR

DPLC_FA_SURF

DPLC_FA_SURF_LIVE

Part Start and End Programming

Tile Size = 256

Tile Columns = CEILING(Pipe Horizontal Source Size / Tile Size)

Part A Start Tile Row = 1

Part A End Tile Row = FLOOR(Pipe Vertical Source Size / 2 / Tile Size) + 1

Part A Tile Rows = Part A End Tile Row

Part B Start Tile Row = Part A End Tile Row + 1

Part B End Tile Row = CEILING(Pipe Vertical Source Size / Tile Size)

Part B Tile Rows = 1 + Part B Start Tile Row - Part B End Tile Row

The row number programming is zero based, so subtract 1 from these values when programming PART CTL.

Resolutions with less then 3 tile rows in part A or part B are not supported.

Example

5120x3200 pipe source size

Tile Size = 256

Tile Columns = 20

Part A Start Tile Row = 1 (program 0)

Part A End Tile Row = 7 (program 6)

Part A Tile Rows = 7

Part B Start Tile Row = 8 (program 7)

Part B End Tile Row = 13 (program 12)

Part B Tile Rows = 6

Pointer Config Programming

The pointer config and read length are used by hardware to read out the histogram at the end of each part.

Part A X, Y, and DW Indexes = 0.

Part B X and DW Indexes = 0. Part B Y Index = Part B Start Tile Row.

Part A read length = Part A Tile Rows * Tile Columns * DWords per tile

Part B read length = Part B Tile Rows * Tile Columns * DWords per tile

The index values are programmed in the pointer config register, using the same format as DPLC_HIST_INDEX.

DSB Programming

DPLC_IE_INDEX

Part A X, Y, and DW Indexes = 0.

Part B X and DW Indexes = 0. Part B Y Index = Part B Start Tile Row - 1.

DPLC_IE_DATA

Part A number of IE data DWords written by DSB = (Part A Tile Rows - 1) * Tile Columns * DWords per tile

Part B number of IE data DWords written by DSB = (Part B Tile Rows + 1) * Tile Columns * DWords per tile

Overall Sequence

1. Driver setup LACE with fast access
    a. Load DMC Fast LACE access program.
    b. Program DPLC_FA_SURF_PARTA with the memory base address for part A histogram.
    c. Program DPLC_FA_SURF_PARTB with the memory base address for part B histogram.
    d. Write DPLC_CTL to enable LACE function, Fast Access Enable, select enhancement mode. IE Enable is OFF.
    e. Write DPLC_PART_CTL, DPLC_RDLENGTH_LIVE, and DPLC_PTRCFG with partA/PartB programming.
    f. Write DPLC_FA_IMR to unmask histogram copy done interrupts. Other interrupts are optional for error handling.
    g. Configure pipe interrupts to unmask and enable LACE interrupt.
    h. Write DPLC_CTL to enable correction - IE Enable.
2. DMC execution triggered at the end of part A
    a. Read DPLC_RDLENGTH_LIVE_PARTA to obtain number of reads to issue.
    b. Move DPLC_PTRCFG_LIVE_PARTA to DPLC_HIST_INDEX to start histogram reads for part A.
    c. Read DPLC_HIST_DATA as many times as programmed in DPLC_RDLENGTH_LIVE_PARTA.
    d. Write as many cachelines to system memory for part A to the base address specified in DPLC_FA_SURF_PARTA.
    e. Flush writes with a read to the last write location, ignoring the read return value.
    f. Write DPLC_FA_STATUS to clear Part A Histogram Ready and set Part A Histogram Copy Done.
3. Driver
    a. Receive Part A Histogram Copy Done interrupt.
    b. Read histogram part A from system memory at the base address specified in DPLC_FA_SURF_PARTA.
    c. Calculate PART A IET.
    d. Configure DSB with IET part A programming.
    e. Poll for DSB Idle.
    f. Load DSB HTP.
4. DSB
    a. Write DPLC_IE_INDEX to start IET data load.
    b. Write DPLC_IE_DATA entries.
    c. Write DPLC_FA_STATUS to set Part A Load IE.
5. DMC execution triggered at the end of part B
    a. Read DPLC_RDLENGTH_LIVE_PARTB to obtain number of reads to issue.
    b. Move DPLC_PTRCFG_LIVE_PARTB to DPLC_HIST_INDEX to start histogram reads for part B.
    c. Read DPLC_HIST_DATA as many times as programmed in DPLC_RDLENGTH_LIVE_PARTB.
    d. Write as many cachelines to system memory for part B to the base address specified in DPLC_FA_SURF_PARTB.
    e. Flush writes with a read to the last write location, ignoring the read return value.
    f. Write DPLC_FA_STATUS to clear Part B Histogram Ready and set Part B Histogram Copy Done.

6. Driver

a. Receive Part B Histogram Copy Done interrupt.
b. Read histogram part B from system memory at the base address specified in DPLC_FA_SURF_PARTB.
c. Calculate PART B IET.
d. Configure DSB with IET part B programming.
e. Poll for DSB Idle.
f. Load DSB HTP.

7. DSB

a. Write DPLC_IE_INDEX to start IET data load.
b. Write DPLC_IE_DATA entries.
c. Write DPLC_FA_STATUS to set Part B Load IE.

8. Go to 2

Fast Access Status and Interrupts

Histogram Ready

Status

Used by LACE to know when DMC has finished reading the histogram

Set by LACE

Clear by DMC after reading histogram

IIR

No current usage.

Set by LACE when FA status bit sets

Clear by driver on servicing interrupt

Load IE

Status

Used by LACE to know when to load the IE.

Set by DSB after copying IE into LACE

Clear by LACE after load complete

IIR

No current usage.

Set by LACE when FA status bit sets

Clear by driver on servicing interrupt

Histogram Copy Done

Status

Used by DMC to pass the copy done to the interrupt

Set by DMC after copying histogram to memory

Cleared immediately by LACE

IIR

Used by driver to know when histogram is ready in memory and which part to work on

Set by LACE when FA status bit sets

Clear by driver on servicing interrupt

Histogram/IET Late/Overlap

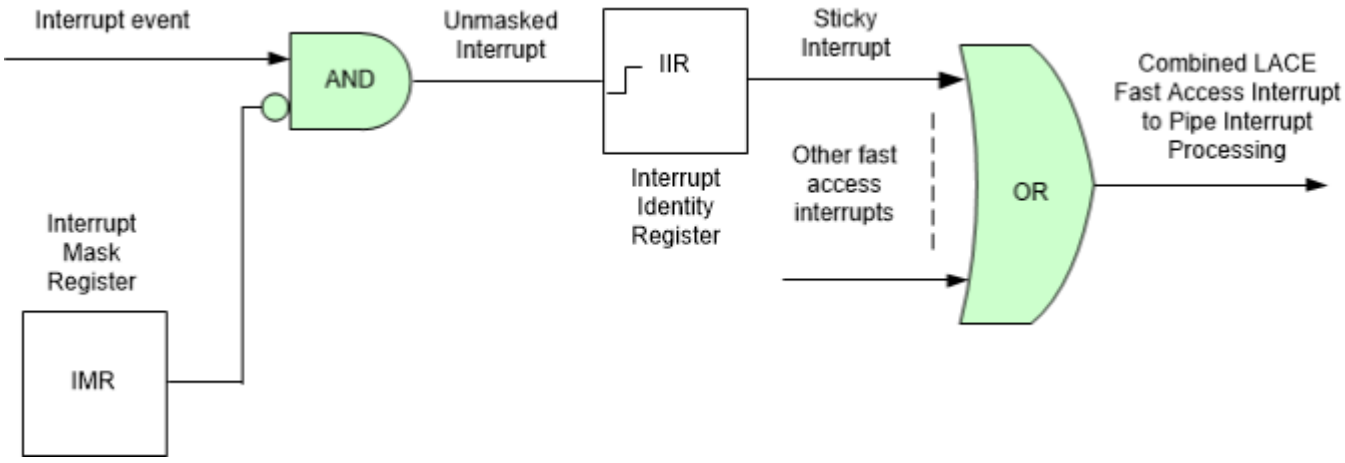
IIR

Used by driver to detect error cases

Set by LACE when error detected

Clear by driver on servicing interrupt

Interrupt Structure



DSB Engine Programming

Display State Buffer Programming

A DSB (Display State Buffer) is a queue of MMIO instructions in the memory which can be offloaded to DSB HW in Display Controller. DSB HW is a DMA engine that can be programmed to download the DSB from memory. It allows driver to batch submit display HW programming. This helps to reduce loading time and CPU activity, thereby making the context switch faster.

It can be chiefly used as an extension of the current flip programming

There are three DSB DMA engines per pipe.

Different DSB's are required if different tasks need to be accomplished simultaneously

DSB's can access only the pipe, plane, and transcoder Data Island Packet registers.

DSB cannot access any registers outside pipe registers, including, but not limited to FBC, clocks, audio, Aux channel, Power Wells, HPD.

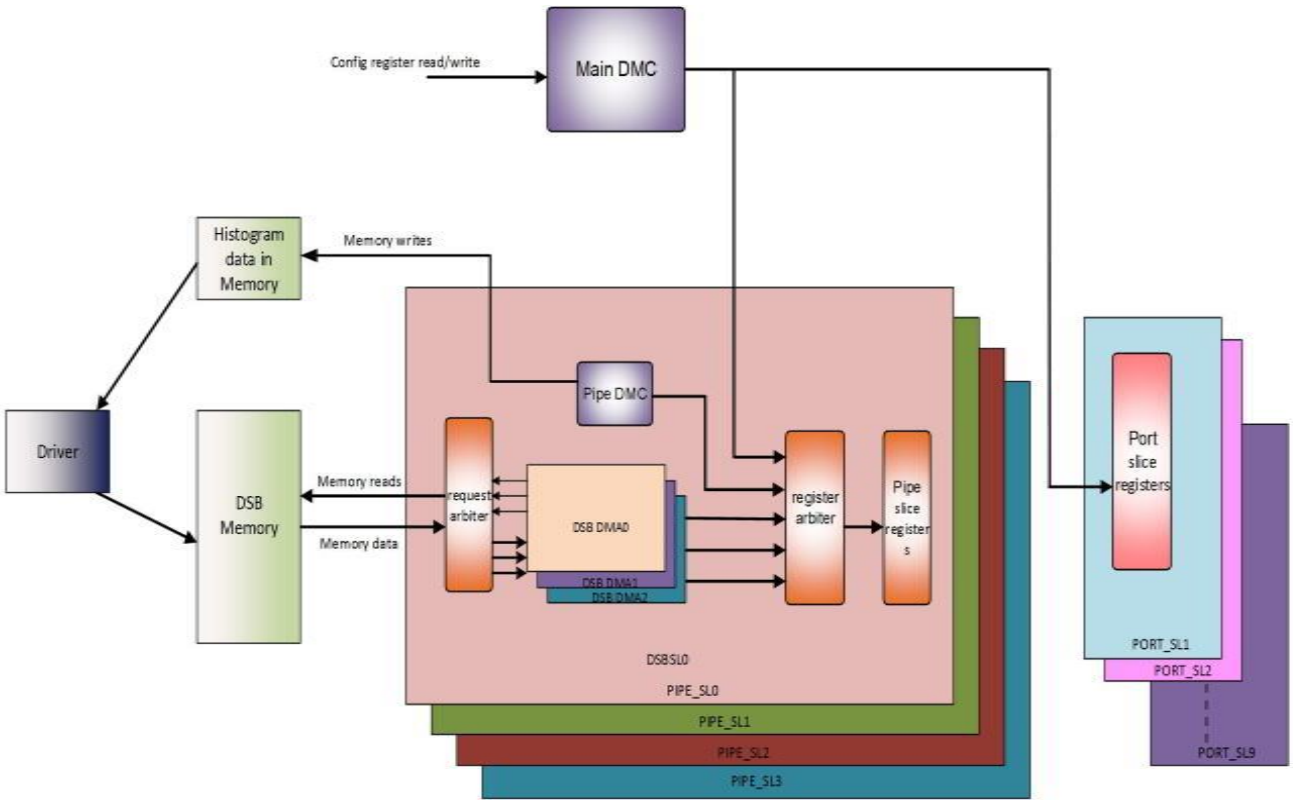
DSB HW can support only register writes (both indexed and direct MMIO writes). There are no registers reads possible with DSB HW engine.

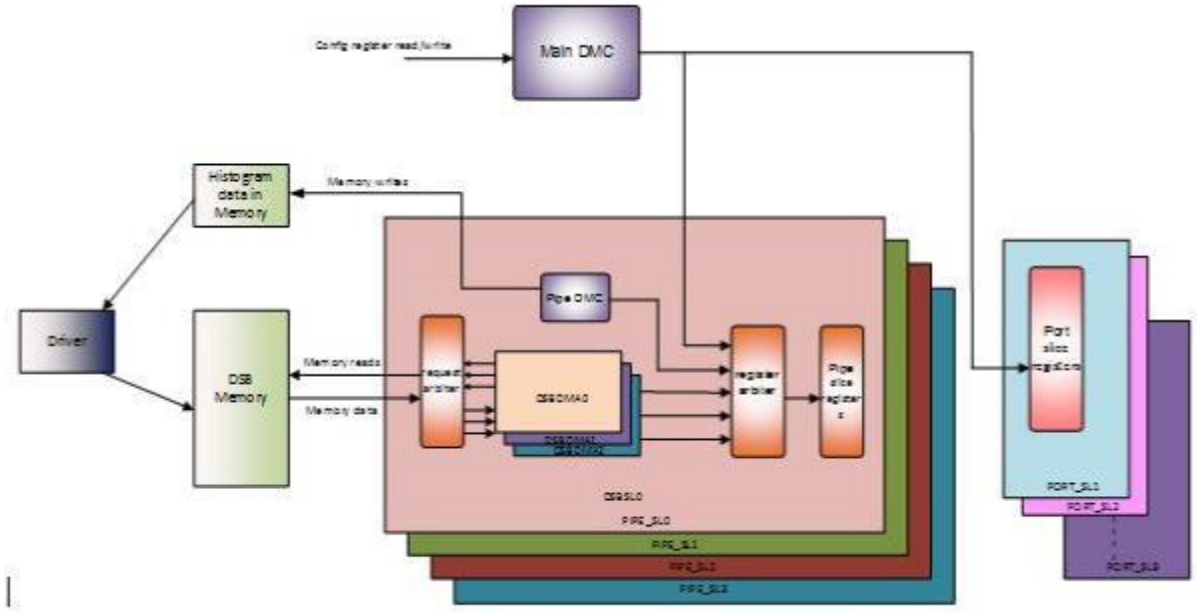
DSB HW has poll function capability.

DSB HW can be used in FAST LACE programming.

DSB HW can be used in HDR Meta data programming.

Below is the block diagram of the DSB HW and shows the general flow of the HW and SW flow. Driver will upload the Display State Buffer with instructions to program the Pipe registers.





Following registers will be used in the DSB HW. Software must ensure correct programming of these registers for the proper function of the HW. Illegal/incorrect programming of these register may result in unexpected behavior of hardware.

DSB HW Registers

DSB_CTRL

DSB_BUFRPT_CNT

DSB_HEAD_PTR

DSB_TAIL_PTR

DSB_MMIOCTRL

DSB_POLLMASK

DSB_POLLFUNC

DSB_INTERRUPT

DSB_PF_LN_LOWER

DSB_PF_LN_UPPER

DSB_PMCTRL

DSB_PMCTRL_2

DSB_CURRENT_HEAD_PTR

DSB_RM_TIMEOUT

DSB_RMTIMEOUTREG_CAPTURE

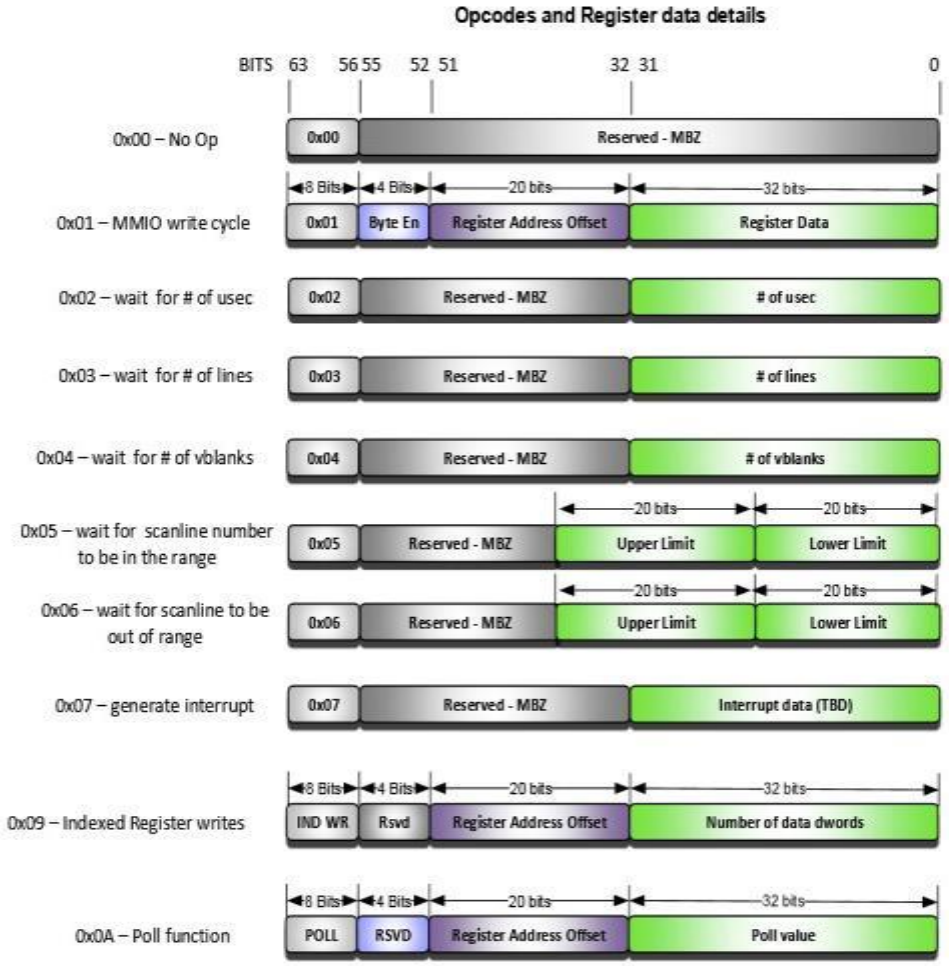
DSB_INTERRUPT

DSB_STATUS

PIPEDMC_CONTROL

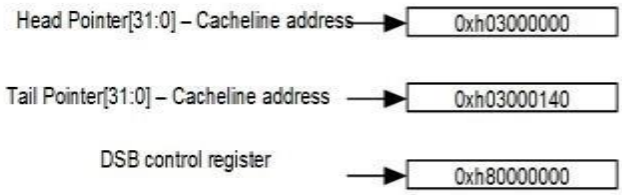
DSB Instructions

The following DSB instructions are supported by the DSB HW. Software must upload the instructions in the defined format shown below.



Below are some examples of the DSB instructions format in the memory.

63	31	0	
20000000 10000000	09F7227c 00000003	00000008 00000007 00000006 00000005 00000004 00000003 00000002 00000001	09F7227c 00000008	0xh03000000
01F72250 8003801d	01F7224c 80020011	01F72248 80010009 01F72244 80004003 01F72240 80004003 01F721c8 00000000	00000000 30000000	0xh03000040
01F721cc 00002000	01F7219c 91440000	01F721a4 00000000 01F72268 00004007 01F7225c 8005c02f 01F72258 80050029	01F72254 80044023	0xh03000080
07000000 FFFFFFFF	00000000 00000000	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	01F72180 84000408	0xh03000100



General Programming Sequence of the DSB HW

1. Driver will form a display state buffer using the supported DSB instructions to program the display registers and upload it in the memory.

2. Driver must program the **DSB_CTRL** register to enable the DSB engine. SW may choose to start the DSB DMA engine to start the memory requests right away or wait for a vblank to trigger the DSB DMA engine. SW may also chose to repeat to play the the same buffer again on every vblank by setting re-iterate bit of the **DSB_CTRL** register. SW must follow the programming rules defined in the **DSB_CTRL** register bits.

3. Driver cannot update or modify the DSB registers within the DSB program.

4. Driver must program the **DSB_HEAD_PTR** with the first cacheline address of the program and **DSB_TAIL_PTR** with the last cacheline address of the DSB program plus one. DSB HW DMA engine will be triggered on the tail pointer update.

- Head and Tail Pointers (32 bits each).
- SW must make sure that there must atleast one cacheline in the DSB program when updating Head and Tail pointer. i.e. head and tail pointer cannot be equal.
- Cacheline aligned graphics address.
- TP points to last location+1 cacheline address.
- SW should not modify these DSB registers when DSB is busy.
- Each cacheline can have 8 instructions.
- 64 bits per instruction.
- Fill incomplete cachelines with NoOps
- Cacheline data is in little Endian.
- One DSB can write to the registers of the another DSB engine within the pipe.

5. Driver can poll on the DSB Status bit in the **DSB_CTRL** register to check if the DSB engine has completed. If DSB program has an interrupt instruction as the last instruction then an interrupt will generated to SW which can be used to check the status the DSB DMA engine.

- SW can enable the interrupts by programming the appropriate enabled in **DSB_INTERRUPT** register.
- A pipe interrupt at **DE_PIPE_INTERRUPT** will be generated when interrupts are correctly enabled in the IMR and IER registers. Bits 13,14,15 shows the status of interrupts from each DSB engine.
- SW must clear the sticky bits in the **DSB_INTERRUPT** first and then clear the ISR bits in the **DE_PIPE_INTERRUPT**

6. Each of the instructions are described below in detail. Each of the supported functions/capabilities are also described that SW can use to program DSB HW. Every instruction must start at a 64 bit boundary.

NOOPs

The opcode of this instruction is 0x00. All the instructions must start and end on the cacheline boundaries. NoOPs can be used to complete the cachelines. All the 64 bits of this instruction are 0's.

MMIO Writes

The opcode of this instruction is 0x01. Lower 32 bits of the instruction is the MMIO write data. Bits 51:32 is the offset address of the MMIO write. Bits 55:52 are the byte enables of the mmio write instruction.

Wait for number of Microseconds

The opcode of this instruction is 0x02. The lower 32 bits of the instructions will have the number of microseconds to wait. DSB HW will wait for the programmed number of micro seconds before processing the next instruction.

Wait for number of lines

The opcode for this instruction is 0x03. The lower 32 bits of the instruction has the number of lines to wait. DSB HW will wait for the programed number of lines before processing the next instruction.

Wait for number of vblanks

The opcode for this instruction is 0x04. The lower 32 bits of the instruction has the number of vblanks to wait. DSB HW will wait for the programed number of vblanks (pipe undelayed vblank) before processing the next instruction.

Wait for scanline number in range

The opcode for this instruction is 0x05. The lower 40 bits of the instruction has the upper and lower scanline numbers. DSB HW will wait for the scanline number to be within the programmed range of the line numbers before processing the next instruction.

Wait for scanline number out of range

The opcode for this instruction is 0x06. The lower 40 bits of the instruction has the upper and lower scanline numbers. DSB HW will wait for the scanline number to be out of the programmed range of the line numbers before processing the next instruction.

Generate Interrupt

The opcode for this instruction is 0x07. DSB HW, after decoding this instruction, will generate a interrupt when interrupt is enabled in **DSB_INTERRUPT** register and the IMR ad IER register bits are correctly programmed in **DE_PIPE_INTERRUPT** .

Indexed MMIO register writes

The opcode for this instruction is 0x09. The lower 32 bits of this instruction will have the number of the data dwords of the indexed MMIO writes. Bits 51:32 of this instruction will have the address offset.

- The data dwords are followed by this instructions.
- If the number of the data dwords is odd then the last dword of the instruction will be 0's to complete the 64 bit instruction. Example is shown below

Example of Indexed Register Writes with odd number of dwords

Poll Function

The opcode for this instruction is 0x0A. Lower 32 bits of the instruction will have the poll value to be compared and bits 51:32 have the offset of the register that needs to be polled on.

- SW must program the **DSB_POLLMASK** before this instruction is used. Poll mask will have the enable on each of the 32 bits that the data will be compared against.
- SW must also program the **DSB_POLLFUNC** register prior to using the poll function instruction. Poll function has to be enabled before using the poll function.
  - The default value of number of micro seconds to wait before a poll retry is set to 2us.
  - The default number of retries is set to 50 times before a timeout or poll fails
- SW can chose to add these two registers as part of DSB program itself before using the POLL instruction. But SW must add 5 NOOPs after MMIO writes instructions to these (**DSB_POLLMASK** and **DSB_POLLFUNC**) registers to allow some time for DSB engine to process the register writes to itself.

Special programming

The following transcoder registers need to special programming if they are part of the DSB program in the buffer. SW must follow the sequence listed below in the DSB program before adding programming of these registers.

1. Set the non-posted bit 8 to 1 in the **DSB_CTRL** register as part of the DSB program.
2. Add four No-Op instructions.
3. Add the transcoder register programming.
4. Set the non-posted bit 8 to 0 in the **DSB_CTRL** register.
5. Add four No-Op instructions
6. Add the rest of the programming (if any).

The list of these special registers is below.

DDI_BUF_CTL
TRANS_DDI_FUNC_CTL2
TRANS_DDI_FUNC_CTL
VIDEO_DIP_CTL
VIDEO_DIP_PPS_DATA_*
VIDEO_DIP_ADAPTIVE_SYNC_DATA_*
TRANS_CONF
TRANS_STEREO3D_CTL
TRANS_DPT_PAT
TRANS_SET_CONTEXT_LATENCY
TRANS_VRR_CTL
TRANS_VRR_FLIPLINE
TRANS_VRR_VMAXSHIFT
TRANS_VRR_VMAX
TRANS_VRR_VMIN
TRANS_VRR_VSYNC
TRANS_LINKN1
SRD_CTL
SRD_PERF_CNT
PSR_MASK
DP_COMP_CTL
DP_TP_CTL
PSR2_CTL

DSB Atomic Usage

The following is the framework for using DSB to program pipe and plane registers to atomically update (all update together for the same frame). DSB can also be used to update registers non-atomically or to update single resources that are self-atomic, with simpler sequences that don't require any wait for vblank.

For this atomic usage, transcoder timings must be programmed to create a window between start of undelayed vblank and delayed vblank. The typical requirement is a 100 microsecond window to provide enough time for DSB to program all the pipe and plane registers. Smaller values may be used for testing. Larger values may exceed required time for filling the display pipeline during vblank when using reduced blanking. ROUNDUP(desired window time/line time) = number of lines to program delayed vblank larger than undelayed vblank.

Fixed Refresh Rate

This requires VRR to be disabled.

Before enabling DSB, set bit 23=1b in the following register for the pipe and DSB to skip waits when PSR is entered.

Pipe A: DSB0 0x70BF0, DSB1 0x70CF0, DSB2 0x70DF0

Pipe B: DSB0 0x71BF0, DSB1 0x71CF0, DSB2 0x71DF0

Pipe C: DSB0 0x72BF0, DSB1 0x72CF0, DSB2 0x72DF0

Pipe E: DSB0 0x73BF0, DSB1 0x73CF0, DSB2 0x73DF0

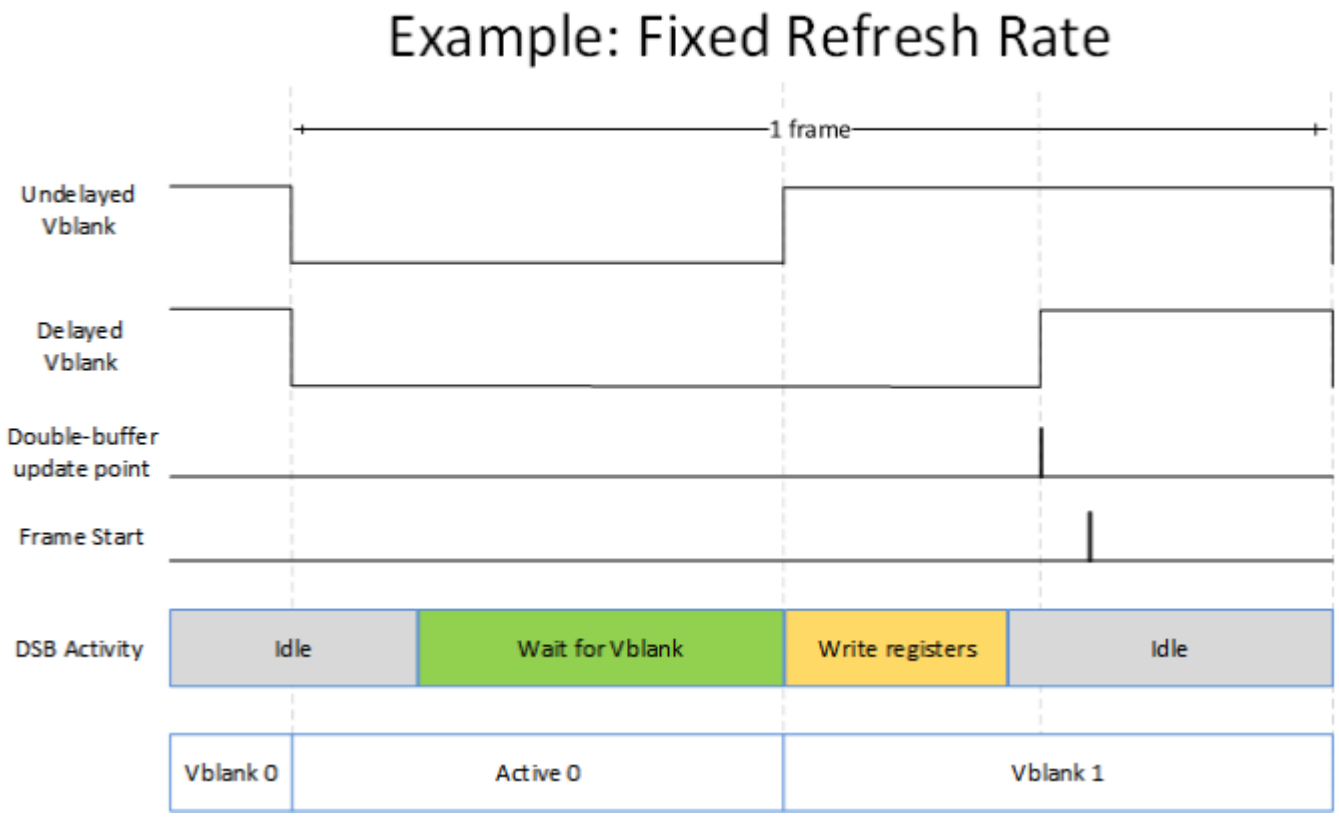
DSB will need to align instruction execution to the start of vblank. Either enable DSB_CTRL Wait for VBLANK or use the wait for vblank instruction.

Follow the General Programming Sequence of the DSB HW to setup and initiate DSB.

Use the following DSB instructions

1. If DSB_CTRL Wait for VBLANK not enabled, Wait for 1 vblank.
2. Write 1 to DSB_STATUS bit 16 to clear any previous indication that DSB was busy during delayed vblank.
3. Write pipe and plane registers to be updated for the next frame.
    - Depending on the registers to write, this can include register writes and indexed register writes.
    - Requirements for writing some registers in certain orders to arm them for double-buffer updates are still required.
4. Optional: Generate Interrupt

After DSB completes, found through interrupt or polling on DSB status, read DSB_STATUS bit 16 to find if delayed vblank started before DSB finished. If it did, then there was an error and the pipe and plane programming may not be complete. Software may log the error and attempt recovery.



Variable Refresh Rate

This requires VRR with push bit mode.

Before enabling VRR, set bit 31 in the following register for the transcoder attached to this pipe to enable VRR safe window generation.

Transcoder A: 0x420C0

Transcoder B: 0x420C4

Transcoder C: 0x420C8

Transcoder D: 0x420D8

Before enabling DSB, set bits 23=1b, 15:14=11b, bits 7:6=11b in the following register for the pipe and DSB to enable VRR safe window to be used instead of vblank and to skip waits when PSR is entered.

Pipe A: DSB0 0x70BF0, DSB1 0x70CF0, DSB2 0x70DF0

Pipe B: DSB0 0x71BF0, DSB1 0x71CF0, DSB2 0x71DF0

Pipe C: DSB0 0x72BF0, DSB1 0x72CF0, DSB2 0x72DF0

Pipe E: DSB0 0x73BF0, DSB1 0x73CF0, DSB2 0x73DF0

DSB will need to align instruction execution to the VRR safe window. Either enable DSB_CTRL Wait for VBLANK during DSB setup or use the wait for vblank instruction.

Follow the General Programming Sequence of the DSB HW to setup and initiate DSB.

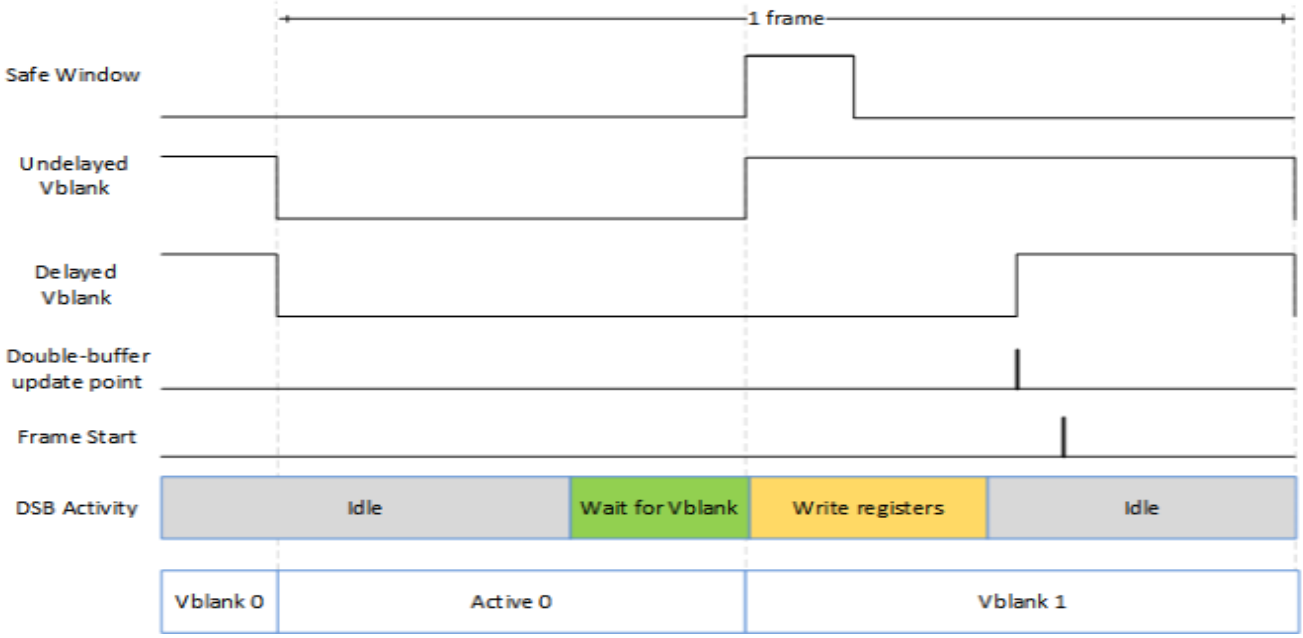
Use the following DSB instructions

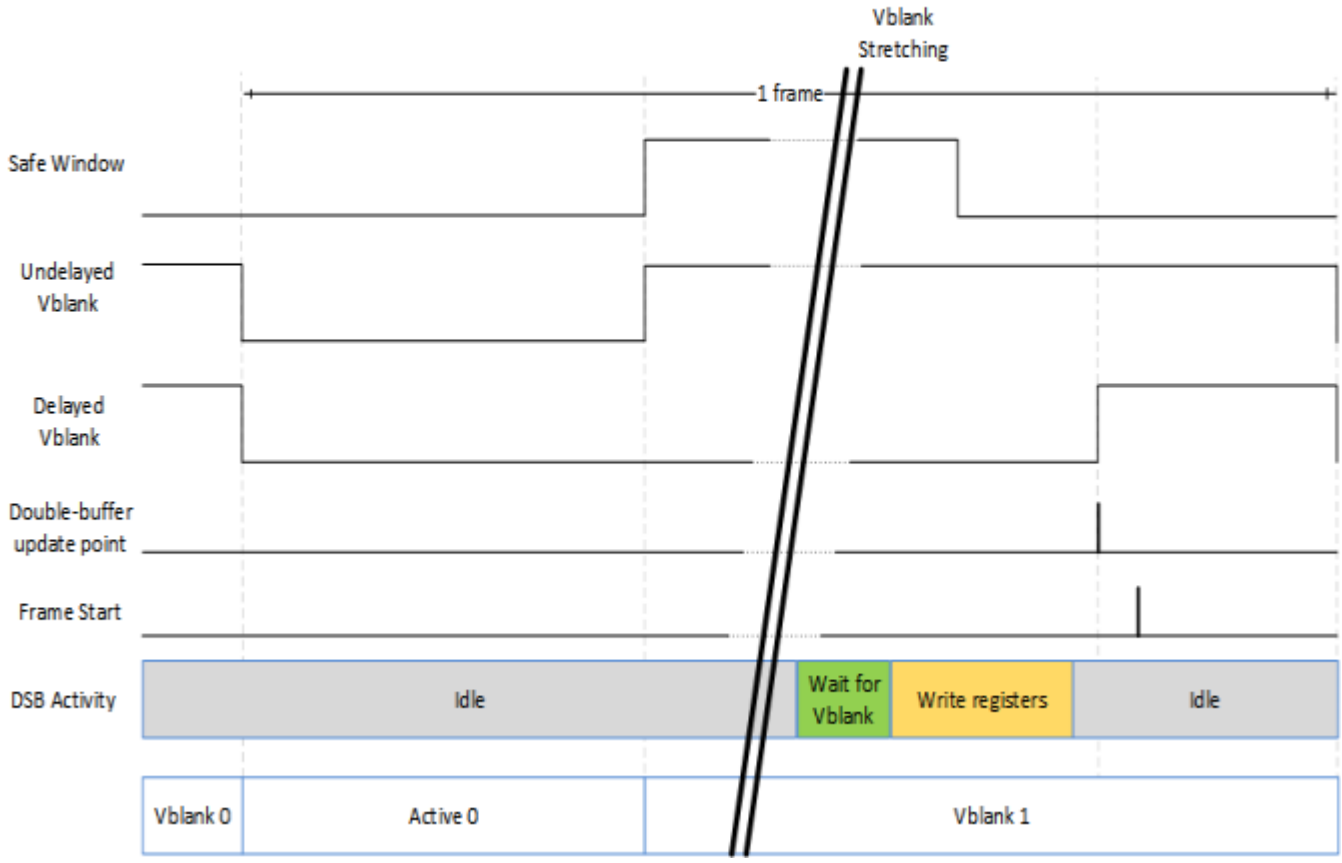
1. If DSB_CTRL Wait for VBLANK not enabled, Wait for 1 vblank.
2. Write 1 to DSB_STATUS bit 16 to clear any previous indication that DSB was busy during delayed vblank.
3. Write 1 to transcoder push bit*. This step can be done here or in step 5. Initiating push at this point will allow the vblank to end sooner, but is different from the non-DSB programming flow that does push at the end.
4. Write pipe and plane registers to be updated for the next frame.
    - Depending on the registers to write, this can include register writes and indexed register writes.
    - Requirements for writing some registers in certain orders to arm them for double-buffer updates are still required.
5. Write 1 to transcoder push bit* if not done in earlier step.
6. Optional: Generate Interrupt

After DSB completes, found through interrupt or polling on DSB status, read DSB_STATUS bit 16 to find if delayed vblank started before DSB finished. If it did, then there was an error and the pipe and plane programming may not be complete. Software may log the error and attempt recovery.

*If driver knows that push is already guaranteed to be set outside of this DSB, such as by another DSB running in parallel, then this push can optionally be skipped. This requires very careful sequencing to ensure that this DSB does not get out of alignment with whatever is issuing the push.

Example: VRR – DSB Wait Starts Before Vblank





Adding Dynamic Metadata

If dynamic metadata will be sent just once, then it can be programmed together with the pipe and plane registers in the previous sequences.

Color Space Conversion

CSC_COEFF

CSC COEFFICIENT FORMAT

CSC_PREOFF

CSC_POSTOFF

CSC_MODE

OUTPUT_CSC_COEFF

OUTPUT_CSC_PREOFF

OUTPUT_CSC_POSTOFF

The high color channel is the most significant bits of the color. The low color channel is the least significant bits of the color. The medium color channel is the bits between high and low. For example: In

RGB modes Red is in the High channel, Green in Medium, and Blue in Low. In YUV modes, V is in the High channel, Y in Medium, and U in Low.

The color space conversion registers are double buffered and are updated on the start of vertical blank following a write to the CSC Mode register for the respective pipe.

The matrix equations are as follows:

OutputHigh = (CoefficientRY * InputHigh) + (CoefficientGY * InputMedium) + (CoefficientBY * InputLow)

OutputMedium = (CoefficientRU * InputHigh) + (CoefficientGU * InputMedium) + (CoefficientBU * InputLow)

OutputLow = (CoefficientRV * InputHigh) + (CoefficientGV * InputMedium) + (CoefficientBV * InputLow)

Example programming for RGB to YUV is in the following table:

The input is RGB on high, medium, and low channels respectively and the desired YUV output is VYU on high, medium, and low channels respectively.

Program CSC_MODE to put gamma before CSC.

Program the CSC Post-Offsets to +1/2, +1/16, and +1/2 for high, medium, and low channels respectively.

The coefficients and pre and post offsets can be scaled if desired.

	Bt.601		Bt.709	
	Value	Program	Value	Program
RU	0.2990	0x1990	0.21260	0x2D98
GU	0.5870	0x0968	0.71520	0x0B70
BU	0.1140	0x3E98	0.07220	0x3940
RV	-0.1687	0xAAC8	-0.11460	0xBEA8
GV	-0.3313	0x9A98	-0.38540	0x9C58
BV	0.5000	0x0800	0.50000	0x0800
RY	0.5000	0x0800	0.50000	0x0800
GY	-0.4187	0x9D68	-0.45420	0x9E88
BY	-0.0813	0xBA68	-0.04580	0xB5E0

Example programming for YUV to RGB is in the following table:

The input is VYU on high, medium, and low channels respectively.

The output is RGB on high, medium, and low channels respectively.

Program CSC_MODE to put gamma after CSC.

Program the CSC Pre-Offsets to -1/2, -1/16, and -1/2 for high, medium, and low channels respectively.

The coefficients and pre and post offsets can be scaled if desired.

	Bt.601 Reverse		Bt.709 Reverse	
	Value	Program	Value	Program
GY	1.000	0x7800	1.000	0x7800
BY	0.000	0x0000	0.000	0x0000
RY	1.371	0x7AF8	1.574	0x7C98
GU	1.000	0x7800	1.000	0x7800
BU	-0.336	0x9AC0	-0.187	0xABF8
RU	-0.698	0x8B28	-0.468	0x9EF8
GV	1.000	0x7800	1.000	0x7800
BV	1.732	0x7DD8	1.855	0x7ED8
RV	0.000	0x0000	0.000	0x0000

Pipe 3D LUT

The 3D LUT is a pixel modification function which resides in the post blend color processing section of the display pipeline. It is used to apply non-linear transforms on each color component on a per pixel basis. Our LUT implementation uses a 17x17x17 three dimensional matrix of color points, with each point holding a 30 bit pixel value (10 bpc).

3D LUT functionality is supported only in pipe A and pipe B.

LUT_3D_CTL

LUT_3D_INDEX

LUT_3D_DATA

Programming

Enabling 3D LUT

Software should follow the sequence below.

1. Check if the "New LUT Ready" bit is clear. If set, software must wait till the bit is clear. LUT entries must not be change the LUT entries when the "New LUT Ready" is set.
2. Load the desired 3D LUT entries.
3. Set the "Enable" and the "New LUT Ready" bits.

The LUT buffer is double buffered. When 3D LUT functionality is enabled, the hardware observes the "New LUT Ready" bit on every vblank start. If the "New LUT Ready" bit is set, hardware loads the LUT entries in to its working RAM and clears the bit. The 3D LUT functionality works with programmed LUT values in the following frames until it gets disabled. When the "New LUT Ready" bit is clear, the software is allowed to modify the LUT entries.

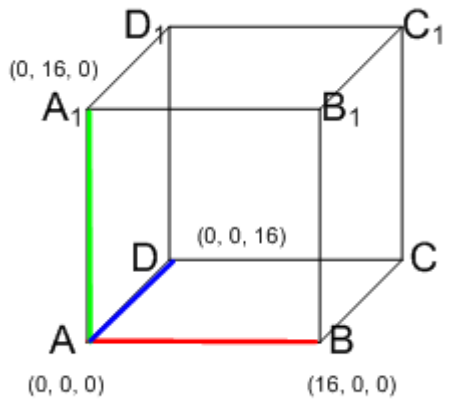
Disabling 3D LUT

1. Clear the "Enable" bit to '0'.

Programming 3D LUT Entries

The LUT array is accessed by an index/data register pair. The index register is read/writable and auto-increments after each read/write to the data register. Write '0' into the index register, followed by 4913 LUT entry writes to the data register.

Each LUT 3D entry is 30 bits and programmed as R10G10B10 (msb... lsb) value in the LUT_3D_DATA register. Since 10 bit values are used for all 17 points, the max value programmed is limited to 1023. A 1:1 mapping should use [0, 64, 128, 192, 256, 320, 384, 448, 512, 576, 640, 704, 768, 832, 896, 960, 1023].



The LUT entries should start at A and end at C_1 following the sequence specified below.

```
Iterate on Red axis from 0 - 16 {
 Iterate on Green axis 0 - 16 {
 Iterate on Blue axis 0 - 16 {
program 3D LUT entry
}
 }
 }
```

Pipe DPST

Registers

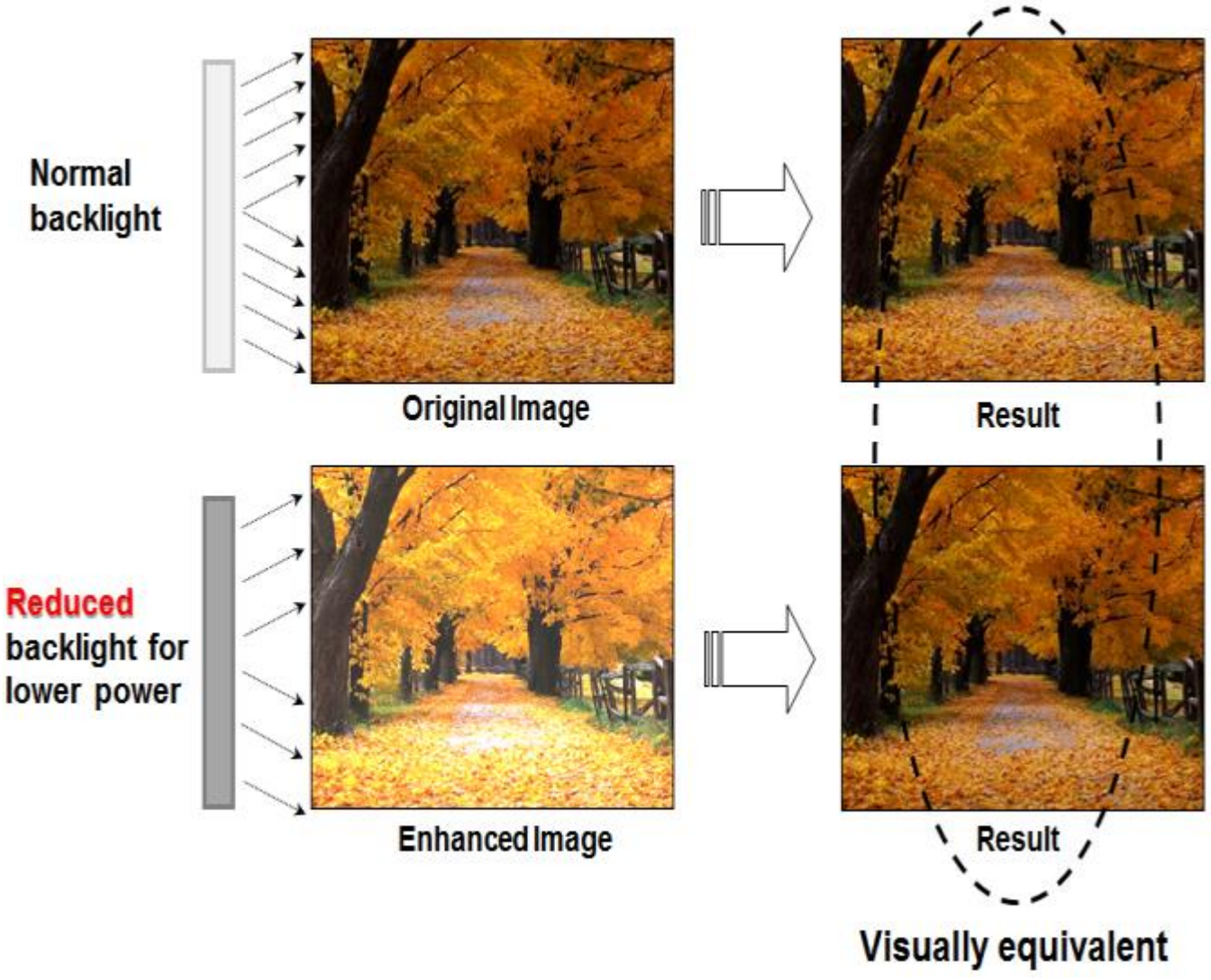
DPST_CTL

DPST_GUARD

DPST_BIN

Overview

Display Power Savings Technology (DPST) achieves significant platform average power savings by dynamically decreasing the display backlight brightness, while increasing the pixel values in the displayed image by a corresponding factor. The goal of DPST is to provide equivalent end-user-perceived image quality at a decreased backlight power level.



DPST generates statistics (histogram) for each image frame that is sent to the display. These statistics are used to determine if, and by how much, the backlight level can be reduced, saving backlight power. In order to maintain the same image brightness, the image pixel values are increased by an amount related to the backlight level reduction.

DPST is composed of three blocks.

Histogram Block

The hardware histogram block generates image statistics based on the pixel stream input. These statistics are used by the Processing block to determine how much the backlight level can be reduced.

The histogram block has the following requirements.

- Generate a histogram for each frame of display data
  - The histogram is generated based on the brightness of each pixel.
    - DPST_CTL Histogram Mode Select selects how the brightness is determined; either HSV max(RGB_{in}), or the luma after converting RGB_{in} to YUV.
  - The histogram is composed of 32 bins with each bin covering a range of 8 values for 8 bit pixel component values. The first bin covers the values 0 thru 7.
    - If the pixel component values use more than 8 bits, the most significant 8 bits are used to form the histogram.
  - Each bin value has enough bits to count every pixel within an image.
  - Enabled by DPST_CTL IE Histogram Enable
- Two sets of histograms
  - The histogram currently being generated and a saved histogram from a previous frame.
  - The current histogram is moved to the saved histogram at the end of a frame in which a threshold event occurs.
  - Each bin of the saved histogram is readable by software.
  - Clear DPST_CTL Bin Register Function Select to TC
  - Wait for vertical blank for switch to TC mode, can skip if step 1 was done more than 1 vblank previously
  - Set DPST_CTL Bin Register Index to 0
  - Read DPST_BIN
  - If DPST_BIN Busy Bit is 1, go to step 3
  - Store DPST_BIN Data
  - Go to step 4 until all 32 bins are read
- Threshold register for comparing the histogram of the current frame to the histogram of the saved frame.
  - If any bin in the current histogram differs from the same bin in the saved histogram by more than the value in DPST_GUARD Threshold Guardband, then a threshold event is generated at the start of vertical blank.
  - DPST_GUARD Guardband Interrupt Delay specifies the number of consecutive frames the threshold must be exceeded before generating a threshold event.
    - This allows filtering out momentary variations from generating a threshold event.
  - DPST_GUARD Histogram Interrupt Enable enables the threshold event interrupt.
  - DPST_GUARD Histogram Event Status is a sticky bit that is set with the interrupt and must be cleared to receive more histogram events.

Enhancement Block

The hardware enhancement block adjusts the pixel values sent to the display, compensating for the brightness loss due to lowering of the display backlight level.

The enhancement block has the following requirements.

- Located after the histogram block
- Enabled by DPST_CTL IE Modification Table Enable
- Find the enhancement factor from a Look Up Table (LUT) with 33 entries
    - Switch (DPST_CTL Enhancement mode)
        - Case Direct: LUT address = most significant bits of RGB_{input}
        - Case Multiplicative: LUT address = most significant bits of HSV max(RGB_{input})
        - Case Additive: LUT address = most significant bits of Y channel after converting RGB_{input} to YUV
    - DPST_CTL IE Table Value Format selects if the enhancement factor is a 1 integer and 9 fractional bits format, or a 2 integer and 8 fractional bits format.
        - The 2 integer and 8 fractional bits format allows for brightness increases nearly to 4x, but with reduced precision.
    - The final enhancement factor is derived by interpolating between the addressed LUT entry and the next entry, using the lower bits of the input.
    - Each entry of the LUT is programmable by software.
    - Set DPST_CTL Bin Register Function Select to IE
    - Wait for vertical blank for switch to IE mode, can skip if step 1 was done more than 1 vblank previously
    - Set DPST_CTL Bin Register Index to 0
    - Write enhancement factor to DPST_BIN Data
    - Go to step 4 until all 33 entries are written
        - The IE values are double-buffered and will update on the next start of pipe vblank. The values are not atomically updated or tied into the global double buffering control (disable/stall), so if programming straddles the pipe vblank start, some values will update in the upcoming frame and some in the frame after that.
- The enhancement factor modifies each input pixel component value with the method selected by DPST_CTL Enhancement Mode.
    - Direct lookup mode replaces the input pixel value with the enhancement factor.
    - Additive mode increases the input pixel value by the enhancement factor.
    - Multiplicative mode multiplies the input pixel value by the enhancement factor.

Processing Block

The software processing block responds to the histogram threshold interrupts, determines how much the backlight level can be reduced, then sets the backlight level and programs the Enhancement block.

The Processing block has the following requirements.

- Enable or disable the hardware blocks based on OS and user control inputs.
- Respond to the histogram threshold interrupts by reading the histogram and calculating a new backlight level based on statistics from the histogram.
    - The calculation is proprietary. In general, darker images will allow greater backlight reduction, and increased power savings.
    - The aggressiveness of backlight reduction can be controlled by the OS and user.
    - The final backlight reduction amount must be combined with backlight level requirements set by the OS, applications, and other power saving technologies.
- Program the Enhancement block to compensate for the brightness loss due to reducing the backlight level.
    - Because the maximum RGB component value is limited, not all pixel values can be perfectly compensated.
    - The number of pixel values which cannot be perfectly compensated is a function of the aggressiveness level.
- Phase-in the backlight level change and pixel enhancement gradually, in order to avoid flickering artifacts.

Pipe Palette and Gamma

The display palette provides a means to correct the gamma of an image stored in a frame buffer to match the gamma of the monitor or presentation device. Additionally, the display palette provide a method for converting indexed data values to color values for VGA and 8-bpp indexed display modes. The display palette is located after the plane blender. Using the individual plane gamma enables, the blended pixels can go through or bypass the palette on a pixel by pixel basis.

PAL_LGC

PAL_PREC_INDEX

PAL_PREC_DATA

PAL_GC_MAX

PAL_EXT_GC_MAX

GAMMA_MODE

PAL_EXT2_GC_MAX

PRE_CSC_GAMC_INDEX

PRE_CSC_GAMC_DATA

PAL_PREC_MULTI_SEG_INDEX

PAL_PREC_MULTI_SEG_DATA

If any gamma value to be programmed exceeds the maximum allowable value in the associated gamma register, then the programmed value must be clamped to the maximum allowable value.

Programming Modes

The display palette can be accessed through multiple methods and operate in one of four different modes as follows.

8 bit legacy palette/gamma mode:

This provides a palette mode for indexed pixel data formats (VGA and primary plane 8 bpp) and gamma correction for legacy programming requirements.

All input values are clamped to the 0.0 to 1.0 range before the palette/gamma calculation. It is not recommended to use legacy palette mode with extended range formats.

For input values greater than or equal to 0 and less than 1.0, the input value is used to directly lookup the result value from one of the 256 palette/gamma entries. The 256 entries are stored in the legacy palette with 8 bits per color in a 0.8 format with 0 integer and 8 fractional bits.

The legacy palette is programmable through both MMIO and VGA I/O registers. Through VGA I/O, the palette can look as though there are only 6 bits per color component, depending on programming of other VGA I/O registers.

Direct lookup (10 bit) gamma mode:

This provides the highest quality gamma for pixel data formats of 30 bits per pixel or less.

For input values greater than or equal to 0 and less than 1.0, the input value is used to directly lookup the result value from one of the first 1024 gamma entries. The first 1024 entries are stored in the precision palette with 10 bits per color in a 0.10 format with 0 integer and 10 fractional bits.

For input values greater than or equal to 1.0 and less than 3.0, the input value is used to linearly interpolate between the 1024th and 1025th gamma entries to create the result value. The 1025th entry is stored in the PAL_EXT_GC_MAX register with 19 bits per color in a 3.16 format with 3 integer and 16 fractional bits.

For input values greater than or equal to 3.0 and less than 7.0, the input value is used to linearly interpolate between the 1025th and 1026th gamma entries to create the result value. The 1026th entry is stored in the PAL_EXT2_GC_MAX register with 19 bits per color in a 3.16 format with 3 integer and 16 fractional bits.

All input values are clamped to the greater than -7.0 and less than 7.0 range before the gamma calculation.

For negative input values, gamma is mirrored along the X-axis, giving the same result as positive input values, except for a negative sign. When gamma input may be negative, the first gamma point should be programmed to a value of 0.0 in order to have a symmetric mirroring.

Interpolated gamma mode:

This mode uses up to 515 gamma entries and the gamma output gets computed through interpolation between the neighboring LUT entries.

The gamma correction curve is represented by specifying a set of gamma entry reference points spaced equally along the curve for values between -1 and 1. For extended values there is an extended gamma entry reference point at the maximum allowed input value.

For input values greater than or equal to 0 and less than 1.0, the input value is used to linearly interpolate between two adjacent points of the first 513 gamma entries to create the result value. The first 512 entries are stored in the precision palette with 16 bits per color in a 0.16 format with 0 integer and 16 fractional bits (upper 10 bits in odd indexes, lower 6 bits in even indexes). The 513th entry is stored in the PAL_GC_MAX register with 17 bits per color in a 1.16 format with 1 integer and 16 fractional bits.

For input values greater than or equal to 1.0 and less than 3.0, the input value is used to linearly interpolate between the 513th and 514th gamma entries to create the result value. The 514th entry is stored in the PAL_EXT_GC_MAX register with 19 bits per color in a 3.16 format with 3 integer and 16 fractional bits.

For input values greater than or equal to 3.0 and less than 7.0, the input value is used to linearly interpolate between the 514th and 515th gamma entries to create the result value. The 515th entry is stored in the PAL_EXT2_GC_MAX register with 19 bits per color in a 3.16 format with 3 integer and 16 fractional bits.

All input values are clamped to the greater than -7.0 and less than 7.0 range before the gamma calculation.

For negative input values, gamma is mirrored along the X-axis, giving the same result as positive input values, except for a negative sign. When gamma input may be negative, the first gamma point should be programmed to a value of 0.0 in order to have a symmetric mirroring.

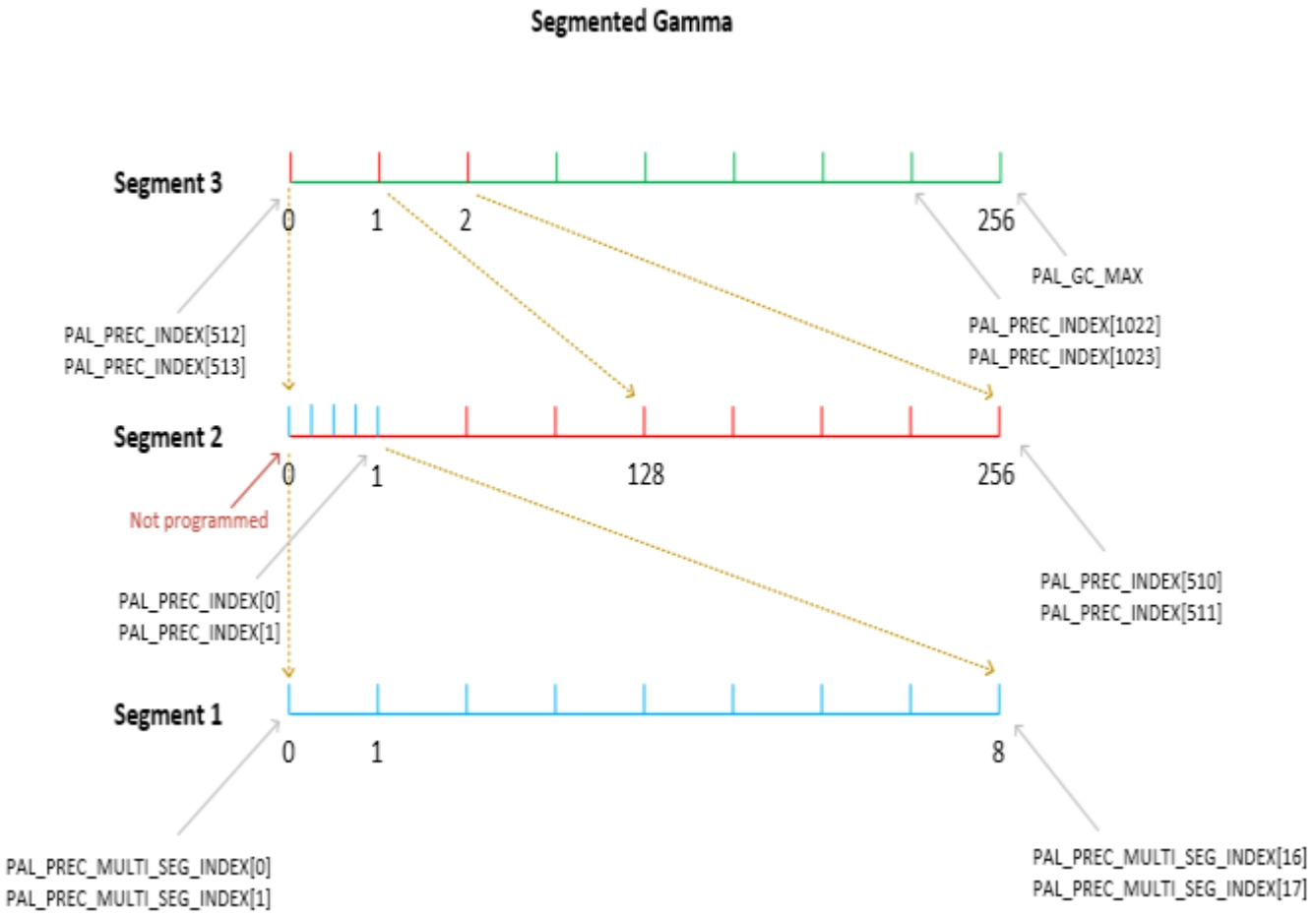
To program the gamma correction entries, calculate the desired gamma curve for inputs from 0 to 3.0. The curve must be flat or increasing, never decreasing. For inputs of 0 to 1.0, multiply the input value by 512 to find the gamma entry number, then store the desired gamma result in that entry. For inputs greater than 1.0 and less than or equal to 3.0, store the result for an input of 3.0 in the 514th gamma entry.

Multi-segmented interpolated gamma mode:

This provides the highest quality gamma for HDR with three segments, each addressing different parts of the gamma curve. The gamma output gets computed through interpolation between the neighboring LUT entries.

The gamma correction curve is represented by specifying a set of gamma entry reference points, spaced equally within each segment, along the curve for values between -1 and 1. For extended values there is an extended gamma entry reference point at the maximum allowed input value.

All input values are clamped to the greater than -7.0 and less than 7.0 range before the gamma calculation.



For input values greater than or equal to 0 and less than 1.0, the input value is used to linearly interpolate between two adjacent points of the gamma entries to create the result value. The gamma entries in all segments are stored in the precision palette with 16 bits per color in a 0.16 format with 0 integer and 16 fractional bits (upper 10 bits in odd indexes, lower 6 bits in even indexes). The 513th entry is stored in the PAL_GC_MAX register with 17 bits per color in a 1.16 format with 1 integer and 16 fractional bits.

For input values greater than 1.0 and less than 3.0, the input value is used to linearly interpolate between the 513th and 514th gamma entries to create the result value. The 514th entry is stored in the PAL_EXT_GC_MAX register with 19 bits per color in a 3.16 format with 3 integer and 16 fractional bits. For input values greater than or equal to 3.0 and less than 7.0, the input value is used to linearly interpolate between the 514th and 515th gamma entries to create the result value. The 515th entry is stored in the PAL_EXT2_GC_MAX register with 19 bits per color in a 3.16 format with 3 integer and 16 fractional bits.

For negative input values, gamma is mirrored along the X-axis, giving the same result as positive input values, except for a negative sign. When gamma input may be negative, the first gamma point should be programmed to a value of 0.0 in order to have a symmetric mirroring.

To program the gamma correction entries, calculate the desired gamma curve for inputs from 0 to 7.0. The curve must be flat or increasing, never decreasing. For inputs of 0 to 1.0, program the gamma segments 1, 2 and 3. For inputs greater than 1.0 and less than or equal to 3.0, store the result for an input of 3.0 in the 514th gamma entry.

The Segment 1 or the super fine segment has 9 entries covering the input range from 0 to 1/(128*256) => [0, 1/(8* 128*256), 2/(8* 128*256) ... 1/(128*256)] . The 9 entries of Segment 1 gamma values gets programmed using the PAL_PREC_MULTI_SEG* registers ranging from index 0 to index 17.

The Segment 2 or the fine segment has 257 entries covering the input range from 0 to 1/128 => [0, 1/(128*256), 2/(128*256) ... 1/128]. The Segment 2 gamma values gets programmed using the PAL_PREC* registers ranging from index 0 to index 510. The Segment 2 zero value is implied in hardware and is not explicitly programmed by software. Segment 2 first value is programmed in PAL_PREC* index 0 as shown in the figure above.

The Segment 3 or the coarse segment has 257 entries covering the input range from 0 to 1 => [0, 1/256, ... 255/256, 1.0]. The first 256 entries of Segment 3 gamma values gets programmed using the PAL_PREC* registers ranging from index 512 to index 1023. The 257th value (1.0) is programmed in PAL_GC_MAX register.

Pipe Control

PIPE_SRCSZ

PIPE_SCANLINE

PIPE_SCANLINECOMP

PIPE_MISC

PIPE_FRMTMSTMP

PIPE_FLIPTMSTMP

PIPE_BOTTOM_COLOR

PIPE_FLIPCNT

PIPE_FRMCNT

PIPE_FLIPDONETMSTMP

PIPE_MISC2

PIPE_STATUS

Pixel Passthrough Operation

For modes of operation where the input image CRC needs to match the output image CRC (i.e. the pixels need to flow through the pixel pipe unmodified from frame buffer to the port), the following programming needs to be done:

1. Use only a single Plane with no cursor (**CUR_CTL**)
2. Use a fixed point, non-planar pixel format without Alpha (**PLANE_CTL**). Make sure the frame buffer format matches the port output format

3. Disable all color correction (i.e. CSC, Gamma, etc), Lace, Scaling, compression and dithering within the Plane and Pipe. See the other sub-sections of the Pipe/Planes chapters for more details:
    1. Universal Plane
    2. Color Space Conversion
    3. Pipe Color Gamut Enhancement
    4. Pipe Palette and Gamma
    5. Pipe LDPST
    6. Pipe 3D LUT
    7. Pipe DPST
    8. Pipe Scaler
    9. DSC (Display Stream Compression)
4. If the Plane that is being used (i.e. enabled) is an HDR Plane, then the HDR Mode bit within **PIPE_MISC** must be set. Otherwise, the HDR Mode bit of PIPE_MISC must be **cleared**.
5. Disable (i.e. truncate) Pixel Rounding (**PIPE_MISC**)

Pipe Scaler

Each scaler has its own set of registers.

Scaler
PS_PWR_GATE
PS_WIN_POS
PS_WIN_SZ
PS_CTRL
PS_VSCALE
PS_HSCALE
PS_VPHASE
PS_HPHASE
PS_ECC_STAT
PS_ADAPTIVE_CTRL
PS_COEF_INDEX
PS_COEF_DATA
SCALER_COEFFICIENT_FORMAT
PS_PROG_HSCALE PS_PROG_VSCALE

Scaler Programmed Coefficients

Two sets of programmed coefficients are available for each scaler. The horizontal filter and vertical filter can be configured individually to use one of these 2 sets. When used for YUV planar format plane scaling, the Y and UV scalers can be configured individually to use one of the 2 sets. Scaler coefficients

are accessed through their respective index and data registers following the mapping shown below. The coefficients must be programmed in the SCALER_COEFFICIENT_FORMAT.

17 phase of 7 taps requires 119 coefficients in 60 dwords per set. The letter represents the filter tap (D is the center tap) and the number represents the coefficient set for a phase (0-16).

Coefficient Set		
Index Value	**Data Value Coefficient 2**	**Data Value Coefficient 1**
00h	B0	A0
01h	D0	C0
02h	F0	E0
03h	A1	G0
04h	C1	B1
...	...	...
38h	B16	A16
39h	D16	C16
3Ah	F16	E16
3Bh	Reserved	G16

Nearest-neighbor scaling (Integer scaling ratios)

Nearest neighbor scaling can be used to maintain the rendering intent of some classic games in integer upscaling scenarios. For enabling nearest-neighbor scaling, the scaler must be set to use "programmed" mode with the center tap (Dxx) values set to 1 and all other values set to 0. The following coefficients values must be used with the coefficients programmed in the SCALER_COEFFICIENT_FORMAT.

Coefficient Set		
Index Value	**Data Value Coefficient 2**	**Data Value Coefficient 1**
00h	B0 = 0	A0 = 0
01h	D0 = 1	C0 = 0
02h	F0 = 0	E0 = 0
03h	A1 = 0	G0 = 0
04h	C1 = 0	B1 = 0
...	...	...
38h	B16 = 0	A16 = 0
39h	D16 = 1	C16 = 0
3Ah	F16 = 0	E16 = 0
3Bh	Reserved	G16 = 0

Register Double Buffering

Prior to D11, the Scaler has the following double buffer trigger points:

1. When the Pipe is disabled:
    a. Any write to the PS_CTRL register (will refer to this as Control DB trigger)
    b. Any write to the PS_WINSZ register (will refer to this as WinSize DB trigger)
2. When the Pipe is enabled:
    a. For registers that are part of the Control DB trigger group, the DB trigger point is on the rising edge of V. Blank when double buffering is armed
    b. For registers that are part of the WinSize DB trigger group, the DB trigger point is after the Frame Start when double buffering is armed

From D11 and onwards, the Scaler has the following double buffer trigger points:

1. When the Pipe is disabled:
    a. Any write to the PS_CTRL register (Control DB trigger)
    b. Any write to the PS_WINSZ register (WinSize DB trigger)
2. When the Pipe is enabled, the DB trigger point is on the rising edge of V. Blank when double buffering is armed and double buffering is allowed (Allow Double Buffer Update Disable = 1 in PS_CTRL)

When the Pipe is enabled, any write to the PS_WINSZ will arm the double buffering for the Scaler registers.

Once double buffering is armed, the disarming point is dependent on the Display generation.

- Prior to D11: Double buffering is disarmed on the next WinSize DB trigger (i.e. after Frame Start)
- D11+ : Double buffering is disarmed on the next rising edge of V. Blank where double buffering is allowed

As implied above, the double buffering for each of the Scaler registers is dependent on the trigger group it is within (i.e. Control or WinSize).

Register	DB Trigger
PS_PWR_GATE	Control
PS_CTRL	Control
PS_VPHASE	WinSize
PS_HPHASE	WinSize
PS_ADAPTIVE_SET_*_CTRL	WinSize
PS_WINPOS	WinSize
PS_WINSZ	WinSize

Note that the Programmable Coefficient sets accessed through the PS_COEF_SET_*_INDEX and PS_COEF_SET_*_DATA registers are on the Control DB trigger.

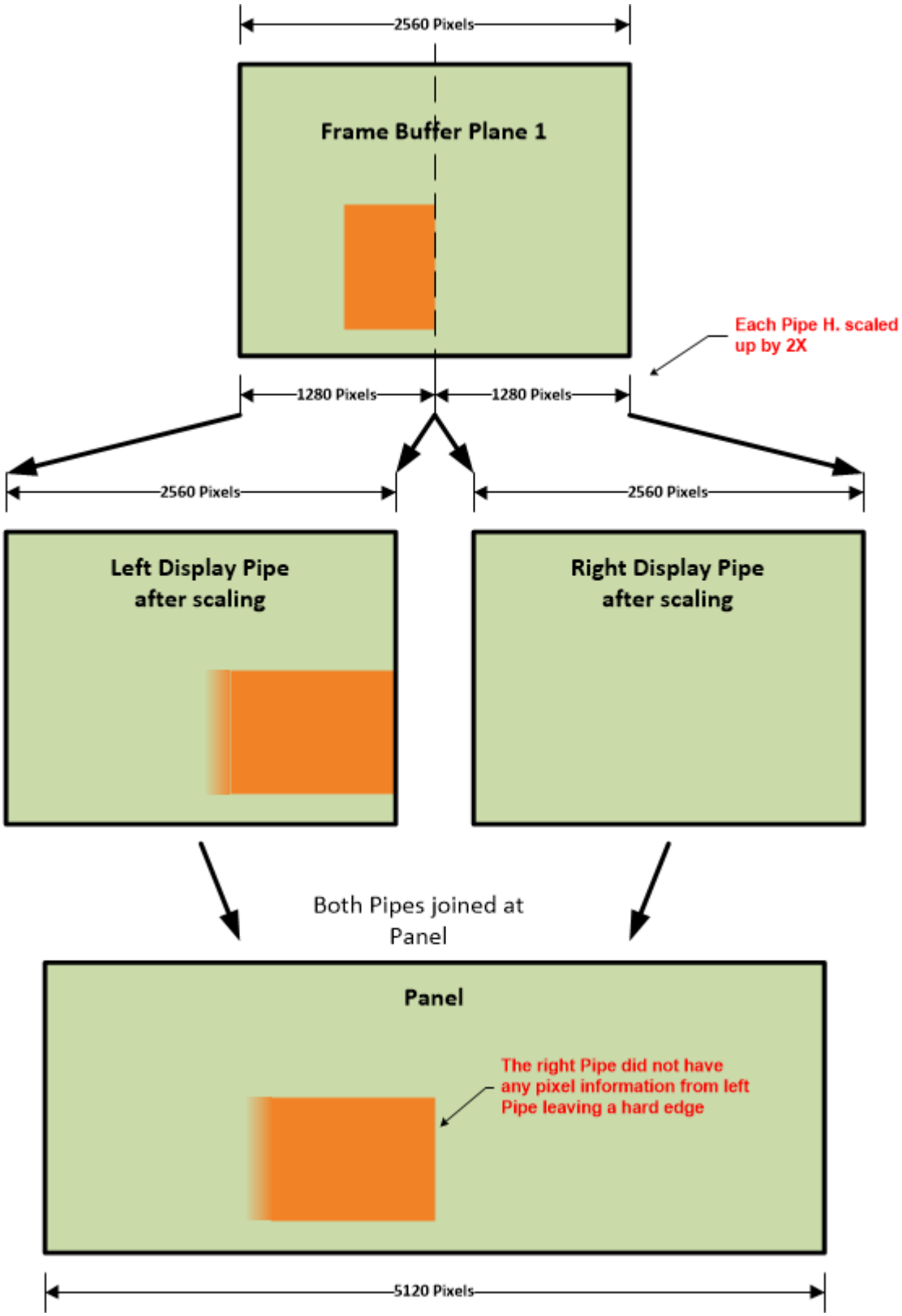
Programming Sequence

The below sequence should be followed when programming the Scaler to ensure proper double buffering:

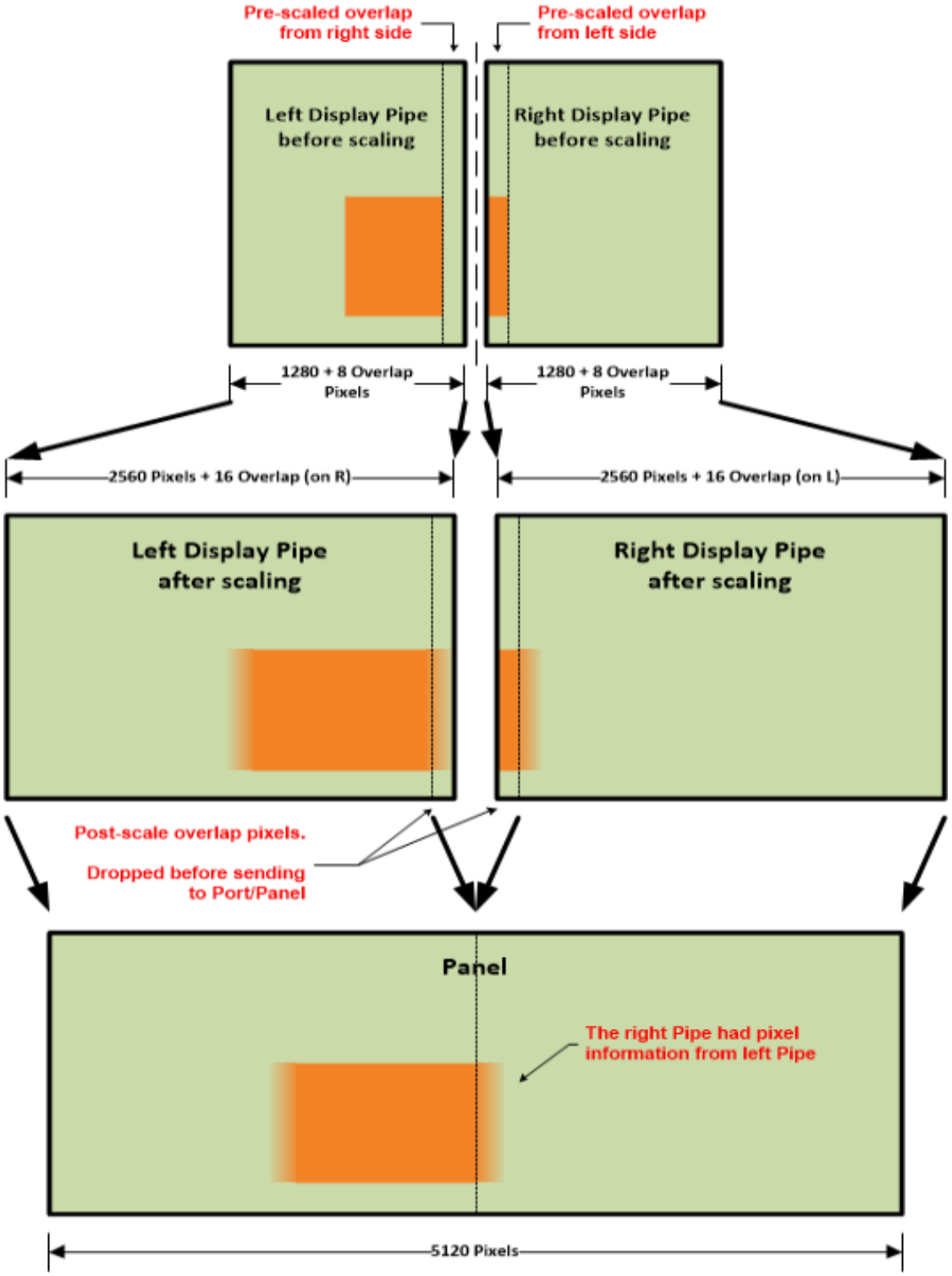
- Configure the Scaler's Programmable Coefficient sets through PS_COEF_SET_*_INDEX and PS_COEF_SET_*_DATA registers, if using Programmable Coefficients
- Configure power gating control of the Scaler SSA's (PS_PWR_GATE), if the Scaler is not already enabled
- Enable and/or configure Scaler (PS_CTRL), if needed
    - If the Scaler is not already enabled, the Scaler will begin the process of powering up the Scaler SSA's
    - Software doesn't have to write to this register to enable Control DB trigger registers (i.e. Programmable Coefficient sets) if the Scaler and Pipe are both enabled
- Configure initial phases (PS_*PHASE), adaptive control (PS_ADAPTIVE_SET_*_CTRL), and window position (PS_WINPOS), if needed
- Configure/write to the PS_WINSZ
    - For pre-D11 products, the write to this register should only be done outside of the V. Blank regime if the Pipe is enabled. There are no restrictions for D11+ products
    - If Software has changed any PS_* register programming and the Pipe is enabled, then Software has to write to this register regardless of whether the value is changing, or not.

Tiled Scaling (Seam Removal)

There are certain usage cases where an image from memory will be horizontally split across two Pipes, scaled up, and then joined at the Port/Panel. When the image is being horizontally scaled up across the seam of the split image if the Scalers within each Pipe do not have some additional pixels from the other Pipe's image, then an artifact at the seam can occur.


2560 Pixels
Frame Buffer Plane 1
Each Pipe H. scaled up by 2X
1280 Pixels
1280 Pixels
2560 Pixels
2560 Pixels
Left Display Pipe after scaling
Right Display Pipe after scaling
Both Pipes joined at Panel
Panel
The right Pipe did not have any pixel information from left Pipe leaving a hard edge
5120 Pixels

By adding some overlapping pixels of the split image around the seam, the Scalers within each Pipe will be able to correctly filter across the seam. At the end of the Pipe (before the image is delivered to the Port) the post-scaled overlap pixels will be dropped.



To perform the cross seam scaling, Software will be responsible for the following:

- It will calculate the pre and post scale excesses needed for each Pipe
- It will include the pre-scale excess within the pre-scaled Horizontal image sizes (e.g. the Horizontal Source Size of the PIPE_SRCSZ register)
- It will include the post-scale excess within the Scaler's Window Size register (PS_WIN_SZ)
- It will program the post-scale excess within the PIPE_SEAM_EXCESS register
- It will program the Horizontal Active size at the transcoder (TRANS_HTOTAL) to be the Panel width
    - This should **not** include any seam pixels

The following sections will discuss how Software will determine the pre/post excess sizes and the initial phase needed for the Scaler processing the right side image.

Location of Seam and Splitting Source Image

When the location of an image in memory is going to cross the seam between the two Pipes, then the PLANE_SIZE, PLANE_POS, and PLANE_OFFSET registers of the Planes carrying the image (one within each Pipe) will determine how that image in memory is split across the two Pipes.

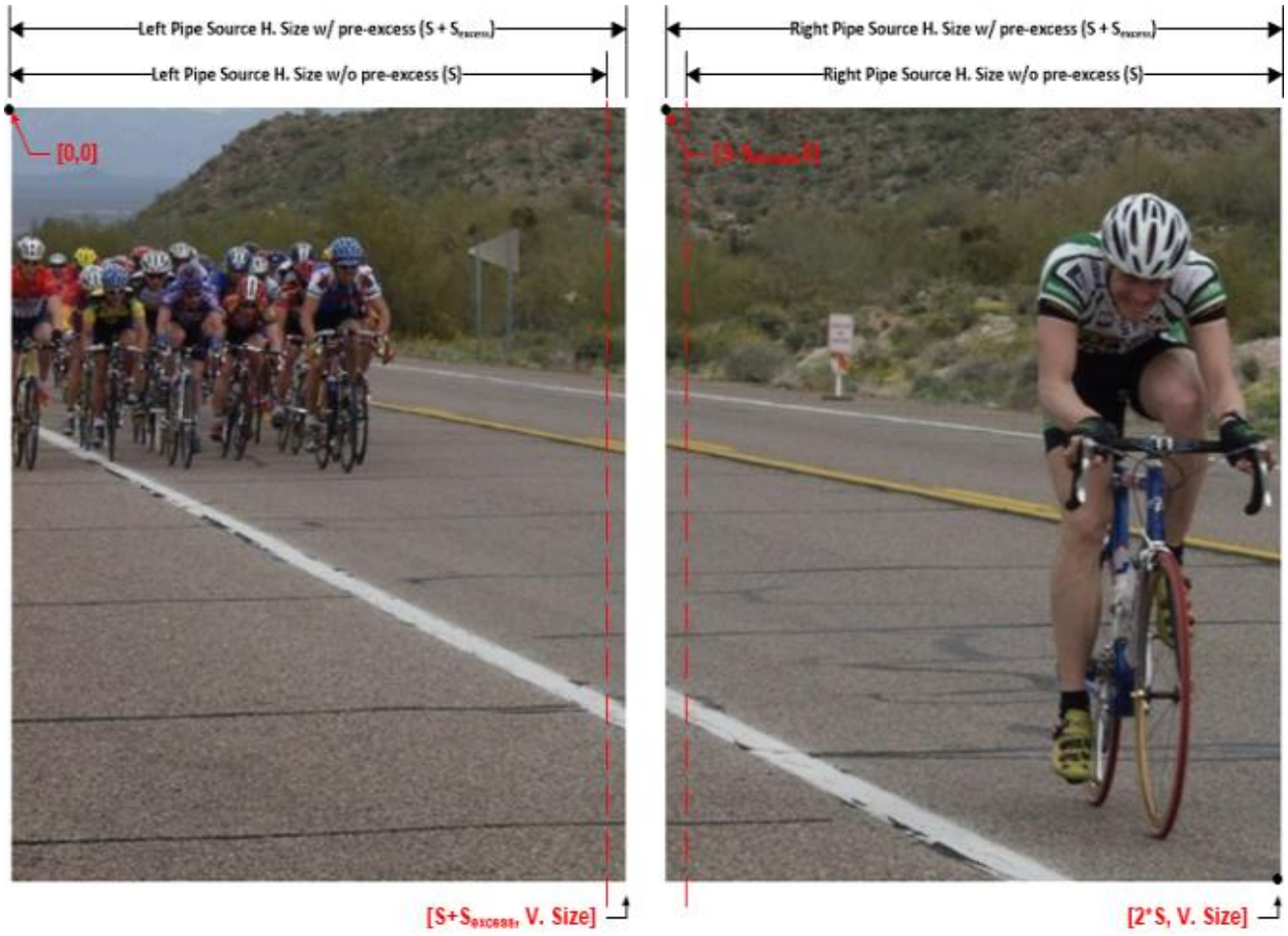
Seam Location

The location of the seam is relative to the left or right side of the Pipe Source Size window where the horizontal Pipe Source Size will include the pre-scaled excess (a.k.a. Source Excess, or S_{excess})

H. Pipe Source Size = Target H. Pipe Source Size (S) + Source Excess (S_{excess})

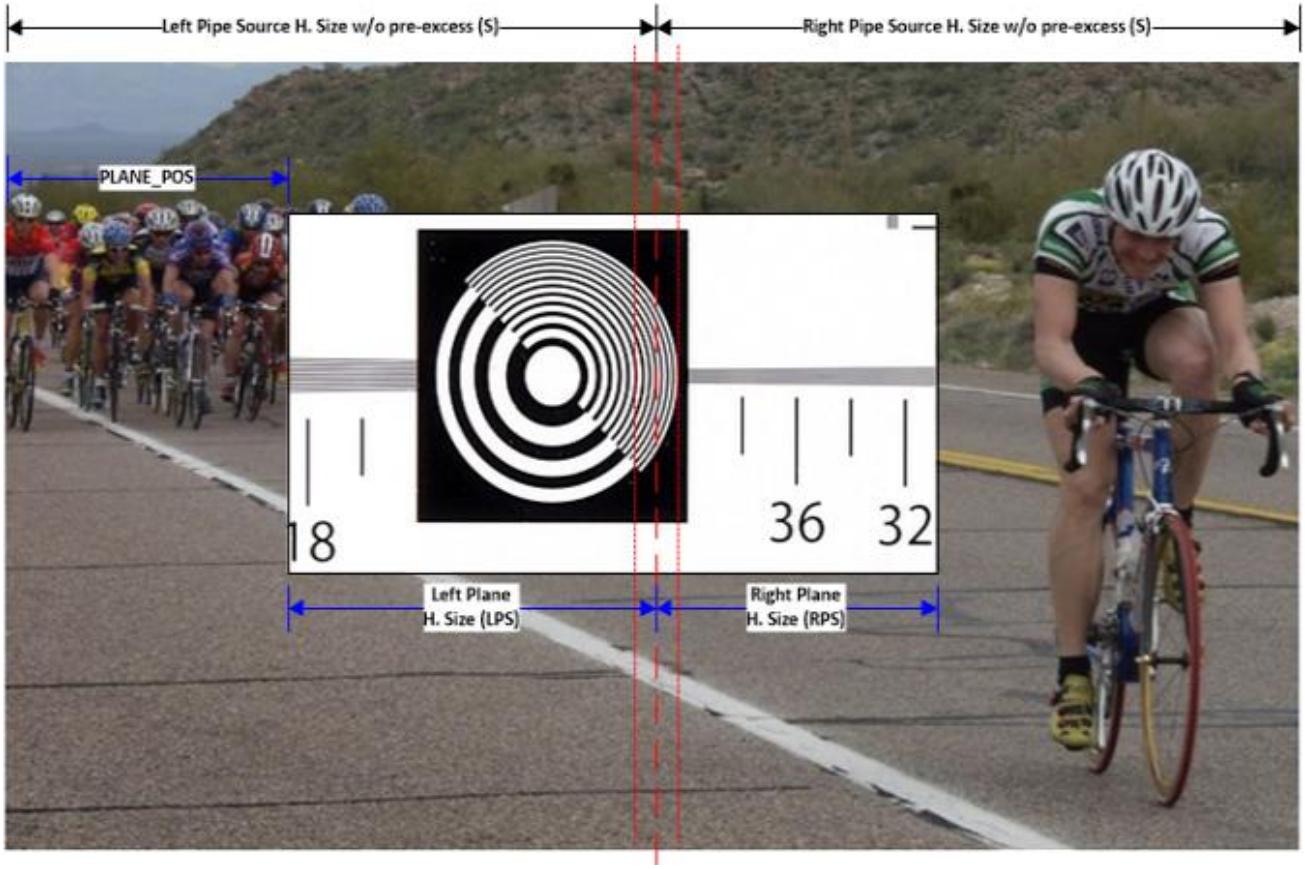
For the left-side Pipe, the seam will be located on the right-side of its source window, and for the right-side Pipe, the seam will be located on the left-side of its source window.

[0,0]
Seam
[2*S, V. Size]



Note that this discussion assumes symmetric horizontal Pipe Source Sizes without the source excess (i.e. S) across both Pipes. This is not a requirement, but it will be left to the reader to extrapolate the equations within this section for asymmetric S terms.

Locating the seam within an image that spans the horizontal Pipe Source Size across both Pipes (e.g. the desktop/background image) is simply a function of the Pipe. But, if a Plane image is smaller than the combined horizontal Pipe Source Sizes, then the location of the seam is dependent on the offset of the Plane within the Pipe Source Window.



So, if the addition of the Plane Position (PP) and the Full H. Plane Size (HPS) is greater than S, then the following equations define where the seam is located within the image in memory.

HPS = Left Plane Size (LPS) + Right Plane Size (RPS)

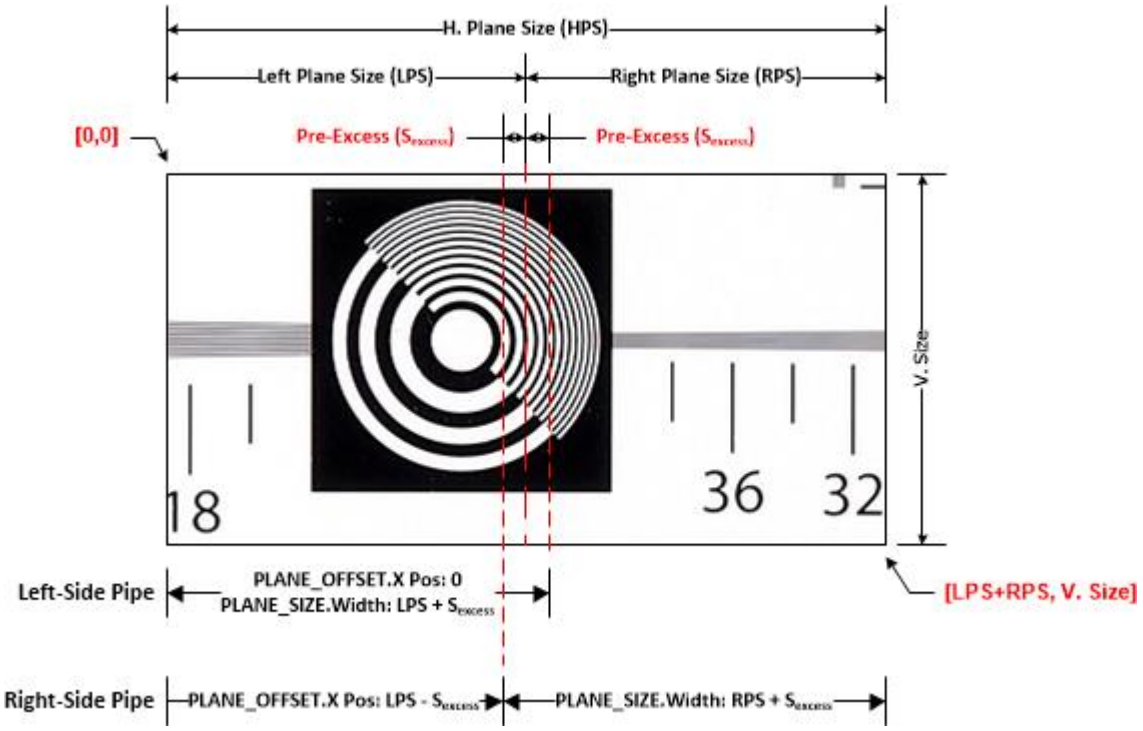
S = PP + LPS => **LPS = S - PP**

RPS = HPS - (S - PP)

Note that the Source Excess (S_{excess}) is a constant regardless of where the Plane image is located.

Splitting the Image

Now that the location of the seam is known, splitting the image across both Pipes can be performed. This is done using the PLANE_OFFSET and PLANE_SIZE registers.



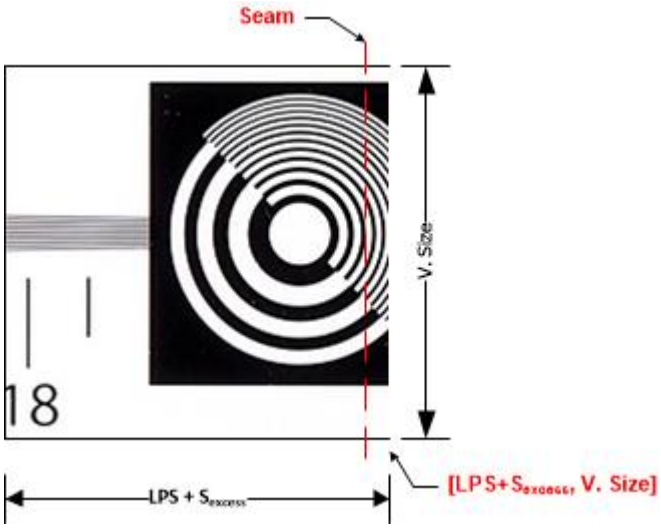
Using the figure above as a reference, to split the image in memory across the two Pipes.

Left PLANE_OFFSET:

- Starting X Position = H. Cropping Position (HCP) + 0
- Starting Y Position = V. Cropping Position (VCP) + 0

Left PLANE_SIZE:

- Width = Left Plane Size (LPS) + Pre-Excess Size (S_{excess})
- Height = V. Size

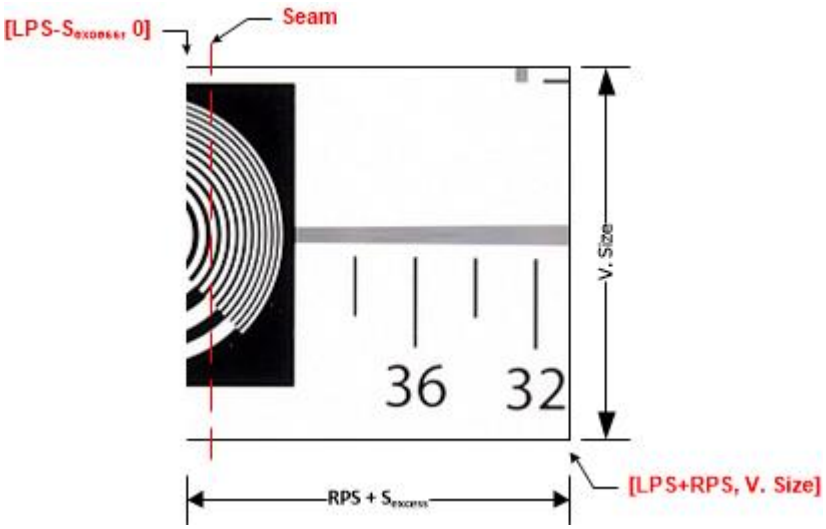


Right PLANE_OFFSET:

- Starting X Position = HCP + LPS - S_{excess}
- Starting Y Position = VCP + 0

Right PLANE_SIZE:

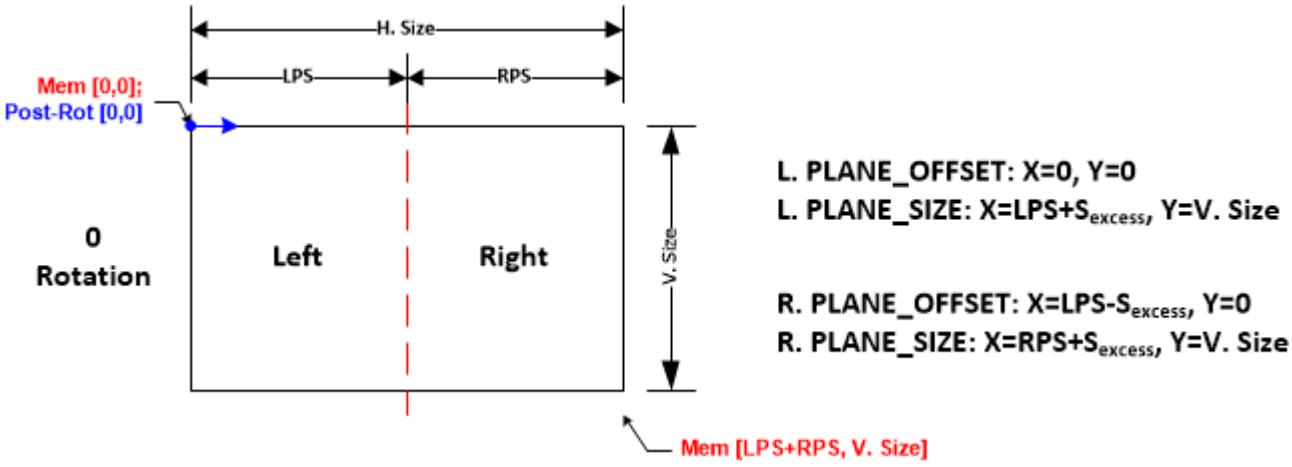
- Width = Right Plane Size (RPS) + S_{excess}
- Height = V. Size



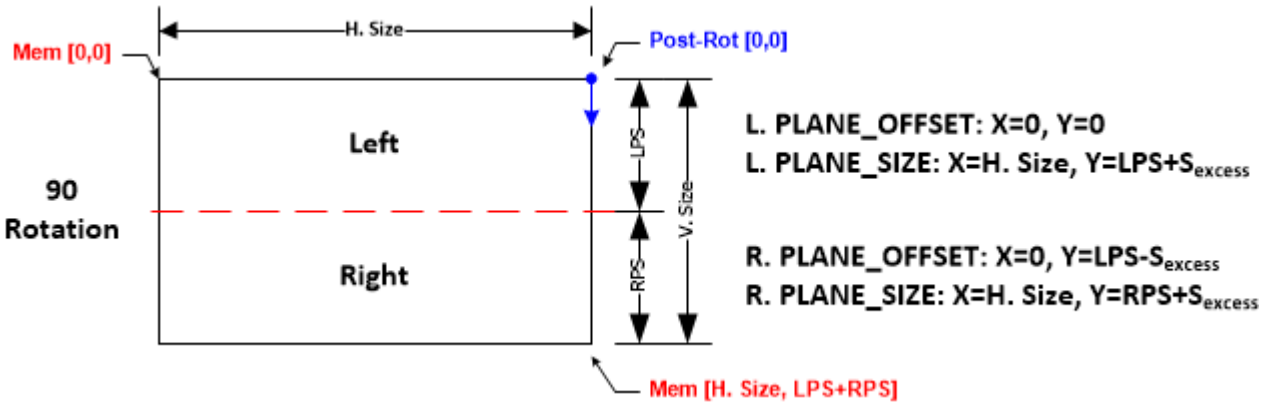
Rotation

The above discussion for splitting an image is when there is no rotation. If the image is to be rotated, then the programming of the PLANE_OFFSET and PLANE_SIZE needs to be adjusted accordingly. Since rotation is applied after the Plane Position and Size, the programming of these registers "rotates" to account for the image rotation.

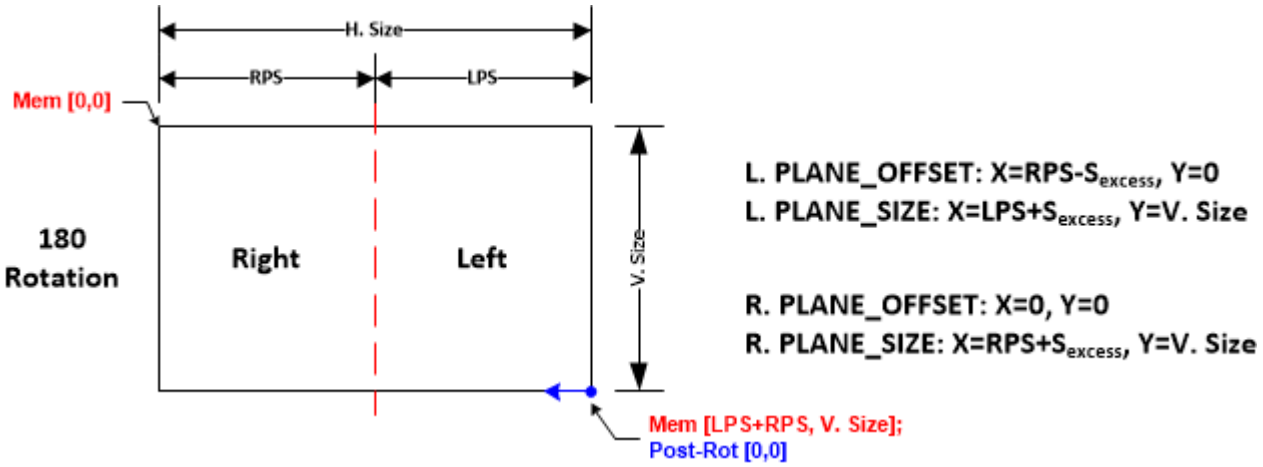
If we look at an image in memory that is going to be split without rotation, the regions of the image that are sent down each Pipe is shown below. Note that the blue dot and arrow represent the post rotation origin of the image and direction of scan.



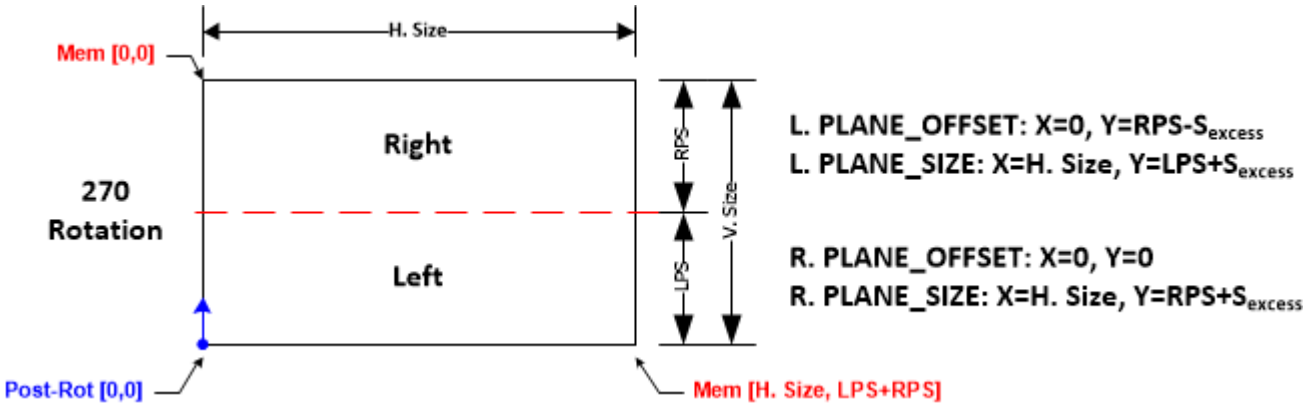
When the image is rotated by 90^o, then the horizontal and vertical sizes programmed within the Plane/Pipe are swapped. So, the left and right regions swap as well.



When rotating another 90^o (i.e. 180^o rotation), then the left and right regions swap positions with respect to the no rotation scenario.



Finally, when the image is rotated 270^o then the left and right regions swap positions with respect to the 90^o rotation scenario.

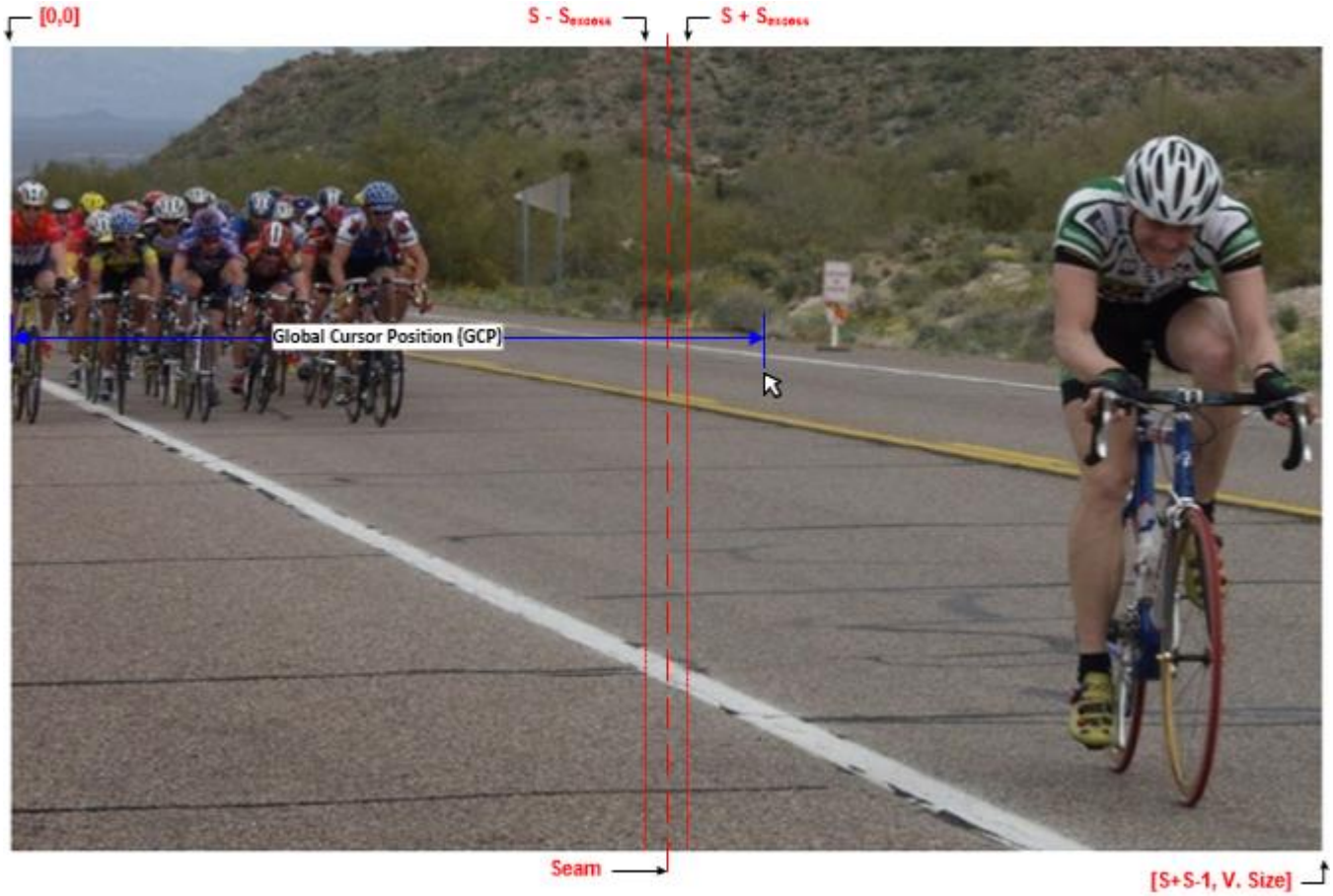


Note that Software should ensure that the horizontal Plane position for the image in the right Pipe is zero (i.e. there should be no positional offset of the image with respect to the seam)

Refer to the table within the Source Image Manipulation below for a summary of the Plane offset and size programming.

Cursor

The mechanism that Software uses to determine the horizontal position of the Cursor across both Pipes is beyond the scope of this document. However, to explain how the Cursor is enabled/programed around the seam, this document will use a global coordinate system that spans across both Pipe source windows.



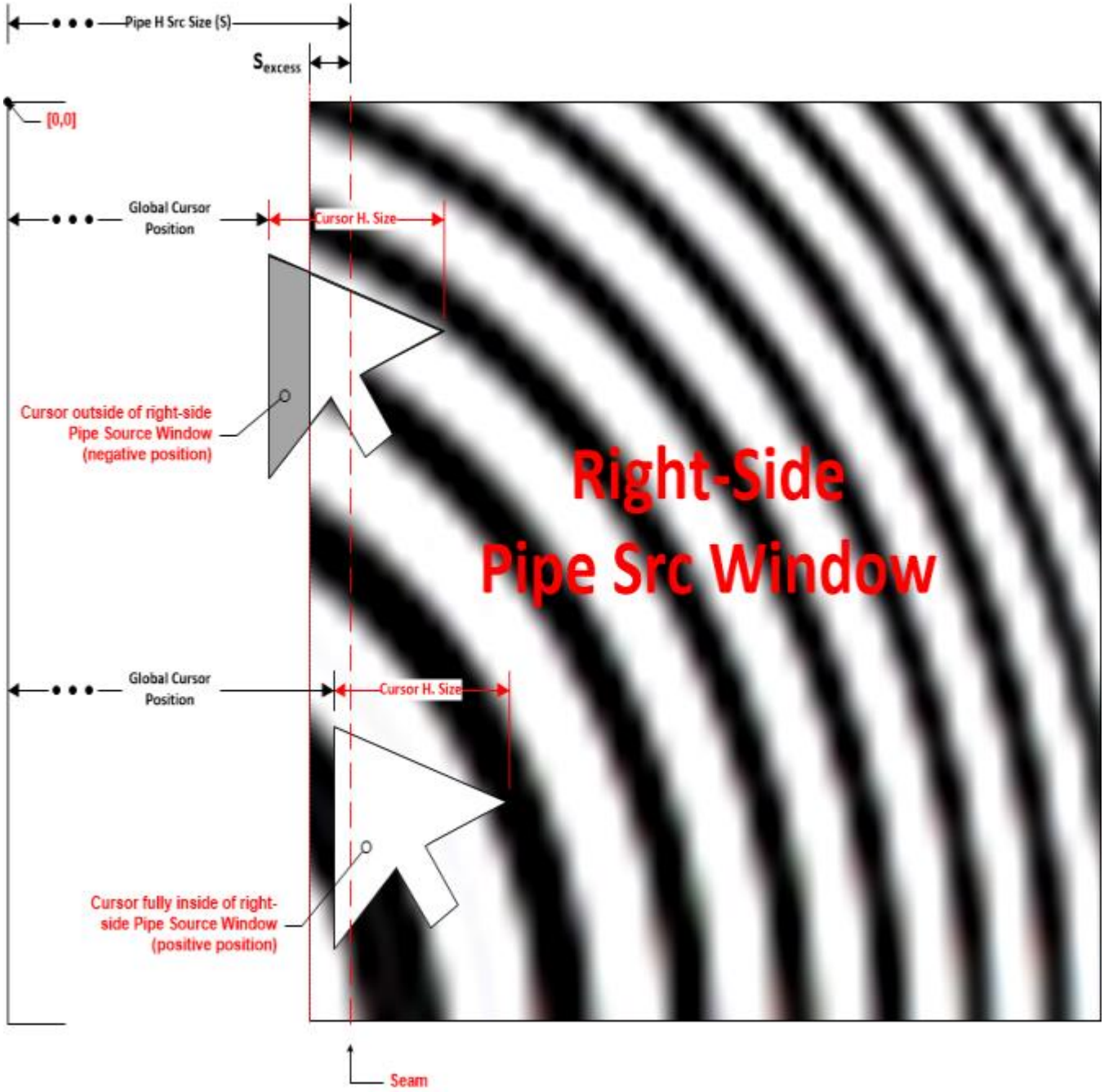
The horizontal position of the Cursor will be referred to as the global cursor position (GCP) that will span across both Pipe source windows from 0 to (2*S)-1.

When the Cursor is outside of the seam "window" (S +/- S_{excess}), then only one Pipe will have its Cursor enabled. As the Cursor moves into the seam window, then both Pipes will need to enable its Cursor. Determining when the Cursor is within this window is dependent on the global cursor position and the Cursor size (CS).

Global Cursor Position	Left Pipe Cursor	Right Pipe Cursor
GCP < (S - S_{excess} - CS)	Enabled	Disabled

(S - S_{excess} - CS) <= GCP < (S + S_{excess})	Enabled	Enabled
GCP >= (S + S_{excess})	Disabled	Enabled

The positioning of the Cursor will be dependent on where the Cursor is within the seam window. The left-side Cursor position is directly based off of the global cursor position, but the right-side Cursor is a little more involved.



The following table illustrates how to determine the Cursor position when it is crossing the seam.

Global Cursor Position	Left Pipe Cursor Position	Right Pipe Cursor Position
GCP < (S - S_{excess} - CS)	Magnitude = GCP Sign = SW determines	N/A (Disabled)
(S - S_{excess} - CS) <= GCP < (S - S_{excess})	Magnitude = GCP Sign = Positive	Magnitude = S - S_{excess} - GCP Sign = Negative
(S - S_{excess}) <= GCP < (S + S_{excess})		Magnitude = GCP - S - S_{excess} Sign = Positive
GCP >= (S + S_{excess})	N/A (Disabled)	

Calculating Pre and Post Excess Sizes

To get an accurate sampling across the seam, the number of pre-scaled excess pixels (a.k.a. Source Excess, S_{excess}) needs to (at a minimum) fill the number of FIR taps to the left or right of the horizontal FIR's Center Tap.

$$\mathbf{S_{excess} >= (NUMTAPS - 1)/2}$$

Note that the Display Pipe requires an even number of line pixels, so the source excess may need to be rounded up to the next even number.

To determine the number of post-scaled excess pixels (a.k.a. Destination Excess, D_{excess}) we can use the following equation:

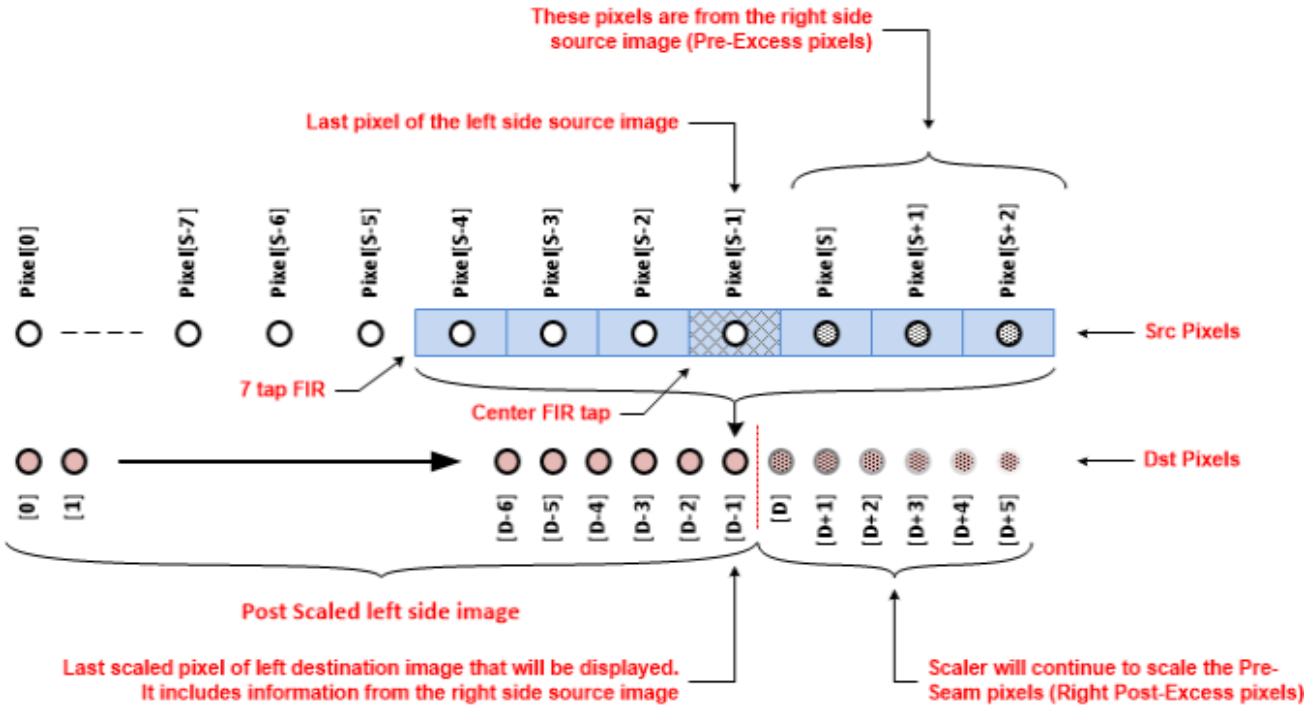
$$SF_{target} = (S + S_{excess}) / (D + D_{excess})$$

$$\mathbf{D_{excess} = Round(((S + S_{excess}) / SF_{target}) - D)}$$

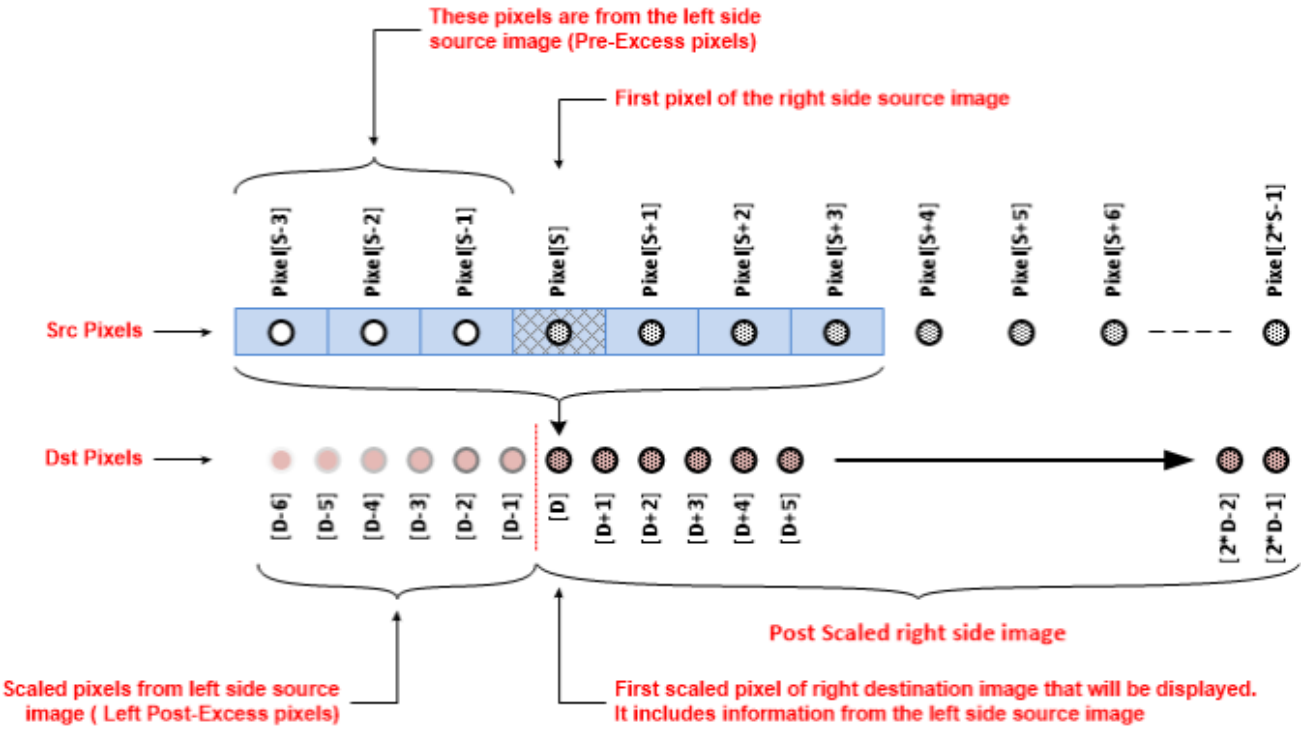
Where:

- S is the Source size in pixels per Pipe (one-based and without excess pixels)
- D is the Destination size in pixels per Pipe (one-based and without excess pixels)
- SF_{target} is the target Scale Factor (i.e. S/D)

Note that the D_{excess} must be a multiple of DISP_NUMPPC since the pixel dropping logic works at that granularity.



The above illustrates the Scaler output for the left side image (seam is on the right side). As the last pixel of the left side source image moves into the center tap of the Scaler's FIR, all of the taps to the right of the center tap contain pixels from the right side source image (Pixels S to S+2). The resulting destination pixel (this is the last displayable pixel) will contain all of the necessary information from the right side of the source image. All of the pixels beyond the last pixel of the destination image are the post excess pixels and those will be dropped.

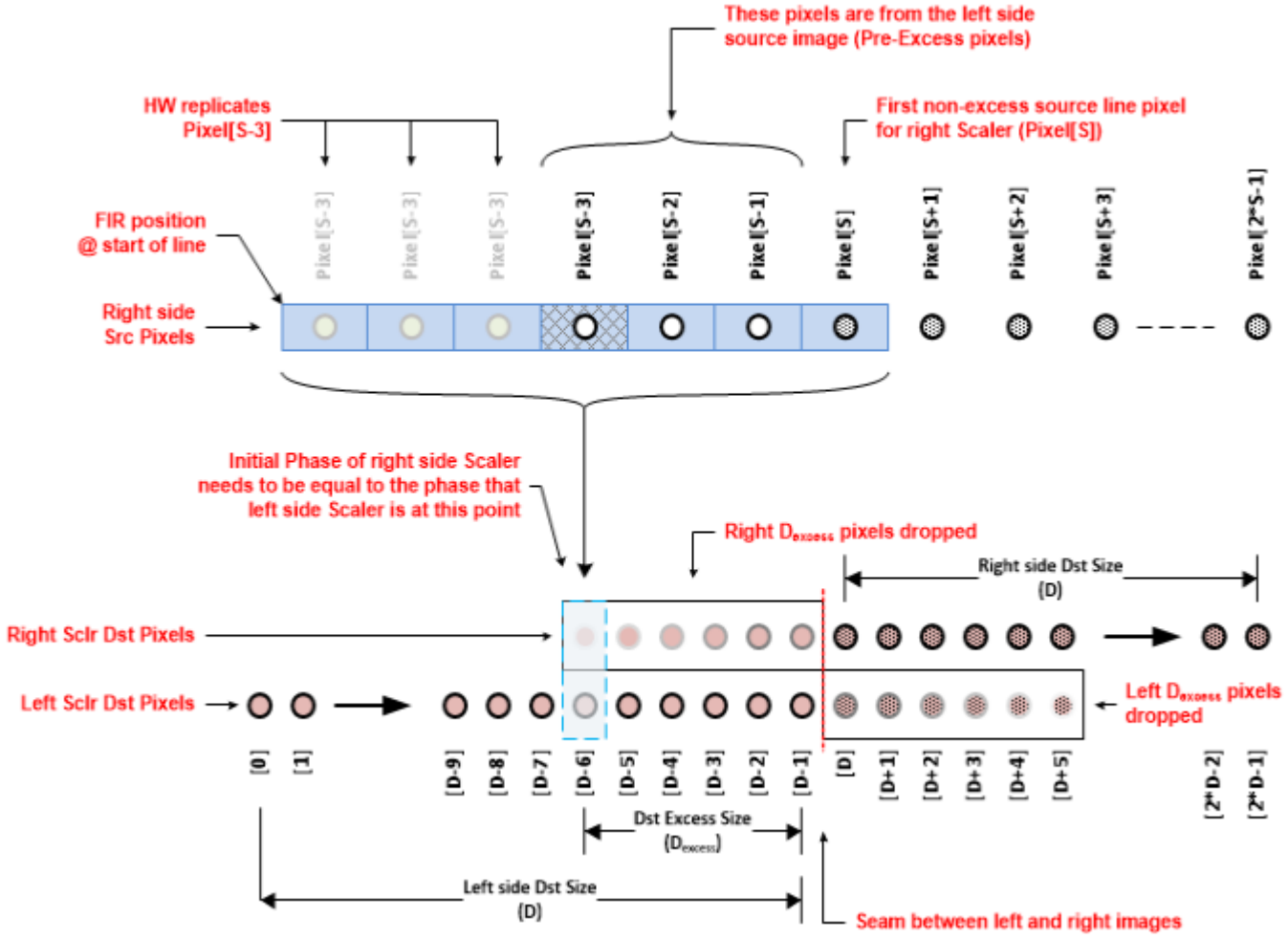


The above illustrates the Scaler output for the right side image (seam is on the left side). As the FIR progresses, the first pixel of the right side source image eventually moves into the center tap and all of the taps to the left of the center tap contain pixels from the left side source image (Pixels S-3 to S-1). All pixels up to this point are the post excess pixels that will be dropped. The resultant destination pixel when the first right side source image pixel makes it to the center pixel contains all of the necessary information from the left side of the source image.

Calculating Initial Phases

To ensure the image from both Pipes appears to be scaled by a single Scaler, the horizontal initial phase for the Scaler processing the right-side image will need to be set such that the phase of the horizontal FIR of the right Scaler picks up where the horizontal FIR of the left Scaler leaves off.

When the horizontal filter starts processing a line, it will load the first source pixel of the line into the Center Tap of the FIR and then apply an initial phase from that point (note that the Scaler can only apply positive initial phases). For the right-side Scaler, the first pixel of the source line will be the left most Source Excess pixel (i.e. a pixel from the left source image). To match the phase of the horizontal FIR of the right-side Scaler at the **beginning** of its source line with the phase of the horizontal FIR of the left-side Scaler at the **end** of its source line we need to determine the phase of the left-side Scaler's horizontal FIR (D_{excess}-1) pixels **before** it reaches the final Destination pixel (D-1).



The following equation is used to determine the phase that the right-side Scaler needs to start at.

$$\mathbf{P_{right} = SF_{actual} * (D - D_{excess})}$$

Notes:

1. To get an accurate initial phase, need to use the actual scale factor (SF_{actual}) that the Scaler on the left is using. Depending on configuration, this is not always the target scale factor
   a. If the Scaler HW is calculating the Scale Factor: SF_{actual} = (S + [Left, Right] S_{excess}) / (D + D_{excess})
   b. If the Scaler HW is using the programmed Scale Factor: SF_{actual} = PS_PROG_HSCALE
      i. The PS_PROG_HSCALE register should equal the target Scale Factor (SF_{target})
2. The Scale Factor has a precision of 2.14, regardless of how it is calculated. However, HW will automatically round the Scale Factor (Software should do the same)

The absolute phase difference between the left and the right will be used as the Horizontal Initial Phase (HIP) to be applied to either the left or right side.

$$\mathbf{Absolute\ Horizontal\ Initial\ Phase = |\ (S - S_{excess}) - P_{right}\ |}$$

The side that the initial phase is applied will depend on which side is ahead of the other.

- If the left-side Scaler phase is less than or equal to the source pixel that the right-side Scaler starts at (i.e. (S - S_{excess}) - P_{right} is positive) then the HIP will be applied to the left-side Scaler
- Otherwise, the HIP will be applied to the right-side Scaler

Notes:

1. Make sure to set the Initial Phase Trip bit of PS_HPHASE
2. The act of applying an initial phase will cause the image (on the left or the right) to appear to "shift". The shifting will be more evident for the left-side.

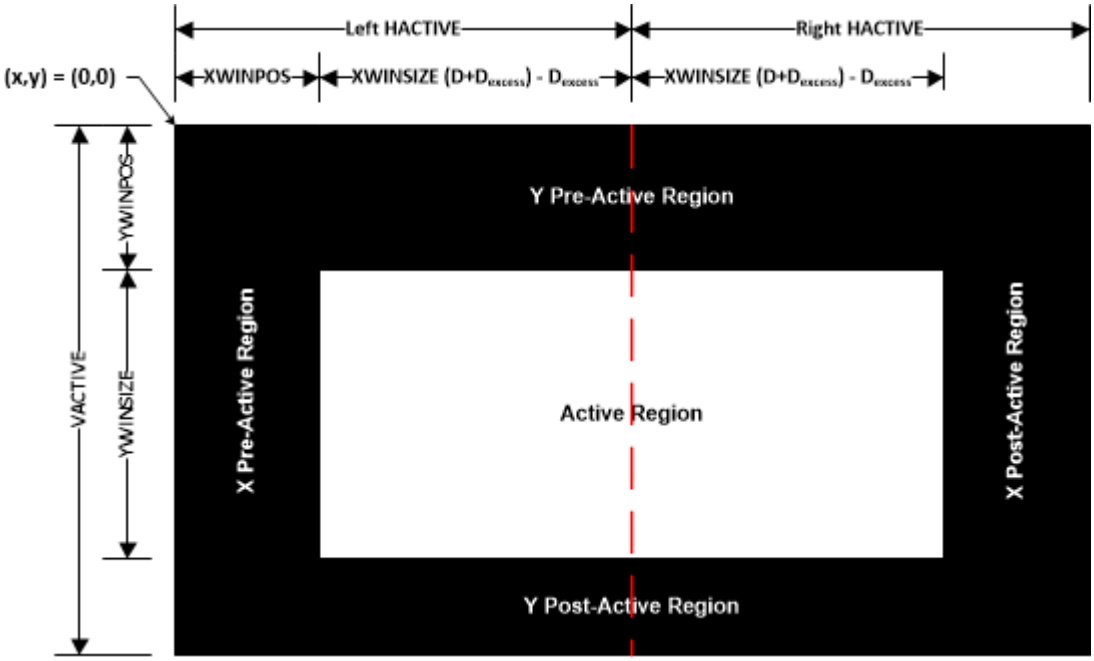
The following table illustrates an example of a 1920x1080 source image that is split across two Pipes with Pipe Source Sizes of 960 + S_{excess}.

		Notes
Source Size (S)	960	Horizontal Pipe Source Size w/o S_{excess}
Source Excess (S_{excess})	4	EVEN((NUMTAPS -1)/2)
Destination Size (D)	2576	Number of destination pixels per Pipe w/o D_{excess}
Target Scale Factor (SF_{target})	0.372670807	S/D
Actual Scale Factor (SF_{actual})	0.372497559	HW calculated by left Scaler: (S + S_{excess}) / (D + D_{excess}) Result is rounded to 2.14 precision
Destination Excess (D_{excess})	12	ROUND(((S + S_{excess}) / SF_{target}) - D)
Initial Phase needed on Right (P_{right})	955.0837402	SF_{actual} * (D - D_{excess})
First Src Pixel on Right	956	S - S_{excess}
Absolute Horizontal Initial Phase	0.9162598	| (S - S_{excess}) - P_{right} |

Left Scaler Initial Phase	0.9162598	(S - S_{excess}) >= P_{right}
Right Scaler Initial Phase	0.0	(S - S_{excess}) >= P_{right}

Pillar Box Windowing

If pillar-box borders are going to be applied by the Scaler, then Software will need to account for this across both Pipes when programming the PS_WIN_POS register.



As we can see from the above illustration, the left-side border is applied by the PS_WIN_POS programming of the left-side Scaler and the right-side border is applied by the H. Active programming of the right-side transcoder.

Left Side:

- PS_WIN_POS.X Position = Left side border size
- TRANS_HTOTAL.H Active = Destination image size w/o excess (D) + Left border size

Right Side:

- PS_WIN_POS.X Position = 0
- TRANS_HTOTAL.H Active = Destination image size w/o excess (D) + Right border size

Summary:

The following is a summary of the above discussion that Software will need to perform for tiled Pipe scaling.

Some general notes:

1. Adding excess pixels for tiled Pipe scaling is only necessary when upscaling the tiled images. If downscaling the tiled images, then excess pixels do not need to be added.
2. This document assumes that the pre-excess horizontal Pipe Source Size (i.e. S) is symmetric across both Pipes (i.e. S is the same for both the left and right-side Pipes). It is not a requirement that this term be symmetric, but it will be left to the reader to extrapolate the equations within this document for asymmetric S terms.

Source Image Manipulation

- **S is the horizontal Pipe Source Size without S_{excess} pixels (one-based)**
- **S_{excess} >= (NUMTAPS - 1)/2**
    - **NUMTAPS = 7**
    - **Round up the result to the nearest NUMPPC (NUMPPC=2)**
- **If the sum of the Plane Position (PP) and the Full H. Plane Size (HPS) is greater than S (i.e. (PP + HPS) > S):**
    - **HPS = Left Plane Size (LPS) + Right Plane Size (RPS)**
    - **LPS = S - PP**
    - **RPS = HPS - (S - PP)**

Note that the "Full H. Plane Size" is the size of the Plane image after rotation.

The following table summarizes how the Plane offset, position, and size programming changes with rotation. Some notes:

1. The programming reflected in this table is only applicable for Plane images that straddle the seam (i.e. (PO + HPS) > S)
2. The Plane offset programming in the table is not accounting for any image cropping that software may be applying to the image in memory
3. Software should ensure that the horizontal Plane position for the image in the right Pipe is zero (i.e. there should be no positional offset of the image with respect to the seam)
4. As the diagrams above illustrate, depending on rotation the LPS and RPS terms are either based off the horizontal or vertical size of the image in memory. The terms will be denoted with "h" or "v" subscripts to differentiate which size the term is based off.

Rotation	Side	Register	Field	Programming
0	Left	PLANE_OFFSET	Starting X Pos	0
			Starting Y Pos	0
		PLANE_SIZE	Width	$LPS_h + S_{excess}$
			Height	V. Size

	Right	PLANE_OFFSET	Starting X Pos	LPS_h - S_{excess}
			Starting Y Pos	0
		PLANE_SIZE	Width	RPS_h + S_{excess}
			Height	V. Size
90	Left	PLANE_OFFSET	Starting X Pos	0
			Starting Y Pos	0
		PLANE_SIZE	Width	H. Size
			Height	LPS_v + S_{excess}
	Right	PLANE_OFFSET	Starting X Pos	0
			Starting Y Pos	LPS_v - S_{excess}
		PLANE_SIZE	Width	H. Size
			Height	RPS_v + S_{excess}
180	Left	PLANE_OFFSET	Starting X Pos	RPS_h - S_{excess}
			Starting Y Pos	0
		PLANE_SIZE	Width	LPS_h + S_{excess}
			Height	V. Size
	Right	PLANE_OFFSET	Starting X Pos	0
			Starting Y Pos	0
		PLANE_SIZE	Width	RPS_h + S_{excess}
			Height	V. Size
270	Left	PLANE_OFFSET	Starting X Pos	0
			Starting Y Pos	RPS_v - S_{excess}
		PLANE_SIZE	Width	H. Size
			Height	LPS_v + S_{excess}

	Right	PLANE_OFFSET	Starting X Pos	0
			Starting Y Pos	0
		PLANE_SIZE	Width	H. Size
			Height	RPS_v + S_{excess}

Cursor

Global Cursor Position	Left Pipe Cursor	Right Pipe Cursor
GCP < (S - S_{excess} - CS)	Enabled	Disabled
(S - S_{excess} - CS) <= GCP < (S + S_{excess})	Enabled	Enabled
GCP >= (S + S_{excess})	Disabled	Enabled

Global Cursor Position	Left Pipe Cursor Position	Right Pipe Cursor Position
GCP < (S - S_{excess} - CS)	Magnitude = GCP Sign = SW determines	N/A (Disabled)
(S - S_{excess} - CS) <= GCP < (S - S_{excess})	Magnitude = GCP Sign = Positive	Magnitude = S - S_{excess} - GCP Sign = Negative
(S - S_{excess}) <= GCP < (S + S_{excess})		Magnitude = GCP - S - S_{excess} Sign = Positive
GCP >= (S + S_{excess})	N/A (Disabled)	

Post-Excess

- **D is the number of non-excess destination pixels per Pipe (one-based)**
- **SF_{target} = S/D**
- **SF_{actual}**
    - **Scaler calculates Scale Factor: (S + [Left, Right] S_{excess}) / (D + D_{excess})**
    - **Programmed Scale Factor: SF_{target} (PS_PROG_HSCALE)**
    - Hardware maintains the Scale Factor at a precision of **2.14. When the Scaler calculates the Scale Factor, it will round the result.**
- **D_{excess} = ROUND(((S + Left S_{excess}) / SF_{target}) - D)**
    - **Round up to the nearest NUMPPC**

Initial Phase

- **P_{right} = SF_{actual} * (D - D_{excess})**
- **Absolute H. Initial Phase = | (S - S_{excess}) - P_{right} |**
    - **Absolute HIP applied to left-side Scaler if (S - S_{excess}) >= P_{right}**
    - **Absolute HIP applied to right-side Scaler if (S - S_{excess}) < P_{right}**

Impacts of Plane Scaling

If a Plane is being scaled across a seam, then the following variables will be impacted:

- The S_{excess} in the PLANE_SIZE will be unchanged
- The S_{excess} in the Pipe Source size will be equal to the D_{excess} from the Plane Scaler
- The D_{excess} at the outlet of the Pipe will be calculated based off the S_{excess} in the Pipe Source
- The Pipe Scaler's actual scale factor will be based off the S_{excess} in the Pipe Source and the D_{excess} at the outlet of the Pipe

Affected Registers

The following table summarizes the registers directly affected when multi-Pipe Scaling is being employed

Register	Field	Programming
PLANE_POS	X/Y Position	See above "Source_Image_Manipulation" section
PLANE_OFFSET	Start X/Y Position	See table above in "Source_Image_Manipulation"
PLANE_SIZE	Width/Height	See table above in "Source_Image_Manipulation"
PIPE_SRCSIZE	H. Source Size	S + S_{excess}
PIPE_SEAM_EXCESS		**Left Side:** Right Excess Amount = D_{excess} **Right Side:** Left Excess Amount = D_{excess}
PS_WIN_SZ	X Size	D + D_{excess}
PS_HPHASE	RGB Initial H. Phase (Integer & Fraction)	Programming dependent on comparison of (S - S_{excess}) and P_{right} Note that the Initial H. Phase Trip bit should be set
PS_WIN_POS	X Position	**Left Side:** Left side border size, **if a border is present** **Right Side:** This should always be zero
TRANS_HTOTAL	H. Active	**Left Side:** D + Left side border size (if border present) **Right Side:** D + Right side border size (if border present)

Planes

Flips

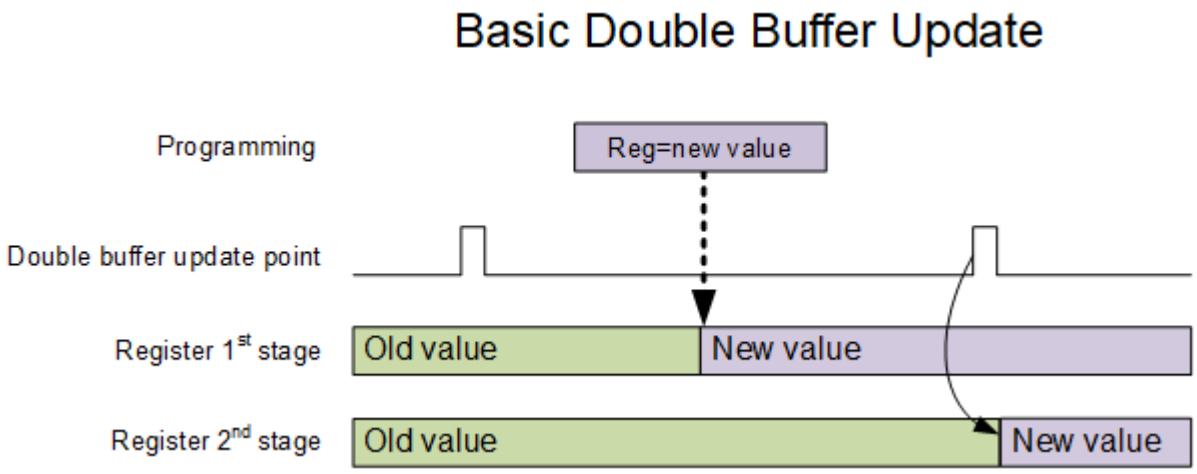
Overview

Display flips originally referred to switching between frame buffer surfaces to change the image. The term has evolved to refer more generally to an update to a plane configuration. Cursor configuration changes can also be considered to have flips, although it's more rare to describe a cursor update that way. Even updates to an enabled display pipe, such as new color correction settings, can be considered a flip, but usually only when combined with changes to the planes. A flip can be as small as updating one plane's surface address to a new frame buffer location, or as large as a synchronized update to planes, cursors, color correction, scaling, and transcoder metadata, across multiple running pipes.

Double Buffering

Most configuration that is allowed to be changed while a function is enabled, such as flips, is done through double buffered registers. Double buffer registers have two stages of registers to align the configuration update to a safe point, typically the start of vertical blank.

Writes and reads to double buffer registers access the first stage. The first stage is transferred to the second stage at the double buffer update point. The second stage is the "live value" that goes to the function being configured.



Some functions, such as planes, have read only registers to read the "live value" from the second stage.

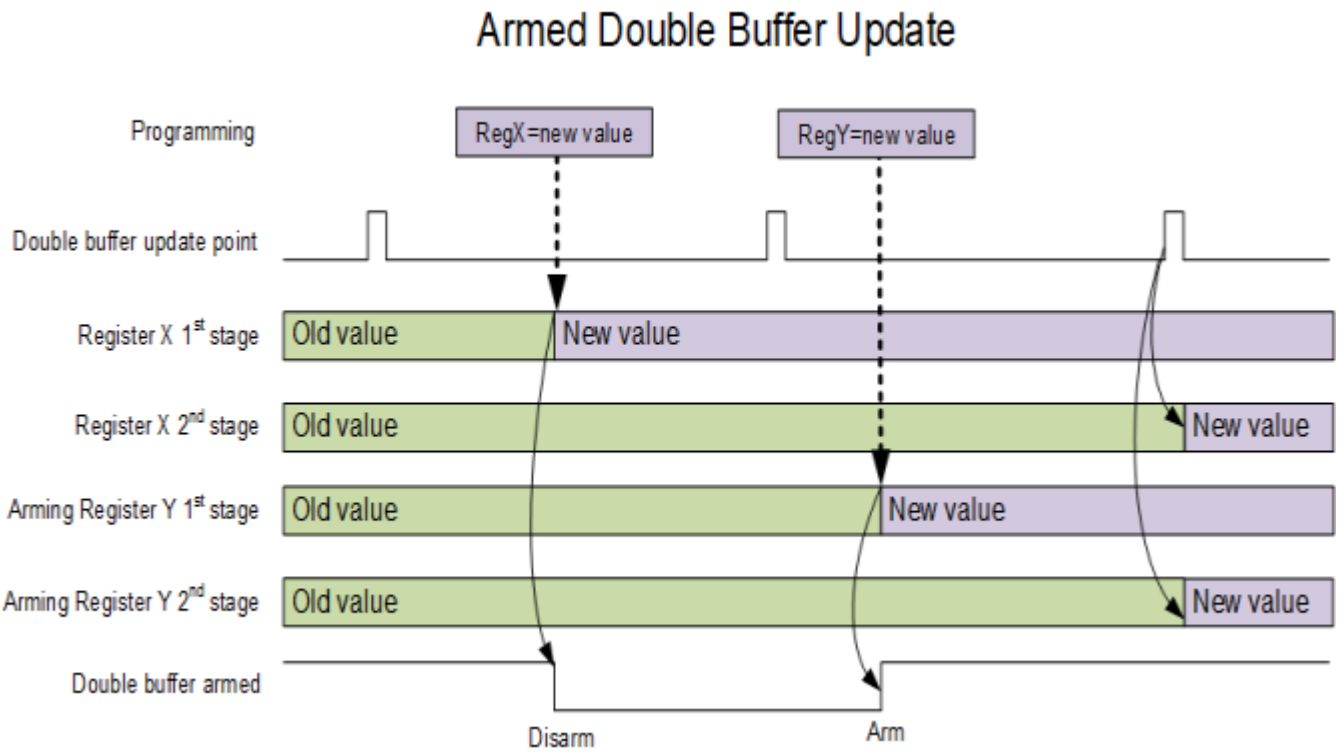
Arming and double buffer stalling are used for some registers to control the double buffer update point for atomic updates across multiple registers and functions.

Atomic Updates

Updates to multiple registers of a single function can be done atomically using the arming mechanism that some functions, such as planes, use. One of the registers for the function is defined as the arming register, described in the double buffer fields of the registers. Writes to the arming register 1st stage will

arm the double buffer to update at the next double buffer update point. Writes to the other double buffered registers for that function will disarm and prevent updates at the double buffer point. The programming sequence is to write the disarming registers first and the arming register last, then all the registers update at the next double buffer update point.

Updates across multiple planes and other resources require double buffer stalling as explained in the Double Buffer Control section.



Note: Once armed, by writing to the armed by register, the registers controlled by this arming should not be changed until the double buffer update point is reached. If changed, this will disarm the sequence and will require another write to the armed by register to get it to the armed status again.

Flip Types

There are two broad types of flip; sync and async.

Synchronous Flips

Synchronous flips (sync flips) update the plane surface base address, and other double-buffered registers, at the frame boundary (start of next vertical blank). These can be used for updates across multiple functions, not just planes.

Asynchronous Flips

Asynchronous flips (async flips) update the plane surface base address as soon as possible (next TLB boundary reached or start of next vertical blank). This can cause the image to change mid-frame with a tear, but gives better responsiveness than sync flips.

Cursors cannot do async flips.

Asynchronous flip can update only the plane surface (PLANE_SURF).

Asynchronous flip completion time depends greatly on how much data has been prefetched for power savings, and can take up to 1 full frame to complete. For faster flip completion, disable FBC and render/unified compression and allocate a minimum amount of data buffer for the plane.

Flip Programming

There are two paths to programming a flip; MMIO and command ring.

MMIO Flips

MMIO writes may be used to flip.

General MMIO Sequence for Plane or Cursor Flips

1. If interrupt will be used to wait for the flip to complete, unmask and enable the flip done interrupt for the plane that will be flipped. See the Display Interrupts section. For cursors, use the vblank interrupt.
2. Write registers to update plane configuration fields besides the arming register, including PLANE_CTL Async Address Update Enable to select sync or async flip.
3. Arm the update: For planes, write PLANE_SURF with the plane surface address. For cursors, write CUR_BASE. For other resources, write the arming register as specified in registers for that resource. Some resources do not have arming.
4. Optional: Wait for the interrupt for the flipped plane. Not required if software uses other methods to find when the old surface is no longer being used.

Double buffering control does not apply to PLANE_SURF updates that occur when the plane is disabled so an interrupt event is generated immediately when the PLANE_SURF is written.

Command Ring Flips

The render command MI_DISPLAY_FLIP may be used to flip. Typically it is only used for plane surface address updates, but it can be extended to more complicated flips.

MI_DISPLAY_FLIP can be combined with LOAD_REGISTER_IMMEDIATE, which behaves like a MMIO write, to update other parts of a plane configuration.

General Command Sequence for Plane Flips

1. If MI_WAIT_FOR_EVENT will be used to wait for the flip to complete, unmask flip done for the plane that will be flipped in DE_RRMR using MMIO or LOAD_REGISTER_IMMEDIATE. See the Render Response section.
2. Optional: LOAD_REGISTER_IMMEDIATE to write registers to update plane configuration fields that are not included in the MI_DISPLAY_FLIP_COMMAND.
3. MI_DISPLAY_FLIP to update plane surface address and other configuration listed in the command details (stride, tile format, and more), and select sync or async flip. The flip is self arming.
4. Optional: MI_WAIT_FOR_EVENT to wait for the flip to complete for the flipped plane. Not required if software uses other methods to find when the old surface is no longer being used.
5. Re-mask the flip done in DE_RRMR using MMIO or LOAD_REGISTER_IMMEDIATE. It does not need to be immediately re-masked, but unmasked events will wake GT as they occur, so for improved power savings it is recommended to only unmask events as they are required.

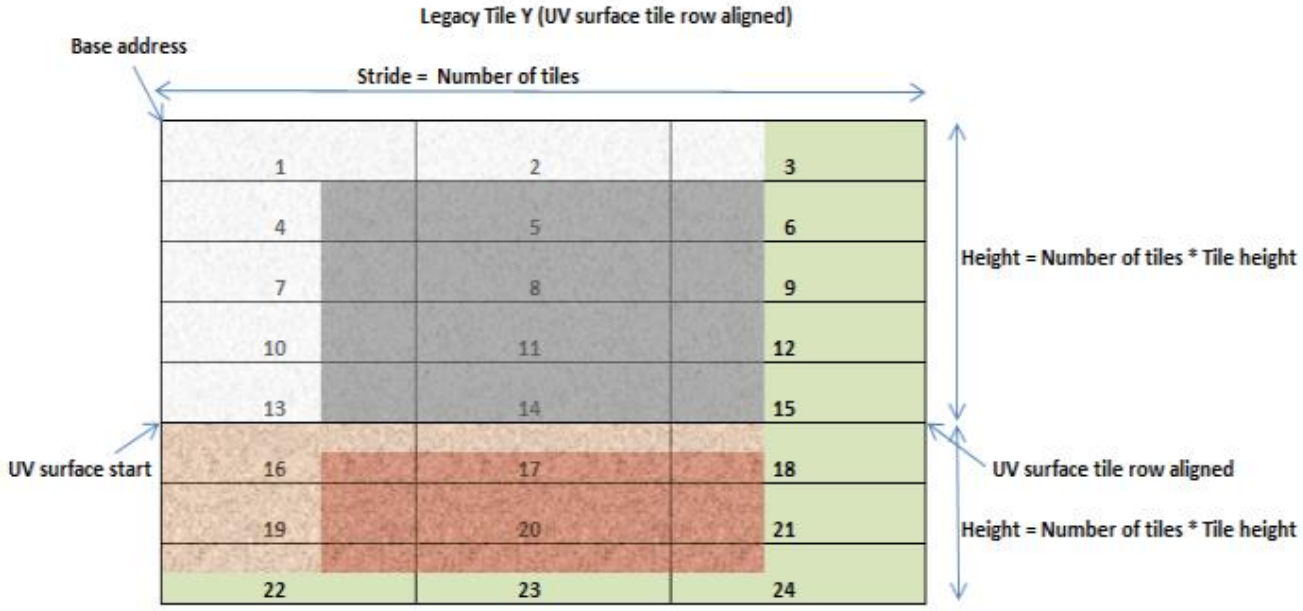
Flips to a plane that is disabled or on a disabled pipe will complete immediately and send the flip done.

Plane Planar YUV programming

YUV 420 hybrid planar formats (NV12, P0xx) have Y and UV surfaces stored separately. Display supports YUV 420 planar frame buffers in linear and tiled surface formats. Each display pipe can support up to two hybrid planar YUV 420 frame buffers. Display hardware uses two planes to handle each of these framebuffers - one for Y surface data and the other for UV surface. In addition to the two planes, a plane scaler must be enabled for YUV420 to YUV444 upsampling. See the Plane Capability and Interoperability section for UV and Y surface plane assignments.

Y and UV surface Base Address, Stride and Surface Offset (Start X and Y position) must be programmed separately for Y and UV planes. Software must make sure that the Y and UV plane programming gets applied together in the same frame. The Y and UV plane programming can be synchronized using the double buffer synchronization mechanism defined in DOUBLE_BUFFER_CTL. In the case of DisplayPort VRR mode (Variable Refresh Rate), the VRR push must be used for the Y and UV plane synchronization. Software must make sure that there is no push between the Y and UV plane flips.

With tiled surfaces, the UV surface must start on a new tile row.



Y Surface

Base Address, Stride and Surface Offset (Start X and Y position) should follow the standard plane programming.

UV Surface

The UV surface has some additional requirements.

Base Address:

For UV surfaces, the programmed Base Address should satisfy the following alignment requirements:

Linear:

The UV surface Base Address programmed in the PLANE_SURF register should be aligned to Stride in bytes of the Y surface * 64.

Tiled surfaces:

The start of the UV surface programmed in the PLANE_SURF register should be Tile Row Aligned.

Surface Offset:

The UV surface offset is programmed in the PLANE_OFFSET register. In the 0/180 rotate cases, the programmed X and Y start position should be half of the Y surface start X and Y positions.

For 0/180 rotate cases:

UV surface Start X Position = half of Y plane X start position

UV surface Start Y Position = half of Y plane Y start position

For 90/270 rotate cases:

UV surface Start X Position = (UV surface height in tiles * tile height) - UV plane Y start position (non-rotated) - UV plane height (non-rotated)

UV surface Start Y Position = UV plane Start X Position (non-rotated)

90/270 Rotation

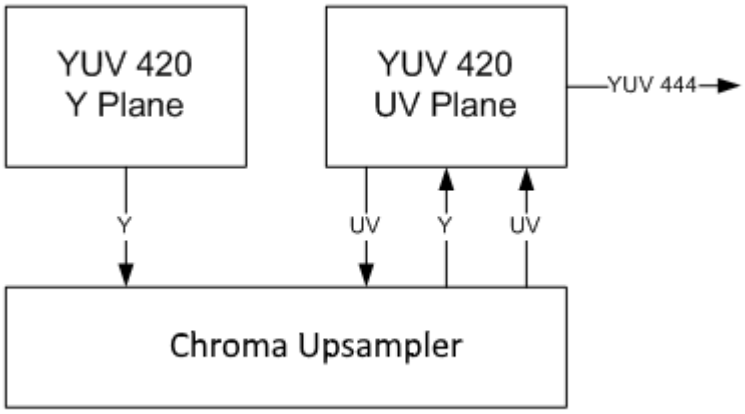
Plane 90/270 rotation support requires GTT remapping. GTT remapping allows the hardware to walk the pages sequentially. The Y and UV surfaces should be remapped separately, and the pages that contain portions of both Y and UV data will get GTT remapped twice, once for Y surface and the other for UV surface.

Upsampling and further processing

Planes 1-3 have dedicated chroma upsampler that is programmed in PLANE_CUS_CTL register.

The Y and UV plane pixels go through the chroma upsampler, gets upsampled, scaled and merged to form the YUV444 pixels for further processing.

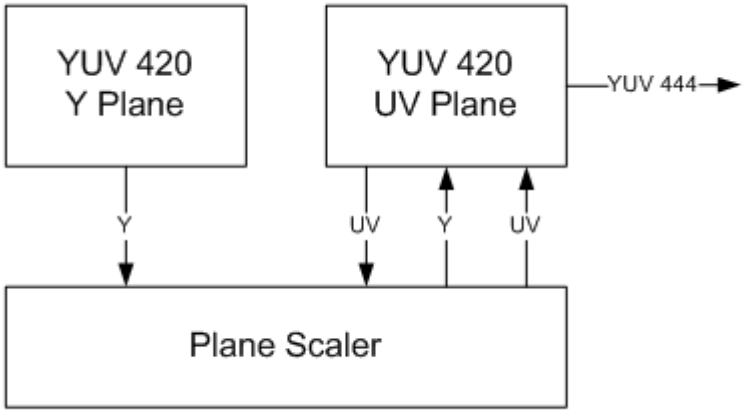
The Y plane binding is specified in the 'PLANE_CUS_CTL->Y Binding' field.



Planes 4-5 use the plane scaler for chroma upsampling.

The Y and UV plane pixels go through the plane scaler, gets upsampled, scaled and merged to form the YUV444 pixels for further processing.

The Y plane binding is specified in the 'PS_CTRL->Scaler Binding Y' field.



Destination keying, Color correction and Gamma must be programmed in the UV plane. Destination keying, Color correction and Gamma programming in the Y plane gets ignored. The UV plane position is used for blending order determination.

Flips

Since two different planes are involved in handling the pixels from a single planar YUV 420 frame buffer, we have to ensure that both the Y and UV planes' flip gets applied at the same time.

Global double buffer update disable mechanism can be used to synchronize and commit the flips atomically.

Follow the following sequence for CS, BCS or GUC flips

1. Use LRIs to set the 'Allow Double Buffer Update Disable' field for Y and UV planes.
2. Use LRI to set the 'Global Double Buffer Update Disable' field.
3. Use LRIs to update the 'Surface Base Address' and 'Stride' of the Y surface as required.
4. Send the flip messages for the UV surface.
5. Use LRI to clear the 'Global Double Buffer Update Disable' field.
6. Use LRIs to clear the 'Allow Double Buffer Update Disable' field for the Y and UV planes.

The flips will get applied in the following frame.

Follow the following sequence for MMIO flips

1. MMIOs to set the 'Allow Double Buffer Update Disable' field for Y and UV planes.
2. MMIO to set the 'Global Double Buffer Update Disable' field.
3. MMIOs to update the 'Surface Base Address' and 'Stride' of the Y surface as required.
4. MMIOs to update the 'Surface Base Address' and 'Stride' of the UV surface as required.
5. MMIO to clear the 'Global Double Buffer Update Disable' field.
6. MMIOs to clear the 'Allow Double Buffer Update Disable' field for the Y and UV planes.

The flips will get applied in the following frame.

Watermarks

Watermarks should be calculated and programmed for Y and UV planes separately. Refer to Watermarks page for further details.

Plane Capability and Interoperability

Plane Assignments and Capabilities

Plane capabilities:

- There are 7 planes + 1 cursor/pipe.
- Each planar YUV 420 surface uses 2 planes - Y and UV surfaces are handled by seperate planes which gets combined later in the pipeline. Each pipe can support up to 2 YUV 420 planar surfaces.
- HDR mode supports up to 3 planes per pipe.

	Pipe A	Pipe B	Pipe C	Pipe D
Cursor	Cursor	Same as pipe A	Same as pipe A	Same as pipe A
Plane 7	Render Decompression Planar YUV 420 Y Surface	Same as pipe A	Same as pipe A	Same as pipe A
Plane 6	Render Decompression Planar YUV 420 Y Surface	Same as pipe A	Same as pipe A	Same as pipe A
Plane 5	Render Decompression Media Decompression Planar YUV 420 UV Surface OLED Compensation	Same as pipe A	Same as pipe A	Same as pipe A
Plane 4	Render Decompression Media Decompression Planar YUV 420 UV Surface	Same as pipe A	Same as pipe A	Same as pipe A
Plane 3	Render Decompression Media Decompression Planar YUV 420 UV Surface HDR	Same as pipe A	Same as pipe A	Same as pipe A

Plane 2	Render Decompression Media Decompression Planar YUV 420 UV Surface HDR	Same as pipe A	Same as pipe A	Same as pipe A
	Render Decompression Media Decompression Planar YUV 420 UV Surface HDR Frame Buffer Compression	Render Decompression Media Decompression Planar YUV 420 UV Surface HDR	Render Decompression Media Decompression Planar YUV 420 UV Surface HDR	Render Decompression Media Decompression Planar YUV 420 UV Surface HDR

Display Plane Mapping to Command Streamer Plane Number

Display Plane Name	Command Streamer Plane Number
Plane 1 A	Plane 1
Plane 1 B	Plane 2
Plane 1 C	Plane 3
Plane 2 A	Plane 4
Plane 2 B	Plane 5
Plane 2 C	Plane 6
Plane 3 A	Plane 7
Plane 3 B	Plane 8
Plane 3 C	Plane 9
Plane 4 A	Plane 10
Plane 4 B	Plane 11
Plane 4 C	Plane 12
Plane 5 A	Plane 13
Plane 5 B	Plane 14
Plane 5 C	Plane 15
Plane 6 A	Plane 16
Plane 6 B	Plane 17
Plane 6 C	Plane 18
Plane 7 A	Plane 19
Plane 7 B	Plane 20
Plane 7 C	Plane 21

Display Plane Name	Command Streamer Plane Number
Plane 1 D	Plane 22
Plane 2 D	Plane 23
Plane 3 D	Plane 24
Plane 4 D	Plane 25
Plane 5 D	Plane 26
Plane 6 D	Plane 27
Plane 7 D	Plane 28
N/A	Plane 29
N/A	Plane 30
N/A	Plane 31
N/A	Plane 32

Display Plane Mapping to Command Streamer Plane Number - Legacy Mode

Display Plane Name	Command Streamer Plane Number
Plane 1 A	Plane 1
Plane 1 B	Plane 2
Plane 1 C	Plane 3
Plane 2 A	Plane 4
Plane 2 B	Plane 5
Plane 2 C	Plane 6
Plane 3 A	Plane 7
Plane 3 B	Plane 8
Plane 3 C	Plane 9
Plane 4 A	Plane 10
Plane 4 B	Plane 11
Plane 4 C	Plane 12

Plane Feature Interoperability

Display Features / Surface Formats:

	RGB32 2:10:10 :10	RGB 32 8:8:8 :8	RGB 32-bit XR_BIAS 10:10: 10	RGB64 16:16:16 :16 FP16	RGB64 16:16:16 :16 UINT	RGB 16-bit 5:6:5	Indexed 8-bit	YUV4 44 8 bpc	YUV4 22 8 bpc	YUV4 20 - NV12	YUV4 20 - P0xx	YUV4 22 - Y210, Y212, Y216	YUV4 44 - Y410	YUV4 44 - Y412, Y416
Linear - 0/180 rotation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Linear - 90/270 rotation														
X Tiling - 0/180 rotation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
X Tiling - 90/270 rotation														
Y Tiling (legacy) - 0/180 rotation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Y Tiling (legacy) - 90/270 rotation	✓	✓	✓			✓		✓	✓	✓	✓		✓	
Yf Tiling - 0/180 rotation														
Yf Tiling - 90/270 rotation														
Frame Buffer Compression		✓				✓ 90/270 rotation not supported								
Render Compression	✓	✓		✓										
Media Compression	✓	✓		✓				✓	✓	✓	✓	✓	✓	✓

Plane Scaling	✓	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
Per-pixel Alpha	✓	✓	✓	✓	✓									
Color Keying (supported only in non-HDR mode)	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Horizontal Flip	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Async Flip	✓	✓	✓	✓	✓	✓								

Tiling Modes / Display Features:

	Rotation 0/180	**Rotation 90/270**	**Horizontal Flip**	**Render Decompression**	**Media Compression**	**Frame Buffer Compression**	**Interlacing (Interlaced Fetch)**
Linear	✓					✓	✓
X Tiling	✓		✓			✓	✓
Y Tiling Legacy	✓	✓ (some surface formats not supported)	✓	✓	✓	✓	
Yf Tiling							

Rotation Modes / Display Features:

	Interlacing	**Frame Buffer Compression**	**Render Decompression**	**Media Compression**
0/180 rotation	✓	✓	✓	✓
90/270 rotation		✓ RGB 16 bit format not supported		
Horizontal Flip	✓	✓	✓	✓

Render Decompression

GT Render engines uses a lossless scheme to compress the color Render targets. The goal of the compression is to reduce the memory bandwidth. The memory foot print increases slightly due to the need of the control surface.

- Decompression is supported with RGB8888, RGB1010102 and FP16 formats.

- Decompression is supported only with legacy tile Y or tile 4 surfaces.
- Decompression support is limited to left-right cache line pair mode. Top-bottom mode is not supported.
- Decompression is not supported with 90/270 degree rotation.
- Compressed displayable surfaces must be 16KB aligned and have pitches padded to multiple of 4 tiles.

Color Control Surface

The Color Control Surface (CCS) contains the compression status of the cache-line pairs**.** CCS stride is programmed separately independent of the main surface stride. The compression state of the cache-line pair is specified by 4 bits in the CCS. The address of CCS surface is specified as an offset from the start of the Render Target main surface.

Decompression Programming

When compressed Render targets are presented to Display, the display decompression must be enabled. Along with main surface programming, the following additional programming is required to enable the decompression

- **Decompression Enable**

  Decompression is enabled by programming the 'Render Decomp' bit in the PLANE_CTL register.

- **Color Control Surface Distance**

  The start of the CCS surface is programmed as the distance from the start of the main surface in the PLANE_AUX_DIST register. The CCS is always placed after the main surface and is 4K page aligned.

- **Clear color value**

  When clear color is enabled in PLANE_CTL register, the clear color value must be programmed in the PLANE_CC_VAL register.

Media Decompression

The goal of the compression is to reduce the memory bandwidth. The memory foot print increases slightly due to the use of the control surface.

Display supports decompression of compressed media surfaces. 4 horizontally adjacent cache lines form a compression unit and its compression status is stored in the control surface (Tile Status Surface). The address of the control surface is specified as an offset from the start of the main surface. The control surface is always linear.

With Y tiling and planar YUV surfaces UV surface, the UV surface starts right after the Y surface and it can be in the middle on the tile row. The only guarantee is that the Y surface start will be 4 lines (cache line) aligned. Y and UV surfaces have their own individual control surface and they both have to be

programmed independently. In cases where the UV plane start is not aligned to the tile row start, the UV main surface start is programmed to the previous tile row aligned address with a plane offset to the start of the UV. In this case, the UV control surface will match the main surface start and hence will have the tile status for all compression units from the previously tile row aligned main surface address.

Restrictions
• Only Y tiling is supported. • 90/270 rotation not supported. • Async flips not supported.

Decompression Programming

When compressed media targets are presented to display, the media decompression must be enabled. Along with main surface programming, the following additional programming is required for both Y and UV planes individually.

- **Decompression Enable**

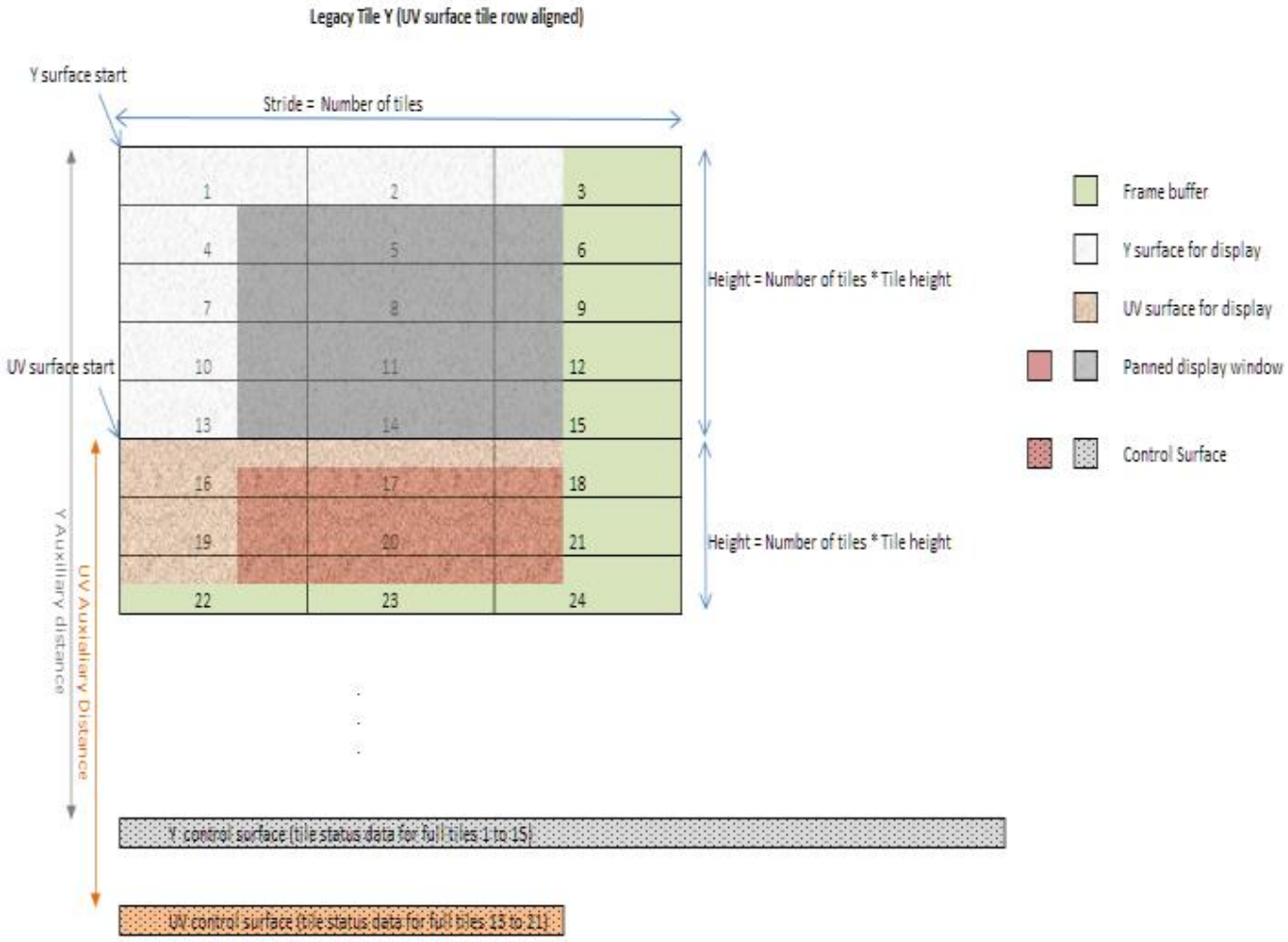
  Decompression is enabled by programming the 'Media Decomp' bit in the PLANE_CTL register.

- **Control Surface Distance**

  The start of the control surface is programmed as the distance in bytes from the start of the main surface in the PLANE_AUX_DIST register. The control surface must always be placed after (not necessarily immediately) the main surface and must be 4K page aligned.

  Control surface stride is not used for media compression.

  Y and UV control surfaces must both be 4k aligned.



Plane Rotation Programming

This topic provides programming information for plane rotation.

The 180 rotation mode is unchanged and will continue to use the same programming. In the 180 rotation mode display hardware is responsible for walking the pages in the reverse order. The cacheline walk within the page is also reversed. The 90 and 270 rotation modes require more complicated page walk mechanism. The page walk is made transparent to the hardware by providing a different set of page translations (remapped GTT) for the same rendered surface. The remapping completely abstracts the page walk away from the hardware and the hardware walks the pages as if there is no rotation. Hardware is still responsible for handling the the walk within the page appropriately. Also, the 90, 270 Rotation requires a new parameter - surface height. The changes needed in the driver programming is discussed below.

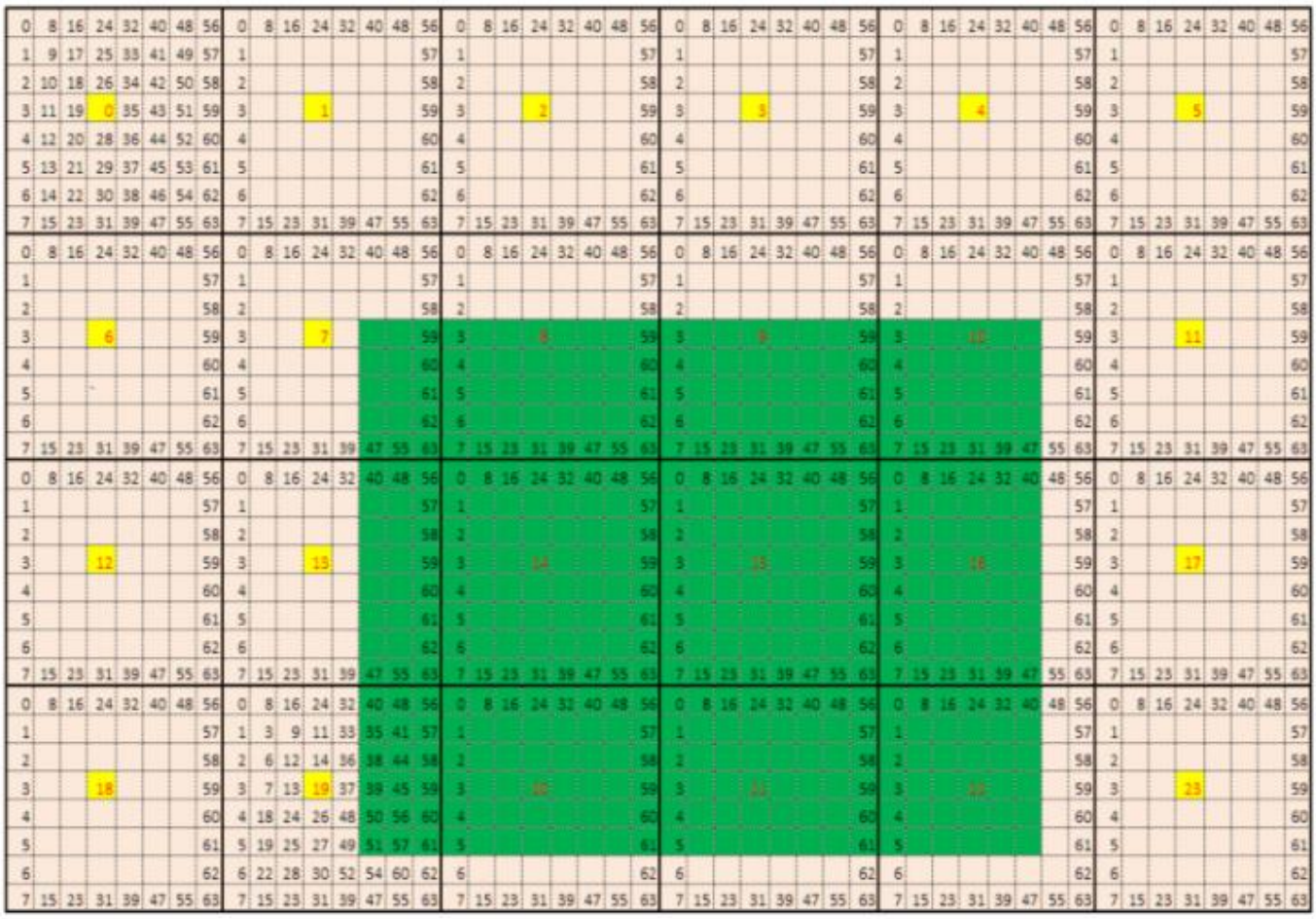
90 Rotation

For the 90 rotation programming, the plane parameters must use the following mapping

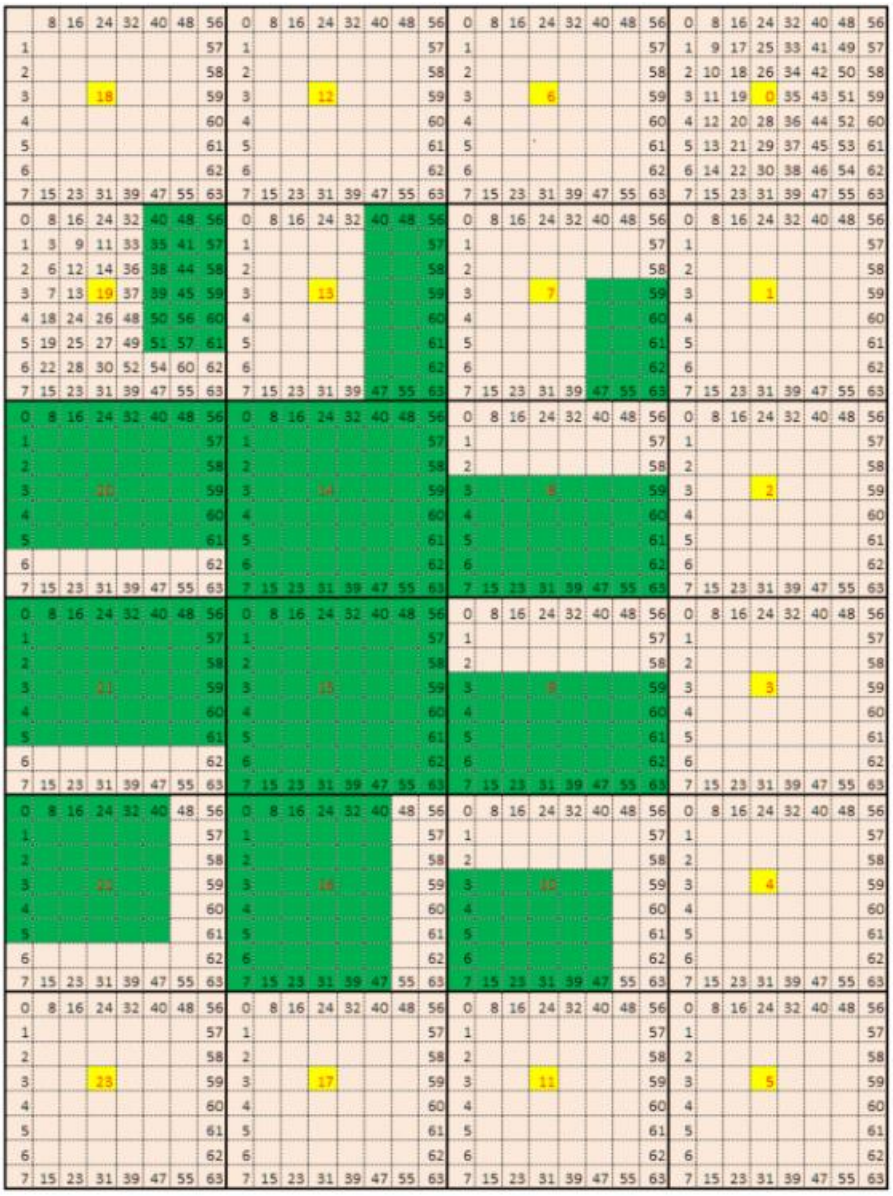
- Base address = New address (remapped GTT)

- Stride = Surface height in tiles
- Plane Width = Plane Height
- Plane Height = Plane Width
- X offset = (Surface height in tiles * tile height) - Plane Y offset - Plane Height *[Note: the calculated X offset will always be >= 0 since Plane Y Offset + Plane Height <= Surface Height in lines]*
- Y offset = X offset
- Driver can enable the Scaler, as needed, to fit the rotated content in to the pipe active display area.

Sample frame buffer surface



Page access sequence in the rotated 90 mode



Example

Let us assume the following display programming for a single pipe - single plane, Y tiled, non-rotated, 1920 x 1200, 4Bpp with the plane panned (100, 150), covering full active area and scaler not enabled.

GTT mapping

Here is a sample GTT mapping for 90 rotation mode.

Original GTT

Assumed Surface base = 0x200000

0x200000 = page 0

0x201000 = page 1

0x202000 = page 2

0x203000 = page 3

...

Remapped Display GTT - 90 rotation

Assumed new Surface base = 0x400000

0x400000 = page 18

0x401000 = page 12

0x402000 = page 6

0x403000 = page 0

Register programming for non-rotated scenario

- *PLANE_SURF->Surface Base Address = 0x200000*
- *PLANE_STRIDE->Stride = 60* [(1920 * 4)/128] [(width *bpp)/tile width]
- *PLANE_SIZE->Width = 1920*
- *PLANE_SIZE->Height = 1200*
- *PLANE_OFFSET->Start X Position = 100*
- *PLANE_OFFSET->Start Y Position = 150*
- *Surface Height in tiles (assumed) = 50 (allocated surface height in number of scan lines/tile height. For plane height = 1200, the surface height should be a minimum of 38 tiles (ceiling (1200/32). When panning is used, the rendered frame buffer surface will be larger than the plane size. Here, let us assume that the rendered surface height in tiles = 50).*

The programming changes to following for a 90 rotation scenario

- *PLANE_SURF->Surface Base Address = 0x400000* [uses remapped GTT]
- *PLANE_STRIDE->Stride = 50* [Surface height in tiles (assumed earlier)]
- *PLANE_SIZE->Width = 1200* [non-rotated Height]
- *PLANE_SIZE->Height = 1920* [non-rotated Width]
- *PLANE_OFFSET->Start X Position = 250* [(50*32)-150-1200] [(Surface height * tile height) - non rotated Y position - non rotated Height]
- *PLANE_OFFSET->Start Y Position = 100* [non-rotated X position]

The scaler should be programmed appropriately to fit the rotated plane in the pipe active area and the window position should be adjusted if it is desired to maintain the same apparent position on a physically rotated display.

270 rotation

Uses the same GTT remapping and register programming as 90 rotation mode with the Plane control register rotation mode set as 270.

NV12 (YUV 420) rotation

For NV12 90/270 rotation, the Y and UV surfaces should be treated as separate surfaces and thus the GTT remapping for rotation should be done separately.

Display Buffer Programming

Display Buffer Allocation

Allocation of the display buffer is programmable for each display plane, using the buffer start and buffer end values in **PLANE_BUF_CFG**.

Proper display buffer allocation is important for Display hardware to function correctly. Optimal allocation provides better display residency in memory low power modes. Display Buffer allocation must be recalculated and programmed when pipes/planes get enabled or disabled.

Display Buffer Size

Buffer Count	Total Buffer Size	Total Buffer Blocks	Blocks Available for Programming
2	1 MB (512 KB per buffer)	2048 (1024 per buffer)	2048

Each display buffer block is 8 cache lines.

Allocation Requirements

Allocation must not overlap between any enabled planes.

A minimum allocation is required for any enabled plane. See Minimum Allocation Requirements below.

A gap between allocation for enabled planes is allowed.

The allocation for enabled planes should be as large as possible to allow for higher watermarks and better residency in memory power saving modes.

Minimum Allocation Requirements

Allocation for each enabled plane must meet these minimum requirements.

Planes using Linear or X tiled memory formats must allocate a minimum of 8 blocks.

Planes using Y tiled memory formats must allocate blocks for a minimum number of scanlines worth of data. The formula and table of minimum scanlines is below.

Y tiled minimum allocation = Ceiling [(4 * Plane source width * Plane Bpp)/512] * MinScanLines/4 + 3

Plane Bpp	Minimum Scanlines for Y Tile	
	0/180 Rotation	90/270 Rotation
1	8	32
2	8	16
4	8	8
8	8	4

YUV 420 Planar Surface Format	Bpp
NV12	1 Bpp for Y and 2 Bpp for UV
P0xx	2 Bpp for Y and 4 Bpp for UV

Multi-Buffer Enabling and Allocation Requirements

There are two display buffers DBUF_S1 and DBUF_S2.

Display Buffer	Buffer Start	Buffer End
DBUF_S1	0	1023
DBUF_S2	1024	2047

Enable DBUF_S1 with the display initialization sequence so it will be always available for VGA and backwards compatibility. Enable DBUF_S2 when any planes are allocated to it as per the following table.

The table ensures that pipes are using the closest DBUF when there are multiple pipes enabled.

Pipe and DBUF ordering: PipeD - DBUF_S2 - PipeC - PipeA - DBUF_S1 - PipeB

When a pipe is allowed to allocate from 2 DBUFs, a plane on that pipe may use allocation that straddles the 2 DBUFs.

Pipe D Planes DBUF Allocation	Pipe C Planes DBUF Allocation	Pipe A Planes DBUF Allocation	Pipe B Planes DBUF Allocation	Pipes with Enabled Planes
N/A	N/A	N/A	N/A	None or VGA
N/A	N/A	S1+S2	N/A	A
N/A	N/A	N/A	S1+S2	B
N/A	N/A	S2	S1	A+B
N/A	S1+S2	N/A	N/A	C
N/A	S2	S1	N/A	A+C
N/A	S2	N/A	S1	B+C
N/A	S2	S1	S1	A+B+C
S1+S2	N/A	N/A	N/A	D
S2	N/A	S1	N/A	A+D
S2	N/A	N/A	S1	B+D
S2	N/A	S1	S1	A+B+D
S2	S1	N/A	N/A	C+D

Pipe D Planes DBUF Allocation	Pipe C Planes DBUF Allocation	Pipe A Planes DBUF Allocation	Pipe B Planes DBUF Allocation	Pipes with Enabled Planes
S2	S2	S1	N/A	A+C+D
S2	S2	N/A	S1	B+C+D
S2	S2	S1	S1	A+B+C+D

Basic Allocation Method

These are basic methods that can be used for single and multi-pipe modes. For optimal power usage, the display driver can choose to use more advanced allocation techniques as desired.

Example Method 1:

Single Pipe

Enable display buffer(s). Refer to DBUF_CTL register for display buffer enabling.

If only DBUF_S1 is enabled

TotalBlocksAvailable = 1024

else if both DBUF_S1, DBUF_S2 enabled

TotalBlocksAvailable = 2048

Allocate a fixed number of blocks to cursors and then allocate the remaining blocks among planes, based on each plane's data rate.

BlocksAvailable = TotalBlocksAvailable

1. Allocate 32 blocks for cursor

   The driver frequently enables and disables the cursor or changes the cursor pixel format. Fixed allocation is preferred for cursor to minimize the buffer re-allocation. More allocation might be required to support deeper low power states (based on the results of watermark calculations).

   CursorBufAlloc = 32

   BlocksAvailable = BlocksAvailable – 32

2. Check for minimum buffer requirement

   For each enabled plane

   If Y tiled

   MinScanLines = Look up minimum scanlines needed from the table

   *PlaneMinAlloc = Ceiling [(4*Plane width*Bpp)/512] * MinScanLines/4 + 3*

   Else

   PlaneMinAlloc = 8

If sum of PlaneMinAlloc > BlocksAvailable

Enable DBUF_S2

BlocksAvailable = 2048 - CursorBufAlloc

If sum of PlaneMinAlloc > BlocksAvailable

Error - Display Mode can't be supported.

The driver can change the number of enabled planes or the plane configuration and rerun the algorithm.

3. Calculate Relative Data Rate for planes

   In this step the driver may want to use the expected maximum plane source sizes so it does not have to reallocate for a plane that is changing size.

   For each enabled plane

   If PlaneScalerEnabled

   *PlaneScaleFactor = (Plane width/Scaler window X size) * (Plane height/Scaler window Y size)*

   Else

   PlaneScaleFactor = 1

   *PlaneRelativeDataRate = Plane height * Plane width * plane source bytes per pixel * PlaneScaleFactor*

4. Allocate blocks for enabled planes as per the Data rate ratio.

   For each plane that needs allocation (PlaneBlockAllocFinal == false)

   *PlaneBufAlloc = floor (BlocksAvailable * PlaneRelativeDataRate/Sum of PlaneRelativeDataRate of all planes that need allocation).*

   *floor - rounds down to an integer value dropping the fractional part.

5. Adjust for minimum allocation requirement

   AdjustmentRequired = false

   For each plane needs allocation (PlaneBlockAllocFinal == false)

   If PlaneBufAlloc < PlaneMinAlloc

   AdjustmentRequired = true

   PlaneBufAlloc = PlaneMinAlloc

   PlaneBlockAllocFinal = true

   BlocksAvailable = BlocksAvailable - PlaneMinAlloc

   If AdjustmentRequired = true

   Go back to step 4

Multi-Pipe Option 1:

Enable both display buffers DBUF_S1 and DBUF_S2.

TotalBlocksAvailable = 2048

NumPipes = Total number of display pipes in the hardware

Allocate a fixed number of blocks to cursors, allocate 1/*NumPipes* of the remaining blocks to each pipe, then calculate each pipe individually as in the Single Pipe case.

1. Allocate 8 blocks for cursor per pipe

   The driver frequently enables and disables the cursor or changes the cursor pixel format. Fixed allocation is preferred for cursor to minimize the buffer re-allocation. More allocation might be required to support deeper low power states (based on the results of watermark calculations).

   For each enabled cursor

   CursorBufAlloc = 8

   *BlocksAvailable = TotalBlocksAvailable - (8 * NumPipes)*

2. Distribute the blocks equally among the pipes

   BlocksAvailable = BlocksAvailable/NumPipes

3. Assign blocks to the planes

   For each pipe

   Perform Single Pipe sequence, starting from step 2.

Multi-Pipe Option 2:

Allocate a fixed number of blocks to cursors, use the Single Pipe case for all the other planes across all pipes.

Example Method 2:

This allocation is based on the Watermark calculations and helps to distribute the buffer more optimally to achieve consistent latency levels supported across all planes.

- For each enabled plane, calculate buffer allocation needed for all Latency levels 1 to 7.
- Calculate the total buffer allocation needed for each latency level by adding the individual allocation of all enabled planes.
- Choose the max latency level that can be supported with the available display buffer. For each enabled plane, program/enable all watermarks up to that latency level.
- Allocate the buffer to the planes as required by the latency level chosen.

Use method 1 to allocate any remaining buffer.

Buffer Allocation Re-distribution

When an additional pipe is getting enabled, or an existing pipe requires more buffer to support a new mode or is disabled, buffer reallocation may be necessary for proper display functionality. Whenever a portion of the allocated buffer is taken away from one pipe and allocated to a different pipe, the following sequence should be followed to make sure that there are no buffer allocation overlaps at any point of time.

1. *For each pipe whose allocation is reduced*
    a. *Program the new buffer allocation.*
    b. *Wait for VBlank of that pipe for new allocation to update.*
2. *For each pipe whose allocation is increased*
    a. *Program the new buffer allocation.*
    b. *Wait for VBlank of that pipe for new allocation to update.*

Display Buffer Allocation and Watermark Programming Prior to OS Boot

Basic programming of the display buffer and watermarks to allow limited display usage prior to OS boot:

This will not allow package power saving states.

Supported usages:

- Up to 4 pipes enabled at once
- Up to one universal plane enabled per pipe. No cursor.
- Linear or Xtile memory
- Any RGB frame buffer pixel format 32bpp or less, without compression
- Any supported screen resolution
- Downscaling less than or equal to 12.5%

Enable DBUFs following the Multi-Buffer Enabling and Allocation Requirements.

Allocate 160 blocks per pipe

- Pipe A: 0-159, Pipe B: 160-319, Pipe C: 320-479, Pipe D: 480-639
- PLANE_BUF_CFG_<plane number>_A = 0x009F0000
- PLANE_BUF_CFG_<plane number>_B = 0x013F00A0
- PLANE_BUF_CFG_<plane number>_C = 0x01DF0140
- PLANE_BUF_CFG_<plane number>_D = 0x027F01E0

Set level 0 watermarks for any enabled plane to 160 blocks and 2 lines.

- PLANE_WM_<plane number>_<pipe>_0 = 0x800080A0

The higher level watermarks for any enabled plane must have bit 31=0 to keep the low power watermarks disabled.

VGA

The VGA Control register is located here. The VGA I/O registers are located in the VGA Registers document.

Register
VGA_CONTROL

Cursor Plane

Planes
CUR_CTL
CUR_BASE
CUR_POS
CUR_PAL
CUR_FBC_CTL
CUR_SURFLIVE
PLANE_BUF_CFG
PLANE_WM
DPRC_INSTANCES
CUR_COLOR_CTL
CUR_CSC_COEFF
CUR_PRE_CSC_GAMC_INDEX
CUR_PRE_CSC_GAMC_DATA

The CUR_CTL and CUR_FBC_CTL active registers will be updated on the vertical blank or when pipe is disabled, after the CUR_BASE register is written, or when cursor is not yet enabled, providing an atomic update of those registers together with the CUR_BASE register.

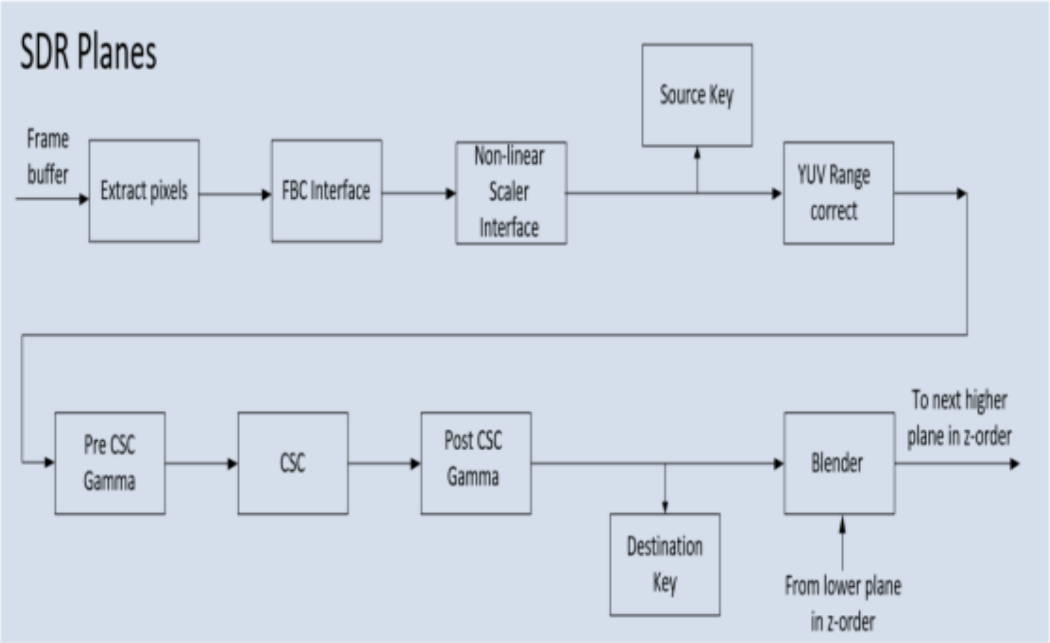
Universal Plane

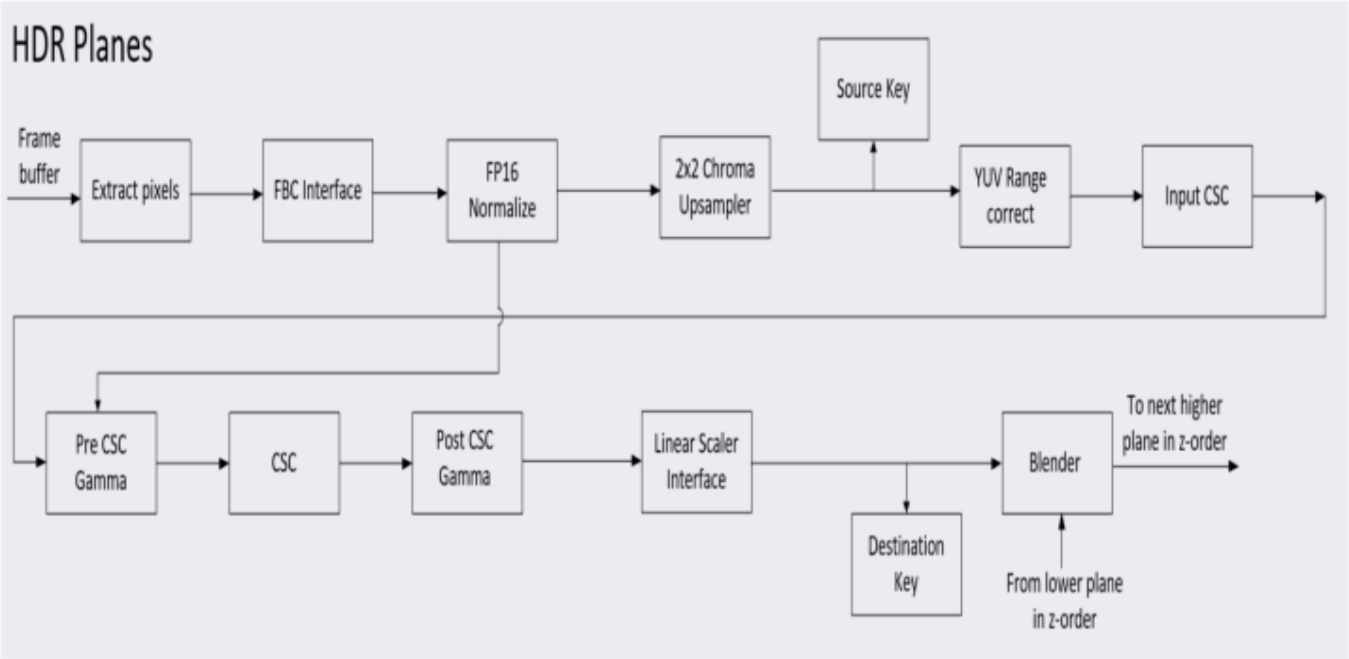
Planes
PLANE_CTL
PLANE_STRIDE
PLANE_POS
PLANE_SIZE
PLANE_SURF
PLANE_LEFT_SURF
PLANE_AUX_DIST
PLANE_SURFLIVE
PLANE_WM
PLANE_BUF_CFG
PLANE_OFFSET
PLANE_KEYVAL

Planes
PLANE_KEYMSK
PLANE_KEYMAX
PLANE_PRE_CSC_GAMC_INDEX
PLANE_PRE_CSC_GAMC_DATA
PLANE_POST_CSC_GAMC_INDEX
PLANE_POST_CSC_GAMC_DATA
PLANE_COLOR_CTL
PLANE_PIXEL_NORMALIZE
PLANE_INPUT_CSC_COEFF
PLANE_INPUT_CSC_PREOFF
PLANE_INPUT_CSC_POSTOFF
PLANE_CSC_COEFF
PLANE_CSC_PREOFF
PLANE_CSC_POSTOFF
PLANE_PRE_CSC_GAMC_DATA_ENH
PLANE_PRE_CSC_GAMC_INDEX_ENH
PLANE_POST_CSC_GAMC_DATA_ENH
PLANE_POST_CSC_GAMC_INDEX_ENH
PLANE_CUS_CTL
PLANE_CC_VAL
PLANE_POST_CSC_GAMC_SEG0_DATA_ENH
PLANE_POST_CSC_GAMC_SEG0_INDEX_ENH
SEL_FETCH_PLANE_CTL

Many of the plane control active registers will be updated on the vertical blank or when pipe is disabled, after the surface base address register is written, or when the plane is not yet enabled, providing an atomic update of those registers together with the surface base address register.

Data Flow Through the Plane





FBC not on all planes or pipes.

HDR planes have higher bit precision throughout the processing stages.

PIPE_MISC HDR Mode must be enabled to get the higher precision output from the HDR planes, bypassing the SDR planes in blending.

Planes 1-3 are the HDR planes. The other planes are SDR planes.

Plane Pixel Formats

ARGB

Name	Alpha	Red	Green	Blue
RGB 32-bit 8:8:8:8 BGRA	31:24	23:16	15:8	7:0
RGB 32-bit 8:8:8:8 RGBA	31:24	7:0	15:8	23:16
RGB 32-bit 2:10:10:10 BGRA	31:30	29:20	19:10	9:0
RGB 32-bit 2:10:10:10 RGBA	31:30	9:0	19:10	29:20
RGB 64-bit 16:16:16:16 Float BGRA (FP16) Each component is 1:5:10 MSb-sign:exponent:fraction	63:48	47:32	31:16	15:0
RGB 64-bit 16:16:16:16 Float RGBA (FP16) Each component is 1:5:10 MSb-sign:exponent:fraction	63:48	15:0	31:16	47:32
RGB 64-bit 16:16:16:16 UINT BGRA Each component is 16 bit unsigned integer	63:56	47:32	31:16	15:0
RGB 64-bit 16:16:16:16 UINT RGBA Each component is 16 bit unsigned integer	63:56	15:0	31:16	47:32
RGB 32-bit XR_BIAS 2:10:10:10	31:30	9:0	19:10	29:20
16-bit BGR 5:6:5	N/A	15:11	10:5	4:0
64-bit formats supported only on the HDR planes.				

YUV 420 Planar

Name	Y	U	V
YUV 4:2:0 8 bpc - NV12	7:0	15:8	7:0
YUV 4:2:0 10 bpc - P010	15:6	31:22	15:6
YUV 4:2:0 12 bpc - P012	15:4	31:20	15:4
YUV 4:2:0 16 bpc - P016	15:0	31:16	15:0

YUV 422 Packed

Name	Y1	U	Y2	V
YUV 4:2:2 YUYV 8 bpc	7:0	15:8	23:16	31:24
YUV 4:2:2 UYVY 8 bpc	15:8	7:0	31:24	23:16
YUV 4:2:2 YVYU 8 bpc	7:0	31:24	23:16	15:8
YUV 4:2:2 VYUY 8 bpc	15:8	23:16	31:24	7:0
YUV 4:2:2 YUYV 10 bpc - Y210	15:6	31:22	47:38	63:54
YUV 4:2:2 YUYV 12 bpc - Y212	15:4	31:20	47:36	63:52
YUV 4:2:2 YUYV 16 bpc - Y216	15:0	31:16	47:32	63:48

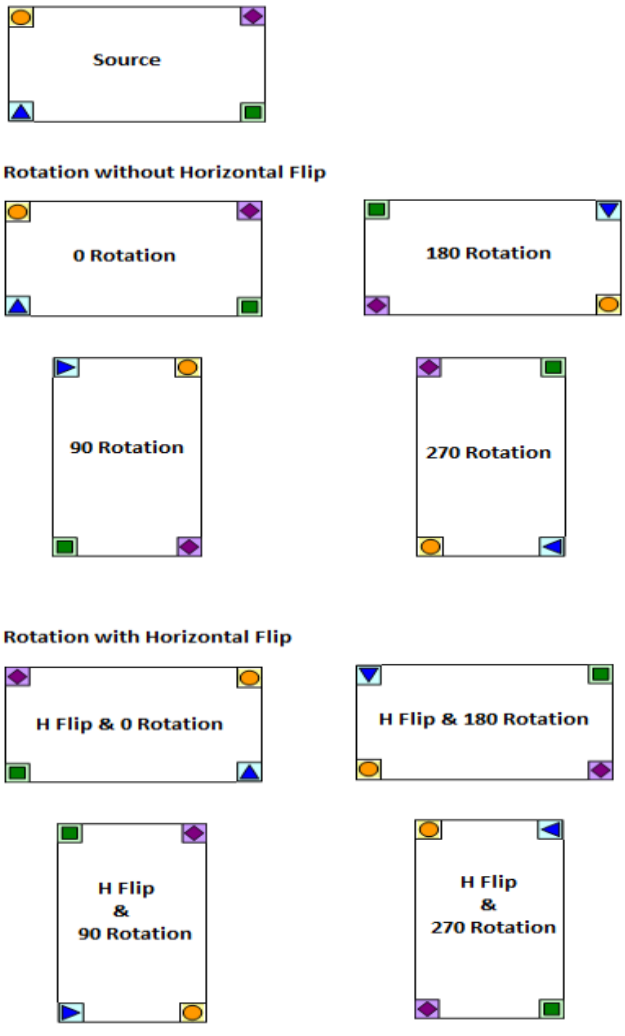
YUV 444 Packed

Name	Ignored	Y	U	V

Name	Ignored	Y	U	V
YUV 4:4:4 8 bpc	31:24	23:16	15:8	7:0
YUV 4:4:4 10 bpc - Y410	31:30	19:10	9:0	29:20
YUV 4:4:4 12 bpc - Y412	63:52	31:20	15:4	47:36
YUV 4:4:4 16 bpc - Y416	63:48	31:16	15:0	47:32

Horizontal Flip

When plane horizontal Flip is enabled with rotation, the horizontal flip operation is logically performed first followed by the rotation operation. The sample results are shown below.



DSC

DSC Source Policies

Port Type	Source Policy
eDP	DSC output BPP is as defined in DPCD. If eDP Sink reports capability for multiple link rates, choose the optimal link rate based on bandwidth calculations.
DP	DSC is enabled only when link bandwidth requires it. DSC output BPP is set to the highest possible value within link bandwidth.
HDMI	DSC BPP and link bandwidth are selected as per the Specification (VIC).

Transport mandatory DSC features

The following features are options at DSC level but are mandatory at transport level.

Design requirements

Design parameter	RGB/YUV 444	YUV 422	YUV 420
slice width	no restriction	multiple of 2	multiple of 2
	minimum = 18	minimum = 36	minimum = 36
slice height	no restriction	no restriction	multiple of 2
Pixels in Slice	Must be >= 15000	Must be >= 30000	Must be >= 30000

VDSC compression engines do not support interlaced mode of operation for any transport protocol.

Below are the list of DSC encoder specific registers in Bspec. There are 2 copies of these registers, one for each VDSC engine.

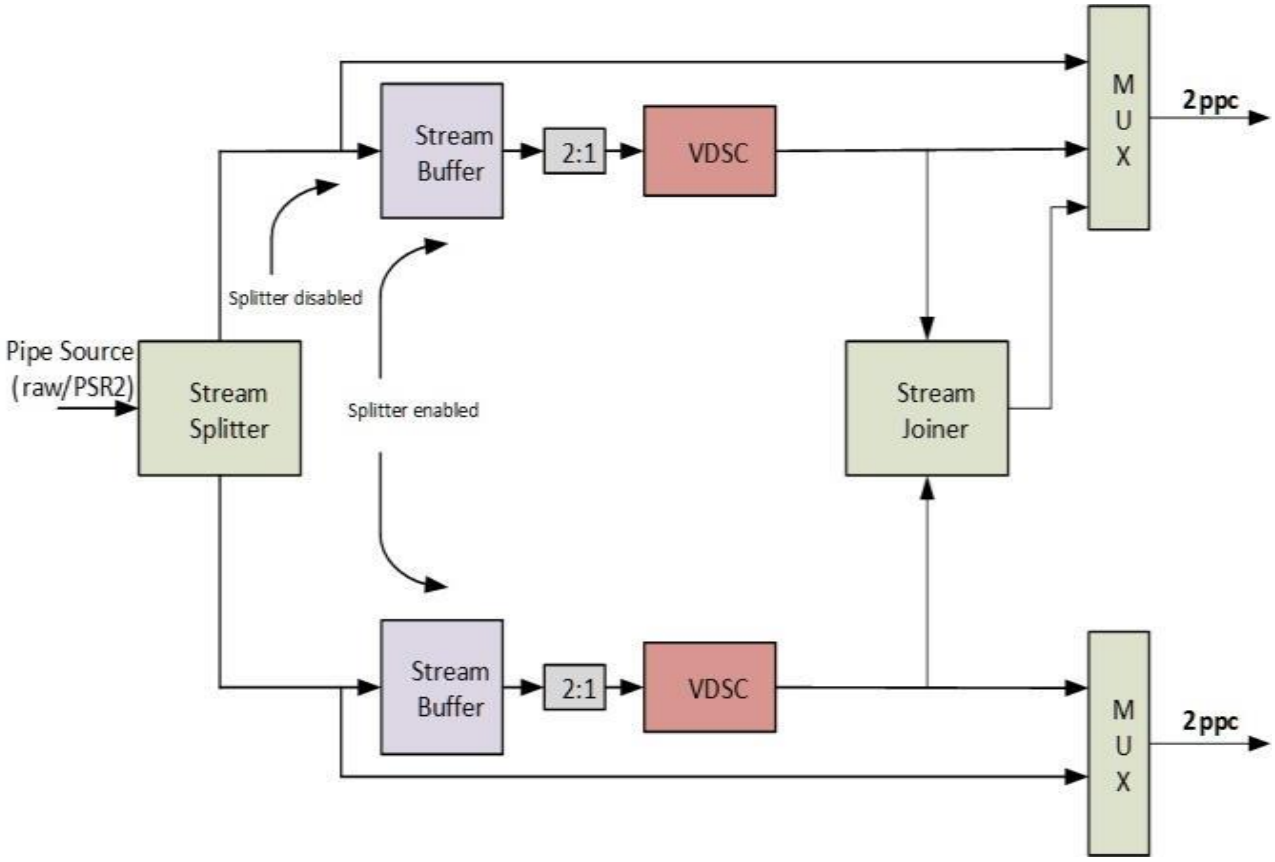
Register List
PIPE_DSS_CTL1
PIPE_DSS_CTL2
DSC_PICTURE_PARAMETER_SET_0 Field vbr_enable setting as "1" is not supported.
Set 0 Field "Allow double buffer update disable" is not from DSC standard.
DSC_PICTURE_PARAMETER_SET_1
DSC_PICTURE_PARAMETER_SET_2
DSC_PICTURE_PARAMETER_SET_3
DSC_PICTURE_PARAMETER_SET_4

Register List
DSC_PICTURE_PARAMETER_SET_5
DSC_PICTURE_PARAMETER_SET_6
DSC_PICTURE_PARAMETER_SET_7
DSC_PICTURE_PARAMETER_SET_8
DSC_PICTURE_PARAMETER_SET_9
DSC_PICTURE_PARAMETER_SET_10
DSC_PICTURE_PARAMETER_SET_11
DSC_RC_BUF_THRESH_0
DSC_RC_BUF_THRESH_1
DSC_RC_RANGE_PARAMETERS_0
DSC_RC_RANGE_PARAMETERS_1
DSC_RC_RANGE_PARAMETERS_2
DSC_RC_RANGE_PARAMETERS_3
DSC_PICTURE_PARAMETER_SET_12
DSC_PICTURE_PARAMETER_SET_13
DSC_PICTURE_PARAMETER_SET_14
DSC_PICTURE_PARAMETER_SET_15
DSC_PICTURE_PARAMETER_SET_16 Fields "slice_row_per_frame" and "slice_per_line" are not from DSC standard.

VDSC

VDSC feature compresses the raw pixel bytes into compressed byte stream as per VESA DSC specification. The pixel stream can be injected into VDSC as shown below at any of the 3 different input ports. In a typical use case, VDSC can split the pipe frame into left/right (front/back in MIPI) halves and uses 2 VDSC engines to compress each half and then joins the compressed byte streams from 2 VDSC engines into a single compressed byte stream. Each VDSC engine is configured with its own PPS register set.

Each VDSC operates with 1 ppc throughput and this becomes a limitation when the pixel clock is higher than the VDSC clock (cdclk). These cases are supported by running 2 VDSC engines in parallel, effectively realizing 2ppc throughput.



The following datapath options are supported in the VESA based compression engine. The top mux output is designated as left (front) and bottom mux output is designated as right (back) in the case of split streams. Stream joiner output can only feed into the top mux output. If the input frame is divided into an even number of slices (say, N) by enabling the splitter, then the corresponding branch parameters such as picture width and slice width need to be adjusted accordingly. For example, if the input frame (pipe source in the diagram above) is divided into 4 slices per scanline then the slice width on each branch will be HACTIVE/4 and the picture width on each branch will be HACTIVE/2.

Compression Engine Modes of Operation

Mode	Splitter Enabled	VDSC Enabled	Joiner Enabled	Notes
Bypass mode	No	No	No	Non-PSR2 stream eDP PSR2: same as bypass except for input source to DSC engine Output on left VDSC branch only.
VDSC enabled mode	No	Yes	No	MIPI: see the MIPI table below Pixel rate is limited w/o splitter, see resolution support page Output on left VDSC branch only.
Split mode	Yes	No	No	Required for eDP CoG/MSO without VDSC Required for MIPI dual link (front/back and interleaving) w/o VDSC Output on left and right VDSC branches.
Split/Compress mode	Yes	Yes	No	Required for eDP CoG/MSO with VDSC Required for MIPI dual link with VDSC Output on left and right VDSC branches.
Joiner enabled mode (small joiner)	Yes	Yes	Yes	Cannot work with CoG Output on left VDSC branch only.

Big Joiner

The above DSC engine can achieve compression over multiple pipes to compress 8K image size as shown below. A frame can be divided and processed by 2 adjacent pipes, compressed by their associated VDSC engines, then the compressed pixels streams from 2 pipes will be joined into a single compress pixel stream. In this 8K scenario the frame is compressed by a total of 4 VDSC engines in parallel. We have separate PPS DIP packet defined on per transcoder basis.

In the case of big joiner mode, video timings are determined by the attached transcoder.

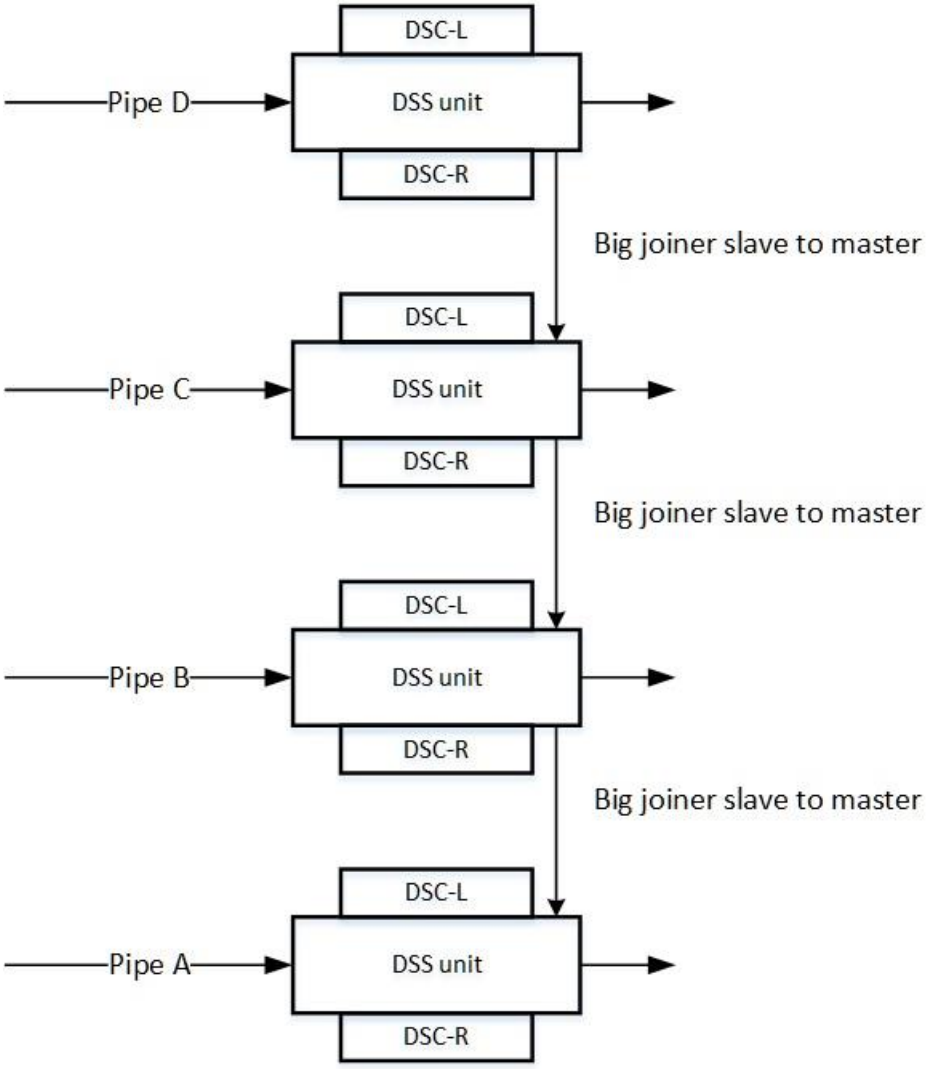
Big joiner mode cannot be used if compression is bypassed in VDSC.

Big joiner mode cannot be used if small joiner is not enabled.

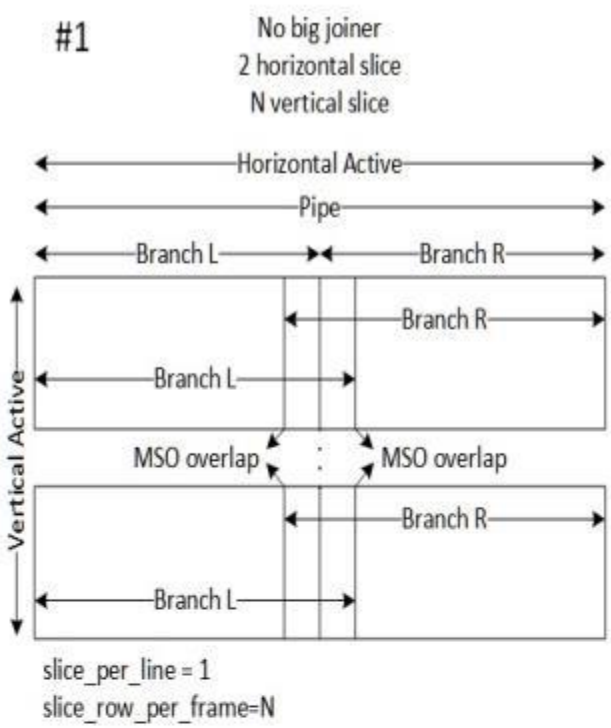
The front-end display requirements in the case of big joiner mode are same as in dual port 8K configuration. Audio works same as in other display configurations.

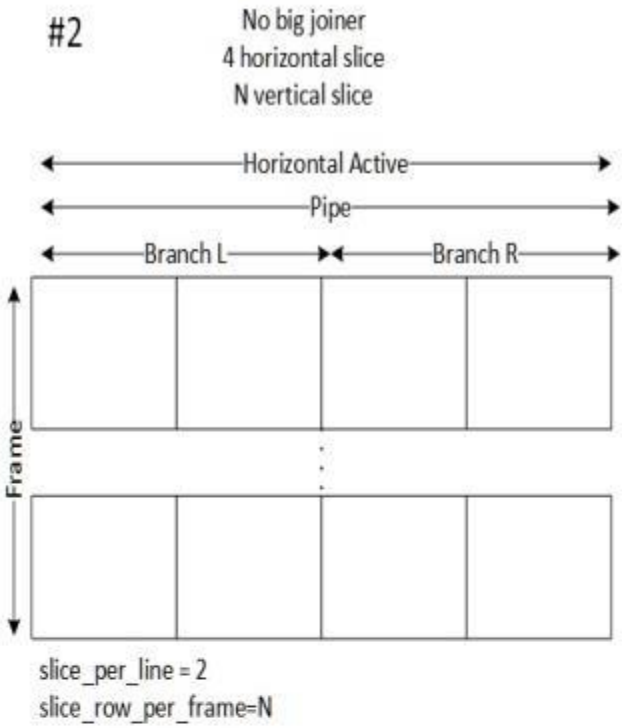
In DSC:

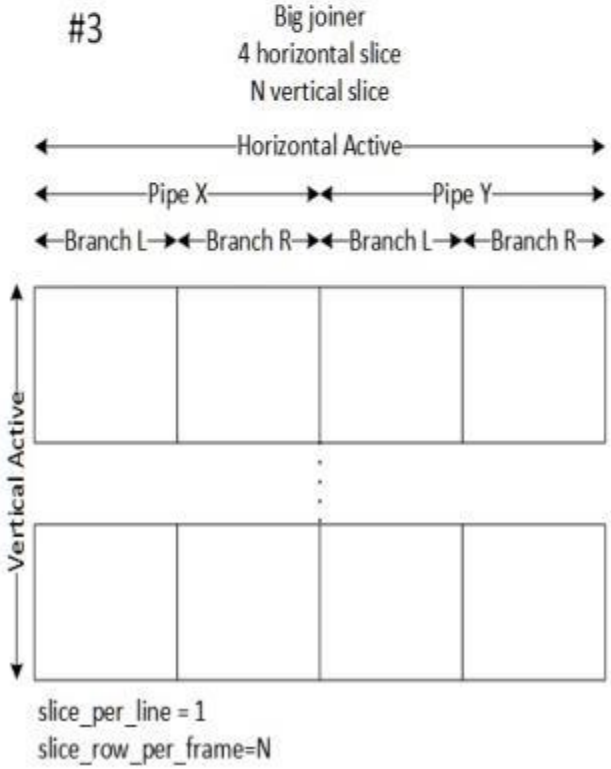
- Compression engine is moved to the pipe and all pipes support VDSC.
- Any two adjacent pipes can be joined together as shown below to support even higher resolutions (example: 8k).

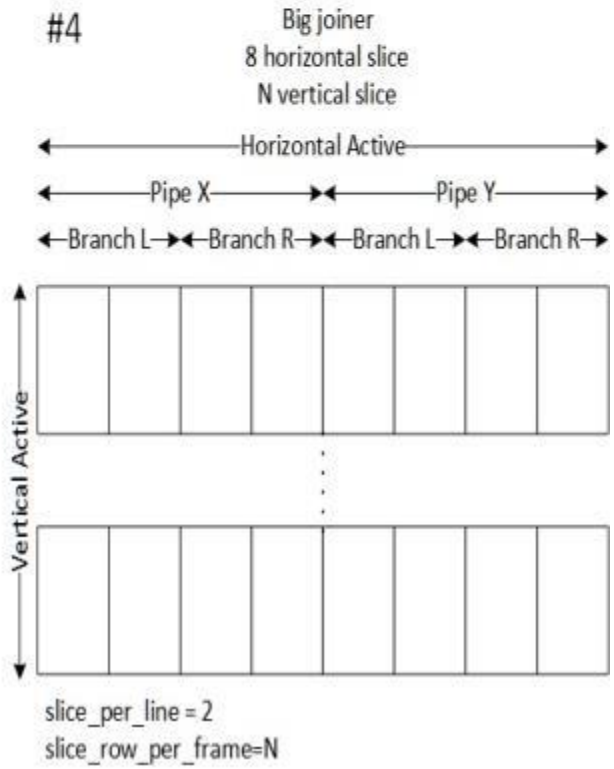


VDSC Slice Options









- Each DSC branch supports 1 or 2 horizontal slices.
- Both branches must be enabled with small joiner, so pipe output is 2 or 4 horizontal slices
- Final port output without big joiner (1 pipe alone) is 2 or 4 horizontal slices.
- Final port output with big joiner (2 pipes combined) is 4 or 8 horizontal slices.
- For non-MSO use cases, Horizontal active must be an integer multiple of slice width in pixels. For MSO use cases, overlap pixels need to be included.
- Each DSC branch slice_per_line = number of horizontal slices this DSC branch is processing = number of horizontal slices for full frame / number of DSC branches enabled for this video stream
- Minimum vertical slice (slice row) count is 1.
- Maximum vertical slice (slice row) count is 4095.
- Vertical active must be an integer multiple of slice height in lines.
- Each DSC branch slice_row_per_frame = number of vertical slices = vertical active / slice height
- VDSC spec implies that 108 lines is an optimal slice height, but any size can be used as long as vertical active integer multiple and maximum vertical slice count requirements are met.
- Each DSC branch pic_height = vertical active
- non-MSO: Each DSC branch pic_width = horizontal active / number of DSC branches enabled for this video stream. PPS transmitted to receiver must use full width.
- MSO: Each DSC branch pic_width = horizontal active / number of DSC branches enabled for this video stream + overlap pixels. PPS transmitted to receiver must use full width.

Programming considerations and restrictions

DP 1.4a RESTRICTION: For every VDSC instance, active video height must be an integer multiple of slice height.

DSC 1.2a slice height recommendation: Based on empirical data the optimal slice height is 108.

The default slice height can be set to a value that satisfies the above as follows.

slice height = ~108 and picture height = N * slice height where N is an integer.

Below are the checklist items to program VDSC.

- DSS registers to configure VDSC branches
- Picture parameter set register programming for each VDSC branch
- Link M/N programming for compression enabled use cases
- Transcoder data island packet for PPS

Note that there is no implicit order to program compression engine's PPS and DSS registers other than that they need to be programmed prior to a VBLANK.

DSI Configuration	Splitter	Joiner	DSC to DSI Mapping	Notes
Single link - single pipe	Yes	Yes	DSC (L) + DSC (R) -> DSIx	Small joiner output always on left VDSC branch.
Dual independent links	Yes	Yes	PipeX -> DSC (L) + DSC (R) -> DSI0 PipeY -> DSC (L) + DSC (R) -> DSI1	Each pipe will instantiate its own DSS unit to drive pixels to one of the DSI ports.
Dual link - single pipe	Yes	No	Pipe A DSC(L) - DSI0 Pipe A DSC(R) - DSI1	Both ports must be bound to the same pipe AND DSI0 must be configured for Port Sync Mode. Only pipe A can be used.

The following rules determine the highest output bpp that the design can support in various compression scenarios.

Pipe BW check: Pixel clock < PPC * CDCLK * number of pipes

Link BW check: output bpp < number of lanes * DDICLK * 8 / Pixel clock -- DisplayPort

Big joiner BW check
output bpp <= PPC * CDCLK * 24 bits / Pixel clock

Small joiner RAM check: output bpp <= joiner RAM (bits) / Horiz. width

Note that horizontal width has to account for overlap and dummy pixels in CoG use cases.

Greatest allowed DSC output BPP = INT(MIN (output BPP from Link BW check, output BPP from Big joiner BW check, output bpp from small joiner RAM check))

For cases where FEC is enabled, pixel clock is replaced by pixel clock/0.972261 in the above calculations.

Smaller Joiner RAM Size
7680 bytes per pipe, 15360 bytes for two joined pipes Example: CDCLK = 336 MHz, DDICLK = 810 MHz, PPC = 2 pixels/cycle, DP lanes = 4, input Pixel BPP = 30, Pixel clock (CVT) = 1026 MHz, Horiz. width = 3840 pixels Greatest allowed DSC output BPP = INT(MIN (output BPP from Link BW check, output BPP from Big joiner BW check, output bpp from small joiner RAM check)) = INT(MIN (25.26, 15.71, 32)) = INT(15.71) = 15

Design Support

With dual pipe configuration, HW is capable of supporting maximum of 8 slice per line. (Big joiner mode and dual pipe cases)

With single pipe configuration, HW is capable of supporting maximum of 4 slice per line. (Small joiner and single pipe cases)

Input pixel format	bpp	Max input width	Output bits per pixel[1]	DSC Standard
RGB444	24 and 30 bpp	5120 pixels per pipe, 8192 across two joined pipes	8 to 23[2]	v1.1
RGB444	24, 30 and 36 bpp	5120 pixels per pipe, 8192 across two joined pipes	8 to 23[2]	v1.1

[1]BPP at max input width

[2]Use the rules above to calculate maximum bpp

[3]FEC is enabled. See the DDI FEC section for FEC support.

[4] Fractional bpp mode enabled.

[5] Some joiner modes limit output bpp to half of input bpp.

Transcoder

Transcoder DSI Function

The following registers are specific to the MIPI Display Serial Interface (DSI) ports. Both DSI Controller (i.e. the DSI Protocol Layer) and D-PHY (i.e. the DSI Physical Layer) registers are listed here.

Registers
DSI specific functional configuration registers:
TRANS_DSI_FUNC_CONF
Control for DSI protocol timers:
DSI_PWAIT_TO
DSI_HTX_TO
DSI_LRX_H_TO
DSI_TA_TO
DSI_CALIB_TO
DSI interrupt registers:
DSI_INTER_MSK_REG
DSI_INTER_IDENT_REG
DSI timing control registers:
DSI_CLK_TIMING_PARAM
DSI_DATA_TIMING_PARAM
DSI_TRIG_TX_TIME

Registers
DSI_ESC_CLK_DIV
DSI_T_INIT_PRIMARY
DSI_T_WAKEUP
DSI_IO_MODECTL
D-PHY control registers (Located in Combo-PHY):
DPHY_CLK_TIMING_PARAM
DPHY_DATA_TIMING_PARAM
DPHY_TA_TIMING_PARAM
DPHY_TRIG_EXT
DPHY_ESC_CLK_DIV
DSI command transmit/receive control:
DSI_CMD_FRMCTL
DSI_CMD_TXCTL
DSI_CMD_RXCTL
DSI_LP_MSG
DSI receive data (header and payload):
DSI_CMD_RXHDR
DSI_CMD_RXPYLD
DSI transmit data (header and payload):
DSI_CMD_TXHDR
DSI_CMD_TXPYLD
DC state control:

The following registers are common across all ports, in which the DSI port is one of them.

Registers
Transcoder Control
TRANS_CONF
TRANS_STEREO3D_CTL
Transcoder Timing
TRANS_HTOTAL
TRANS_HSYNC
TRANS_VTOTAL
TRANS_VSYNC
TRANS_SPACE
TRANS_VSYNCSHIFT
Transcoder DDI Function
TRANS_DDI_FUNC_CTL
TRANS_DDI_FUNC_CTL2

Burst Mode Considerations

The MIPI DSI Burst Mode is a Video Mode concept where the transcoder delivers pixels to the Panel at a higher bandwidth than the pixel rate required to maintain the display timings. Note that there is no concept of Burst Mode in Command Mode since the Host is not maintaining the display timings and should be sending the pixels to the Panel as fast as possible.

Burst mode can be achieved in Video Mode by one of the following methods:

1. Program the Link speed to the minimum frequency needed to satisfy the resolution pixel rate and compress the pixel stream. This is assuming the minimum frequency is less than or equal to the maximum frequency supported.
2. Program the Link speed to a supported frequency that will generate a pixel rate greater than what is required to satisfy the resolution pixel rate

It is important to understand that the DSI transcoder uses the Horizontal Total, Sync Start, and Sync End programming (programmed in terms of pixels) to calculate the number of 1X clocks needed to attain the desired H. timings to the Panel. Hardware uses the following equation to perform this conversion:

H. Time (in 1X clocks) = ceiling((H. Size * bpp) / (Operating Width * 8)

Some notes on this equation:

1. The "bpp" term is the pixel format being transmitted to the Panel
2. 24bpp should be used for compressed pixel streams
3. Hardware does not know what frequency the Link is running at, so it is the responsibility of Software to program the Link frequency appropriately to meet the desired pixel rate

In a non-Burst Mode of operation, Software will program the Link speed to a frequency that will generate the pixel rate on the Link needed to satisfy the Panel's resolution. In this case the H. timings that Hardware calculates will correlate directly to the H. timings (in pixels) that Software programs. In other words, the 1X clock period is equal to the pixel rate needed for the resolution. However, for a Burst Mode method where the Link speed is running at a higher frequency, the period of the 1X clock is going to be less than the theoretical pixel rate period, so Software will need to account for that by adjusting the H. Total, Sync Start, and Sync End timings. In other words, the H. timings will need to be increased to offset the faster 1X clock frequency.

Calculating the Horizontal burst timings in pixels is simply taking the normal H. timing parameter and multiplying it by the burst ratio, where the burst ratio is:

Burst Frequency Ratio = $f_{8X\ Burst}$ / $f_{8X\ Minimum}$

H. Timing Burst = ceiling(H. Timing Non-Burst * Burst Frequency Ratio)

Example:

H. Active	3072	pixels
H. Blank	160	pixels
H. Front Porch	48	pixels
H. Sync	32	pixels
V. Active	1920	lines
V. Blank	55	lines
Frames per Second	60	fps
Pixel Format	18	bits per pixel
Operating Width	4	lanes
Pixel Frequency	382.99	Mega pixels per frame
8X Min Frequency	1723	MHz
8X Burst Frequency	2500	MHz
Burst Frequency Ratio	1.451	

H. Total Burst= ceiling((3072 + 160) * 1.451) = 4690 pixels

H. Sync Start Burst = ceiling((3072 + 48) * 1.451) = 4527 pixels

H. Sync End Burst = ceiling((3072 + 48 + 32) * 1.451) = 4574 pixels

Programming Horizontal Timings in Video Mode when operating with compressed pixels

When a Horizontal resolution requires a Link speed that is greater than the maximum supported frequency, then compression can be used to reduce the Link speed to a frequency that is within a supported frequency. When the transcoder is operating within this regime, then similar to the Burst Mode described above, the Horizontal timings (H. Total, Sync Start, and Sync End) will need to be adjusted to account for a Link speed that is slower than what the non-compressed Link speed would need to be running at.

H. Timing Compressed = ceiling(H. Timing Non-Compressed * Compression Frequency Ratio)

Compressed Frequency Ratio = $f_{8X\ Compression}$ / $f_{8X\ Non\text{-}Compressed}$

Example:

H. Active	5120	pixels
H. Blank	160	pixels
H. Front Porch	48	pixels
H. Sync	32	pixels
V. Active	3200	lines
V. Blank	91	lines
Frames per Second	60	fps

Pixel Format	24	bits per pixel
Operating Width	4	lanes
Pixel Frequency	1042.5888	Mega pixels per second
8X Non-Cmp Frequency	6255.5	MHz
8X Cmp Frequency	2211.6	MHz
Cmp Frequency Ratio	0.3535	

H. Total Burst= ceiling((5120 + 160) * 0.3535) = 1867 pixels

H. Sync Start Burst = ceiling((5120 + 48) * 0.3535) = 1828 pixels

H. Sync End Burst = ceiling((5120 + 48 + 32) * 0.3535) = 1839 pixels

Determining Minimum Horizontal Blanking Regions

When in Video Mode, there are instances where the DSI controller does not have enough time to let the link transition to the LP state before the next HS burst. In these instances, the controller transmits a Blanking packet to keep the link in the HS state. However, Blanking packets are long packets (i.e. a packet header + payload + payload CRC) which means that the time between non-Blanking packet HS bursts (i.e. blanking regions) must be big enough to transmit a minimum sized Blanking packet.

Per the DSI specification, a long packet can carry zero bytes of payload (i.e. a packet header + payload CRC), but to simplify the hardware the transcoder expects the Blanking packet to carry a payload, and to keep the CRC boundary conditions to a minimum, the transcoder expects the payload to be **greater** than 4 bytes. Therefore, the minimum sized Blanking packet size that the controller can support is **11 bytes**.

In addition to the minimum Blanking Packet size, the blanking region must also account for protocol overhead (i.e. Pixel Packet header and payload CRC, and Sync Packets). The overhead within each horizontal blanking region differs slightly as shown within the following table.

Blanking Region	Size (Bytes)	Notes
Horizontal Front Porch	17	11B for minimum Blanking Packet, 6B for Pixel Packet header and payload CRC
Horizontal Sync	15	11B for minimum Blanking Packet, 4B for Horizontal Sync Start packet
Horizontal Back Porch	15-19	11B for minimum Blanking Packet, 4B for Horizontal Sync End packet, 4B for Pixel Packet header. Additional notes: 1. HW subtracts the Pixel Packet header from this region which is why the additional 4B is added to the minimum. 2. During Sync Event model the H. Sync End packet is not

Blanking Region	Size (Bytes)	Notes
		transmitted, so those 4B can be recovered.

Given the above, the number of byte clocks for a minimum sized blanking region can be calculated with the following equation.

Min Blanking Region (Byte clocks) = Ceiling(Region Size / Operating Width)

The following table lists the blanking region size in number of Byte clocks across the different operating widths:

Blanking Region	x1	x2	x3	x4
Horizontal Front Porch	17	9	6	5
Horizontal Sync / Horizontal Back Porch (Event)	15	8	5	4
Horizontal Back Porch (Pulse)	19	10	7	5

All the horizontal timings to the transcoder's timing generator is defined in pixels, so the minimum blanking region needs to be translated to the pixel domain. Recall from above, we have the following equation to translate the horizontal timings from the pixel domain to the number of 1x (i.e. Byte) clocks.

H. Time (in 1X clocks) = Ceiling((H. Size * Format bpp) / (Operating Width * 8))

Minimum H. Size (in pixels) = Ceiling((Min Blanking Region * Operating Width * 8) / Format bpp)

Where:

1. The equation is assuming the minimum horizontal pixel size result is one-based. Software will need to convert the size to zero-based when programming the horizontal timing registers
2. "H. Time (in 1X clocks)" = "Min Blanking Region"
3. The numerator of the equation will be a constant for each operating width and represents the blanking region size in terms of bits

Determining Vertical Blank in DBI

When operating in Command Mode (i.e. DBI) the DSI transcoder is not responsible for the timings to the Panel, but it does need to maintain some timings to the Display Engine. Specifically, the Display Engine requires a certain amount of Vertical blanking time to fetch enough of the image from memory to supply the required bandwidth to the DSI transcoder. The minimum Vertical blanking time (in lines) that the Display Engine needs is:

V. Blank = ceiling(400us / Line Time)

The time it takes the DSI transcoder to transmit a line (i.e. Line Time) is given by the following equation:

Line Time = # Byte clocks per line * Byte clock period

Line Time = [(H. Total * Bytes per Pixel) / Operating Width] * Byte clock period

Example:

H. Total = 4256 pixels

8X Clock = 2500 MHz = 3.2ns Byte clock period

Pixel Format = RGB888 = 3 Bytes per Pixel

Operating Width = 4

Line Time = [(4256 * 3) / 4] * 3.2ns = 10.214us

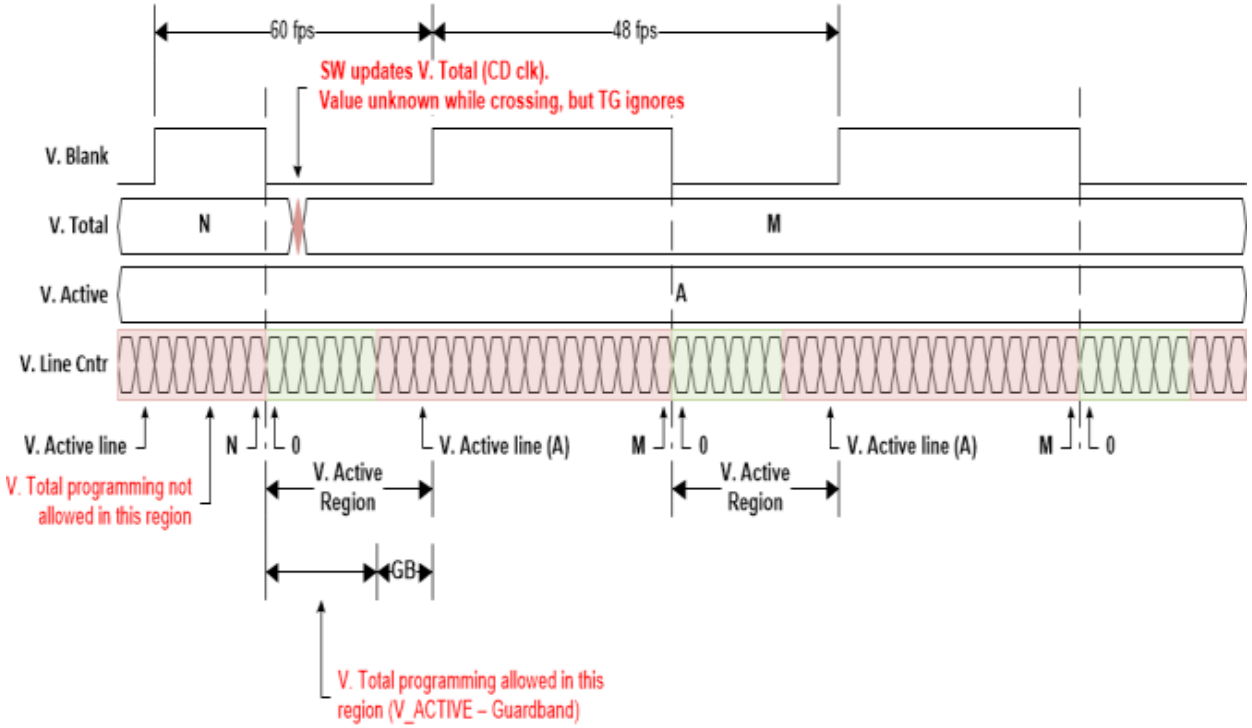
V. Blank = ceiling(400 / 10.214) = 40 lines

Extending Vertical Blank for Lower Frame Rates in DPI

A power saving option when the DSI transcoder is running in Video Mode is to extend the V. Blank time of the frames while running the clocks at the frequency required for the highest refresh rate. Normally, the transcoder expects the frame timings to be programmed before the timing generator is enabled and it expects the timings to remain unchanged while the timing generator is enabled, but for the Vertical timings if Software programs them at the right time, the transcoder should be able to handle the change seamlessly.

Specifically for this functionality, Software will be changing the V. Total field of the **TRANS_VTOTAL** register to either extend or reduce the V. Blank duration to get the desired frame rate. Since the DSI transcoder's timing generator only looks at this field when it is in the Vertical Back Porch, Software shall program the desired V. Total during the vertical active region.

To ensure the write to the TRANS_VTOTAL finishes before the V. Blank region, Software should ensure that it is not closer than 10 lines from V. Blank before issuing the write (i.e. lines 0 to (V. Active-10)). **Also, to ensure there is no glitching on the V. Active field of the TRANS_VTOTAL register, Software shall disable the byte enables for that portion of the register when re-programming the V. Total field.**



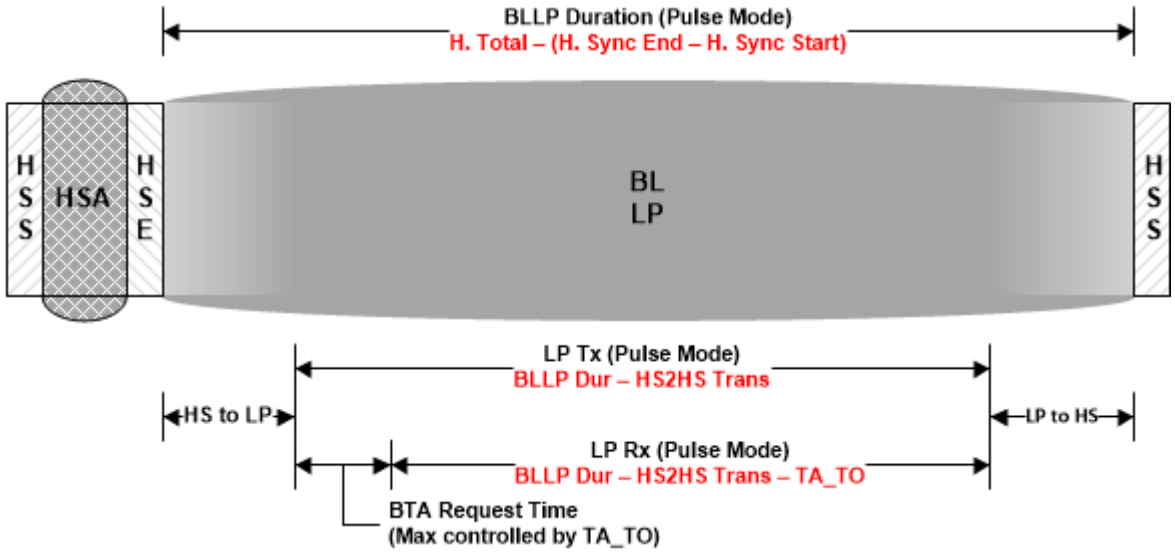
Performing LP Transactions in Video Mode

When in Video Mode the Host is responsible for transmitting the timing information to the Panel. Low Power communication is a very low bandwidth mode of operation on the Link (about 10 Mbps), so it is critical that there is enough time between scheduled synchronous timing packets to perform any LP transactions.

The following latencies need to be accounted for if a LP transaction is going to be performed:

- The available time between Sync packets in the Vertical blanking region
- HS to LP and LP back to HS Link transition latencies (HS to HS latency)
- Bus Turn-Around (BTA) latency, if doing a BTA
- LP transmission

The below diagram illustrates the durations available for LP communication within a Vertical blank line time.



The amount of time available for the LP transactions within the V. blanking region is dependent on whether the transcoder is in the Sync Pulse of Event mode.

Sync Mode	BLLP Duration
Pulse	H. Total - (H. Sync End - H. Sync Start)
Event	H. Total

Since the Horizontal timings are defined in terms of pixels, the BLLP Duration needs to be converted to the Byte clock domain using the following equation:

Number of Byte Clocks = Ceiling((H. Size * Bytes per Pixel) / Operating Width)

The HS to HS transition latencies are given in the following table:

CLK Lane in LP?	HS2HS Transition Latency
Yes	(N * 17) + 12
No	(N * 7.5) + 4

Where N is the number of Byte clocks per Escape clock

N = ceiling(f_{Link} (MHz) / 160)

Note that the above Link transition latencies assumes the default HW maintained timing parameters. If any of the timing parameters are being overridden by SW, than that needs to be accounted for.

So, given the above, the amount of time available for the actual LP transmissions is given as the following:

Direction of LP Transaction	Allowed Duration
Tx	BLLP Duration - HS2HS Latency
Rx	BLLP Duration - HS2HS Latency - Turnaround Timeout

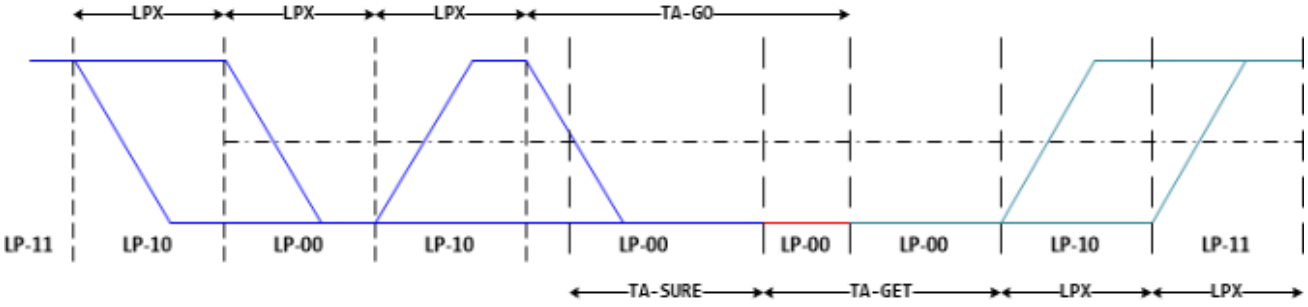
Some things to note on the above durations:

1. For the receive direction, the maximum amount of time given to perform a BTA is defined by the Turnaround Timeout (DSI_TA_TO). This timeout is in Escape clocks, so it will need to be multiplied by N
2. For the receive direction, Software should program the LP RX Timer Timeout (DSI_LRX_H_TO) to the allowed Rx duration.
3. For the receive direction, Software should program the Set Maximum Return Packet Size to account for the allowed Rx duration. Note that the value programmed should take into account the overhead of the Escape Mode entry, exit, packet header, and payload CRC.

Programming DSI_TA_TO and DSI_LRX_H_TO

The TA_TO and LRX_H_TO timers specify how long the DSI controller (i.e. the Host) will allow the panel to own the bus when there is a bus turn-around in progress.

For the turnaround time (DSI_TA_TO), the sequence to turn the Data Lane around is given below:



The table below shows the latency of a bus turn-around.

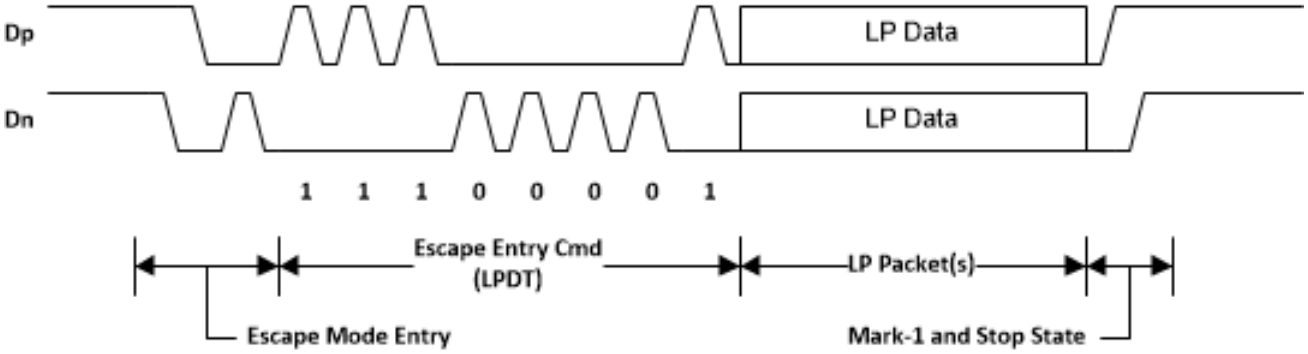
Sequence Step	HW Maintained (1)	SW Override (1)
LP10	1	1
LP00	1	1
LP10	1	1
V_{OL} (2)	1	1
LP00 (3)	6	TA-SURE + TA-GET

Sequence Step	HW Maintained (1)	SW Override (1)
LP10	1	1
LP11	1	1
Guardband (4)	2	2
TOTAL:	**14**	**8 + TA-SURE + TA-GET**

Notes:

1. The latencies are in terms of Escape clocks (i.e. T_{LPX})
2. From when the Physical layer starts driving the TA-SURE step of the sequence (i.e. LP10 to LP-00), there is a short latency for the 1->0 transition to pass the V_{OL} threshold. This latency is less than one T_{LPX}, but since the granularity of the timeout timer is at the Escape clock granularity, this step is rounded up to one Escape clock.
3. TA-SURE, and TA-GET timing parameters are maintained by the Host and can, therefore, be overridden by Software. HW maintains a TA-SURE of 1 T_{LPX} and a TA-GET of 5 T_{LPX}
4. The guardband is to account for any clock crossings

For the amount of time that the panel should be allowed to own the link (DSI_LRX_H_TO), the timeout value is dependent on the maximum amount of data that the panel could have to send back to the Host. The worst case amount of data the panel can send back to the Host is a READ return followed by an Acknowledge and Error packet.



Sequence Step	Duration (1,2)
Escape Mode Entry	5
Escape Entry Cmd	16
LP Packets	READ Hdr: 64 READ Pyld: 512 (3) READ Pyld CRC: 32

Sequence Step	Duration [1,2]
	Ack & Error Hdr: 64 [4]
Mark-1 and Stop State	2
Bus Turn-Around	14
Total:	**709**

Notes:

1. When in the Escape Control domain, it takes two Escape clocks (T_{LPX}) to transmit a given bit (e.g. the Escape Entry Command is 8 bits, so it takes 16 T_{LPX} to transmit the command)
2. This analysis is assuming the panel is not stalling
3. The Host can accept up to 32 bytes of READ payload data from the panel (i.e. the Set Maximum Return Packet Size can be set up to 32). Software can control how much READ data the panel sends for each BTA.
4. This analysis is assuming the panel sends the Acknowledge and Error packet immediately after the READ packet.

Transcoder Control

Control
TRANS_CONF
TRANS_STEREO3D_CTL

Transcoder LRR

LRR (Low refresh rate) is mainly about dynamically reducing panel refresh rate to lowest possible refresh rate or to media multiple refresh rate (e.g. 48Hz) for idle and media playback scenarios. LRR provides predictable VBI unlike flip based refresh rate. LRR feature relies on software programmed Vtotal functionality and can be enabled on VRR capable panels.

Entry Sequence

1. If VRR is enabled, driver disables VRR.
2. Driver updates Vtotal and Vblank-end to any refresh rate supported by the panel.

Exit Sequence

1. Driver updates Vtotal and Vblank-end to nominal refresh rate.

Here is an example scenario with idle and LRR.

1. Configure eDP/DP at 60 Hz.

2. LRR idle entry: Update Vtotal and Vblank-end for 40 Hz
3. LRR idle exit: Update Vtotal and Vblank-end for 60 Hz
4. Enalbe VRR and program flips
5. Disable VRR
6. LRR media playback entry: Program flip and then update Vtotal and Vblank-end for 48 Hz
7. LRR media playback exit: Update Vtotal and Vblank-end for 60 Hz

LRR Restrictions

- After push, software must wait for push done before starting another push.
- Global Double Buffer Disable functionality is not supported with LRR.
- No asynchronous flips are allowed when LRR is enabled.
- Interlaced mode, DRRS, Stereo 3D, and PSR2 are incompatible with LRR.
- Since panels may have a restriction on the maximum change in refresh rate between two consecutive frames, disabling LRR (TRANS_VRR_CTL[31]==0) does not immediately bring the current refresh rate to the nominal refresh rate.
- When using the transcoder port sync mode, LRR must use Pipeline Full Override Programmed Pipeline Full Line Count setting in TRANS_VRR_CTL.
- PSR is incompatible with LRR.

Transcoder Timing

Transcoder
TRANS_HTOTAL
TRANS_HBLANK
TRANS_HSYNC
TRANS_VTOTAL
TRANS_VBLANK
TRANS_VSYNC
TRANS_SPACE
TRANS_VSYNCSHIFT
TRANS_MULT

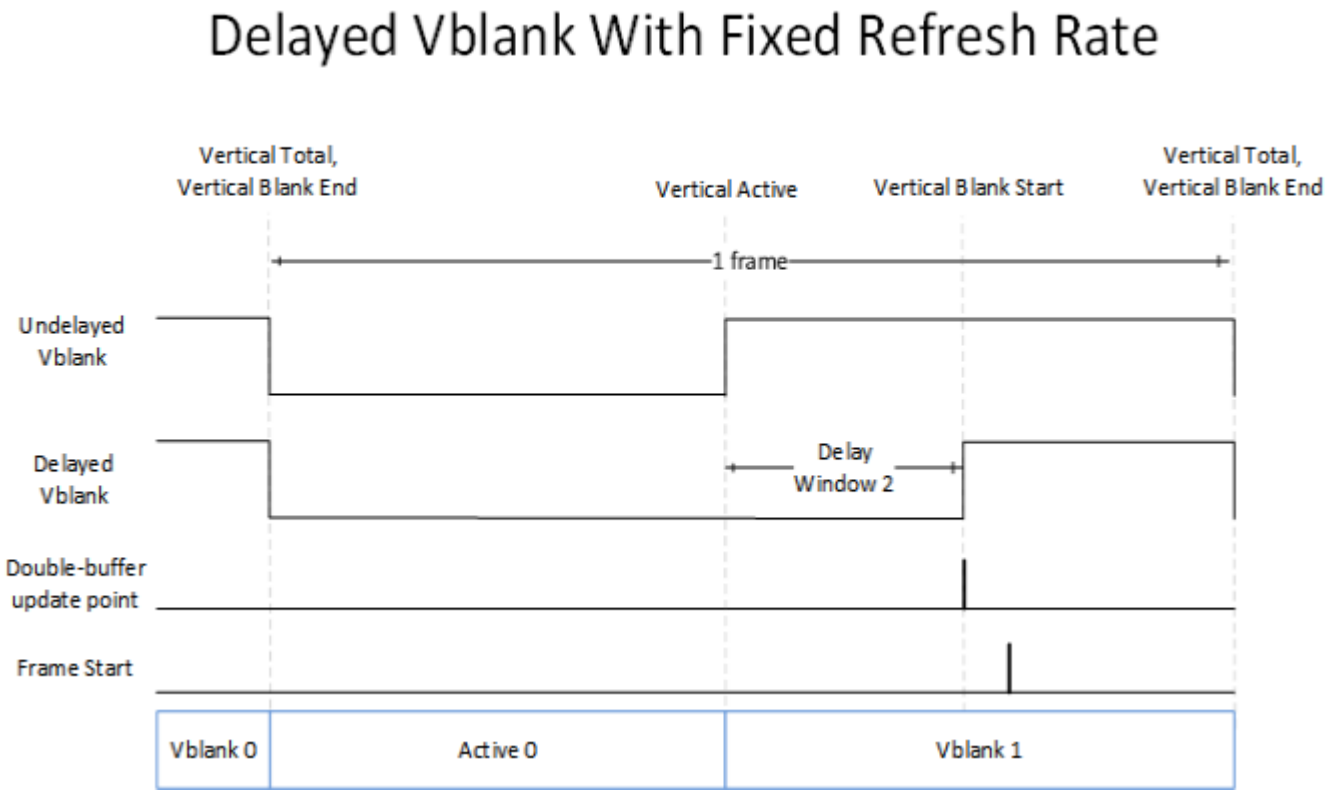
The transcoder timing generators create two versions of vertical blank (vblank) that are sent to the display pipes. The undelayed vblank asserts at the end of vertical active (TRANS_VTOTAL Vertical Active). The delayed vblank asserts at the vertical blank start (TRANS_VBLANK Vertical Blank Start) and must be programmed greater than or equal to the end of vertical active. Both de-assert at the end of vblank (TRANS_VBLANK Vertical Blank End), which must be programmed to match the vertical total (TRANS_VTOTAL Vertical Total).

The rising edge of the delayed vblank is the double-buffer register update point. The frame start signal asserts after that point, then pixel processing begins for the next frame.

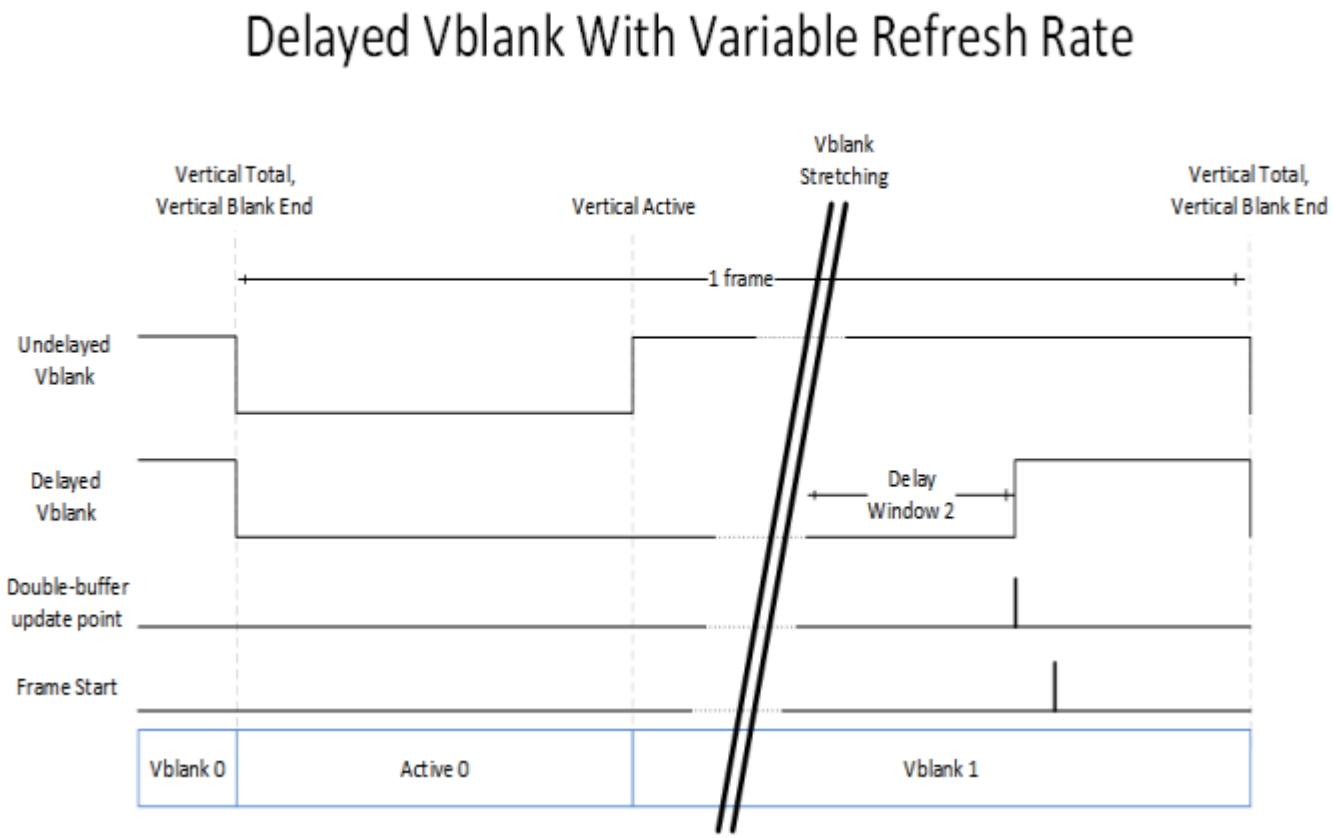
TRANS_VBLANK Vertical Blank Start can be programmed greater than TRANS_VTOTAL Vertical Active to create a window between start of undelayed vblank and delayed vblank where pixels are not being processed (called window 2) and programming can safely update double-buffered and non-double-buffered registers.

Window 2 size = TRANS_VBLANK Vertical Blank Start - TRANS_VTOTAL Vertical Active

Window 2 is typically only configured to be greater than zero when DSB will be used to program display registers. Without DSB, driver MMIO programming usually is too slow to fit in window 2, and instead must update only the double-buffered registers and use capabilities like the double buffer update disable to ensure atomic updates, and then there is no need to delay the vblank. See the DSB Engine Programming for window 2 size requirements.



When variable refresh rate is used, the vblank stretches while waiting for the indication to complete this vblank and start the next frame. The stretching happens before window 2.



Transcoder MN Values

There is one instance of these registers per each transcoder.

For dynamic switching between multiple refresh rates, M/N values may be reprogrammed on the fly. The link N should be programmed last to trigger the update of all the data and link M and N registers and then the new M/N values will be used in the next frame that is output.

DATAM

DATAN

LINKM

LINKN

Clocks:

ls_clk is the link symbol clock. i.e. 270 MHz for HBR.

strm_clk is the stream clock, which is the video pixel rate or dot clock.

cdclk is the core display clock.

The link only operates in Synchronous Clock mode. The link clock and stream clock are synchronous and the link M and N values stay constant for a given pixel rate.

Calculation of TU:

TU is the Transfer Unit used in SST.

TU size = 64 (recommended)

Calculation of Data M, and Data N:

Active/TU Size = Payload/Capacity = Data M/N

Note that for 4:2:0 format the number of bytes per pixel will be half the number of bytes of RGB 4:4:4 pixel.

Recommendation: In data M/N calculation, M should be rounded up and N should be rounded down.

Context:	Low Bandwidth SST over High BW link
Active/TU size = Data M/N Select link rate such that there is at least 1 active symbol in every TU for any given dot clock. An example where this check fails is 720p with YUV 4:2:0 pixel format over HBR3 channel: Active/64 = (27 * 1.5) / (810 * 4) results in Active < 1.	

Compression Ratio (CR) = DisplayPort Compression enabled ? min(ratio1,ratio2) : 1

- ratio1 = Compressor BW / Link BW = (cdclk * bytes per pixel) / (ls_clk * number of lanes)
- ratio2 = (Horizontal active in pixels * bytes per pixel / 4) / ((Horizontal active in pixels * bytes per pixel / 8) +2)

Data M/N = (strm_clk * bytes per pixel) / (CR * ls_clk * number of lanes)

The data M/N should be calculated as below for the eDP cases with VESA DSC.

First compute the compression ratio based on the DSC PPS parameters as follows.

without FEC:

Compression Ratio (CR) = DSC input bits per pixel / DSC output bits per pixel

Examples:

RGB444: Compression Ratio (CR) = bits_per_component *3 / bits_per_pixel

If eDP CoG with VESA compression

Data M/N = ($strm\_clk_{CoG}$ * bytes per pixel) / (CR * ls_clk * number of lanes per segment)

else if eDP with VESA compression (no CoG)

Data M/N = (strm_clk * bytes per pixel) / (CR * ls_clk * number of lanes)

Note1: Data M/N remains unchanged between CoG and non-CoG because the reduction in the number of lanes is compensated by the reduction in stream clock.

Note2: For the CoG case above, the actual stream clock on pipe side = ($strm\_clk_{CoG}$ * number of segments).

Note3: $strm\_clk_{CoG}$ is calculated considering a single segment

with FEC:

Compression Ratio (CR) = DSC input bits per pixel / DSC output bits per pixel

Examples:

RGB444: Compression Ratio (CR) = bits_per_component *3 / bits_per_pixel

If eDP CoG with VESA compression

Data M/N = ((strm_clk$_{CoG}$ * bytes per pixel) / (ls_clk * number of lanes per segment)) * 1/(0.972261 * CR)

else if eDP with VESA compression (no CoG)

Data M/N = ((strm_clk * bytes per pixel) / (ls_clk * number of lanes)) * 1/(0.972261 * CR)

Note1: Data M/N remains unchanged between CoG and non-CoG because the reduction in the number of lanes is compensated by the reduction in stream clock.

Note2: For the CoG case above, the actual stream clock on pipe side = (strm_clk$_{CoG}$ * number of segments).

Note3: strm_clk$_{CoG}$ is calculated considering a single segment

The data M/N should be calculated as below for the case FEC is enabled over the link in order to account for the 2.4% overhead.

Data M/N = ((strm_clk * bytes per pixel)/(ls_clk * number_of_lanes)) * 1/(0.972261)

If compression is also enabled, then the following calculation should be used.

Data M/N = ((strm_clk * bytes per pixel)/(ls_clk * number_of_lanes)) * 1/(0.972261*CR)

Calculation of Link M and Link N:

Link M/N = strm_clk / ls_clk - for non-CoG use cases

Link M/N = strm_clk$_{CoG}$ / ls_clk - for CoG use cases

Recommendation: In Link M/N calculation, M should be rounded up and N should be rounded down.

Restriction on clocks and number of lanes:

- Number of lanes >= INT(strm_clk * bytes per pixel / ls_clk)

Restrictions on the Virtual Channel (VC) payload size in DisplayPort MST mode

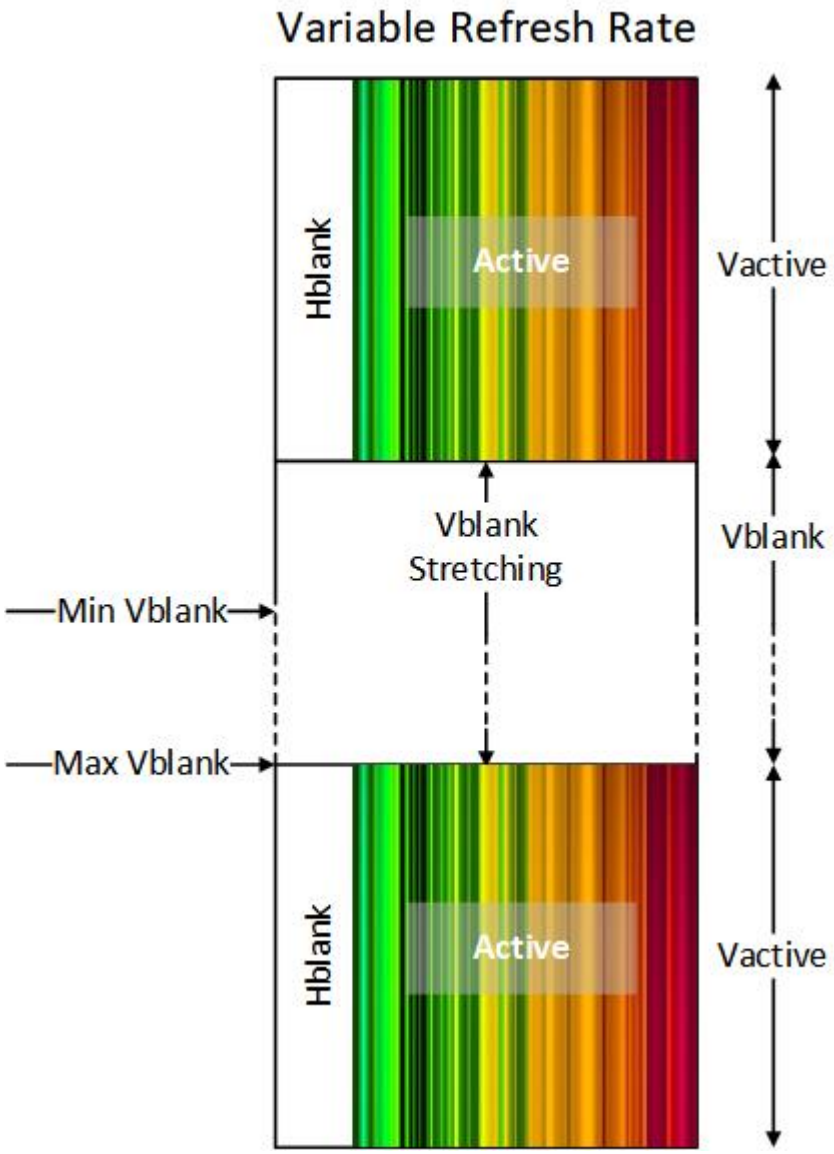
- In a x1 lane config, each pipe stream on the link must use a VC payload size that is a multiple of 4.
- In a x2 lane config, each pipe stream on the link must use a VC payload size that is a multiple of 2.
- In a x4 lane config, each pipe stream on the link must use a VC payload size that is a multiple of 1.

Transcoder VRR Function

VRR Registers
VRR Control TRANS_VRR_CTL
VRR Vmax TRANS_VRR_VMAX
VRR Vmin TRANS_VRR_VMIN
VRR Vmax-shift TRANS_VRR_VMAXSHIFT
VRR Status TRANS_VRR_STATUS
VRR Vtotal Previous TRANS_VRR_VTOTAL_PREV
VRR Flipline TRANS_VRR_FLIPLINE
VRR Status2 TRANS_VRR_STATUS2
TRANS_PUSH

VRR Overview

In the adaptive sync mode (a.k.a. VRR or Variable Refresh Rate), the display engine adapts itself to render speed by adjusting its refresh rate dynamically. Adaptive sync mode is supported both on eDP and DP. The dot clock is set to support the peak desired refresh rate for a given resolution and link clock. A minimum and maximum vertical blank (vblank) period is specified and the display hardware stretches or shrinks the actual vblank period based on the required refresh rate.



VRR Details

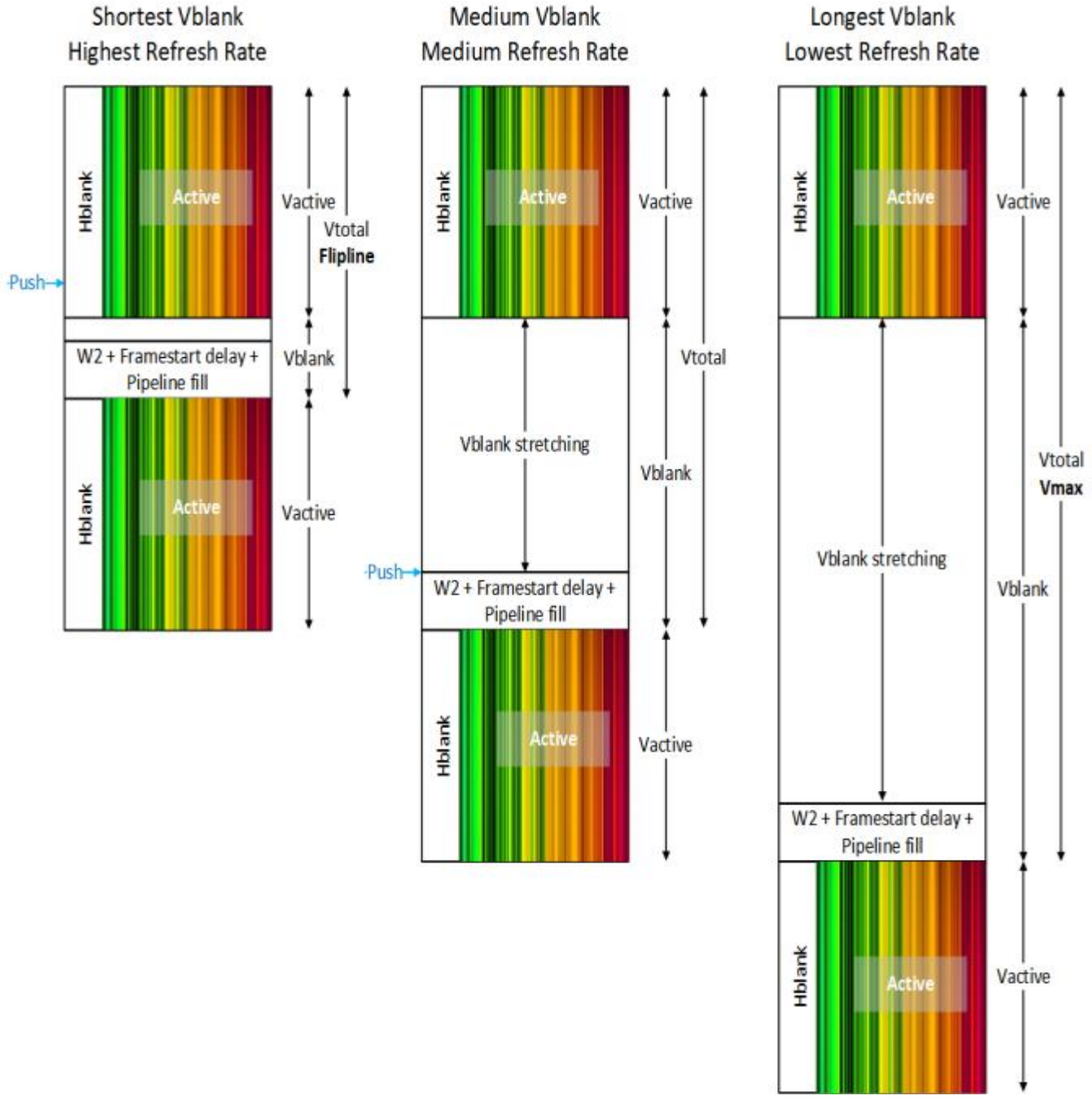
VRR stretches the beginning of vblank until the vblank needs to be terminated, then delay window 2 (W2) happens, double buffer registers are updated (sync flips complete), framestart is triggered (after a delay), the display pipeline fills, and vblank ends.

The vblank period is constrained by the registers specifying vertical total (Vtotal) lines maxium (Vmax) and minimum (Vmin). The Vtotal contains the Vblank and vertical active (Vactive) region of each display frame. The minimum Vblank must at a minimum contain the framestart delay and pipeline fill.

When there is a frame update (flip or other update), software triggers the termination of vblank by setting the transcoder Send Push bit (push bit). The start of termination may be immediate, or delayed, depending on flip line and max shift. Hardware clears the push bit when the vblank termination begins. The vblank termination is the flip decision boundary, or latest time a push will be serviced. If a push is set after flip decision boundary, it will be held until the next vblank and serviced there.

Flip Line provides a frame by frame programmable limit on the minimum Vtotal. The flip line is programmed before the push. Hardware calculates the flip line decision boundary = flip line value - pipeline fill - framestart delay. If push happened before that boundary point, the vblank termination starts at that boundary point. If push happens after that boundary point, the vblank termination starts immediately.

When software does not trigger termination, the vblank automatically terminates when the programmed vertical max (Vmax) will be reached. Hardware calculates the Vmax decision boundary = Vmax - pipeline fill - framestart delay, and starts termination at that boundary point.

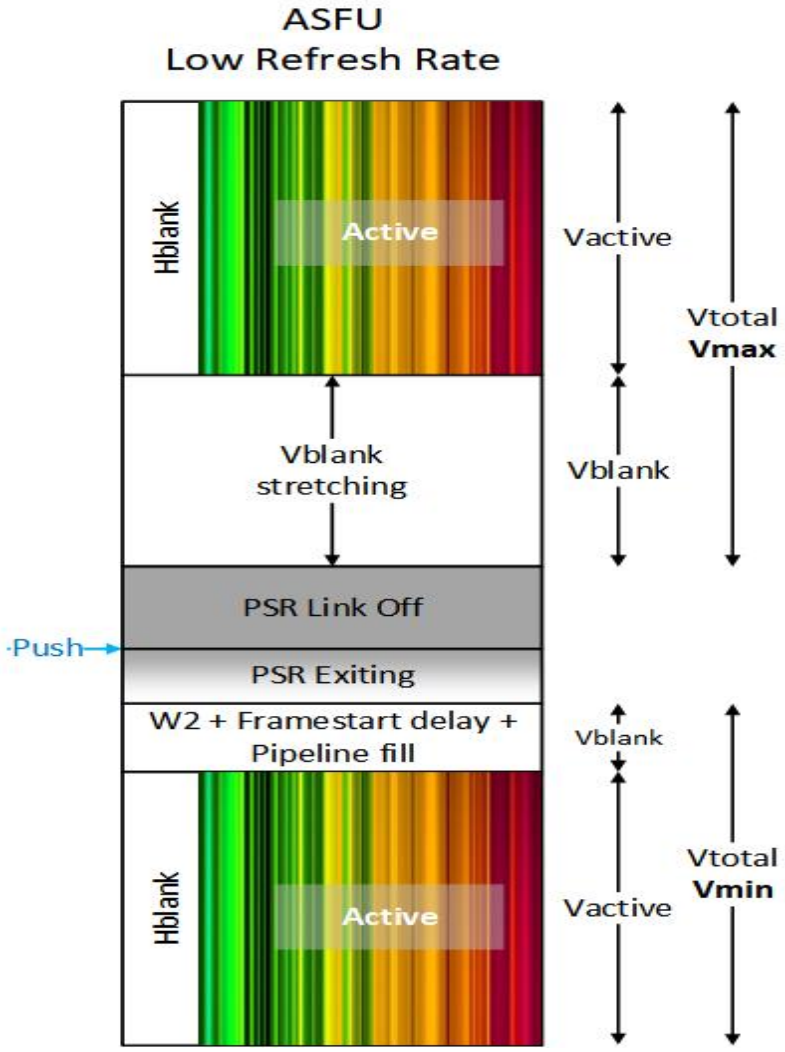


Double-Buffer Behavior

The push mode causes VRR to affect all pipe and plane double-buffered registers. As the vblank sent to the panel (timing generator vblank or undelayed vblank) stretches it delays the assertion of vblank sent to the pipe (delayed vblank), which is used as the double-buffer update for all of the pipe and planes. When VRR begins to terminate the vblank, delay window 2 (W2) happens, then the delayed vblank asserts, triggering the double buffer update before the framestart and pipeline fill. As a result, double-buffer registers will update after the push is set, or in a non-ASFU case, when vmax decision boundary is reached.

ASFU

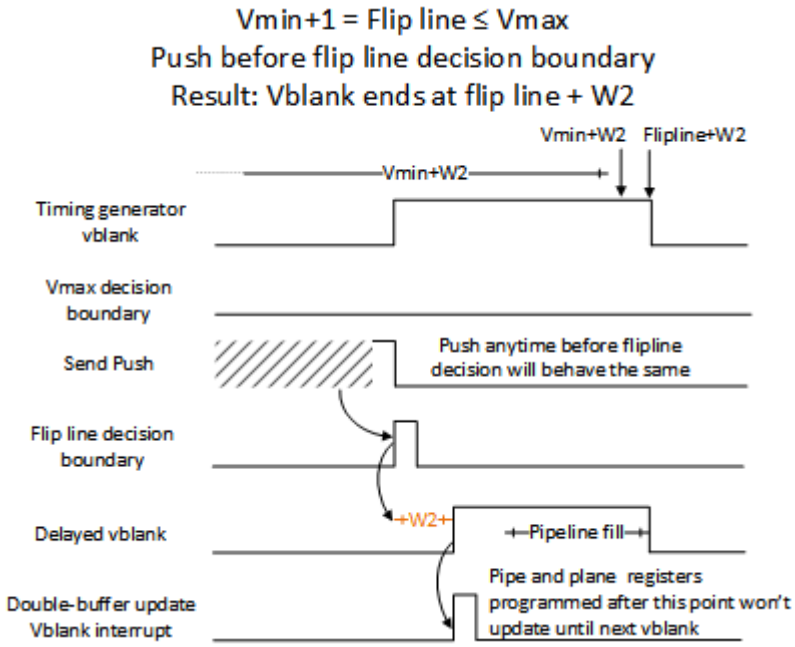
Adaptive Sync Frame Update (ASFU) changes the VRR behavior when max vertical is reached so that it enters PSR instead of terminating vblank and starting another frame. A later push then causes PSR exit with minimum vblank (flip line value ignored at PSR exit). The VRR behavior with push before Vmax decision boundary is unchanged by ASFU.



VRR Scenarios

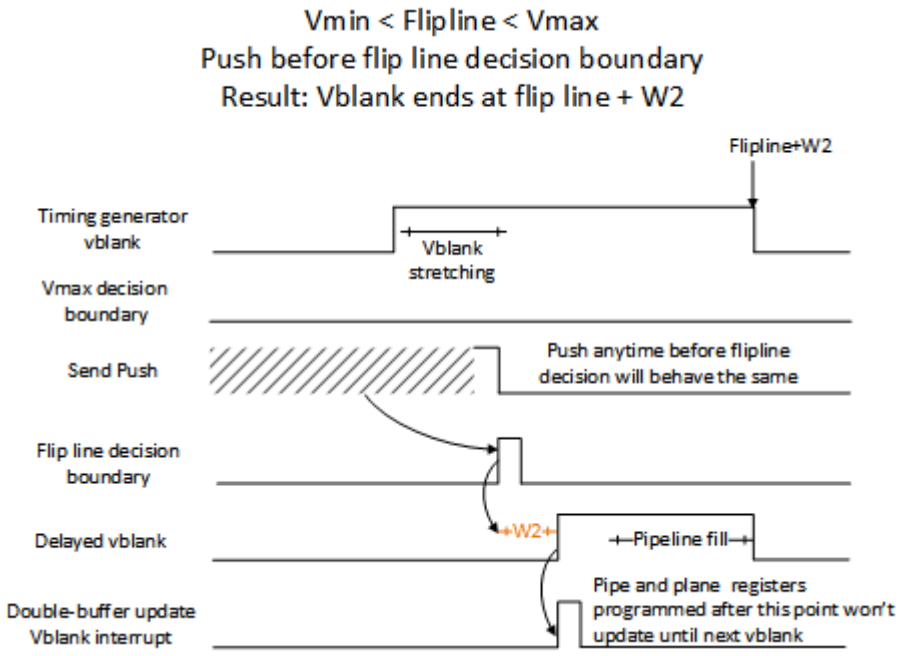
Minimum vblank

If push happens before flip line decision boundary and flip line value is Vmin+1, the vblank is minimized and the plane and pipe double-buffer registers update at the start of timing generator vblank.



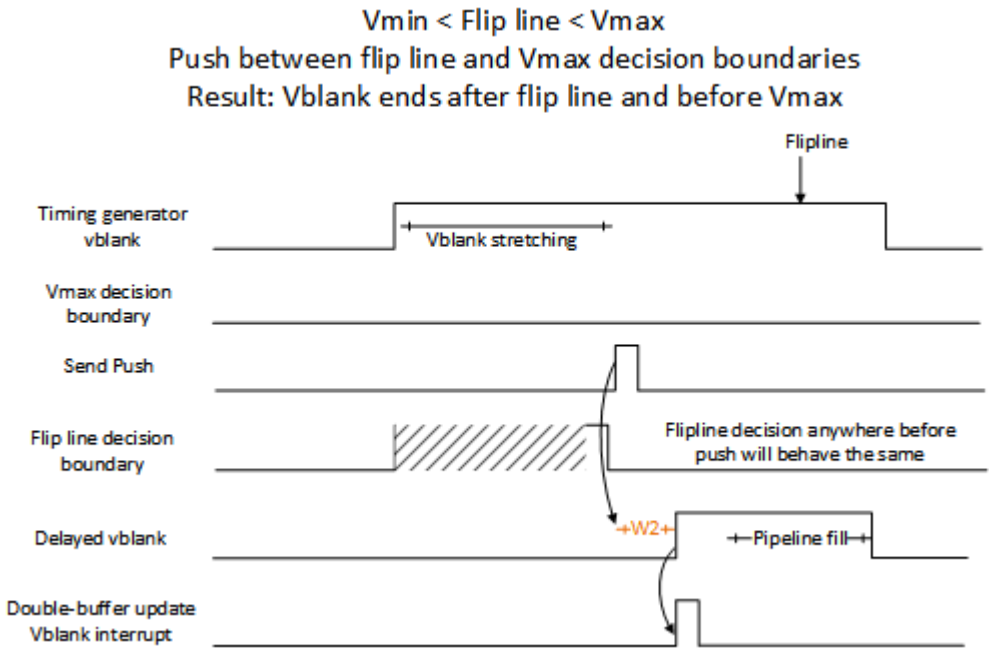
Medium vblank terminated at flip line

If push happens before flip line decision boundary and flip line value is greater than Vmin and less than or equal to Vmax, the vblank will stretch and the plane and pipe double-buffer registers update at the flip line decision boundary.



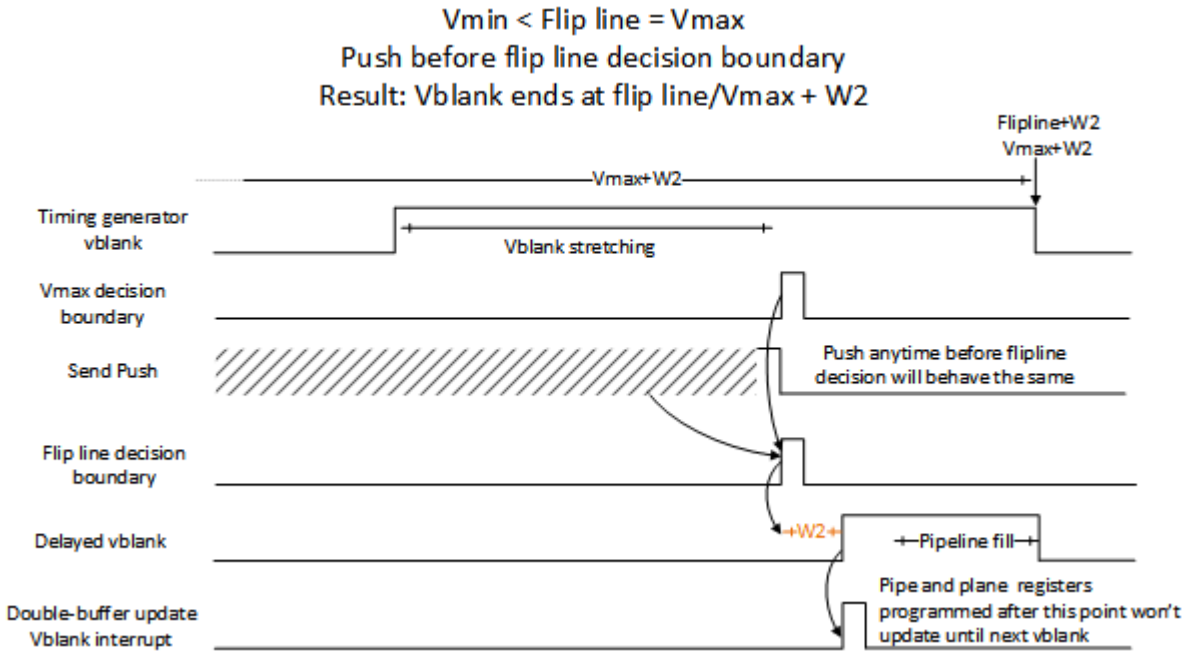
Medium vblank terminated at push

If push happens after flip line decision boundary and before Vmax decision boundary, the vblank will stretch and the plane and pipe double-buffer registers update at the push.



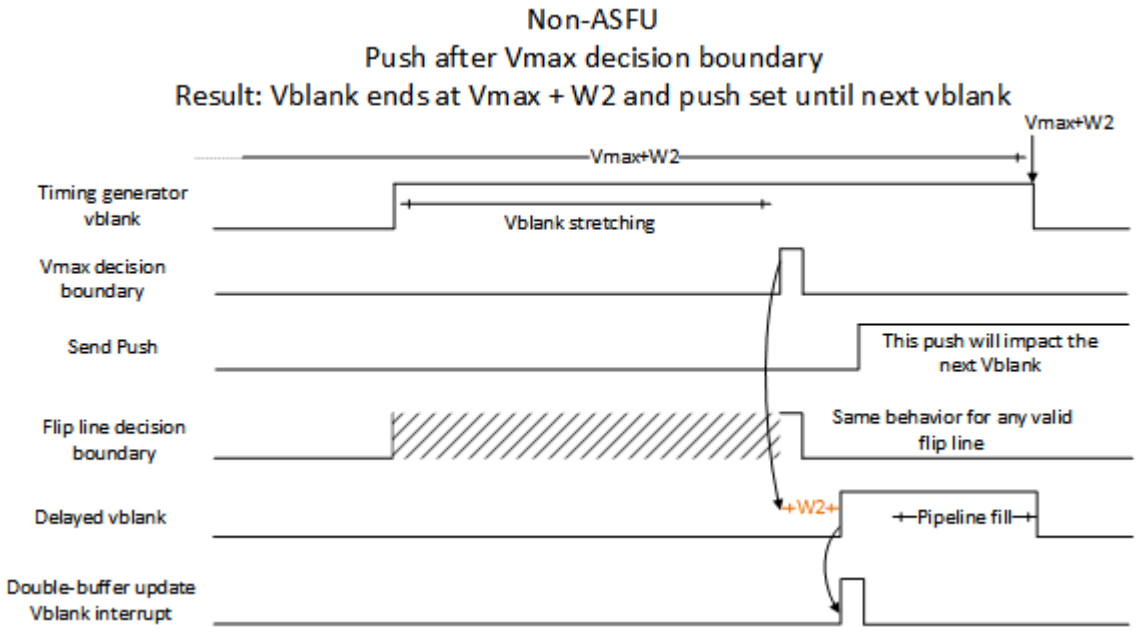
Maximum vblank terminated at flip line

If push happens before flip line decision boundary and flip line value is equal to Vmax, the vblank will stretch to the max and the plane and pipe double-buffer registers update at the Vmax and flip line decision boundary.



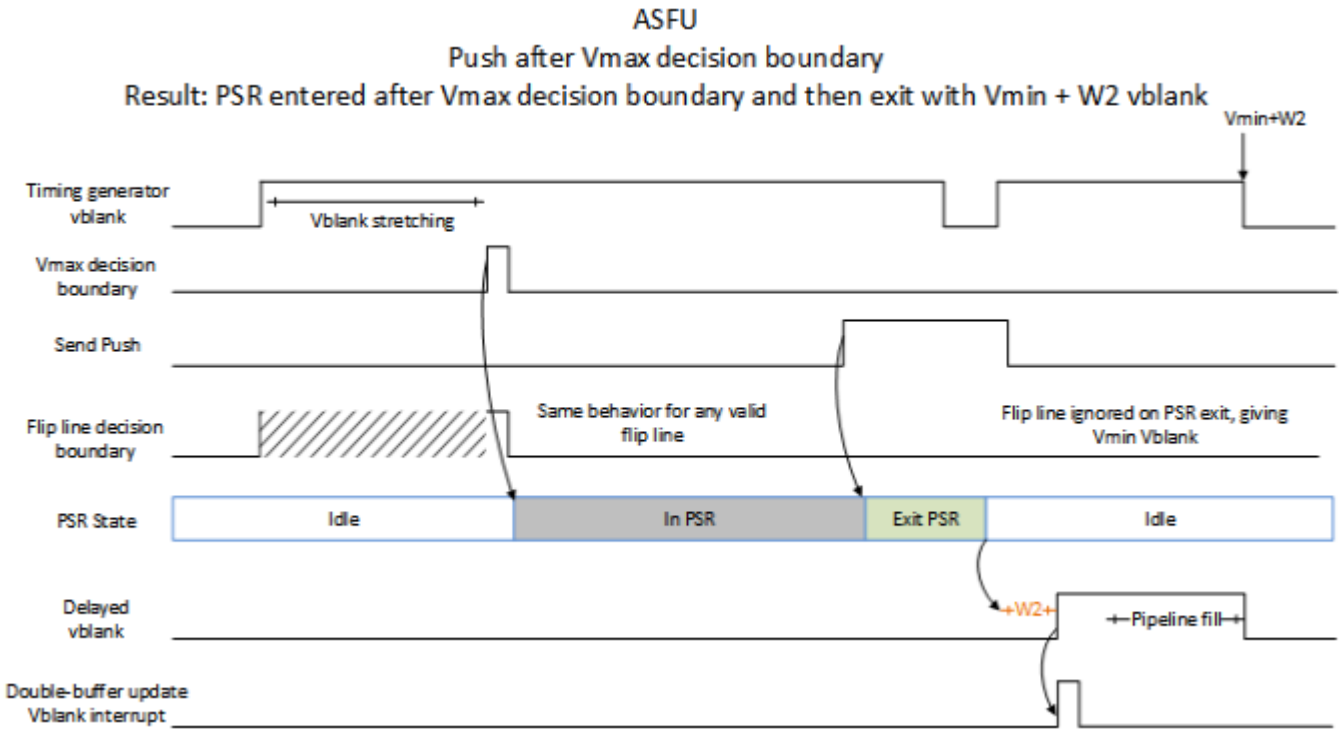
Maximum vblank terminated at Vmax

If PSR is disabled and push happens after Vmax decision boundary, the vblank will stretch and the plane and pipe double-buffer registers update at the Vmax decision boundary.



Maximum vblank enters PSR

If PSR is enabled (for ASFU) and push happens after Vmax decision boundary, the vblank will stretch and enter PSR after the Vmax decision boundary. The push then causes PSR exit with minimum vblank (flip line ignored at PSR exit) and the plane and pipe double-buffer registers update at the start of vblank.



Programming Sequence

1. Enable VRR
    - This can be done after mode set (called VRR Enable Sequence 1), or during mode set in the step that configures transcoder timings and other pipe and transcoder settings so that VRR is enabled from the first frame (called VRR Enable Sequence 2).
    1. Configure VRR timing and control registers
        - TRANS_VRR_CTL Pipeline Full Override must be set and the FrameStart to Pipeline Full LineCount programmed.
        - TRANS_PUSH Push Enable must be set.
        - If max shift, TRANS_VRR_CTL Ignore Max Shift and Flip Line Enable must be clear, else Ignore Max Shift and Flip Line Enable must be set.
    2. If adaptive sync SDP use case
        - Program TRANS_VRR_VSYNC
        - Set "Adaptive Sync SDP Enable" in VIDEO_DIP_CTL register
    3. Set TRANS_VRR_CTL VRR Enable
    4. If ASFU, enable PSR - See Panel Self Refresh chapter

2. Screen updates, repeat as needed for each update
    1. Program the double-buffer registers that need updating
        - There will be a double-buffer update at Vmax decision boundary if ASFU is not used, so care must be taken to ensure multiple resource programming does not straddle the double-buffer update point and cause non-atomic updates.
            - Options
                - Use ASFU so that PSR is entered at Vmax.
                - Align programming to happen before the Vmax decision boundary.
                - Use double buffer disable/stall mechanisms to stall double buffering until all programming is complete.
    2. If not max shift, program Flip Line value in TRANS_VRR_FLIPLINE (this can be programmed multiple times before the next step).
        - Flip Line value cannot be changed between when the push is initiated and the push is done.
    3. Set TRANS_PUSH Send Push
    4. Hardware will clear the Send Push when the double buffer update happens
    5. Poll for TRANS_PUSH Send Push cleared
3. Disable VRR
    1. If ASFU, disable PSR - See Panel Self Refresh chapter
    2. Clear TRANS_VRR_CTL VRR Enable
    3. Stop setting push
        - Push is not needed when VRR is disabling because frames will terminate automatically.
        - Setting push after VRR is disabled will cause the push to be held until later VRR enabling. That is not validated and not supported.
    4. Poll for VRR live status indicating VRR has disabled.
        - VRR live status takes one frame to change after VRR mode is disabled.
        - Depending on VMAXSHIFT, the vblank length may keep changing for several frames after VRR mode is disabled, which is interrupted by disabling the transcoder or re-enabling VRR.

- VRR double buffer update interrupt will remain active until VRR live status is de-asserted.
    1. TRANS_PUSH Push Enable can be cleared at this point or later
    2. VRR can be re-enabled by returning to step 1.
    3. If port will be disabled, continue to mode set disable sequence.
        - Hardware may be capable of transcoder disable with VRR enabled, but that is not validated and not supported.

VRR with Port Sync Mode

Port sync has multiple transcoders running synced together. The primary transcoder will send sync signals to the secondary transcoders.

Program the VRR registers the same on both transcoders when enabling VRR.

> VRR_CTL VRR Enable and TRANS_PUSH Push Enable must be set on primary and secondary transcoders. The other registers may not all be necessary for the secondary transcoders, but programming them differently is not validated and not supported.

When sending screen updates, only the primary transcoder VRR needs to be updated. Only set push on the primary transcoder and only update the flip line value on the primary transcoder.

Starting VRR from Nominal Refresh Rate

1. Driver determines the panel timings and the highest, lowest, and nominal refresh rates to support
2. Driver does mode set to the highest pixel rate and highest refresh rate timings (should be close to CVT1.2 RB), except it extends the vertical total so that the resulting refresh rate is nominal.
3. Driver enables VRR with Vmax for the lowest refresh rate and Vmin for the highest refresh rate. Vmin = the vertical total from the mode set before extending to nominal refresh rate.
4. Hardware will vary refresh rate between the Vmax and Vmin based on the timing of pushes.
5. Hardware will also vary refresh rate between the Vmax and Vmin based on the VMAXSHIFT.

1. When VRR is disabled the refresh rate will return to Vtotal (nominal).

- Steps 2 and 3 may be separated (VRR Enable Sequence 1) or combined so that VRR is enabled during the mode set (VRR Enable Sequence 2).
- The pixel rate and horizontal timings are programmed to match the highest refresh rate and do not change.

Starting VRR from Maximum Refresh Rate

1. Driver determines the panel timings and the highest, lowest, and nominal refresh rates to support.
2. Driver does mode set to the highest pixel rate and highest refresh rate timings (should be close to CVT1.2 RB).
3. Driver enables VRR with Vmax for the lowest refresh rate and Vmin for the highest refresh rate. Vmin = vertical total from the mode set.
4. Hardware will vary refresh rate between the Vmax and Vmin based on the timing of pushes and VMAXSHIFT.

1. When VRR is disabled the refresh rate will return to Vtotal (same as Vmin).
2. It will take several frames to return, depending on VMAXSHIFT.

- Steps 2 and 3 may be separated (VRR Enable Sequence 1) or combined so that VRR is enabled during the mode set (VRR Enable Sequence 2).

- The pixel rate and horizontal timings are programmed to match the highest refresh rate and do not change.

DPCD

Video timing information in the DP MSA (main stream attribute packet) data is designed for video modes where the parameters are static. For modes such as VRR that dynamically change the video timing, the main stream attribute fields cannot be used. Therefore, panel DPCD registers need to be appropriately programmed for VRR use cases.

VRR Restrictions
• Interlaced mode, DRRS, Stereo 3D, and PSR2 are incompatible with VRR.
• Vmin = window 2 + Vactive + framestart delay + pipeline fill
• Vmax must be greater than Vmin. Vmin < Vmax
• After push, software must wait for flip done before starting another push.
• PSR is supported with VRR for the Adaptive Sync Frame Update (ASFU). • ASFU is mutually exclusive with max shift. • Flip Line must always be enabled when VRR is enabled, except if max shift is enabled. • Flip Line is mutually exclusive with max shift. • Flip Line value cannot be changed between when the push is initiated and the push is done. • Flip Line value must be less than or equal to Vmax. Flip Line <= Vmax • Flip Line value must be greater than Vmin. Vmin < Flip Line.   • Flip Line value equal to Vmin is not supported and causes vblank terminated by flip line to have an extra line.

Adjusting for Delayed Vblank Window 2

If vblank is delayed to create window 2 (see Transcoder Timings section), then VRR programming needs to be adjusted to account for window 2. Hardware does not calculate window 2 in the decision boundaries. Window 2 is inserted before the delayed vblank asserts, and is preserved when vblank stretches for VRR, so it has the impact of pushing out vblank by window 2 size. The window 2 size then needs to be subtracted when programming the VRR vblank affecting registers to get the desired result.

VRR Vmin = Desired Vmin - window 2 size

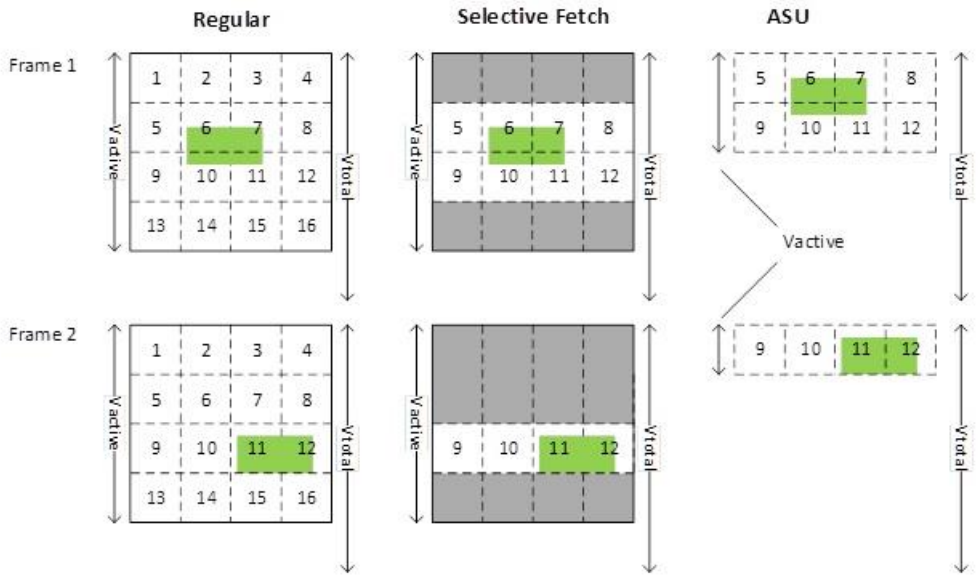
VRR Vmax = Desired Vmax - window 2 size

Flipline = Desired flipline - window 2 size

FrameStart to Pipeline Full LineCount must be calculated using the adjusted VRR Vmin.

Transcoder Asynchronous Selective Update

This feature extends ASFU to partial frame updates and adds the capability of updating one region of the display using the ASFU framework. Only one update region per frame is supported and the capability is only supported with manual tracking mode/selective fetch. A Metadata packet will transfer the co-ordinates of the update region as a proprietary secondary data packet, prior to sending the active data for the frame. All the framework available with ASFU will continue to apply to ASU. If compression is enabled, ASU mandates that the PPS sent to the panel be fixed and it should not change on the fly. Hence, DSS control registers once configured, should not be modified for partial updates. ASU does not support CoG use cases.



- Dashed lines represent DSC slice
- Solid line represents full frame size
- Greyed out area not fetched
- Green are is updated region
- Pipe scaling would require fetch of extra slices around the updated region slices in order to have the data to fill scaler filters before entering the updated slices

ASU flows

1. Mode set enable.
2. PSR enable, wait for 1 Vblank.
3. If selective update, SW updates pipe source size, Vblank start, Vactive and slice_row per frame as per SU size.
4. Set TRANS_PUSH send PUSH.
5. SW to poll for push done bit.
6. HW will consume pipe source size and slice_row per frame @VBI and Vblank start and Vactive @Vtotal.
7. For subsequent selective updates, SW to poll on scanline interrupt, 1st line and go to step (3).

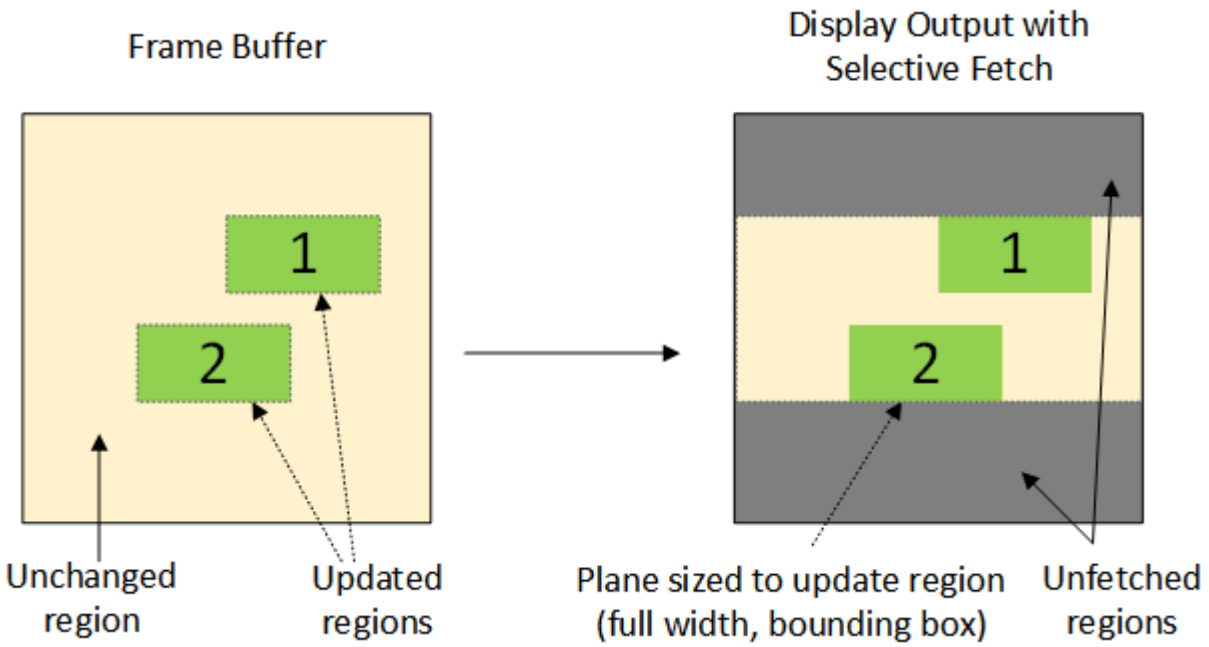
8. If PSR disable, wait for scanline interrupt, 1st line. Restore the original timing parameters.

The complete sequence is as follows.

1. Mode set enable
2. Push enable
3. DB Vactive enable
4. VRR enable
5. PSR enable, wait for 1 Vblank
6. Selective update
7. PUSH
8. Poll for PUSH done
9. SW to poll on scanline interrupt, 1st line for future selective updates
10. PSR disable
11. Restore original timing parameters
12. Program Vtotal to Vnominal.
13. VRR disable
14. DB Vactive disable
15. PUSH disable

Selective Fetch

Selective fetch is a power saving feature to reduce display engine use of memory bandwidth by only fetching (reading from memory) the updated regions of the frame buffer and sending those updated regions to a panel with a remote frame buffer and selective update capability. Software selects the amount of fetch by adjusting the size and position of the display planes to fit the updated regions in the frame, and configures panel selective update manual tracking to output that region to the panel. The result is that the bounding box around the updated regions is read from memory and delivered to the panel remote frame buffer and the other regions are not fetched and not sent to the panel, saving memory bandwidth, IO, and panel power.



Restrictions

• Supported with only full line width update region. Software must adjust the update region to full width. • Supported with only a single update region per frame. Software must create the bounding box around all regions. • Not supported with async flips. The plane size and position cannot be changed with async flips, so selective fetch cannot be used. Software must output a full frame for async flips. • Not supported with command streamer flips. Command streamer flips do not update the selective fetch plane registers without adding in extra LOAD_REGISTER_IMMEDIATE commands, which are too complicated. • Not supported with plane rotation. Software calculations to adjust update region coordinates are too complicated. • Not supported with plane or pipe scaling. Software calculations to account for extra lines of scaler filter input and adjusted scale factor and filter phase are too complicated. • Not fully supported with LACE/LDPST. The histogram accumulates across the frame and gives incorrect results when only part of the frame is updated. Software adjusts the histogram taking algorithm to get partial support.
• Not fully supported with DPST. The histogram accumulates across the frame and gives incorrect results when only part of the frame is updated. Software adjusts the histogram taking algorithm to get partial support.
• Supported with PSR2.
• Not supported with DSC.

Hardware

Hardware support of selective fetch is through the panel selective update feature with manual tracking, the planes and cursor selective fetch registers, and the full frame update triggers.

Software

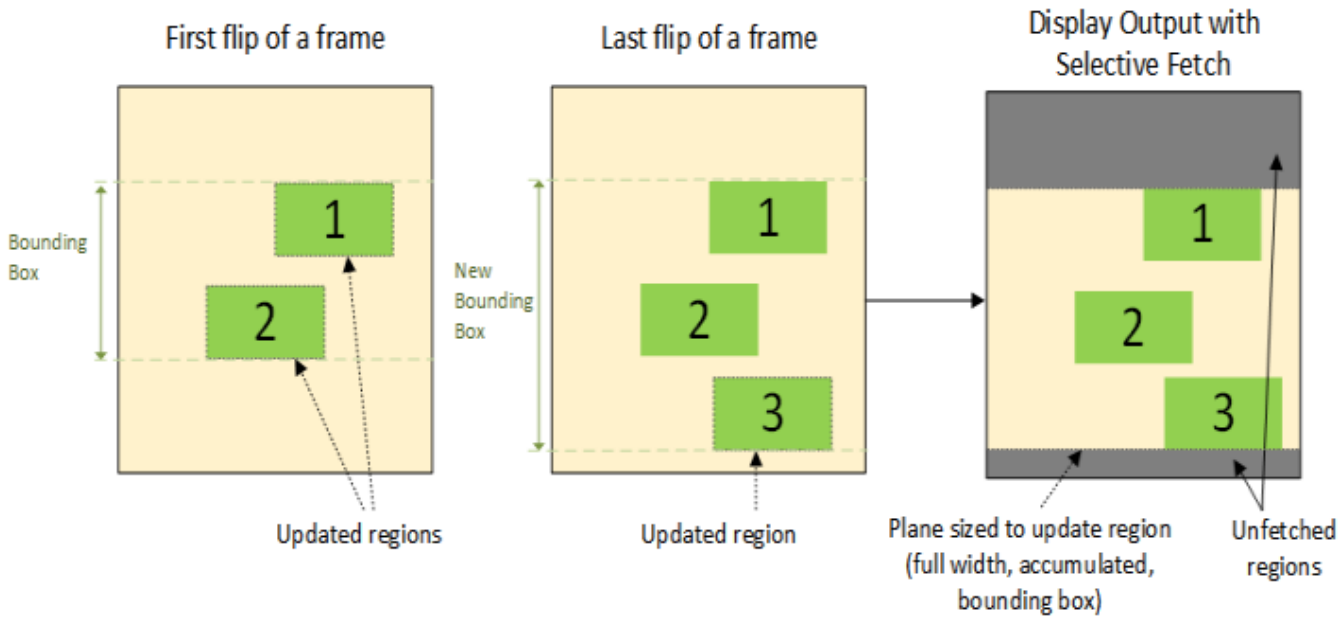
Software support of selective fetch

- Calculate the updated region
- Program the planes and cursor selective fetch registers to adjust the sizes, positions, and panning to fit the update region.
- Program the selective update manual tracking register to fit the update region.
- Program planes, cursor, and manual tracking together to atomically update.
- Trigger full frame updates when selective fetch cannot be used, such as when updating DPST image enhancement.

Update Region

Selective fetch requires software to know which areas of the screen are being updated. The update region coordinates are then used to adjust the planes and cursor to limit the fetch, and to notify the panel of which region has been selectively updated. The update region must be full width and just a single region per frame. Software must calculate the region as a bounding box to encompass all the updated sub-regions.

When there are multiple screen updates (flips) within a single frame, the update region can be different for each flip, but the final update region needs to be a bounding box that encompasses the update regions for all the flips in the frame. Hardware does not accumulate the update region across flips; it only uses the last programmed values. Software must either accumulate the update regions and program hardware with the accumulated result at each flip and reset accumulation at each frame, or trigger a full frame update when multiple flips happen within a single frame.



If there is a flip where the update region is unknown (e.g. application doesn't support this kind of tracking) or there is update that impacts the entire frame (e.g. pipe color correction programming), the update region is the full frame.

Update Region Alignment

Some features require the update region to be aligned to boundaries, then software needs to expand the update region to meet the alignment, increasing the amount of fetch.

For PSR2 selective update, the frame is divided into blocks of four scan lines each. The update region must be expanded so it aligns to the 4 line groups of transcoder vertical active.

Adjusting Plane and Cursor Size and Position

The selective fetch registers are an extra set of registers for each plane and cursor where software programs the adjusted size, position, offset (panning), and control (just the enable) to be used when doing a selective fetch. The regular (non SEL_FETCH) plane registers are used for full frame updates. Both sets must be updated with each flip, so that any frame will appear correct whether it is a selective fetch (uses the plane selective fetch registers) or a full frame fetch (uses the regular plane registers). Hardware will select between the sets of registers automatically. These registers are double-buffered and armed together with the regular plane and cursor registers.

SEL_FETCH_PLANE_CTL

PLANE_POS instances with SEL_FETCH prefix

PLANE_SIZE instances with SEL_FETCH prefix

PLANE_OFFSET instances with SEL_FETCH prefix

CUR_CTL instances with SEL_FETCH prefix

The plane selective fetch registers are programmed with the adjusted size and position of the planes, reduced down to the update region (must be the full width, accumulated, bounding box).

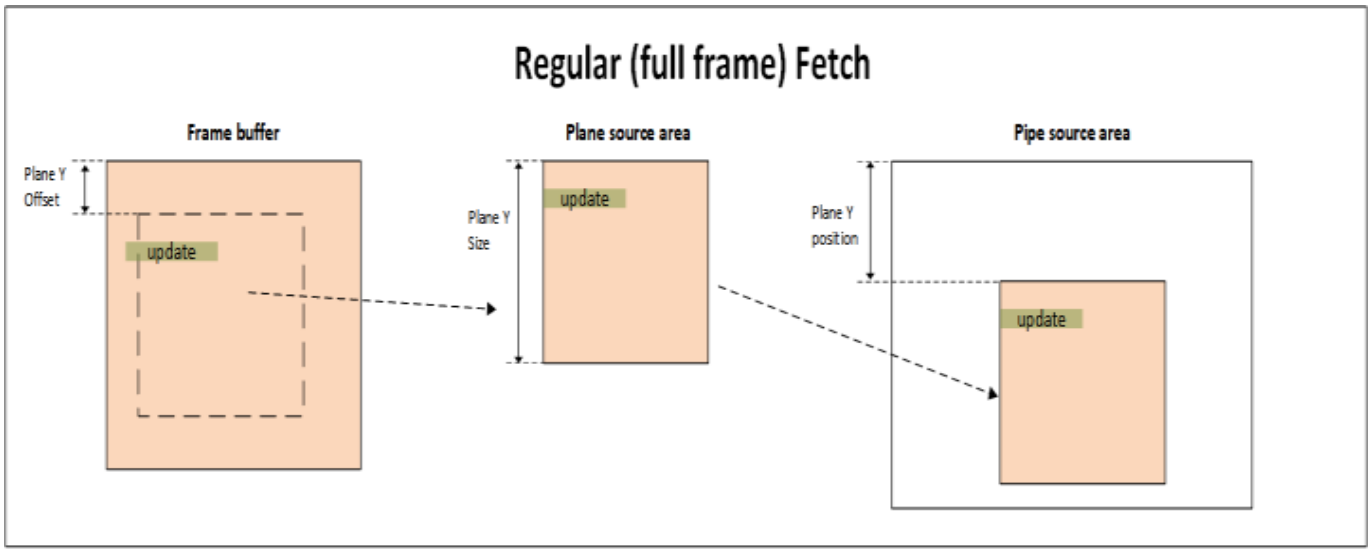
The plane adjustment calculations here are assuming the update region is relative to this plane's frame buffer. If the update is coming from the cursor or another plane, the position of cursor or other plane relative to this plane's frame buffer has to be calculated as the update region, or else the other plane and cursor changes can be simplified to do a full frame fetch if they are relatively infrequent. Other methods can be used to calculate using different starting coordinates.

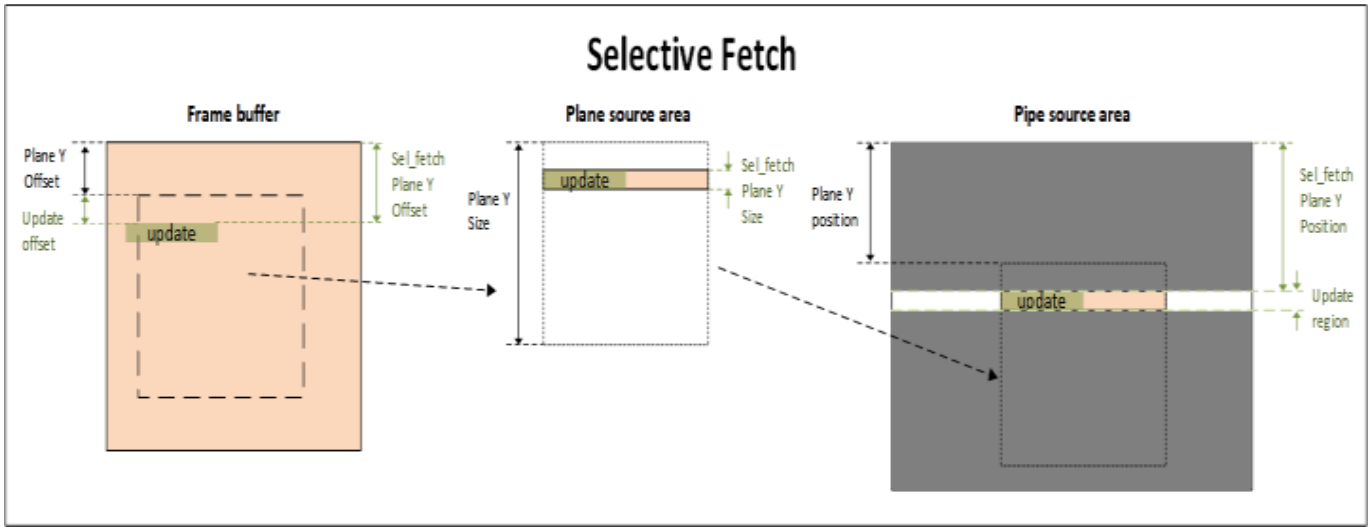
For each plane

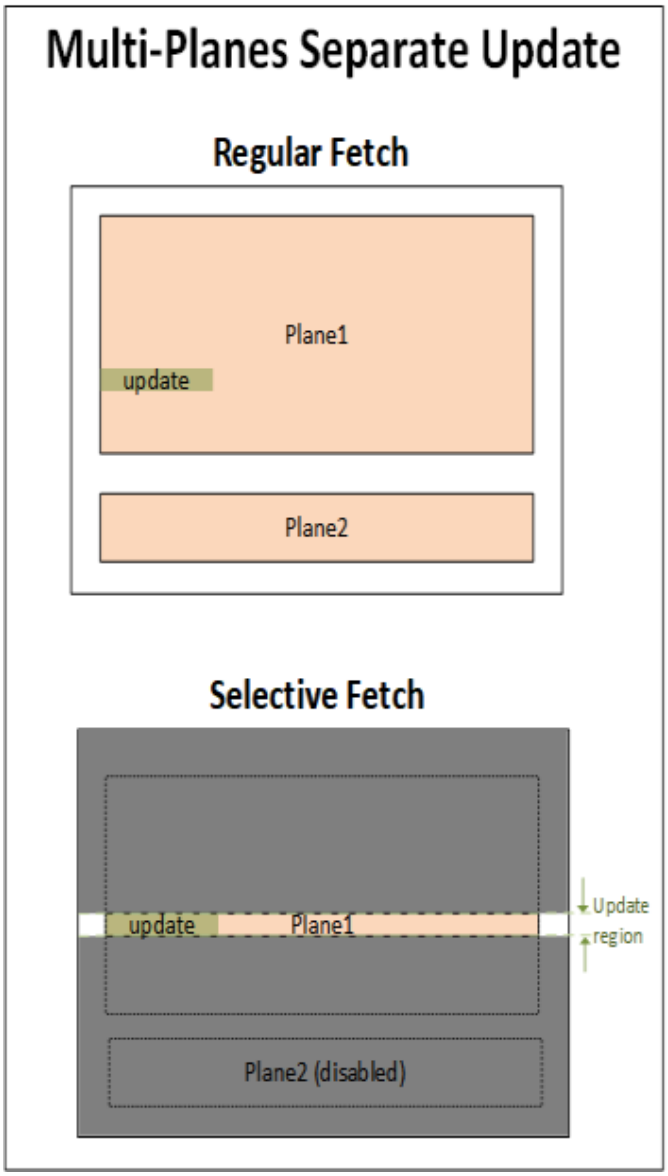
- SEL_FETCH_PLANE_OFFSET Start Y Position = first line of update region relative to start of frame buffer for this plane
- Update offset within this plane = update region vertical offset in this plane = SEL_FETCH_PLANE_OFFSET Start Y Position - PLANE_OFFSET Start Y Position
- SEL_FETCH_PLANE_SIZE Height = vertical size of update region within this plane
- SEL_FETCH_PLANE_POS Y Position = PLANE_POS Y Position + Update offset within this plane
- SEL_FETCH_PLANE_CTL Selective Fetch Plane Enable = If update region is within this plane ? Enable : Disable
    - This plane is disabled when the update region is fully outside of this plane.
- The plane width/X/horizontal values of the position, size, and offset are not changed with selective fetch, so program those fields the same in the normal plane registers and the selective fetch plane registers.

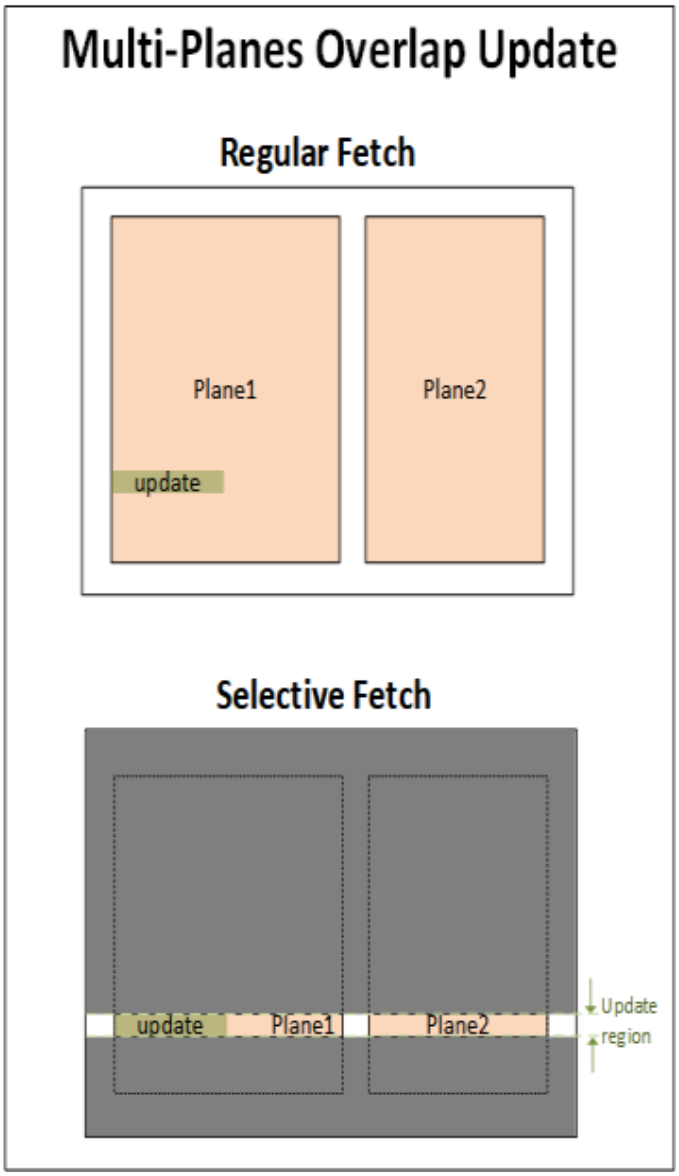
For cursor

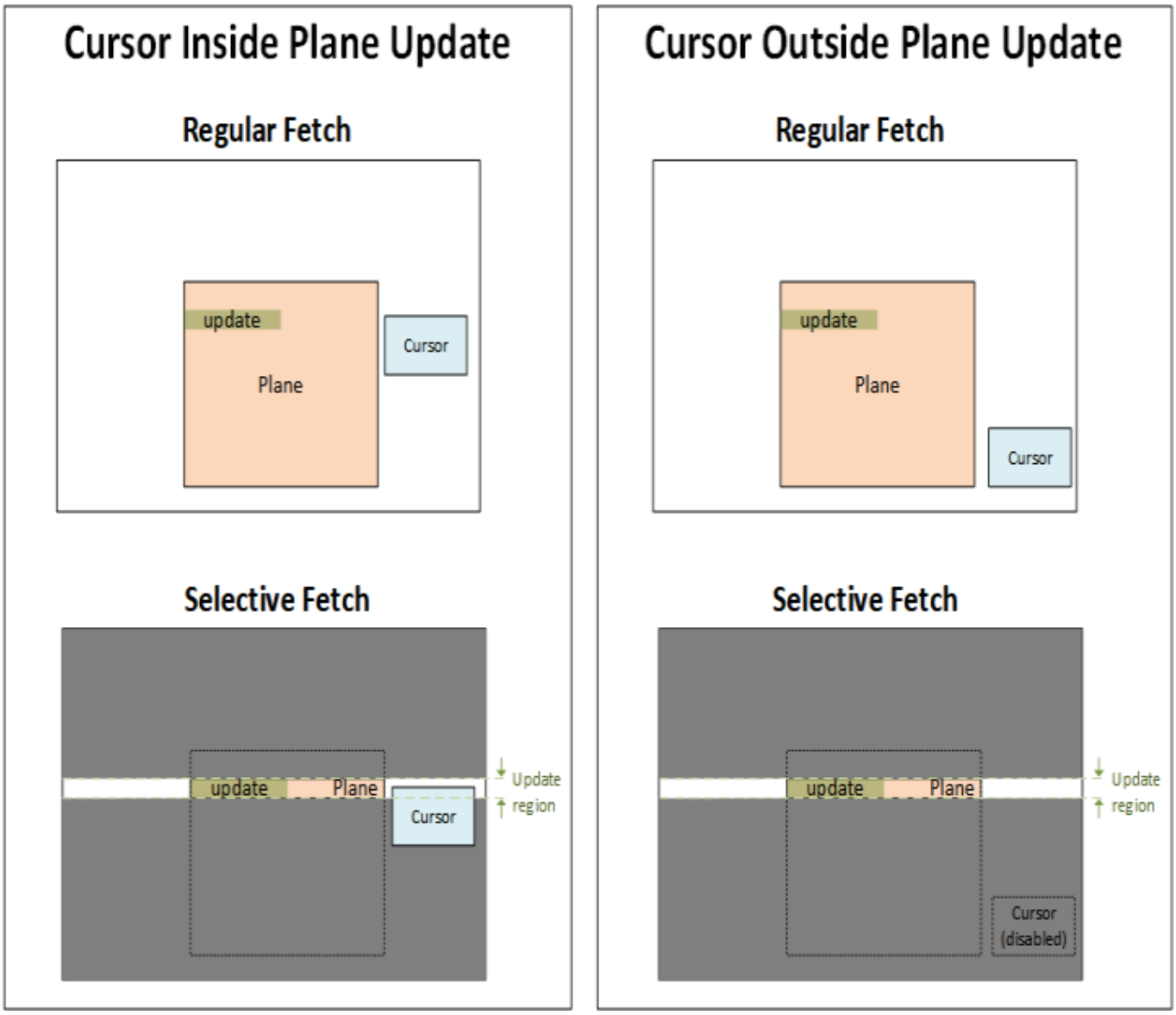
- SEL_FETCH_CUR_CTL Cursor Mode Select = If update region, translated to pipe source coordinates, overlaps this cursor ? CUR_CTL Cursor Mode Select : Disable
    - Cursor size is not adjusted. Cursor is just disabled when the update region is fully outside of cursor.
    - Program the other fields in SEL_FETCH_CUR_CTL to match CUR_CTL.

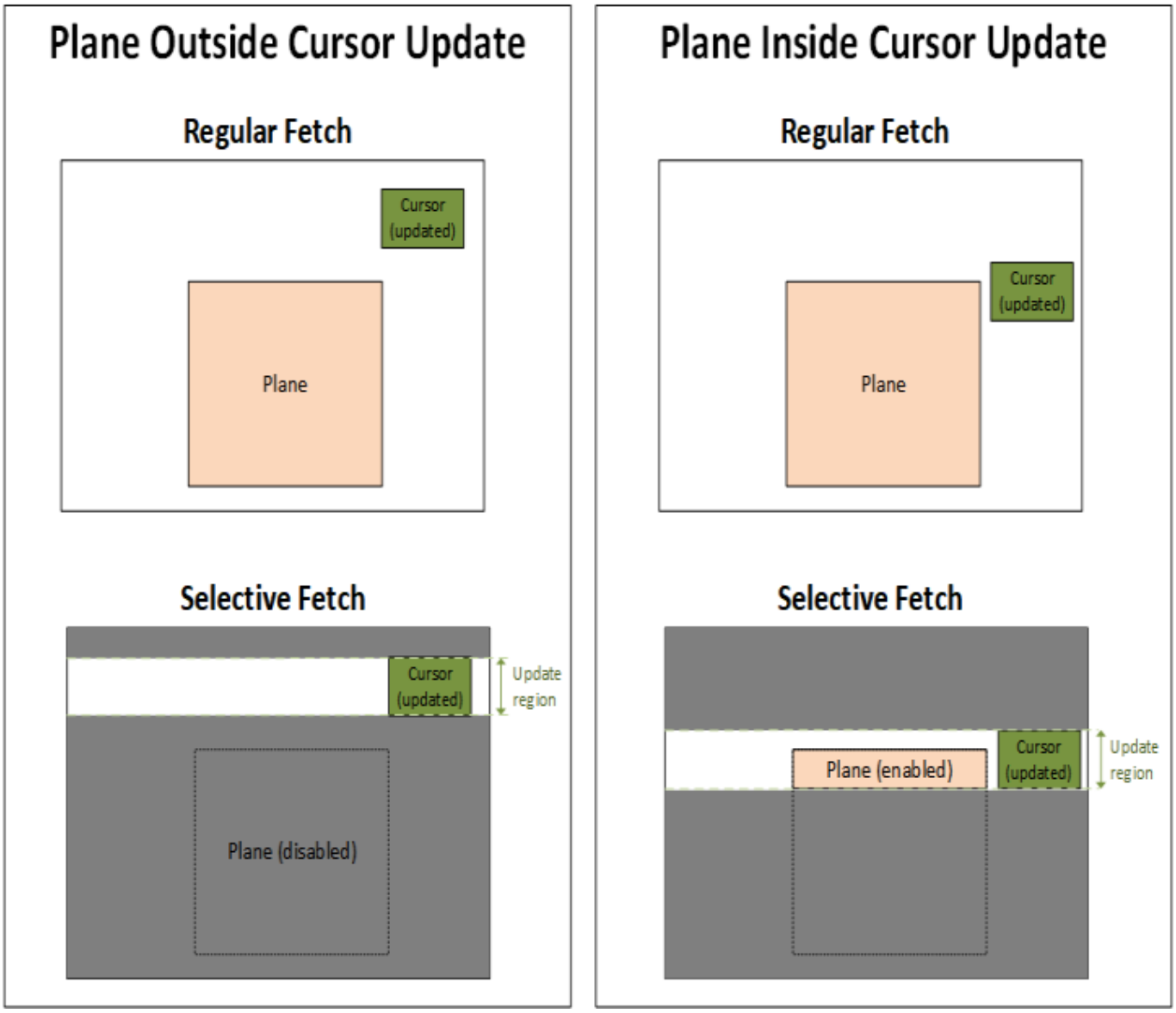

Regular (full frame) Fetch
Frame buffer
Plane Y Offset
update
Plane source area
Plane Y Size
update
Pipe source area
Plane Y position
update


Selective Fetch
Frame buffer
Plane Y Offset
Update offset
update
Sel_fetch Plane Y Offset
Plane source area
Plane Y Size
update
Sel_fetch Plane Y Size
Pipe source area
Plane Y position
update
Sel_fetch Plane Y Position
Update region


Multi-Planes Separate Update
Regular Fetch
Plane1
update
Plane2
Selective Fetch
update
Plane1
Update region
Plane2 (disabled)


Multi-Planes Overlap Update
Regular Fetch
Plane1
update
Plane2
Selective Fetch
update
Plane1
Plane2
Update region


Cursor Inside Plane Update
Regular Fetch
update
Plane
Cursor
Selective Fetch
update
Plane
Cursor
Update region
Cursor Outside Plane Update
Regular Fetch
update
Plane
Cursor
Selective Fetch
update
Plane
Cursor (disabled)
Update region



Selective Update

The selective fetch programming must be combined with panel selective update (SU) programming so that only the selective fetch update region is sent to the panel.

Calculate the update region translated to transcoder vertical active coordinates, then program **PSR2_MAN_TRK_CTL** with that SU region.

- Depending on the project generation, the SU region address is programmed with blocks of either 4 line or 1 line granularity, and with different starting and ending behavior, as noted in the manual tracking register.
- PSR2_MAN_TRK_CTL SF Continuous Full Frame and SF Single Full Frame must be 0 for the SU region to be used.

Full Frame Updates

Sometimes software needs to send a full frame update, such as when async flips happen.

- Software sets PSR2_MAN_TRK_CTL SF Single Full Frame to trigger one full frame to be sent. Hardware will clear the bit after sending the frame.
- Software sets PSR2_MAN_TRK_CTL SF Continuous Full Frame to trigger full frames to be continuously sent until software clears the bit.
- The full frame fields do not cause a frame to be sent, they only set the frame to full size. To send a frame a flip or transcoder push is still needed.

When a full frame is sent, hardware uses the regular plane and cursor registers and not the SEL_FETCH registers, and sets the update region to the full frame, ignoring the programmed SU region.

Hardware will spontaneously send full frame updates on PSR exit, without software necessarily being aware of it, so the regular plane and cursor registers must always be programmed for the full frame.

DPST Handling

The DPST histogram is incorrect when used with selective fetch. DPST will usually give more power savings than selective fetch. Software can still use DPST with selective fetch, at least some of the time, by enabling the histogram only on selected frames (chosen periodicially or algorithmically) and triggering selective fetch full frames while taking the histogram. Based on the histogram result, software can decide if the histogram needs to be taken more frequently for better power savings, with more full frames.

Selective Fetch Programming Sequence

1. DisplayPort enable mode set sequence.
2. For PSR2, configure PSR2_MAN_TRK_CTL SF Partial Frame Enable=1, and depending on project, PSR2_MAN_TRK_CTL PSR2 Manual Tracking Enable=1, then enable PSR2_CTL.
3. For each screen update
    1. If selective fetch for this frame, atomically update planes, cursor, selective fetch, and selective update registers, referring to the Adjusting Plane and Cursor Size and Position and Selective Update sections for the values.
        - All the registers must update atomically together in the same frame. Refer to the Double Buffer Control section, sequence for synchronizing double buffer updates.
            - The fields to allow double buffer stalling are located in PSR2_MAN_TRK_CTL, regular plane control, and regular cursor control registers.
        - The regular plane and cursor registers must be programmed for the full frame size because hardware can spontaneously send a full frame update.
    2. Else, send a full frame update, referring to the Full Frame Updates section.

YUV 4:2:0 Support

Overview

DP and HDMI ports support YUV 4:2:0 output.

YUV 4:2:0 has two modes of operation - bypass mode and full blend mode. Switching between full blend and bypass mode is supported at frame boundaries.

Enabling or disabling YUV 4:2:0 pipe output requires a mode-set.

Full blend mode (HDMI)

In this mode the complete display pipeline is enabled. Input data format can be YUV or RGB. Software has to ensure that the color space is YUV before post blender scaling, since the post blender scaler is only going to be doing the subsampling from YUV 4:4:4 to YUV 4:2:0 (Y & UV scaling). Post blender Y and UV scalers will be used in 5x3 configuration (horizontal taps = 5, vertical taps = 3). A packer block will pack the YUV 4:2:0 data as needed on the HDMI link.

YUV Full blend mode programming requirements (HDMI)

- A scaler should be enabled and bound to pipe.
- The pipe scaler supports downscaling to less than 1.5 (source/destination) in the horizontal direction. However, the pipe scaler can only support downscaling of 1.0 or less in the vertical direction (i.e. no downscaling) without the risk of underflows. A vertical scale factor between 1.0 and 1.5 (non-inclusive) could result in underflows.
- The pipe vertical active display size should be a multiple of 2.
- The horizontal window size must meet the HDMI restriction of a minimum of 128 Y component transfers per scan line.

Additional restrictions (HDMI)

- When YCbCr 4:2:0 pixel encoding is active, pixel repetition is not permitted.
- The transport of 4:2:0 pixels for interlaced video formats is not supported.
- The horizontal active, sync start, sync end, and total should be a multiple of 4 for 8/10/12 bpc.
-

Full blend mode (DP)

In the case of DP/eDP data will be formatted for YUV 4:2:0 before the pixel shifter. Operation is otherwise the same as for Full blend mode with HDMI.

Additional restrictions (DP)

- The transport of 4:2:0 pixels for interlaced video formats is not supported. The other HDMI additional restrictions do not apply.
- This mode is intended for use with external DP to HDMI converters, so VRR, MST, and DP port sync mode are not supported with it.

Bypass mode (HDMI)

In this mode the display pipeline is completely bypassed. YUV 4:2:0 planar data is fetched from memory, then packed according to the YUV 4:2:0 requirements of HDMI 2.0. All the display features in the pipeline including rotation, horizontal flip, blending, plane scaling, and pipe scaling, filtering, dithering, and range correction will not be supported and should not be enabled by software.

Plane size must match pipe active size when bypass mode is enabled.

Only one plane on the pipe can be enabled when bypass mode is enabled.

PIPE_MISC Pixel Rounding must be set to Truncate.

Use only plane 1 for UV and plane 6 for Y

Bypass mode (DP)

In this mode the display pipeline is completely bypassed. YUV 4:2:0 planar data is fetched from memory, then packed according to the YUV 4:2:0 requirements of DP 1.3. Operation and restrictions are otherwise the same as for Bypass mode with HDMI.

Transcoder Video Data Island Packet

Data Island Packet (DIP) is a mechanism that allows data to be sent over a digital port during blanking, according to the HDMI and DisplayPort specifications. This includes header, payload, checksum, and ECC information.

Each type of Video DIP will be sent once each frame while it is enabled.

Video DIP
VIDEO_DIP_CTL
VIDEO_DIP_DATA
VIDEO_DIP_GCP
VIDEO_DIP_ECC
VIDEO_DIP_DRM_DATA
VIDEO_DIP_DRM_ECC
VSC_EXT_SDP_CTL
VSC_EXT_SDP_CONF
VSC_EXT_SDP_HEADER
VSC_EXT_SDP_DATA

Supported DIPs

HDMI	DP	eDP
General Control Packet (GCP) Auxiliary Video Information (AVI) Source Product Description (SPD) Vendor Specific (VS) Gamut Metadata Packet (GMP)	Gamut Metadata Packet (GMP) Video Stream Configuration (VSC)	Video Stream Configuration (VSC)
DRM		
	Picture Parameter Set (PPS)	
		Gamut Metadata Packet (GMP)
	VSC Extension SDP (VSC EXT SDP)	VSC Extension SDP (VSC EXT SDP)

Construction of DIP for AVI, VS, or SPD (HDMI only):

Dword	Byte3	Byte2	Byte1	Byte0
0	Reserved	HB2	HB1	HB0
1	DB3	DB2	DB1	DB0
2	DB7	DB6	DB5	DB4
3	DB11	DB10	DB9	DB8
4	DB15	DB14	DB13	DB12
5	DB19	DB18	DB17	DB16
6	DB23	DB22	DB21	DB20
7	DB27	DB26	DB25	DB24
8 (RO)	Reserved	Reserved	Reserved	HB ECC
9 (RO)	DB ECC 3	DB ECC 2	DB ECC 1	DB ECC 0

HB = Header Byte, DB = Data Byte, RO = Read Only

Dword	Byte3	Byte2	Byte1	Byte0
0	DP: HB3 HDMI: Reserved	HB2	HB1	HB0
1	DB3	DB2	DB1	DB0
2	DB7	DB6	DB5	DB4
3	DB11	DB10	DB9	DB8
4	DB15	DB14	DB13	DB12
5	DB19	DB18	DB17	DB16
6	DB23	DB22	DB21	DB20
7	DB27	DB26	DB25	DB24

Dword	Byte3	Byte2	Byte1	Byte0
8	DB31	DB30	DB29	DB28
9 (RO)	Reserved	Reserved	Reserved	DP: Reserved HDMI: HB ECC
10 (RO)	DP: Reserved HDMI: DB ECC 3	DP: Reserved HDMI: DB ECC 2	DP: Reserved HDMI: DB ECC 1	DP: Reserved HDMI: DB ECC 0
11 (RO)	DP: HB ECC 3 HDMI: Reserved	DP: HB ECC 2 HDMI: Reserved	DP: HB ECC 1 HDMI: Reserved	DP: HB ECC 0 HDMI: Reserved
12 (RO)	DP: DB ECC 3 HDMI: Reserved	DP: DB ECC 2 HDMI: Reserved	DP: DB ECC 1 HDMI: Reserved	DP: DB ECC 0 HDMI: Reserved
13 (RO)	DP: DB ECC 7 HDMI: Reserved	DP: DB ECC 6 HDMI: Reserved	DP: DB ECC 5 HDMI: Reserved	DP: DB ECC 4 HDMI: Reserved

HB = Header Byte, DB = Data Byte, DP = DisplayPort, RO = Read Only

Construction of DIP for VSC (DisplayPort only):

Dword	Byte3	Byte2	Byte1	Byte0
0	HB3	HB2	HB1	HB0
1	DB3	DB2	DB1	DB0
2	DB7	DB6	DB5	DB4
3	DB11	DB10	DB9	DB8
4	DB15	DB14	DB13	DB12
5	DB19	DB18	DB17	DB16
6	DB23	DB22	DB21	DB20
7	DB27	DB26	DB25	DB24
8	DB31	DB30	DB29	DB28
9 (RO)	HB ECC 3	HB ECC 2	HB ECC 1	HB ECC 0
10 (RO)	DB ECC 3	DB ECC 2	DB ECC 1	DB ECC 0
11 (RO)	DB ECC 7	DB ECC 6	DB ECC 5	DB ECC 4

HB = Header Byte, DB = Data Byte, RO = Read Only

Construction of DIP for DRM (HDMI Only)				
DWord	**Byte3**	**Byte2**	**Byte1**	**Byte0**
0	RSVD	HB2	HB1	HB0
1	DB3	DB2	DB1	DB0
2	DB7	DB6	DB5	DB4
3	DB11	DB10	DB9	DB8
4	DB15	DB14	DB13	DB12
5	DB19	DB18	DB17	DB16

6	DB23	DB22	DB21	DB20
7	DB27	DB26	DB25	DB24
8 (RO)	RSVD	RSVD	RSVD	HB ECC
9 (RO)	DB ECC 3	DB ECC 2	DB ECC 1	DB ECC 0

The audio subsystem is also capable of sending Data Island Packets. These packets are programmed by the audio driver and can be read by in MMIO space via the audio control state and audio HDMI widget data island registers.

Video DIP data write sequence:

1. Wait for 1 VSync to ensure completion of any pending video DIP transmissions
2. Disable the video DIP being updated (disable VDSC before updating PPS DIP)
3. Program video DIP data buffer registers for DIP being updated
4. Enable the video DIP

For MMIO programming of dynamic DIPs, Video DIP write sequence is as follows.

1. Wait to be outside the VBlank region so the following programming does not happen during the VBlank.
2. Program video DIP data buffer registers for DIP being updated.

The video DIP data and ECC buffers may be read at any time.

DIP data buffer registers must be programmed with valid data before enabling the DIP.

Partial DIPs are never sent out while the port is enabled. Disabling the DIP at the same time it is being transferred will result in the DIP being completed before the function is disabled.

Shutting off the port on which DIP is being transmitted will result in partial transfer of DIP data. There is no need to switch off the DIP enable bit if the port transmitting DIP is disabled.

When disabling both the DIP port and DIP transmission, first disable DIP and then disable the transcoder before the transcoder clock select is set to none.

Enabling a DIP function at the same time that the DIP would have been sent out (had it already been enabled) will result in the DIP being sent on the following frame.

For HDMI, even if no DIP is enabled, a single Null DIP will be sent at the same point in the stream that DIP packets would have been sent.

Notes on DIP use cases

GMP DIP can be used to transmit the CEA infoframe SDP since it allows full packet header customization.

There is no assumption in HW on the content of the SDPs (except when PSR is enabled).

For all non PSR cases HW transmits SDP using data programmed in the registers.

The controller does not use the data internally.

PPS is a unique packet as it is 128 bytes of data. No other SDP has this length.

The GMP is what is recommended for HDR metadata.

The VSC may also need to be transmitted with HDR content as it contains colorimetry information.

VSC Extension SDP Packets

VSC_EXT_SDP_CTL

VSC_EXT_SDP_CONF

VSC_EXT_SDP_HEADER

VSC_EXT_SDP_DATA

VSC extension packets can support frame synchronous meta data up to 1K bytes. VSC extension SDP packets add capability to support HDR Dynamic Meta Data. HDR dynamic meta data is supported for full frame updates only.

Key Assumptions

- Software should always write metadata in multiples of 8 DWs i.e., enough data to form one SDP packet.
- Hardware does not support padding of incomplete SDP packets.
- DP driver software has to determine whether a particular audio mode can be supported when transmitting extension packets.
- When Audio is enabled Meta Data will be sent only in horizontal active period of vblank.
- When Audio is disabled Meta Data will be sent both in horizontal active and horizontal blanking period of vblank.
- DSB or software should program the metadata and/or set the buffer ready bits when the relevant frame has started.
- Transcoder sends metadata in vblank region as long as at least one buffer is not empty and respective buffer done indication is set.
- Software should not start a new chain in the buffer while the previous chain data is still pending.
- If the chain doesn't fit in one buffer then software can continue to write the data to second buffer in ping-pong fashion. However, packet straddling across buffers is not allowed.
- DSB can continue writing to a buffer until either the chain is complete or 1KB data is programmed.

- If software doesn't set Index auto increment bit then it should program control register with correct index value prior to each data buffer write.
- If software doesn't set Index auto increment bit and start writing from non-zero index then it is the responsibility of software to send as much data as it can fit in (256- Index) entries. Hardware will however start reading from index0 (and continue to read till the pointers match).
- Index value rolls over after reaching maximum possible value.

- Transcoder clock must be enabled while VSC_EXT_SDP_CTL is enabled or VSC_EXT_SDP_DATA is being written.

- TRANS_DDI_FUNC_CTL must be enabled while VSC_EXT_SDP_DATA is being written.

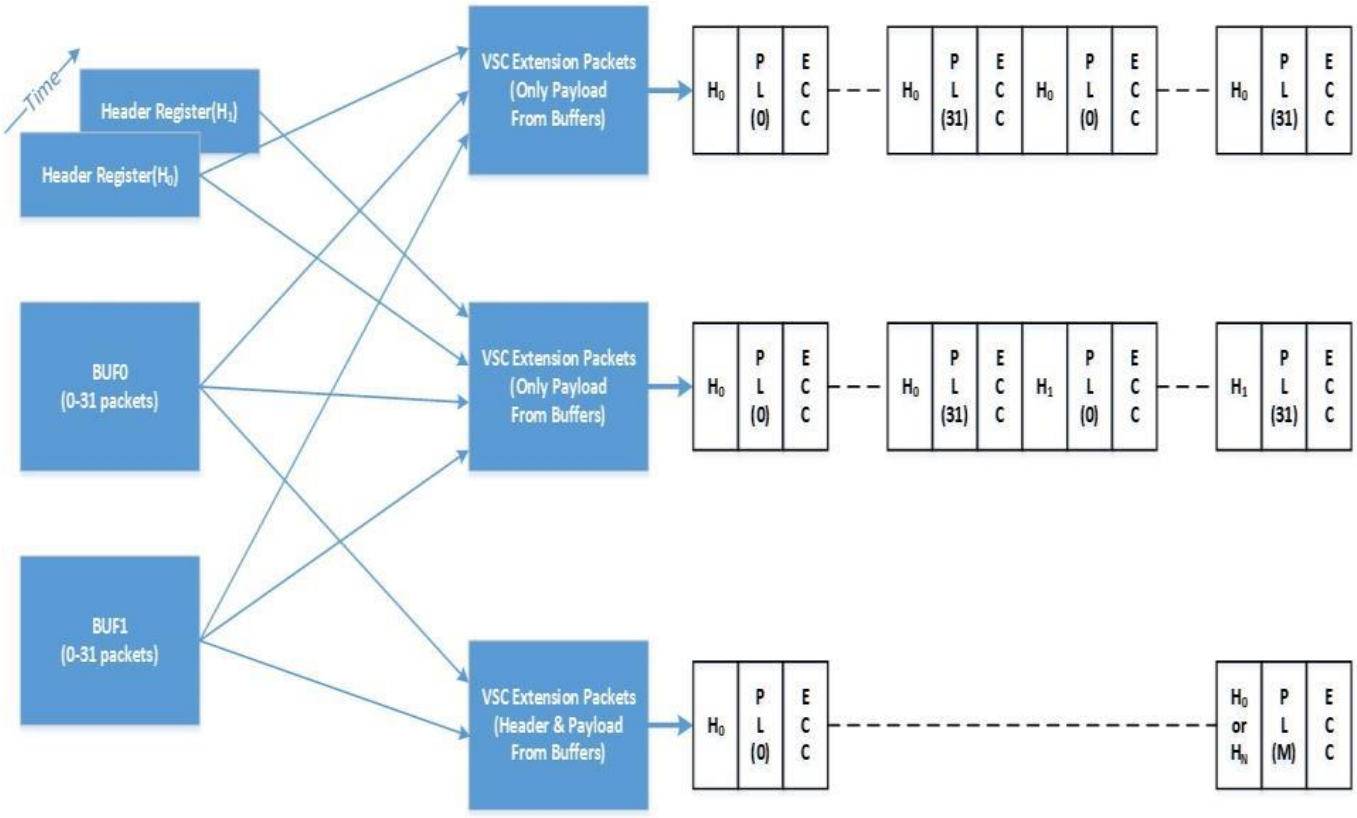
Programming sequence

There are 3 different ways of programming VSC extension packets based on the following considerations.

1. Packet header is from register and payload in buffer
2. Packet header and payload in buffer
3. Multiple VSC extension headers per frame with multiple chains per header

Type of Sequence	Max packets per Buffer
Only payload in buffer	32
Header and payload in buffer	28

The diagram below shows the final VSC extension packets from these 3 different programming methods.



Packet header is from register

In this use case, steps to program any buffer (BUF0 or BUF1) or as follows.

1. Program CTL Register - Auto Index Increment in VSC_EXT_SDP_CTL_0 [14] (VSC_EXT_SDP_CTL_1 [14]) or register bit.
2. Program BUF Register - Meta Data in VSC_EXT_SDP_DATA_0 (or VSC_EXT_SDP_DATA_1) register.
3. Set BUF ready bit - VSC EXT_SDP_CTL0 [16] (or VSC_EXT_SDP_CTL1[16])

The complete sequence is as follows.

1. Program number of packets per chain in VSC_EXT_SDP_CONF[9:0] register.
2. Program Header values in VSC_EXT_SDP_HEADER register.
3. Program BUF0
4. Program BUF1 if needed (payload > 1 KB)
5. Repeat steps 3 and 4 after poll/wait for corresponding BUF empty indication - VSC_EXT_SDP_CTL0[24] (or VSC_EXT_SDP_CTL1[24])

Packet header is from buffer

The number of packets per chain is implicit in the buffer.

In this use case, steps to program any buffer (BUF0 or BUF1) or as follows.

1. Program CTL Register - Auto Index Increment in VSC_EXT_SDP_CTL0 [14] (VSC_EXT_SDP_CTL1 [14]) or register bit.
2. Program BUF Register - Header & Meta Data in VSC_EXT_SDP_Data0 (or VSC_EXT_SDP_Data1) register.
3. Set BUF ready bit - VSC_EXT_SDP_CTL0 [16] (or VSC_EXT_SDP_CTL1[16])

To set this up, program "middle of chaining" bit in the header to '1' and enable "block mode" in VSC_EXT_SDP_DATA_MISC[22].

1. Program BUF0
2. Program BUF1 if needed (payload > 1 KB)
3. Poll/Wait for BUF0 Empty indication - VSC_EXT_SDP_CTL0 [24] .
4. Poll/Wait for BUF1 Empty indication - VSC_EXT_SDP_CTL1 [24] if BUF1 is programmed.
5. Repeat steps 1 to 4 until entire metadata is transmitted.

Multiple headers per frame

In this use case, steps to program any buffer (BUF0 or BUF1) are as follows.

1. Program CTL Register - Auto Index Increment in VSC_EXT_SDP_CTL0 [14] (VSC_EXT_SDP_CTL1 [14]) or register bit.
2. Program BUF Register - Meta Data in VSC_EXT_SDP_Data0 (or VSC_EXT_SDP_Data1) register.

3. Set BUF ready bit - VSC_EXT_SDP_CTL0 [16] (or VSC_EXT_SDP_CTL1[16]) .

The driver will program multiple chains with different number of packets and different header for each chain. As it requires programming of multiple headers per frame, driver has to poll for Chain done bit to avoid overwriting header register before hardware consumes it. Polling of Chain done instruction should be inserted after the programming of entire meta data for each chain is complete.

1. Program number of packets per chain in VSC_EXT_SDP_CONF[9:0] register.
2. Program Header Register - Header values in VSC_EXT_SDP_HEADER register.
3. Program BUF0
4. Program BUF1 if needed (payload per chain > 1 KB)
5. Poll/Wait for BUF0 Empty indication - VSC_EXT_SDP_CTL0 [24] .
6. Poll/Wait for Chain Done indication - VSC_EXT_SDP_CONF[31]
7. If BUF0 is empty but Chain is not done, poll/wait for BUF1 Empty indication. Otherwise, go to step 8.
8. Repeat steps 1 to 7 above until entire meta data is programmed.

DSB side programming considerations

If DSB is being used to program metadata with polling for buffer empty, then the DSB polling timeout (100 uS) will need to be increased, recommending >=500us if metadata programming is started in vblank, or >=frame time if metadata programming is started in vertical active.

Transcoder DDI Function

TRANS_DDI_FUNC_CTL

TRANS_MSA_MISC

TRANS_DDI_FUNC_CTL2

Panel Self Refresh

This section is about Panel Self Refresh (PSR). PSR1 at one point was named Self Refreshing Display (SRD).

PSR1 enable sequence:

- Prerequisite: The associated transcoder and port are running and Aux channel associated with this port has IO power enabled.
- Prerequisite: The associated transcoder, port and at least one plane are running.

1. Configure FBC host and render tracking. The FBC function does not need to be enabled in FBC_CTL.
2. Program Transcoder EDP VSC DIP data with a valid setting for SRD/PSR.
3. Configure and enable SRD_CTL.

PSR1 disable sequence:

- Prerequisite: The associated transcoder and port are running.

1. Disable SRD_CTL.
2. Wait for SRD_STATUS to show SRD is Idle. This will take up to one full frame time (1/refresh rate), plus SRD exit training time (max of 6ms), plus SRD aux channel handshake (max of 1.5ms).

Adaptive Sync Frame Update (ASFU) combines PSR link disable mode, VRR, and PSR single frame update. ASFU allows the vertical blank size to vary as it does for VRR, but then it will enter PSR link disable after the maximum vertical blank, and on PSR exit it can send a single frame before re-entering link disable.

ASFU enable sequence:

- Prerequisite: The associated transcoder and port are running.
- Prerequisite: The associated transcoder, port and at least one plane are running.

1. If SRD_CTL Adaptive Sync Frame Update was not disabled in step 3 of the previous ASFU disable sequence, then disable it.
2. Configure FBC host and render tracking. The FBC function does not need to be enabled in FBC_CTL.
3. Enable VRR - see Transcoder VRR Function chapter
4. Program Transcoder EDP VSC DIP data with a valid setting for SRD/PSR.
5. Configure PSR/SRD Registers and set SRD_CTL Adaptive Sync Frame Update.
6. Set masking bits for PSR to use the VRR push mode
    - PIPE_MISC[Change Mask for Register Write] to '1'
    - PIPE_MISC2[ASFU Flip exception] to '1'.
    - PIPE A: 0x420B0[11] to '1' to mask flips from being frame update events
    - PIPE B: 0x420B4[11] to '1' to mask flips from being frame update events
    - PIPE C: 0x420B8[11] to '1' to mask flips from being frame update events
    - PIPE D: 0x420BC[11] to '1' to mask flips from being frame update events
7. Enable bit 31 of SRD_CTL.

Note that ASFU can be used together with VRR flip line. See the Transcoder VRR Function page for flip line programming since that can be used independent of PSR and ASFU.

ASFU disable sequence:

- Prerequisite: The associated transcoder and port are running.
- Prerequisite: The associated transcoder, port and at least one plane are running.

1. Disable SRD_CTL bit 31.
2. Wait for SRD_STATUS to show SRD is in Idle. This will take up to one full frame time (1/refresh rate), plus SRD exit training time (max of 6ms), plus SRD aux channel handshake (max of 1.5ms).
3. Disable SRD_CTL Adaptive Sync Frame Update, or alternatively, SRD_CTL Adaptive Sync Frame Update can remain enabled until the next ASFU enable sequence is entered.
4. Disable VRR - see Transcoder VRR Function chapter

PSR2 enable sequence:

Prerequisite: The associated transcoder and port are running and Aux channel associated with this port has IO power enabled.

Prerequisite: The associated transcoder, port and at least one plane are running.

1. Configure FBC host and render tracking. The FBC function does not need to be enabled in FBC_CTL.
2. Program Transcoder EDP VSC DIP header with a valid setting for PSR2 and configure VIDEO_DIP_CTL VSC fields.
3. Enable GTC/Aux Frame Sync, if required.
4. Program Aux control register Fast Wake Sync Pulse Count to 8 pulses.
5. Configure Idle Frame, Selective Update Tracking Enable, and Y-coordinate fields, and enable PSR2_CTL.

When configuring PSR2_CTL, also configure the Fast Wake and IO Buffer Wake fields

- IO buffer wake lines = ROUNDUP(PSR2 IO wake time / total line time in microseconds)
- Fast wake lines = ROUNDUP(10us Aux Sync Pattern + 32 microseconds Aux PHY_WAKE transaction time / total line time in microseconds)
    - eDP standard calculation for Aux maximum Fast Wake from FW_SLEEP: 8us Aux preamble + 4us PHY_WAKE pattern + 20us tFW_EXIT_LATENCY
- For both fields limit the minimum to 7 lines and maximum to 12 lines

PSR2 IO wake time = 10us PHY latency + 32us eDP standard maximum Fast Wake from FW_SLEEP

- eDP standard calculation for IO maximum Fast Wake from FW_SLEEP: 11us of valid 8b/10b symbols during Aux preamble and first part of PHY_WAKE pattern + 1us tML_PHY_LOCKsu + 20us tFW_EXIT_LATENCY = 32us

This maximum of 12 lines with 42 wake means PSR2 requires total line time of at least 3.5us.

Panels with relaxed requirements may allow the wake times to be customized.

PSR2 disable sequence:

1. Program PSR2_CTL to clear PSR2 Enable.
2. Disable GTC if required.
3. Wait for PSR2_STATUS to show PSR2 is Idle. This will take up to one full frame time (1/refresh rate), plus exit training time (max of 6ms), plus aux channel handshake (max of 1.5ms), plus one more full frame time if exit began while in a capture frame.
4. If not LRR: Program PSR2_CTL to clear Selective Update Tracking Enable.

Do not enable PSR2 if (TRANS_VBLANK Vertical Blank Start - TRANS_VBLANK Vertical Blank End) < PSR2_CTL Block Count Number value in lines.

This minimum number of lines is required to give time to wake the IO when there is an update starting from the first active line.

This minimum block count of 8 lines means PSR2 requires vblank to be at least 8 lines.

Registers
SRD_CTL
SRD_STATUS
SRD_PERF_CNT
PSR_IMR
PSR_IIR
PSR_MASK
PSR_EVENT
PSR2_CTL
PSR2_MAN_TRK_CTL
PSR2_SU_STATUS
PSR2_STATUS

Transcoder WD Function

Register Links
TRANS_WD_FUNC_CTL
WD_STRIDE
WD_SURF
WD_TAIL_CFG
WD_TAIL_CFG2
WD Interrupt Bit Definition
WD_IMR
WD_IIR
WD_FRAME_STATUS
TRANS_HTOTAL
TRANS_VTOTAL
TRANS_CONF
WD_27_M
WD_27_N

WD is a transcoder for capturing display pipe pixel output to memory. It is generally intended for wireless display, but can be used for other functions.

Like the transcoders for wired ports, WD has a timing generator that initiates each frame, and it formats pipe pixel data output. Unlike wired ports, it does not pass the formatted pixel data to DDI or other port logic, but instead writes it to memory.

The WD timing generator uses TRANS_HTOTAL Horizontal Active and TRANS_VTOTAL Vertical Active to define the active pixel area that is written to memory, with the WD_SURF and WD_STRIDE specifying the memory surface attributes.

The maximum resolution is 3840x2160 60Hz, with any capture color format.

Setup

Enable WD following the Sequences for WD - Enable Sequence, and set TRANS_WD_FUNC_CTL Triggered Capture Mode Enable before or at the same time the WD Function Enable it set. This will put WD into the Triggered Capture Mode where it will wait for a capture to be triggered.

Running (after setup)

1. Trigger a frame capture by writing TRANS_WD_FUNC_CTL with the intended Frame Number and Start Trigger Frame = 1. The vertical blank will start, causing double buffered registers to update, and then capture will begin.
2. Find when capture completes by polling WD_FRAME_STATUS Frame Complete or waiting for WD Frame Complete interrupt.
    - There is no hardware timeout for the triggered capture mode. If capture does not complete within 50 milliseconds, write 1 to TRANS_WD_FUNC_CTL Stop Trigger Frame, then correct the configuration before starting another capture.
3. If using WD_FRAME_STATUS Frame Complete to find the frame completion, write 1 to that field to clear it in preparation for the next frame.
4. If another frame is needed
    a. Update any configuration that needs to be changed for the next frame; like WD_SURF, WD_STRIDE, or pipe and plane double-buffered registers.
    b. Goto 1, using an incremented Frame Number

Do not trigger a frame capture in the last frame when disabling WD. Wait for the last capture frame to finish, or hang and timeout, before disabling WD.

The frame number is used to synchronize capture and encode. Software selects the frame number and starts the capture with it. Transcoder WD includes the frame number in the pointer message sent to the encoder. The encoder reads that to align itself to the correct frame. The frame number should increment on each captured frame.

Selective Update

If only a portion of the screen is changing, WD can be configured to selectively update just that region in the capture buffer, saving power and bandwidth. VDenc will process the full buffer, or a portion, depending on its requirements.

The first capture is full frame size to establish the base image, then subsequent captures can be a selective update region.

The selective update region must be 16 line aligned to match pointer granularity.

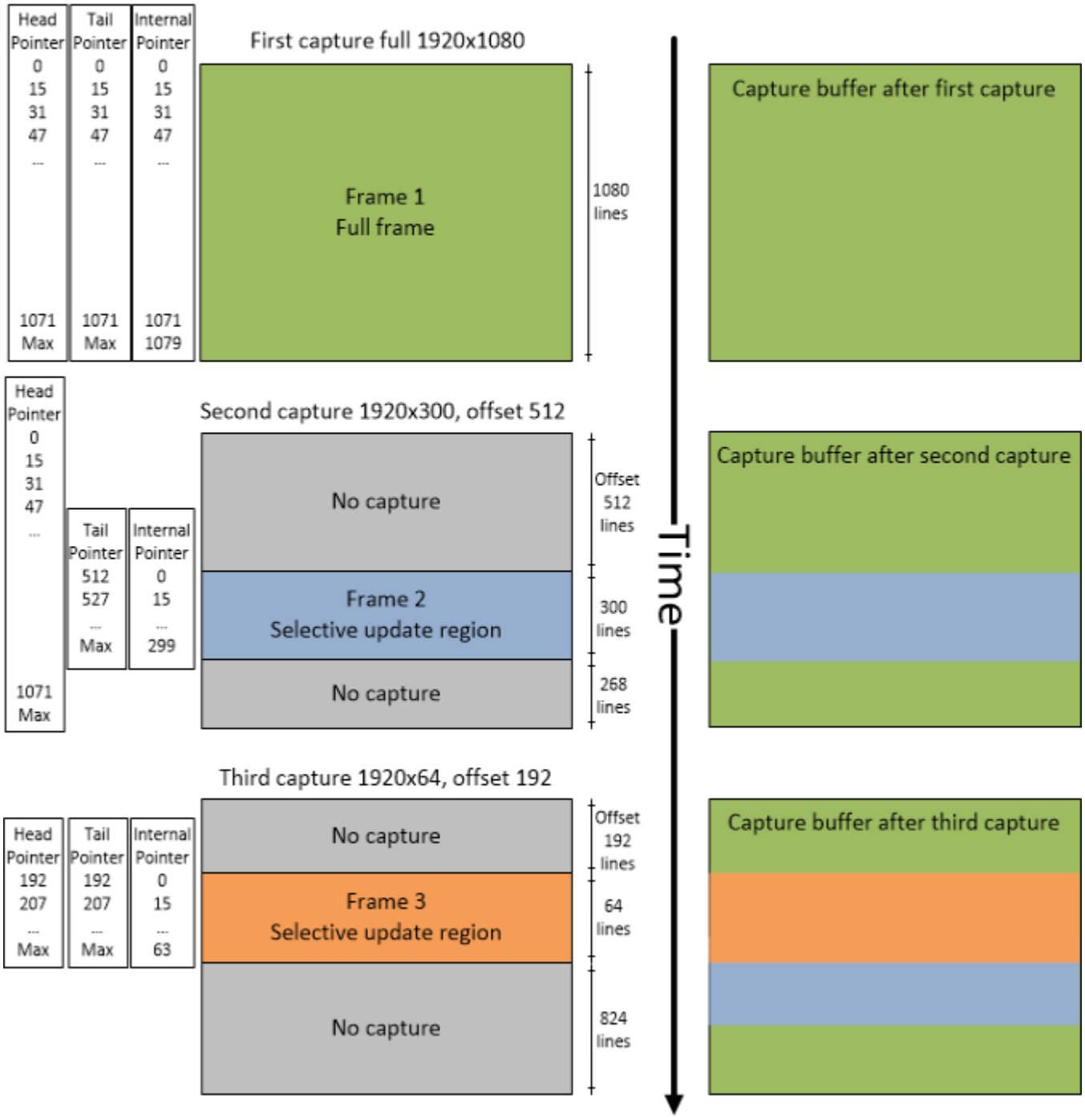
WD applies a programmable offset to the tail pointer to align the pointer to the selective update region within the full frame.

VDenc head pointer is always relative to the start of the full frame.

Programming for selective update region

- Program WD_TAIL_CFG2 Tail Pointer Offset to the number of lines offset from start of the frame to the selective update region, with 16 line granularity.
- Increase surface address to align to the start of the selective update region, with 16 line granularity.
- Reduce pipe and transcoder vertical sizes to match the size of the update region.
- Adjust plane size, position, and vertical panning to align with the update region.

Selective Update Example



Transcoder Port Sync

Feature Description

PORT SYNC is a transcoder level feature. This mode forces two or more transcoders to be in sync with one transcoder primary and one or more transcoder secondaries. In the case of DP/eDP, the primary is unaware that it is operating in Port Sync mode. Only the secondary is aware that it is operating in this mode. Hence, port sync mode is only enabled in the secondary transcoder.

Port Sync mode can be enabled with both DisplayPort SST and MST.

DP/eDP Port Sync Restrictions

1. The secondary and primary transcoders and associated ports must have identical parameters and properties.
2. They must have the same color format, link width (number of lanes enabled), resolution, refresh rate, dot clock, TU size, M and N programming, etc.
3. PSR/PSR2/ASFU/ASU would need to be disabled when port sync mode is enabled.
4. Port Sync Mode Primary Select must be programmed with a valid value when Port sync Mode is enabled.

Port Sync mode on MIPI

When setting up the DSI transcoders for dual link mode (a.k.a. Port Sync Mode) DSI Transcoder 0 is always the primary and DSI Transcoder 1 is always the secondary. The primary transcoder registers will be mirrored to the secondary transcoder when the Port Sync Mode is enabled within the TRANS_DDI_FUNC_CTL register of DSI Transcoder 0 (i.e. the **primary**). The secondary transcoder will ignore the programming within the majority of its registers when Port Sync Mode is enabled within the primary's TRANS_DDI_FUNC_CTL register. The only registers that will not be mirrored between the two transcoders are the registers that allow Software to initiate commands across the DSI Link to the Peripheral.

MIPI Port Sync Restrictions

1. Both DSI transcoders must be connected to the same PLL.

Transcoder Port Sync Support Table

Below is the complete list of primary and secondary transcoders that are supporting this feature.

The 1st column represents the primary transcoders and the 1st row represents the secondary transcoders.

Secondary ->	TC A	TC B	TC C	TC D	TC WD0	TC WD1	TC DSI0	TC DSI1
TC A	N	Y	Y	Y	N	N	N	N
TC B	Y	N	Y	Y	N	N	N	N
TC C	Y	Y	N	Y	N	N	N	N
TC D	Y	Y	Y	N	N	N	N	N
TC WD0	N	N	N	N	N	N	N	N
TC WD1	N	N	N	N	N	N	N	N
TC DSI0	N	N	N	N	N	N	N	Y
TC DSI1	N	N	N	N	N	N	N	N

Multichip Genlock

Cross-feature compatibility

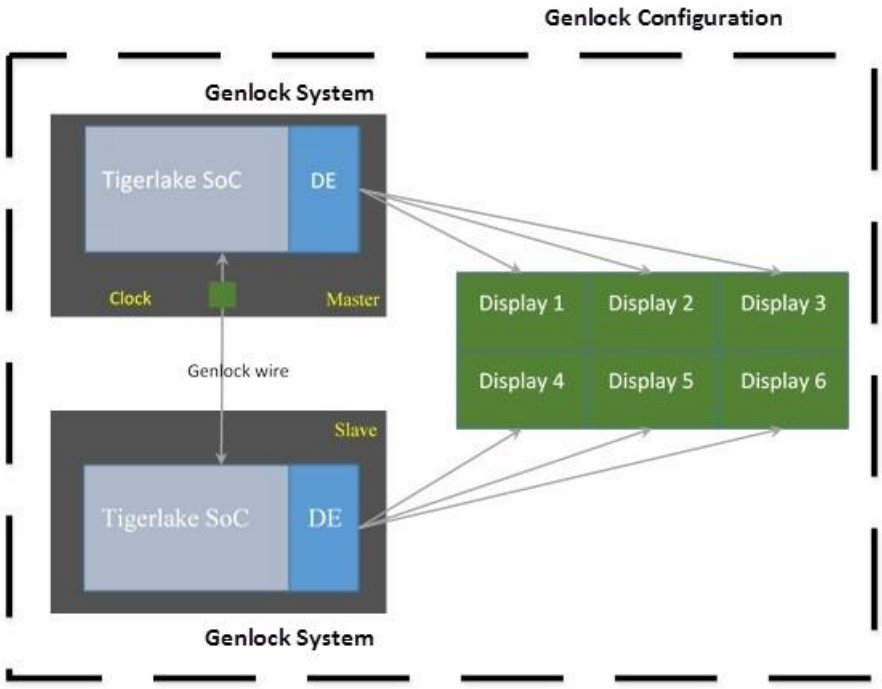
- Multichip genlock does not work with VRR (secondaries do not re-align vblank after enable mode set completes).

Description

With the multichip Genlock feature, platform/application software will synchronize frames (up to N-1 frame) across displays connected to multiple systems. The Nth frame synchronization is what HW portion of this Multi-chip Genlock will accomplish (i.e. all tiles display same scan line of same frame within +/- D pixels, as an example).

Below picture illustrates a wall of 2x3 displays connected to 2 systems in which one acts as genlock primary generating the clock and the other is genlock secondary receiving the clock from primary. DPLLs belonging to display 1, 2 and 3 are using the same PLL reference clock and that same reference clock propagates to DPLLs belonging to display 4, 5 and 6 over genlock wire as shown.

As part of multichip genlock configuration, each Genlock system below guarantees that (a) all display PLLs are running off of the same reference clock and (b) all displays receive their vertical reference (can be a separate wire or combined with reference clock) at the same time. However, flipping the right frame across displays is the responsibility of application software.



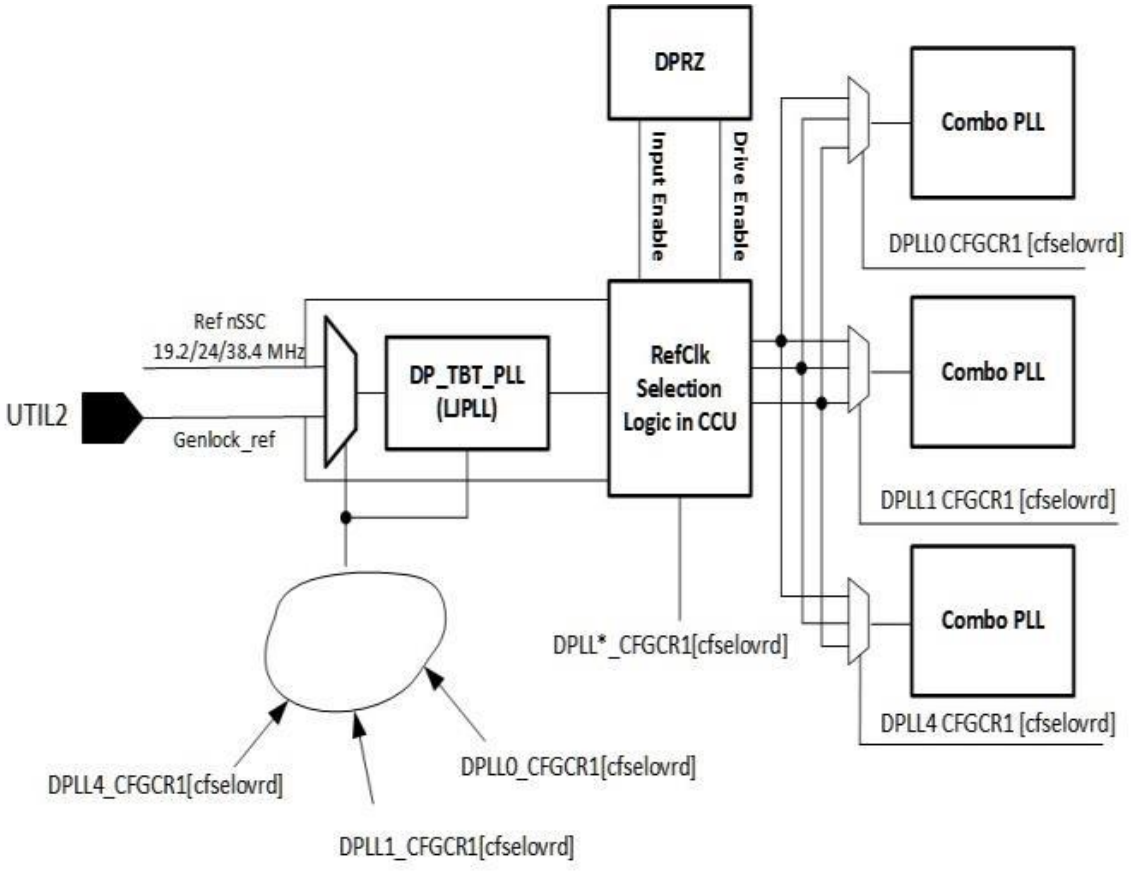
Key functional considerations

- South display is driving genlock enable and direction.
    - BIOS VBT can be configured to target unused GPIO pins.
- Within a genlock configuration, display driver will configure one transcoder as genlock primary (or secondary) and hooks up other transcoders as genlock secondaries.
- Only instances of identical platforms may be genlocked. Different types of platforms may not be mixed in a genlock topology.
    - For example, all platforms must use the same PLL reference clock frequency.
- All displays in a Multichip Genlock system must have the exact same timing requirement and support same transport protocol.
    - As an example, two panels with different 1080p timings are not allowed
- All genlock systems must train their links identically.
- Multichip Genlock is supported with both HDMI and DP (SST only) but not both on the same configuration. DP MST mode is not supported.
- Multichip genlock is not supported with software programmable Vtotal feature (LRR/LRR2).
- After genlock selection is made, chosen system enters genlock primary mode before any other system enters into genlock secondary mode.
    - Both primary and secondary ports finish training and have their transcoder enabled.
    - Primary pipe is enabled.
    - Secondary pipe is enabled.

- Runtime selection of Genlock primary or secondary requires a full mode set.
- Switching PLL reference for multichip Genlock operation requires mode set.
- Modeset across displays connected to the same genlock system can happen in multiple modeset calls.
- Genlock selection is not persistent across reboots.
- A genlock secondary will not generate vblank interrupts until after genlock sync is achieved.
- Genlock secondaries will exit first from genlock secondary mode to default mode before any genlock primary would.
- A genlock primary exits from genlock primary mode to default mode only after all genlock secondaries have switched to default mode.
- Multichip Genlock cannot be enabled at the same time port sync is enabled.
- Note: Genlock hardware may share pins with other mutually exclusive features as follows.

Genlock function	Pin	Pin functionality shared with
Vertical reference (frame sync)	Utility pin 1	MIPI DSI_DE_TE (TE1)
PLL reference clock	Utility pin 2	MIPI DSI_DE_TE2 (TE2)

The genlock reference clock generation for Combo PHY PLLs is shown below. As shown, DPLLs can optionally be filtered by TBT PLL.



Programming

Primary device

- Primary device refers to the display controller that contains the primary transcoder.
- Primary transcoder
  - Primary transcoder outputs the frame sync to be used by local or remote secondaries.
  - TRANS_DDI_FUNC_CTL2 Genlock Enable = 1
  - TRANS_DDI_FUNC_CTL2 Genlock Mode = 10 (Primary)
  - DPLL_CFGCR1[cfselovrd] = 00b (normal XTAL reference)
- Secondary transcoders
  - Secondary transcoders within the primary device are local secondaries.
  - Local secondaries receive frame sync from the primary transcoder in the same device and use the local PLL reference.
  - TRANS_DDI_FUNC_CTL2 Genlock Enable = 1
  - TRANS_DDI_FUNC_CTL2 Genlock Mode = 00 (Local Secondary)
  - DPLL_CFGCR1[cfselovrd] = 00b (normal XTAL reference)
- Pin configuration
  - Frame sync output from primary transcoder to secondary devices
    - UTIL_PIN_CTL Enable = 1
    - UTIL_PIN_CTL Direction = 0 (output)
    - UTIL_PIN_CTL Mode = 0110b (framestart/fsync)
    - UTIL_PIN_CTL Pipe Select = <pipe attached to the primary transcode>
  - Reference clock output to secondary devices
    - UTIL2_PIN_CTL Enable = 1
    - UTIL2_PIN_CTL Direction = 0 (output)

Secondary device

- Secondary device refers to any display controller that contains only secondary transcoders.
- Secondary transcoders
  - Secondary transcoders within the secondary device are remote secondaries.
  - Remote secondaries receive frame sync from the primary transcoder in the primary device and use the primary device PLL reference.
  - Secondary PLL is programmed only after receiving reference clock from the primary device.
  - TRANS_DDI_FUNC_CTL2 Genlock Enable = 1
  - TRANS_DDI_FUNC_CTL2 Genlock Mode = 01 (Remote Secondary)
  - DPLL_CFGCR1[cfselovrd] = 11b (filtered genlock reference)
- Pin configuration
  - Frame sync input from primary device
    - UTIL_PIN_CTL Enable = 1
    - UTIL_PIN_CTL Direction = 1 (input)
  - Reference clock input from primary device

- UTIL2_PIN_CTL Enable = 1
- UTIL2_PIN_CTL Direction = 1 (input)

Sequence to enable a DPLL using filtered genlock reference:

Starting from disabled TBT PLL and DPLL.

1. Configure genlock system to send the genlock reference into this chip
2. Write DPLL CFGCR1 1:0 = 11b (filtered genlock). This automatically sets TBT PLL dividers to the filtered mode and enables TBT PLL.
3. Enable TBTPLL.
4. Poll on TBT PLL lock assertion on the lock signal from CCU to display.
5. Configure DPLL dividers.
6. Enable DPLL.
7. Poll on DPLL lock assertion on the lock signal from CCU to display.

Sequence to disable the DPLL using filtered genlock reference:

1. Disable DPLL.
2. Poll on DPLL lock de-assertion on the lock signal from CCU to display.
3. Write DPLL CFGCR1 1:0 = 00b or 01b. This automatically disables TBT PLL.
4. Disable TBTPLL.
5. Poll on TBT PLL lock de-assertion on the lock signal from CCU to display

Configure genlock system to stop sending the genlock reference into this chip.

Audio

Audio Bios Programming Sequence

Audio Link Settings

When BIOS enables the audio link it must program both the audio controller and audio codec with the same BCLK and T-Mode settings before the link is enabled to ensure enumeration is successful.

In addition the audio codec "Detect Frame Sync Early" value needs to be set per-project in AUD_FREQ_CTL according to the table below.

- The audio codec link setup is performed using the **AUD_FREQ_CTL** register.
  - Program "BCLK" in bits 4:3
  - Program "T-Mode" in bits 15:14
  - Program "Detect Frame sync early" in bits 12:11
  - These settings need to be programmed **before** programming the **AUDIO_PIN_BUF_CTL** register with any analog IO settings.
- The audio controller setup is performed using controller mmio register LCTL (0xc84) and EM1 (0x1000).
  - Please consult with the Audio Controller HW team for official information.
- **For Sx and Connected Standby:**
  - GFX driver programs AUD_FREQ_CTL according to Table 2 below immediately after turning on the power well for audio (PG3).
  - GFX driver disables AUDIO_PIN_BUF_CTL bit 31 after turning off the power well for audio (PG3)

Table 1: *Audio Power Wells*

This table lists the power well containing the audio codec per-project.

Audio Power Well
PG3

Table 2: *Audio Link Settings*

The table below provides the project and stepping specific requirements for the "T-Mode" and "Detect Frame sync early" settings which must be used.

A0		B0 +	
T-Mode	Pull-In Clks	T-Mode	Pull-In Clks
4T	1	8T	0

Codec Verb Table

For each codec present on the High Definition Audio codec link, a corresponding pre-defined "Codec Verb Table" must be available to System BIOS. The Codec Verb Tables are based on codec specific information (coded datasheet) and platform design specific information (schematics) and are built by System BIOS writers and platform designers. The table contains a list of 32-bit Verbs (command and data payload) to be sent to the corresponding codec over the High Definition Audio codec link.

Below is a sample High Definition Audio Codec Verb Table for a platform with 1 codec at codec address 01h.

;Sample HIGH DEFINITION AUDIO Codec Verb Table

;Codec Address (CAd) = 02h

;Codec Vendor: XYZ Company

;VenID DevID:

dd 12345678h

;--

; FrontPanel_Supported? ; 1=Supported ,0=Not supported

db 01h

; # of Rear Panel Pin Complexes

dw 000Ch

; # of Front Panel Pin Complexes

dw 0002h

;--

; Turn on the Audio IO buffer control register to enable the 3 pin link.

Set bit 31 to 1 of AUDIO_PIN_BUF_CTL register.

Note: Set the bit 15 of register offset 0x65F10h of the Display Audio offset. Wait for the Codec to generate the wake event to the controller.

Following verbs should be send to the codec using the PIO method described in the below sections 9.1.3.

VerbTable0:

;The Vendor Node ID and the Pin Node IDs may be updated per project.

;Please refer to the **Node ID descriptions** page to get the node IDs of the vendor node and each supported Pin, and replace the node IDs in the example below.

;All the Pin widgets present in the current project should be programmed with config data as below.

;Enable the third converter and Pin first (NID 02h)

0x20278101h

```
//
// Audio Verb Table
//
// Pin Widget 0x4 - PORT DDI-A
0x20471C10,
0x20471D00,
0x20471E56,
0x20471F18,
// Pin Widget 0x6 - PORT DDI-B
0x20671C20,
0x20671D00,
0x20671E56,
0x20671F18,
// Pin Widget 0x8 - PORT DDI-C
0x20871C30,
0x20871D00,
0x20871E56,
0x20871F18,
// Pin Widget 0xA - PORT USBC1
0x20A71C40,
0x20A71D00,
0x20A71E56,
0x20A71F18,
// Pin Widget 0xB - PORT USBC2
0x20B71C50,
0x20B71D00,
0x20B71E56,
0x20B71F18
// Pin Widget 0xC - PORT USBC3
0x20C71C60,
0x20C71D00,
```

0x20C71E56,

0x20C71F18

// Pin Widget 0xD - PORT USBC4

0x20D71C70,

0x20D71D00,

0x20D71E56,

0x20D71F18

// Pin Widget 0xE - PORT USBC5

0x20E71C80,

0x20E71D00,

0x20E71E56,

0x20E71F18

// Pin Widget 0xF - PORT USBC6

0x20F71C90,

0x20F71D00,

0x20F71E56,

0x20F71F18

Codec Initialization Programming Sequence

After System BIOS has determined the presence of High Definition Audio codecs, it must follow the programming sequence below to update the codec with the correct jack information specific to the platform for the High Definition Audio driver to retrieve and use later.

There are two ways to send verbs to and receive response data from codecs over the High Definition Audio codec link: using CORB/RIRB (Command Output Ring Buffer / Response Input Ring Buffer) or using the Immediate Command/Immediate Response register pair. The sequence below uses the latter which does not require the availability of a memory buffer.

- System BIOS should ensure that the High Definition Audio HDBAR D27:F0:10-17h contains a valid address value and is enabled by setting D27:F0:04h[1].
- System BIOS must ensure program as mentioned in section 9.6, and then the Controller Reset# bit of Global Control register in memory-mapped space (HDBAR+08h[0]) is set to 1b and read back as 1b.
- When clearing this bit and setting it afterward, System BIOS must ensure that minimum link timing requirements (minimum RESET# assertion time, etc.) are met.
- **Note:** To initialize the codec System BIOS should set the display mmio register 0x65F10 bit 15 to 1b. This bit needs to be set after the controller is brought out of reset.

- The codec requires **50 ms** to come out of reset prior to subsequent operations - System BIOS should wait for the Controller to detect the wake event and recognize the Codec.
- System BIOS can poll on display mmio register 0x65F10 bit 14 "Codec Sleep State" for value 0b (codec awake) with a timeout of **50 ms** before proceeding to additional steps.

For each High Definition Audio codec present as indicated by HDBAR + 0Eh[3:0], System BIOS should perform the codec initialization as described below:

1. Read the VendorID/DeviceID pair from the attached codec.

- Verify that the ICB bit, HDBAR + 68h[0], is 0.
- Write verb 200F0000h (dword) to the IC register, HDBAR + 60h, where: '2' (bits 31:28) represents the codec address (CAd).
- Program HDBAR + 68h[1:0] to 11b to send the verb to the codec.
- Poll the ICB bit, HDBAR+68h[0] until it returns 0 indicating the verb has been sent to the codec. System BIOS may write HDBAR + 68h[0] to a 0 if the bit fails to return to 0 after a **50 us** timeout period.
- If HDBAR + 68h[1] = 1b indicating the response data from the codec is now valid, read HDBAR + 64h; the data is the VID/DID value returned by the codec.

2. Check against internal list to determine if there is a stored verb table which matches the CAd/VID/DID information.

Steps 1 and 2 are System BIOS implementation-specific steps and can be done in different ways. If a System BIOS has prior knowledge of a fixed platform/codec combination (e.g., for a System BIOS having 3 stored verb tables for 3 known codecs at known codec addresses on a known platform), a simple pre-defined codec-to-table matching can be used and steps 1 and 2 can be eliminated. For a System BIOS to support multiple codec/platform combinations, an internal match-list might be needed to match a platform/codec combination to a codec verb table.

3. If there is a match, send the entire list of verbs in the matching verb table one by one to the codec.

- Verify the ICB bit, HDBAR + 68h[0] is 0.
- Write the next verb (dword) in the table to HDBAR + 60h.
- Program HDBAR + 68h[1:0] to 11b to send the verb to codec.
- Poll the ICB bit, HDBAR + 68h[0] until it returns 0 indicating the verb has been sent to the codec. System BIOS may write HDBAR + 68h[0] to a 0 if the bit fails to return to 0 after a **50 us** timeout period.
- Repeat the steps until all the verbs in the table have been sent.

Some verbs in the table may be dependent on certain platform-specific conditions. For example, for the sample table above, the verbs for Pin Complex 7 and 8 (NID=14,16 respectively) should be sent only if the Front Panel Jacks are present and connected on the platform, which may be indicated by a software flag that is controlled by a certain GPIO pin.

Audio Programming Sequence for Link Wakeup

The following audio programming sequences are to be used for preventing the Unsolicited responses when 3 pin link is awake.

Display Audio codec generates a wake event whenever the power well (PGx - power well in which Display Audio codec HW resides) is powered up. If the link is already running, this wake event is considered as unsolicited response by audio controller in PCH. This may sometimes be considered as unnecessary URs. To avoid such URs following programming should be followed by SW. This sequence assumes communication between Audio and GFX drivers without HW to indicate Audio codec power well status.

Power down sequence:

1. Unplug event
2. PD goes low
3. PG2 low
4. Audio link should go low (there should be a communication between GFX driver and Audio Driver to turn off the link)
    1. To turn off iDisp-A link:
    2. Power off iDisp-A codec. Follow the PW2 turn off sequence.
    3. In HD Audio Controller (PCH) Program LCTL1.SPA = 0.
    4. Wait for LCTL1.CPA = 0 to indicate the link has clock stopped.
    5. Clear bit 31 to 0 of AUDIO_PIN_BUF_CTL register.

Power up sequence:

1. Plug event
2. PG2 goes high
3. Audio link to be enabled (there should be a communication between GFX driver and Audio Driver to turn on the link)
    1. To turn on iDisp-A link:
    2. In HD Audio Controller (PCH) Program LCTL1.SPA = 1.
    3. Wait for LCTL1.CPA = 1 to indicate the link has clock running.
    4. Set bit 31 to 1 of AUDIO_PIN_BUF_CTL register.
4. Check WAKESTS[2] = 1 to indicate the codec wake up occurs
5. PD bit set
6. Codec awake. Continue with codec init.

Audio Programming Sequence

The following HDMI and DisplayPort audio programming sequences are to be used when enabling or disabling audio or temporarily disabling audio during a display mode set.

- The audio codec and audio controller disable sequences must be followed prior to disabling the transcoder or port in a display mode set.
- The audio codec and controller enable sequences can be followed after the transcoder is enabled and the port is enabled and completed link training (not sending training or idle patterns if DisplayPort).
- The audio controller and audio codec sequences may be done in parallel or serial. In general, the change in ELDV/PD in the codec sequence will generate an unsolicited response to the audio controller driver to indicate that the controller sequence should start, but other mechanisms may be used. SW should make sure to set the Inactive (IA) bit to 0 before setting PD to 1.

In addition, the Audio Bandwidth Checks page provides algorithms and calculations that must be followed during the enabling sequence for audio, to ensure there is adequate bandwidth available for audio transmission.

Disabling Audio

Audio codec disable sequence:

- Disable sample fabrication
  - Set AUD_MISC_CTRL Sample_Fabrication_EN (bit 2) to "0".
- Disable timestamps
  - Set AUD_CONFIG N_value_index (bit 29) to "0" for HDMI or "1" for DisplayPort.
  - Set N_programming_enable (bit 28) to "1"
  - Set Upper_N_value and Lower_N_value (bits 27:20, 15:4) to all "0"s.
- Disable ELDV and ELD buffer
  - Set AUD_PIN_ELD_CP_VLD ELD_valid (bit 0, 4, or 8 based on which port is used) to "0"
- Wait for 2 vertical blanks
- Optional: Disable audio PD (Presence Detect)
  - Software may choose to skip this in order to keep PD enabled during a resolution switch.
  - Set AUD_PIN_ELD_CP_VLD Audio_Inactive (bit 3, 7, or 11) to "1". SW does not need to set this bit to enable Inactive bit.
  - Set AUD_PIN_ELD_CP_VLD Audio_Output_Enable (bit 2, 6, or 10) to "0".

Audio controller disable sequence:

- Program Stream ID to 0 - Verb ID 706
- Disable audio info frames transmission - Verb ID 732
- Disable Digen - Verb ID 70D
- Program the codec to D3 state if needed.
- Audio driver may stop the audio controller DMA engine at this point if needed, but not required.

Enabling Audio

Pre-Enabling Notes:

The following tables (if present) list any project specific sequence requirements to be handled before initiating the audio codec enable sequence.

Programming Notes
• No additional requirements specified.

Audio codec enable sequence:

- Enable audio Presence Detect
  - Set AUD_PIN_ELD_CP_VLD Audio_Inactive (bit 3, 7, or 11) to "0".
  - Set AUD_PIN_ELD_CP_VLD Audio_Output_Enable (bit 2, 6, or 10) to "1".
- Wait for 1 vertical blank
- Load ELD buffer and Enable ELDV
  - Set AUD_PIN_ELD_CP_VLD ELD_valid (bit 0, 4, or 8 based on which port is used) to "1".
- Enable timestamps
  - Set AUD_CONFIG N_value_index (bit 29) to "0" for HDMI or "1" for DisplayPort.
  - Set N_programming_enable (bit 28) to "0".
  - Program Upper_N_value and Lower_N_value (bits 27:20, 15:4) if a non-default N value is needed.
- Enable sample fabrication if this feature is needed
  - Set AUD_MISC_CTRL Sample_Fabrication_EN (bit 2) to "1".
- Enable sample fabrication if this feature is needed

Audio controller enable sequence:

- Program the codec to D0 state if in D3 state.
- Program Stream ID to non zero - Verb ID 706
- Enable audio info frames transmission - Verb ID 732
- Enable Digen - Verb ID 70D
- If audio controller DMA engine is stopped, audio driver can start the DMA engine at this point.

Audio Hblank early enable sequence:

- Bits 20:18 of the AUD_CONFIG_BE register have the Hblank early enable for each pipe for DP Audio. Bits 27:24 of the AUD_CONFIG_BE register have the Hblank early enable for each pipe for DP Audio. When DP Audio is enabled on that pipe, driver must set these bits when Audio presence detect is set for each of these pipe regardless of the video resolution. This programming is not needed for HDMI Audio.

- For the following cases of VDSC, 4K, 5K and 8K resolutions the following programming sequence needs to be performed by the driver before setting the Audio Presence detect of pipe for DP audio. This is not required for HDMI Audio.

  A. Driver must Program "Hblank Early Enable for Pipe X" = 1b always

  B. Determine required pixel clocks between hblank_early rise and hblank rise

  link_clks_available = {ROUDNDOWN[((h_total - h_active) * (link_clk / pixel_clk) - 28)]}

  link_clks_required = {ROUNDUP[(192000 / (refresh_rate * v_total))]} * ((48/lanes) + 2)

  If link_clks_available > link_clks_required

  - For Step C, use hblank_delta = 32

  else

  - For Step C, use hblank_delta = {ROUNDUP[(((5 / link_clk) + (5 / cdclk)) * pixel_clk)]}

  C. Determine hblank_early programming

  tu_data = (pixel_clk * vdsc_bpp * 8) / (link_clk * lanes * fec_coeff)

  tu_line = ((h_active * link_clk * fec_coeff)/(64 * pixel_clk))

  link_clks_active = (tu_line -1)*64 + tu_data

  hblank_rise =((link_clks_active + 6*{ROUNDDOWN[(link_clks_active / 250)]} + 4) * (pixel_clk/link_clk))

  hblank_early_prog = {ROUNDUP[h_active - hblank_rise + hblank_delta]} (from Step B)

  if (hblank_early_prog < 32) then

  - Program "*Hblank_start count for Pipe X*" to select value 32

  elsif (hblank_early_prog > 32 < 64) then

  - Program "*Hblank_start count for Pipe X*" to select value 64

  elsif (hblank_early_prog > 64 < 96) then

  - Program "*Hblank_start count for Pipe X*" to select value 96

  elsif (hblank_early_prog > 96) then

  - Program "*Hblank_start count for Pipe X*" to select value 128

  D. Calculate samples per line required

  samples_room = {ROUNDDOWN[(((h_total - h_active) * (link_clk / pixel_clk) - 12) / ((48/lanes) + 2))]}

if (samples_room < 3) then

- Program "*Number of samples per line for Pipe X*" to select value == samples_room

else

- Program "*Number of samples per line for Pipe X*" to select value == default value (00b = "All Samples available in buffer")

The following variables are referred to in the equations:

Variable	**Units**	**Note**
h_total	pixels	Total horizontal pixels
h_active	pixels	Active horizontal pixels
v_total	pixels	Total vertical pixels
refresh_rate	Hz	Screen refresh rate
input_bpp	bits per pixel	Bits per color "bpc" setting * 8
vdsc_bpp	bits per pixel	If not using compression, vdsc_bpp = input_bpp
fec_coeff	constant	Currently 0.972261. See "**Transcoder MN Values**".
lanes	lanes	1,2, or 4
link_clk	MHz	Link Rate / 10 (i.e. 162,270,540,810 MHz). See "**Clocks**" page per project.
pixel_clk	MHz	h_total * v_total * refresh_rate
cdclk	MHz	See "**Clocks**" page per project.
link_clks_available	link clocks	Link clocks within hblank available for audio use
link_clks_required	link clocks	Link clocks within hblank needed for audio each line

hblank_delta	pixel clocks	Pixel clocks to maintain between hblank_early_rise and hblank_rise
tu_data	link clocks	Link clocks per TU used for pixel data
tu_line	TUs	TUs required per horizontal line
link_clks_active	link clocks	Link clocks required to send active pixels per horizontal line
hblank_rise	pixel clocks	Pixel clock at which hblank will rise, accounting for FEC
hblank_early_prog	pixel clocks	Pixel clocks required to pull hblank_early back far enough from the end of h_active, to ensure there are hblank_delta pixel clocks before hblank rise.
samples_room	audio samples	Maximum number of samples which can fit within hblank, rounded down.

Audio Bandwidth Checks

This page provides the algorithms and calculations needed to ensure there is sufficient bandwidth available with the enabled video configuration for the transmission of audio.

These checks are broken down into DisplayPort and HDMI specific checks, and each section lists the criteria which determine when each check must be performed.

DisplayPort

The following section lists any required checks for audio bandwidth that are applicable in DisplayPort mode.

DP1 MST with DSC Enabled:

The following algorithm and calculation needs to be applied when DP1 MST + DSC + Audio is enabled.

Algorithm

1. Determine max_output_bpp that is supported per the current lanes, link rate, and video mode.
   - Bspec provides this calculation on the Display DSC page under “Programming Considerations and Restrictions”
2. GFX driver prunes all audio modes from the ELD except minimum audio (48KHz 2ch).
3. Check if minimum audio (48KHz 2ch) can work with max_output_bpp from step 1.

- Use Calculation 1 below, with output_bpp = max_output_bpp, aud_freq_khz = 48, aud_chan = 2, and rest of video settings.
    - IF YES -> Proceed to Step 4
    - IF NO -> Disable DSC

4. Determine the min_output_bpp required to support minimum audio (48KHz 2ch).

- For each vdsc_bpp in [8 to max_output_bpp]:
    - Use Calculation 1 below, with output_bpp = vdsc_bpp[i], aud_freq_khz = 48, aud_chan = 2, and rest of video settings.
    - If audio_supported = True, min_output_bpp = vdsc_bpp[i], loop can be stopped.
    - If audio_supported = False, continue loop with next vdsc_bpp[i+1].

5. Enable DSC using at least min_output_bpp from Step 4.

Calculation 1

Inputs:

link_rate_ghz

lanes

pixel_clk_mhz

output_bpp

hactive_pixels

hblank_pixels

aud_freq_khz

aud_chan

Constants:

mtp_size_clks = 64

link_overhead = 0.03

hblank_bytes_avail_overhead = 48

hblank_bytes_req_overhead = 56

General Calc:

```
link_clk_mhz                = (link_rate_ghz / 10) * 1000
line_freq_khz               = (pixel_clk_mhz / (hactive_pixels + hblank_pixels)) * 1000
pixel_bw_gbps               = (pixel_clk_mhz * (output_bpp / 8)) / 1000
link_bw_gbps                = ((link_clk_mhz * lanes) * (1-link_overhead)) / 1000
mtp_size_ns                 = (mtp_size_clks / link_clk_mhz) * 1000
hblank_size_ns              = (hblank_pixels / pixel_clk_mhz) * 1000
vc_slots          = CEIL[ 64 * (pixel_bw_gbps / link_bw_gbps)]
aud_samples_per_line        = CEIL[ aud_freq_khz / line_freq_khz ] + 1
```

Worst Case Hblank vs Audio BW:

hblank_reduced_ns = hblank_size_ns - (((64 - vc_slots) / link_clk_mhz)*1000)

hblank_slots_full_mtps = FLOOR[hblank_reduced_ns / mtp_size_ns] * vc_slots

hblank_slots_partial_mtps = MIN[CEIL[(MOD[hblank_reduced_ns / mtp_size_ns] * link_clk_mhz) / 1000] , vc_slots]

hblank_slots = hblank_slots_full_mtps + hblank_slots_partial_mtps

```
hblank_bytes_available          = (hblank_slots * lanes) - hblank_bytes_avail_overhead
IF (aud_chan > 2):
     hblank_bytes_required      = (aud_samples_per_line*10 + 4)*4
     + hblank_bytes_req_overhead
ELSE:
     hblank_bytes_required      = (CEIL[aud_samples_per_line/2]*5 + 4)*4
     + hblank_bytes_req_overhead
```

audio_supported = IF hblank_bytes_available > hblank_bytes_required:
YES ELSE: **NO**

HDMI

The following section lists any required checks for audio bandwidth that are applicable in HDMI mode.

Currently no audio bandwidth checks are required.

Audio Silent Stream Programming Sequence

The following sequence must be followed by the Display Audio Codec driver to enable/disable the silent stream. This should be part of stream enabling and disabling sequence as described in the Audio programming sequence page here .

Silent Stream disable flow:

1. Program channel index = 0xF, stream id = 0xF through 0x706 verb.
2. Wait for 100us
3. Program channel index = 0xF, stream id = 0 through 0x706 verb

Silent Stream enable flow:

1. Program channel index = 0xF, stream id = DMA stream id through 0x706 verb.
2. Wait for 100us
3. Program channel index = 0x0, stream id = DMA stream id through 0x706 verb

Audio stream playback disable sequence:

1. Program channel index = 0xF through 0x706 verb.
2. Program Stream ID to 0 - Verb ID 706

3. Disable audio info frames transmission - Verb ID 732
4. Disable Digen - Verb ID 70D
5. Program the codec to D3 state if needed.
6. Audio driver may stop the audio controller DMA engine at this point if needed, but not required.

Audio stream playback enable sequence:

1. Program the codec to D0 state if in D3 state.
2. Program Stream ID to non zero - Verb ID 706
3. Program channel index = "actual value" through 0x706 verb.
4. Enable audio info frames transmission - Verb ID 732
5. Enable Digen - Verb ID 70D
6. If audio controller DMA engine is stopped, audio driver can start the DMA engine at this point.

Audio Codec driver must ensure that there are no stream parameters that get modified when silent stream is enabled. Any stream parameters update can be done only after silent stream is disabled as mentioend above.

Audio Configuration

Registers
AUD_CONFIG
AUD_CONFIG_2
AUD_CONFIG_BE
AUD_MISC_CTRL
AUD_VID_DID
AUD_RID
AUD_M_CTS_ENABLE
Audio Power State Format
AUD_PWRST
AUD_EDID_DATA
AUD_FREQ_CNTRL
AUD_INFOFR
AUD_PIN_PIPE_CONN_ENTRY_LNGTH
AUD_PIPE_CONN_SEL_CTRL
AUD_DIP_ELD_CTRL_ST
AUD_PIN_ELD_CP_VLD

DisplayPort Transport

There is one instance of these registers per each DDI.

DP_TP_CTL

DP_TP_STATUS

Digital Display Interface

Combo PHY DDI Buffer

Overview

This section applies to the DDIs going to the combo PHY. DDIs going to FIA thunderbolt/typeC have separate programming.

Registers

DDI_BUF_CTL
PHY_MISC
PORT_COMP_DW0
PORT_COMP_DW1
PORT_COMP_DW3
PORT_COMP_DW8
PORT_COMP_DW9
PORT_COMP_DW10
PORT_CL_DW5
PORT_CL_DW10
PORT_CL_DW12
PORT_CL_DW15
PORT_CL_DW16
PORT_TX_DW1
PORT_TX_DW2
PORT_TX_DW4
PORT_TX_DW5
PORT_TX_DW6
PORT_TX_DW7
PORT_TX_DW8
PORT_PCS_DW1
PORT_PCS_DW9

The group access address can be used to simultaneously write the same value to a register that has instances in all 4 lanes. This is only for use when the same exact register value is applied to all 4 lanes. Reads using a port group address usually cannot return correct data. For read/modify/write to a group, the read should be to one of the lane addresses, then the write to the group address.

Combo PHY Comp Sources

Certain PHYs are connected to compensation resistors and act as compensation sources. Other PHYs and DPLLs sync to their sources to get compensation values.

A comp source PHY must be initialized before its comp sink PHYs are initialized or DPLLs are enabled, and source must be kept initialized while source or sinks are in use.

Source	Sink(s) to this Source
PHY A	DPLLs*, PHY B, PHY C

*DPLLs now have their own comp, but sink programming should be maintained for commonality with past projects.

Combo PHY Initialization Sequence

The PHY must be initialized before enabling combo PHY DDI IO power or Aux IO power. It can be initialized as early as the full display initialization sequence.

DDIA PHY initialization state will be preserved across DC6. Depending on project and SKU, other DDIs may not be preserved and then must be re-initialized for use after DC6 is disabled.

1. If **PORT_COMP_DW0** Comp Init == 1b, skip the rest of this sequence since it is already initialized
2. **DCC susclk divider programming**

   PORT_TX_DW8

   Program PORT_TX_DW8[odcc_clk_sel] to "1" and PORT_TX_DW8[odcc_clk_div_sel[1:0]] to "00".

   DCC continuous mode

   PORT_PCS_DW1

   Program PORT_PCS_DW1[DCC Mode Select] to DCC continuous mode.
3. Clear **PHY_MISC_<DDI that maps to this PHY>** DE to IO Comp Pwr Down to 0b. Skip this step if this DDI does not have a PHY_MISC register instance.
4. Program procmon reference values in **PORT_COMP_DW1**, **PORT_COMP_DW9**, and **PORT_COMP_DW10**.
   - Procmon Reference Values

Voltage	0.85V	0.85V	0.95V	0.95V	1.05V	1.05V
Register/Process	dot-0	dot-1	dot-0	dot-1	dot-0	dot-1
PORT_COMP_DW1 [7:0]	0x00	N/A	0x00	0x00	0x00	0x00
PORT_COMP_DW1 [23:16]	0x00	N/A	0x00	0x00	0x00	0x44
PORT_COMP_DW9 [31:0]	0x62AB67BB	N/A	0x86E172C7	0x93F87FE1	0x98FA82DD	0x9A00AB25
PORT_COMP_DW10 [31:0]	0x51914F96	N/A	0x77CA5EAB	0x8AE871C5	0x89E46DC1	0x8AE38FF1

Voltage and process are found in **PORT_COMP_DW3**

5. If this PHY is a comp source, set **PORT_COMP_DW8** irefgen to 1b.
6. Set **PORT_COMP_DW0** Comp Init to 1b.
7. Set **PORT_CL_DW5** CL Power Down Enable to 1b.

Combo PHY Un-Initialization Sequence

The combo PHY only needs to be un-initialized for DC9.

DDIA PHY is the comp source, so it must not be un-initialized if other combo PHYs are in use.

1. Set **PHY_MISC** DE to IO Comp Pwr Down to 1b.
2. Clear **PORT_COMP_DW0** Comp Init to 0b.

If the PHY is needed again, follow the initialization sequence.

Voltage Swing Programming Sequence

This sequence is used to setup the voltage swing before enabling the DDI, as well as for changing the voltage during DisplayPort link training. The voltage swing values are listed in the tables below. Note that the MIPI DSI voltage swing programming is for the high speed data buffers and hardware automatically handles the voltage swing for the low power data buffers.

1. If port type is eDP or DP, set PORT_PCS_DW1 cmnkeeper_enable to 1b, else clear to 0b.
2. Program loadgen select

Loadgen Values	Conditions		
Register	Bit rate <= 6 GHz and 4 lanes enabled	Bit rate <= 6 GHz and 2 or 1 lanes enabled	Bit rate > 6 GHz
PORT_TX_DW4_LN0 Loadgen Select	0	0	0
PORT_TX_DW4_LN1 Loadgen Select	1	1	0
PORT_TX_DW4_LN2 Loadgen Select	1	1	0
PORT_TX_DW4_LN3 Loadgen Select	1	0	0

- Group access cannot be used here since each lane can have a unique value, and any later writes to PORT_TX_DW4 must not use group access so that they don't overwrite the individual lane values.

3. Set PORT_CL_DW5 SUS Clock Config to 11b.
4. Clear training enable to change swing values
   - Clear PORT_TX_DW5 TX Training Enable to 0b
5. Program swing and de-emphasis
   - Values for each port type are listed in voltage swing programming tables below
   - Set PORT_TX_DW5 Scaling Mode Sel to 010b.
   - Set PORT_TX_DW2 Rcomp Scalar to 0x98 hex
   - Set PORT_TX_DW4 Post Cursor 2 to 0x00 hex
   - Set PORT_TX_DW5
     - Rterm Select to 110b
     - 3 Tap Disable to 1
     - Cursor program to 0
     - Coeff Polarity to 0
   - Program the other fields in PORT_TX_DW2, PORT_TX_DW4, PORT_TX_DW5, and PORT_TX_DW7 using the values specified in the table below.
6. Set training enable to trigger update
   - Set PORT_TX_DW5 TX Training Enable to 1b

Voltage Swing Programming

Protocol	Voltage Swing Level[1]	Pre-emphasis Level[1]	Non-transition mV diff p-p	Transition mV diff p-p	Pre-emphasis dB	Swing Sel DW2 binary	N Scalar DW7 hex	Cursor Coeff DW4 hex	Post Cursor 1 DW4 hex	2 Tap Disable DW5
DP up to HBR	0	0	350	350	0.0	4'b1010	0x32	0x3F	0x00	0'b0
DP up to HBR	0	1	350	500	3.1	4'b1010	0x4F	0x37	0x08	0'b0
DP up to HBR	0	2	350	700	6.0	4'b1100	0x71	0x2F	0x10	0'b0
DP up to HBR	0	3	350	900	8.2	4'b0110	0x7D	0x2b	0x14	0'b0
DP up to HBR	1	0	500	500	0.0	4'b1010	0x4C	0x3F	0x00	0'b0
DP up to HBR	1	1	500	700	2.9	4'b1100	0x73	0x34	0x0b	0'b0
DP up to HBR	1	2	500	900	5.1	4'b0110	0x7F	0x2F	0x10	0'b0
DP up to HBR	2	0	650	700	0.6	4'b1100	0x6C	0x3C	0x03	0'b0
DP up to HBR	2	1	600	900	3.5	4'b0110	0x7F	0x35	0x0a	0'b0
DP up to HBR	3	0	900	900	0.0	4'b0110	0x7F	0x3F	0x00	0'b0
DP HBR2	0	0	350	350	0.0	4'b1010	0x35	0x3F	0x00	0'b0
DP HBR2	0	1	350	500	3.1	4'b1010	0x4F	0x37	0x08	0'b0
DP HBR2	0	2	350	700	6.0	4'b1100	0x63	0x2F	0x10	0'b0
DP HBR2	0	3	350	900	8.2	4'b0110	0x7F	0x2b	0x14	0'b0

Protocol	Voltage Swing Level[1]	Pre-emphasis Level[1]	Non-transition mV diff p-p	Transition mV diff p-p	Pre-emphasis dB	Swing Sel DW2 binary	N Scalar DW7 hex	Cursor Coeff DW4 hex	Post Cursor 1 DW4 hex	2 Tap Disable DW5
DP HBR2	1	0	500	500	0.0	4'b1010	0x47	0x3F	0x00	0'b0
DP HBR2	1	1	500	700	2.9	4'b1100	0x63	0x34	0x0b	0'b0
DP HBR2	1	2	500	900	5.1	4'b0110	0x7F	0x2F	0x10	0'b0
DP HBR2	2	0	650	700	0.6	4'b1100	0x61	0x3C	0x03	0'b0
DP HBR2	2	1	600	900	3.5	4'b0110	0x7B	0x35	0x0a	0'b0
DP HBR2	3	0	900	900	0.0	4'b0110	0x7F	0x3F	0x00	0'b0
DP HBR2	0	0	350	350	0.0	4'b1010	0x35	0x3F	0x00	0'b0
DP HBR2	0	1	350	500	3.1	4'b1010	0x4F	0x36	0x09	0'b0
DP HBR2	0	2	350	700	6.0	4'b1100	0x60	0x32	0x0D	0'b0
DP HBR2	0	3	350	900	8.2	4'b1100	0x7F	0x2D	0x12	0'b0
DP HBR2	1	0	500	500	0.0	4'b1100	0x47	0x3F	0x00	0'b0
DP HBR2	1	1	500	700	2.9	4'b1100	0x6F	0x36	0x09	0'b0
DP HBR2	1	2	500	900	5.1	4'b0110	0x7D	0x32	0x0D	0'b0
DP HBR2	2	0	650	700	0.6	4'b0110	0x60	0x3C	0x03	0'b0
DP HBR2	2	1	600	900	3.5	4'b0110	0x7F	0x34	0x0B	0'b0
DP HBR2	3	0	900	900	0.0	4'b0110	0x7F	0x3F	0x00	0'b0
eDP up to HBR2	0	0	200	200	0.0	4'b0000	0x7F	0x3F	0x00	0'b0
eDP up to HBR2	0	1	200	250	1.9	4'b1000	0x7F	0x38	0x07	0'b0
eDP up to HBR2	0	2	200	300	3.5	4'b0001	0x7F	0x33	0x0C	0'b0
eDP up to HBR2	0	3	200	350	4.9	4'b1001	0x7F	0x31	0x0e	0'b0
eDP up to HBR2	1	0	250	250	0.0	4'b1000	0x7F	0x3F	0x00	0'b0
eDP up to HBR2	1	1	250	300	1.6	4'b0001	0x7F	0x38	0x07	0'b0
eDP up to HBR2	1	2	250	350	2.9	4'b1001	0x7F	0x35	0x0a	0'b0
eDP up to HBR2	2	0	300	300	0.0	4'b0001	0x7F	0x3F	0x00	0'b0
eDP up to HBR2	2	1	300	350	1.3	4'b1001	0x7F	0x38	0x07	0'b0
eDP up to HBR2	3	0	350	350	0.0	4'b1001	0x7F	0x3F	0x00	0'b0
eDP up to HBR2 - optimized[2]					0.0	4'b0110	0x7F	0x3F	0x00	0'b0
eDP HBR3	0	0	350	350	0.0	4'b1010	0x35	0x3F	0x00	0'b0
eDP HBR3	0	1	350	500	3.1	4'b1010	0x4F	0x37	0x08	0'b0

Protocol	Voltage Swing Level[1]	Pre-emphasis Level[1]	Non-transition mV diff p-p	Transition mV diff p-p	Pre-emphasis dB	Swing Sel DW2 binary	N Scalar DW7 hex	Cursor Coeff DW4 hex	Post Cursor 1 DW4 hex	2 Tap Disable DW5
eDP HBR3	0	2	350	700	6.0	4'b1100	0x71	0x2F	0x10	0'b0
eDP HBR3	0	3	350	900	8.2	4'b0110	0x7F	0x2b	0x14	0'b0
eDP HBR3	1	0	500	500	0.0	4'b1010	0x4C	0x3F	0x00	0'b0
eDP HBR3	1	1	500	700	2.9	4'b1100	0x73	0x34	0x0b	0'b0
eDP HBR3	1	2	500	900	5.1	4'b0110	0x7F	0x2F	0x10	0'b0
eDP HBR3	2	0	650	700	0.6	4'b1100	0x6C	0x3C	0x03	0'b0
eDP HBR3	2	1	600	900	3.5	4'b0110	0x7F	0x35	0x0a	0'b0
eDP HBR3	3	0	900	900	0.0	4'b0110	0x7F	0x3F	0x00	0'b0
HDMI			450	450	0.0	4'b1010	0x60	0x3F	0x00	0'b0
HDMI			450	650	3.2	4'b1011	0x73	0x36	0x09	0'b0
HDMI			450	850	5.5	4'b0110	0x7F	0x31	0x0E	0'b0
HDMI: ALS and Retimer option			650	650	0.0	4'b1011	0x73	0x3F	0x00	0'b0
HDMI			650	850	2.3	4'b0110	0x7F	0x37	0x08	0'b0
HDMI			850	850	0.0	4'b0110	0x7F	0x3F	0x00	0'b0
HDMI			600	850	3.0	4'b0110	0x7F	0x35	0x0A	0'b0
MIPI DSI			400	400	0.0	4'b0010	0x7F	0x3F	0x00	**0'b1**

[1]The voltage swing level and pre-emphasis level values follow the naming used in the DisplayPort standard.

[2]eDP extra power optimized setting. This swing setting can be used for eDP HBR2 and lower frequencies if the motherboard has been designed to support it and it is specified through VBT or other OEM selection.

Program the voltage swing as listed in the table and additionally program PORT_CL_DW10 o_edp4k2k_mode_ovrd_en=1 and o_edp4k2k_mode_ovrd_val=1. Program o_edp4k2k_mode_ovrd_val=0 when using any other swing values. If link training fails with this swing, fall back to the regular eDP swing settings.

eDP panels may support lower power, low voltage, swing values using the "eDP" protocol values from the table or higher power, high voltage, swing values using the "DP" protocol values. The selection is generally an OEM decision configured in BIOS VBT.

TypeC PHY DDI Buffer

This programming is used for the typeC PHY when operating in DP alternate mode or the legacy HDMI or DP modes. This programming is not used for thunderbolt as the thunderbolt controller takes care of the PHY programming.

PHY Lane Configuration

The PHY contains 2 lanes, and each lane contains a primary transmitter (TX1) and a secondary transmitter/receiver (TX2 or TX/RX). A single PHY lane can support up to 2 DP or HDMI lanes. Two PHY lanes can be used to support 4 DP or HDMI lanes. Note that the RX function in TX2 is always disabled for DP use cases.

Driver must configure the DP mode to indicate which transmitter is used and voltage swing for each transmitter in use, within each lane in use.

Voltage Swing Programming Sequence

This sequence is used to setup the voltage swing before enabling the DDI, as well as for changing the voltage during DisplayPort link training.

1. Clear DKL_TX_PMD_LANE_SUS_LN0 register if display owns this PHY lane.
2. Clear DKL_TX_PMD_LANE_SUS_LN1 register if display owns this PHY lane.
3. Update registers DKL_TX_DPCNTL0, DKL_TX_DPCNTL1 and DKL_TX_DPCNTL2 registers with appropriate Vswing and de-emphasis levels from table below.
4. Set DKL_TX_DPCNTL2[dp20bitmode] to "0".

DP Voltage Swing Table

Voltage Swing	Pre-emphasis	Non-transition mV diff p-p	Transition mV diff p-p	Pre-emphasis dB	Voltage Level Swing Select	Pre Cursor	Post Cursor	
					DKL_TX_DPCNTL0[2:0] DKL_TX_DPCNTL1[2:0]	DKL_TX_DPCNTL0[17:13] DKL_TX_DPCNTL1[17:13]	DKL_TX_DPCNTL0[12:8] DKL_TX_DPCNTL1[12:8]	
					Vswing_Control_tx1 Vswing_Control_tx2	preshoot_control_l0_tx1 preshoot_control_l0_tx2	de_emphasis_control_l0_tx1 de_emphasis_control_l0_tx2	
							HBR/RBR	HBR3/HBR2
Level 0	0	400	400	0	7	0	0	0
Level 0	1	400	600	3.5	5	0	5	5
Level 0	2	400	800	6	2	0	B	B
Level 0	3	400	1200	9.5	0	0	18	19
Level 1	0	600	600	0	5	0	0	0
Level 1	1	600	800	3.5	2	0	8	8
Level 1	2	600	1200	6	0	0	14	14
Level 2	0	800	800	0	2	0	0	0
Level 2	1	800	1200	3.5	0	0	B	B
Level 3	0	1200	1200	0	0	0	0	0

HDMI Voltage Swing Table

Protocol	Voltage Swing and Pre-emphasis	HDMI Preset Level	Voltage Level Swing Select DKL_TX_DPCNTL0[2:0] DKL_TX_DPCNTL1[2:0] Vswing_Control_tx1 Vswing_Control_tx2	Pre Cursor DKL_TX_DPCNTL0[17:13] DKL_TX_DPCNTL1[17:13] preshoot_control_l0_tx1 preshoot_control_l0_tx2	Post Cursor DKL_TX_DPCNTL0[12:8] DKL_TX_DPCNTL1[12:8] de_emphasis_control_l0_tx1 de_emphasis_control_l0_tx2
HDMI	400mV / 0dB	1	7	0	0
HDMI	500mV / 0dB	2	6	0	0
HDMI : ALS	650mV / 0dB	3	4	0	0
HDMI	800mV / 0dB	4	2	0	0
HDMI : Re-timer	1000mV / 0dB	5	0	0	0
HDMI	Full / -1.5dB	6	0	0	5
HDMI	Full / -1.8dB	7	0	0	6
HDMI : CRLS	Full / -2dB	8	0	0	7
HDMI	Full / -2.5dB	9	0	0	8
HDMI	Full / -3dB	10	0	0	A

DKL_DP_MODE Programming

There are two DP_MODE registers per PHY, one for each PHY lane, containing x1 and x2 fields that are programmed to indicate which transmitter subsystem to enable. The values to program are derived from the pin and lane assignments and port width selection found in the following registers.

- PORT_TX_DFLEXPA1 DPPATC<this port> Display Port Pin Assignment for Type-C Connector
- PORT_TX_DFLEXDPSP DPX4TXLATC<this port> Display Port x4 TX Lane Assignment for Type-C Connector
- DDI_BUF_CTL_<this port> DP Port Width Selection

Orientation flip and lane reversal differences are cancelled out by hardware lane enable signaling shutdown of the unused LN0 or LN1.

DPPATC	DPX4TXLATC	Description	DP Port Width	DP MODE LN1		DP MODE LN0	
				x2_mode	x1_mode	x2_mode	x1_mode
0x1	0xf	A	x4	1	0	1	0
0x1	0xf	A flip	x4	1	0	1	0
0x1	0xf	A	x2	0	0	0	0
0x1	0xf	A flip	x2	0	0	0	0
0x1	0xf	A	x1	0	0	0	0
0x1	0xf	A flip	x1	0	0	0	0
0x1	0x5	A active	x2	0	0	0	0
0x1	0x5	A active flip	x2	0	0	0	0

DPPATC	DPX4TXLATC	Description	DP Port Width	DP MODE LN1		DP MODE LN0	
				x2_mode	x1_mode	x2_mode	x1_mode
0x1	0x5	A active	x1	0	0	0	0
0x1	0x5	A active flip	x1	0	0	0	0
0x2	0xc	B	x2	1	0	1	0
0x2	0x3	B flip	x2	1	0	1	0
0x2	0xc	B	x1	0	0	0	0
0x2	0x3	B flip	x1	0	0	0	0
0x2	0x4	B active	x1	0	0	0	0
0x2	0x1	B active flip	x1	0	0	0	0
0x3, 0x5	0xf	C/E	x4	1	0	1	0
0x3, 0x5	0xf	C/E flip	x4	1	0	1	0
0x3, 0x5	0xf	C/E	x2	1	0	1	0
0x3, 0x5	0xf	C/E flip	x2	1	0	1	0
0x3, 0x5	0xf	C/E	x1	0	1	0	1
0x3, 0x5	0xf	C/E flip	x1	0	1	0	1
0x4, 0x6	0xc	D/F	x2	1	0	1	0
0x4, 0x6	0x3	D/F flip	x2	1	0	1	0
0x4, 0x6	0xc	D/F	x1	0	1	0	1
0x4, 0x6	0x3	D/F flip	x1	0	1	0	1
0x0	n/a	Fixed/static	x4 or HDMI	1	0	1	0
0x0	n/a	Fixed/static reverse	x4 or HDMI	1	0	1	0
0x0	n/a	Fixed/static	x2	1	0	1	0
0x0	n/a	Fixed/static reverse	x2	1	0	1	0
0x0	n/a	Fixed/static	x1	0	1	0	0
0x0	n/a	Fixed/static reverse	x1	0	1	0	0

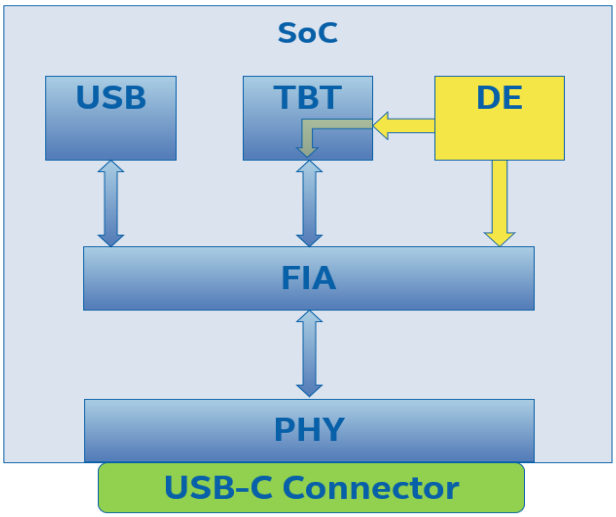
Decode of x1 and x2

DP MODE		Active TX
cfg_dp_x2_mode	cfg_dp_x1_mode	
0	0	TX1 (TX)
0	1	TX2 (TX/RX)
1	0	TX1 + TX2
1	1	TX1 + TX2

TypeC Programming

The USB 3.1, thunderbolt 3, and display (DE) controllers are integrated into the CPU. All these controllers connect to Flexi I/O Adapter (FIA) that muxes data and clocks between controllers and the PHY. The PHY contains the IOs and PLLs for the USB-C connector. Display Engine can also output to thunderbolt (TBT) which then goes to FIA and to PHY. During mode enumeration (DE to FIA path for non-TBT), display software takes into account not only the port bandwidth of the DP receiver (monitor) and any intervening DP re-timers, but also the number of DP lanes being driven out of a given USB Type-C port.

TBT does not impose restrictions on number of DP lanes that can be sourced. Display software reads the monitor capabilities, trains the link, and chooses number of DP lanes to be used per current software policies. However, PHY will not need to be programmed or trained. TBT will use a different PLL than the usual combo PHY DDI PLL or PLL, so software will need to be aware of the TBT connector.



Static/Fixed/Legacy/Native DP or HDMI Configuration

A native DP or HDMI connector can be attached to the TypeC PHY to support a static or fixed configuration instead of dynamic typeC. This configuration must be specified in IFWI strap bits which the SoC micro-controller uses to configure each port, and BIOS VBT which display software uses.

Type C DDI Sequence

The main Type-C flows of interest to display controller are

- PD connect flow
- PD disconnect flow
- PD Re-configuration flow

As part of PD connect flow, whenever a new device is connected over the Type-C connector, PD engine detects the presence of the device over the CC channel and identifies the connector orientation. It also configures PHY common lane and data lane to operate in DP mode. At the end of the PD connect flow,

SOC uC (typically called IOM) also programs FIA Pin Assignment registers, and programs FIA registers with pin assignment (PA), link width, live state, etc. Finally, PD connect flow sends HPD to DE.

The type-C PHYs can be shared between multiple controllers; display, USB, etc. As a result, handshaking through FIA registers is required around connect and disconnect to cleanly transfer ownership with the controller and set the type-C power state.

Display software most not use a disconnected port.

FIA registers updated during PD flows

Register	Function
PORT_TX_DFLEXPA1	FIA arranges the 4 DP lanes in Type-C connector based on 6 possible arrangements called pin assignments A-F in VESA DP Type-C spec.
PORT_TX_DFLEXDPPMS PORT_TX_DFLEXDPCSSS PORT_TX_DFLEXDPSP	Communication between SoC uC and display software.
PORT_TX_DFLEXDPMLE1	Display software configures the number of lanes it is using

Port Mapping

The following table highlights mapping from port to FIA instance.

TypeC Port	2 Connector Modular FIA family
Port1	FIA1, connector 0
Port2	FIA1, connector 1
Port3	FIA2, connector 0
Port4	FIA2, connector 1
Port5	FIA3, connector 0
Port6	FIA3, connector 1

The mapping from connector to port is split into groups of 2. For example, the FIA1 DFLEXDPSP1 register fields referring to connectors 0 through 1 map to ports 1 through 2 and the FIA2 DFLEXDPSP1 register fields referring to connectors 0 through 1 maps to ports 3 through 4. The FIA register fields for connectors >1 are reserved.

Type-C Aux Power Requirements

The aux power function is overloaded to control power states in the type-C subsystem.

Aux power must only be enabled while in the connect state. Do not enable Aux power before connect flow or after disconnect flow.

For DP-alternate and thunderbolt, Aux power must be enabled while using Aux channel or main link (enable mode set) and it should be disabled at other times to save power.

For static/fixed/legacy/native, for DP and for HDMI, it must be enabled during the connect flow and not disabled until disconnect flow.

For every typeC port (static/fixed/legacy/native and DP-alternate, but not thunderbolt), after enabling aux power request and receiving aux power ack, wait for uc_health bit (Dekel PHY register DKL_CMN_UC_DW27 @BASE + 0x236c bit [15]) to assert. Timeout and fail after 10us.

Type-C Hot Plug Detection (HPD)

Type-C ports have separate hot plug detection paths for thunderbolt, DP alternate, and legacy HDMI or DP connectors (static/fixed/legacy/native connection).

The HPD control and status reside in 3 separate sets of display registers

1. Thunderbolt HPD is in the north display hotplug control and interrupt registers, TBT instances.
2. DP alternate HPD is in the north display hotplug control and interrupt registers, TC instances.
3. Legacy HDMI or DP hotplug is detected through the south display hotplug control and interrupt registers.

Display software enables HPD detection in all 3 sets, then responds to whichever interrupt is received. Based on the hotplug source, display software invokes one of the connect/disconnect flows below.

The DFLEXDPSP Live State fields provide the current connect or disconnect status for thunderbolt and DP alternate.

See the Hot Plug Detection page for details.

DP Tunneling - Integrated TBT behavior

On TBT integrated products, DP resource allocation is done by IOM. A USB-C port can connect to TBT discrete device (via a cable) and then a DP source will be allocated to a TBT host to be driven according to TBT protocol. DP resources are allocated according to the IOM "DP_RESOURCE_MNG" register. TBT CM (connection manager) will claim a DP resource via this register.

When IOM claim a resource to itself, TBT connection manager interprets that that corresponding resource is allocated and TBT cannot use it. IOM can also claim a resource even when TBT already claimed the same DP resource, which will resolve in TBT releasing it in favor of IOM.

HPD Processing

HPD messages from TBT to IOM is sent via a HW mechanism. IOM requires TBT to send HPD messages only when a DP resource is allocated to it. TBT CM will build a DP tunneling path only after it gets a resource. At HPD disconnect, TBT CM will tear down the DP connection path (no tunnel) and DP AUX should also be considered as disconnected. TBT will ignore all interface signals to display engine when there is no tunnel. IOM sends acknowledgement to TBT upon receiving HPD disconnect message.

If HPD connect event is detected while TBT port is enabled for DP tunneling a mode set disable is recommended, followed by re-evaluation and re-enabling of the TBT DP tunnel.

DP Connected Standby

DP connected standby is a low power mode in which monitors will be turned OFF actively by IOM (monitors are still physically connected).

NOTE: Registers mentioned below are within Type-C subsystem and not accessible to display software.

Entering connected standby mode

IOM will assert "dp_disconnect_req" on the DP_CONFIG register.

TBT LC (link controller) will detect the above bit assertion and will configure unplug to DP adaptor.

TBT CM will receive the un-plug packet and trigger the basic disconnect flow.

Exiting connected standby mode

IOM will de-assert "dp_disconnect_req" on the DP_CONFIG register.

LC will detect the above bit de-assertion and configure the plug of the DP adapter.

CM triggers the basic connect flow.

TBT DP Path Topology Retention

TBT can route any of its DP inputs to any of its endpoint monitors. In order to keep consistency of resources routing when entering/exiting standby modes, CM will register the routing configuration on its always ON memory. Registering the DP routing configuration should be done whenever a new path is built and CM will always check last state configuration prior to claiming a DP resource.

Whenever the TBT CM detects new plug-in monitor on one of its TBT ports it follows below flow.

- TBT CM will check it's always ON memory for the last DP allocation of the new plug-in TBT port.
- TBT CM will check if DP resource is available and claim it, Otherwise claim for other DP resource. It is intended to restore to the state before entering a standby mode. IOM on its side should prevent DP ALT mode allocation to a source that was previously allocated to TBT.
- If monitors connectivity was physically changed between entering and exiting standby mode, TBT resource routing will not guarantee any sort of compatibility to pre-standby mode.
- If a new DP resource is allocated to TBT port, TBT CM should update the configuration on its always ON memory.

Connect Flows

These flows are used for completing a connection. They can be initiated upon receiving HPD connect interrupt or finding that a previous HPD connect was missed because of reset or power state. IOM FW will configure FIA and send the HPD assert message to display engine to trigger an interrupt. Display software can then complete the handshake and enable the port.

Note that display software can abort the connect flow if it does not intend to use the port main link or Aux channel. By aborting the connect, display software allows IOM to immediately change ownership of the port on a disconnect, without waiting for any display software response.

Display software is required to abort the connect if it knows it will be entering states where hotplug cannot be responded to for a long period, such as when GOP/VBIOS chooses not to enable a display on typeC, entering DC9, D3, S3, S4, and S5, and un-installing driver. Software can start the connect flow after exiting these states.

TCCOLD

The typeC subsystem can enter the TCCOLD power saving state when the state has been enabled by BIOS, there is no DP alternate or thunderbolt connection, and display software has not blocked it with a GT Driver Mailbox command. TCCOLD must be blocked by IOM or display software when display software accesses FIA or typeC PHY registers or uses Aux channel or the main link.

The general usage is for display software to use GT Driver mailbox to block TCCOLD before first accessing FIA registers during the hotplug and connect flows, then use GT Driver mailbox to unblock TCCOLD only after it is done accessing FIA registers and there is no longer a connection (connect abort complete or disconnect complete).

DP-alternate specific usage

Display software uses GT Driver mailbox to block TCCOLD before accessing FIA registers to during type-C hotplug detection.

If there is no longer a connection, display software uses GT Driver mailbox to unblock TCCOLD after finishing accessing FIA registers, and does not follow any connect flow.

If there is a DP-alternate connection, display software proceeds to the connect flow for DP alternate and leaves the mailbox in the TCCOLD blocking state.

Display software uses GT Driver mailbox to unblock TCCOLD after last accessing FIA registers when aborting the connect flow or completing the disconnect flow.

IOM will be doubly blocking TCCOLD while there is a receiver connected or once DFLEXDPPMS.DPPMSTC and DFLEXDPCSSS.DPPMSTC are both 1 in the connect flow until DFLEXDPCSSS.DPPMSTC is cleared in the disconnect flow, but display software will not rely on that.

Thunderbolt specific usage

Display software uses GT Driver mailbox to block TCCOLD before accessing FIA registers to find if a thunderbolt receiver is connected.

If there is no longer a connection, display software uses GT Driver mailbox to unblock TCCOLD after finishing accessing FIA registers, and does not follow any connect flow.

If there is a thunderbolt connection, display software proceeds to the connect flow for thunderbolt and leaves the mailbox in the TCCOLD blocking state.

Display software uses GT Driver mailbox to unblock TCCOLD when aborting the connect flow or completing the disconnect flow.

IOM and the thunderbolt controller will be doubly blocking TCCOLD while there is a receiver connected, but display software will not rely on that.

Static/fixed/legacy/native specific usage

Display software uses GT Driver mailbox to block TCCOLD before accessing FIA registers during the connect flow and leaves the mailbox in the TCCOLD blocking state.

Display software uses GT Driver mailbox to unblock TCCOLD after last accessing FIA registers when aborting the connect flow or completing the disconnect flow.

IOM is not aware of connection and will not be doing any TCCOLD blocking of its own.

GT Driver Mailbox to Block TCCOLD

This mailbox command triggers TCCOLD to exit and not re-enter.

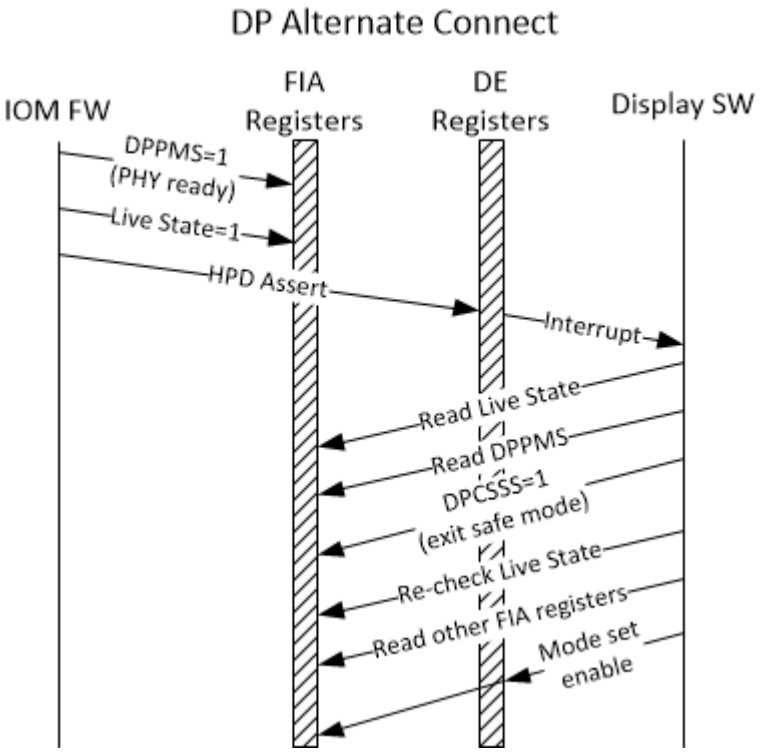
1. Ensure any previous GT Driver Mailbox transaction is complete.
2. Write GT Driver Mailbox Data Low = 0x00000000.
3. Write GT Driver Mailbox Data High = 0x00000000.
4. Write GT Driver Mailbox Interface = 0x80000026.
5. Poll GT Driver Mailbox Interface for Run/Busy indication cleared (bit 31 == 0).
    - Timeout and fail after 200 us. Do not use typeC ports if there is a timeout.
6. Read GT Driver Mailbox Data Low, if bit 0 is 0x0, continue, else go to step 2.
    - If the condition in step 6 is not satisfied after cycling through steps 2-6 for 600us total (typically <150 us), timeout and fail and do not use typeC ports.
    - Data 0 = not in TCCOLD. Data 1 = in TCCOLD.

GT Driver Mailbox to Unblock TCCOLD

This mailbox command allows TCCOLD to re-enter.

1. Ensure any previous GT Driver Mailbox transaction is complete.
2. Write GT Driver Mailbox Data Low = 0x00000001.
3. Write GT Driver Mailbox Data High = 0x00000000.
4. Write GT Driver Mailbox Interface = 0x80000026.
5. Poll GT Driver Mailbox Interface for Run/Busy indication cleared (bit 31 == 0).
    - Timeout and fail after 200 us. Do not use typeC ports if there is a timeout.

Connect Flow for DP Alternate



This flow is for DP alternate mode. PHY lane 0, lane 1, or both, will be used, depending on pin assignment.

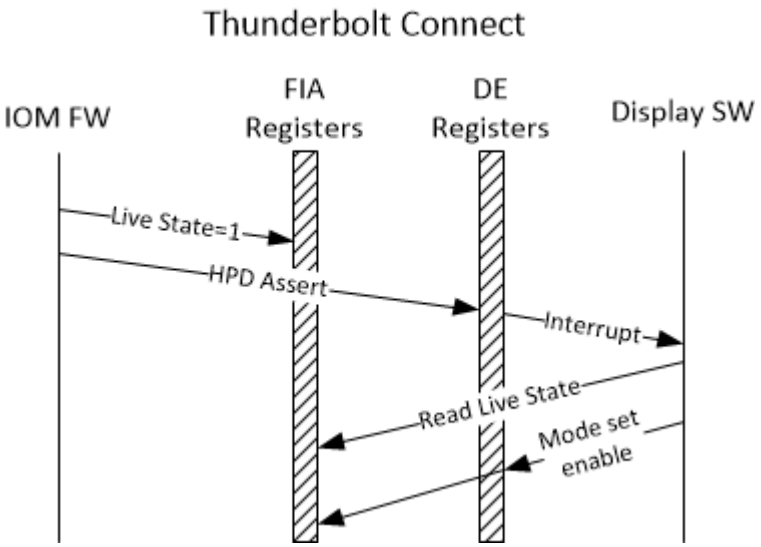
1. If display software will not use the port main link or Aux channel
    1. Abort remainder of the flow
2. Else
    1. Display software uses GT driver mailbox to block TCCOLD
    2. Display software reads DFLEXDPPMS.DPPMSTC which should be '1' to indicate the SOC uC has switched the Lane into DP Mode, else abort connect flow.
    3. Display software sets DFLEXDPCSSS.DPPMSTC to '1' indicating that display controller is not in safe mode anymore.
    4. Display software reads DFLEXDPSP to verify port has not become disconnected
        - If Live State is "No HPD Connect for TypeC or TBT", nothing is connected, so abort the connect flow.
        - If Live State is "HPD Connect for TBT", clear DFLEXDPCSSS.DPPMSTC to '0' and go to the Thunderbolt connect flow.
        - If Live State is "HPD Connect for TypeC", proceed with DP Alternate connect flow.
    5. Display software reads lane assignment from DFLEXDPSP.DPX4TXLATC register.
    6. Display software issues AUX reads for EDID/DPCD.
        - Set DDI_AUX_CTL IO Select field to legacy

- AUX power is controlled through PWR_WELL_CTL_AUX non-Thunderbolt IO Power Request
- See Type-C Aux Power Requirements for main link usage

7. Based on DFLEXDPSP.DPX4TXLATC and info from DPCD/EDID, display software knows the maximum number of lanes supported.
8. If software and the OS decide to enable display output on this connected port, software performs the DP mode set enable sequence to bring display controller up and running.

- If connect flow is aborted at any point, clear DFLEXDPCSSS.DPPMSTC to '0' if it had been set, then use GT driver mailbox to unblock TCCOLD.

Connect Flow for Thunderbolt



This flow is for DP tunneling through thunderbolt. The thunderbolt controller will configure the PHY.

1. If display software will not use the port main link or Aux channel
    1. Abort remainder of the flow
2. Else
    1. Display software uses GT driver mailbox to block TCCOLD
    2. Display software issues AUX reads for EDID/DPCD.
        - Set DDI_AUX_CTL IO Select field to TBT
        - AUX power is controlled through PWR_WELL_CTL_AUX TBT IO Power Request
        - See type-C aux power requirements for main link usage
    3. Based on info from DPCD/EDID, display software knows the maximum number of lanes supported.
    4. If software and the OS decide to enable display output on this connected port, software performs the DP mode set enable sequence to bring display controller up and running.

- If connect flow is aborted at any point, use GT driver mailbox to unblock TCCOLD.

Connect Flow for Static/Fixed/Legacy/Native

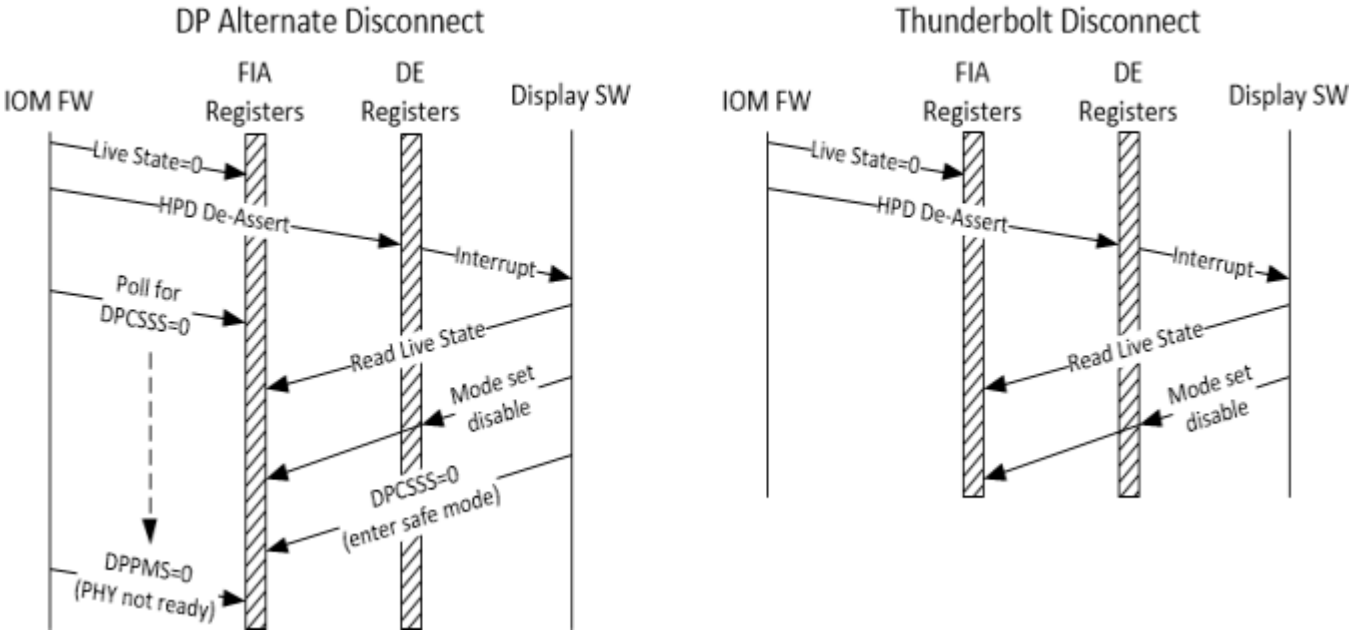
This flow is for DP/HDMI as dedicated connection. PHY lane 0 and lane 1 are dedicated to this connection.

Note: IOM is not processing static/fixed/legacy/native connections, but the handshake with FIA DFLEX registers should be followed for consistency with DP alternate mode flow.

1. If display software will not use the port main link or Aux channel
    1. Abort remainder of the flow
2. Else
    1. Display software uses GT driver mailbox to block TCCOLD
    2. Display software reads DFLEXDPPMS.DPPMSTC which should be '1' to indicate the SOC uC has switched the Lane into DP Mode, else abort connect flow.
    3. Display software sets DFLEXDPCSSS.DPPMSTC to '1' indicating that display controller is not in safe mode anymore.
    4. Set PWR_WELL_CTL_AUX non-Thunderbolt IO Power Request.
    5. Poll for PWR_WELL_CTL_AUX non-Thunderbolt IO Power State == Enabled, timeout and fail after 1.5 milliseconds (typically takes <150us).
    6. Display software issues AUX reads for EDID/DPCD.
        - Set DDI_AUX_CTL IO Select field to legacy for DP. Don't care for HDMI.
        - See Type-C Aux Power Requirements for main link usage, even for HDMI.
    7. Based on info from DPCD/EDID, display software knows the maximum number of lanes supported for DP case. For HDMI, it is always 4 lanes.
    8. If software and the OS decide to enable display output on this connected port, software performs the DP or HDMI mode set enable sequence to bring display controller up and running.

- If connect flow is aborted at any point, clear DFLEXDPCSSS.DPPMSTC to '0' if it had been set, then use GT driver mailbox to unblock TCCOLD.

Disconnect Flow



This flows is used for completing a disconnection. This can be initiated upon receiving HPD disconnect interrupt, finding that a HPD disconnect was missed because of reset or power state, or for an early disconnect where software decides it will no longer require the port main link or Aux channel until the next connect flow or power state exit.

By giving an early disconnect, display software allows IOM to immediately change ownership of the port on a real disconnect, without waiting for any display software response.

Display software is required to give an early disconnect if it knows it will be entering states where hotplug cannot be responded to for a long period, such as DC9, D3, S3, S4, S5, and un-installing driver. Software can start the connect flow after exiting these states.

1. Display software follows the mode set disable sequence. This can be done prior to disconnect flow and skipped if there was never a mode set enable.
2. Disable software disables Aux power and waits for Aux power state to disable. This cannot be done prior to mode set disable sequence because Aux power is required while Aux channel or main link are in use. Do not change the Aux control IO select field until the connect flow.
3. If not thunderbolt, display software clears DFLEXDPCSSS.DPPMSTC to '0' to tell PD FW that it had put the display Controller into Safe State.
    - Note that thunderbolt has no DFLEXDPCSSS.DPPMSTC handshake, so it is possible to receive a HPD connect interrupt before this disconnect flow is finished. Display software needs to complete the disconnect flow before moving to connect flow.
4. Display software uses GT driver mailbox to unlock TCCOLD

Link Training

- In Type-C, the connection defines the mode. Hence link training in Type-C is bound by pin assignment.
- It is ok for display engine to use fewer lanes than are available, same as if it were not a Type C connection.
- Display software is responsible for shutting off assigned but unused PHY lane(s) through DFLEXDPMLE.DPMLETC* as described in the mode set section.

Lane Configuration

Lane configuration in the case of Type-C connector (including non Type-C use cases) involves programming DE, FIA, and PHY registers. The programming is described in the mode set sequences and DDI Buffer sections.

IOM handles FIA and PHY register programming for lane configuration, except for shutting off unused lanes.

Lane reversal

Lane reversal is part of the service handled by the FIA, but only when using a type C connector.

With a legacy DP or HDMI connector, reversal is handled via the display engine DDI_BUF_CTL as selected by VBT i.e., it's a boot time configuration which will be done by display software. FIA is agnostic to any lane reversal occuring within the display controller.

Port clock programming

For DP alternate and static/fixed/legacy/native connections, a PLL in the PHY is used. For thunderbolt, a DPLL is used. The programming is described in the port clock programming section.

DDI Buffer specific registers

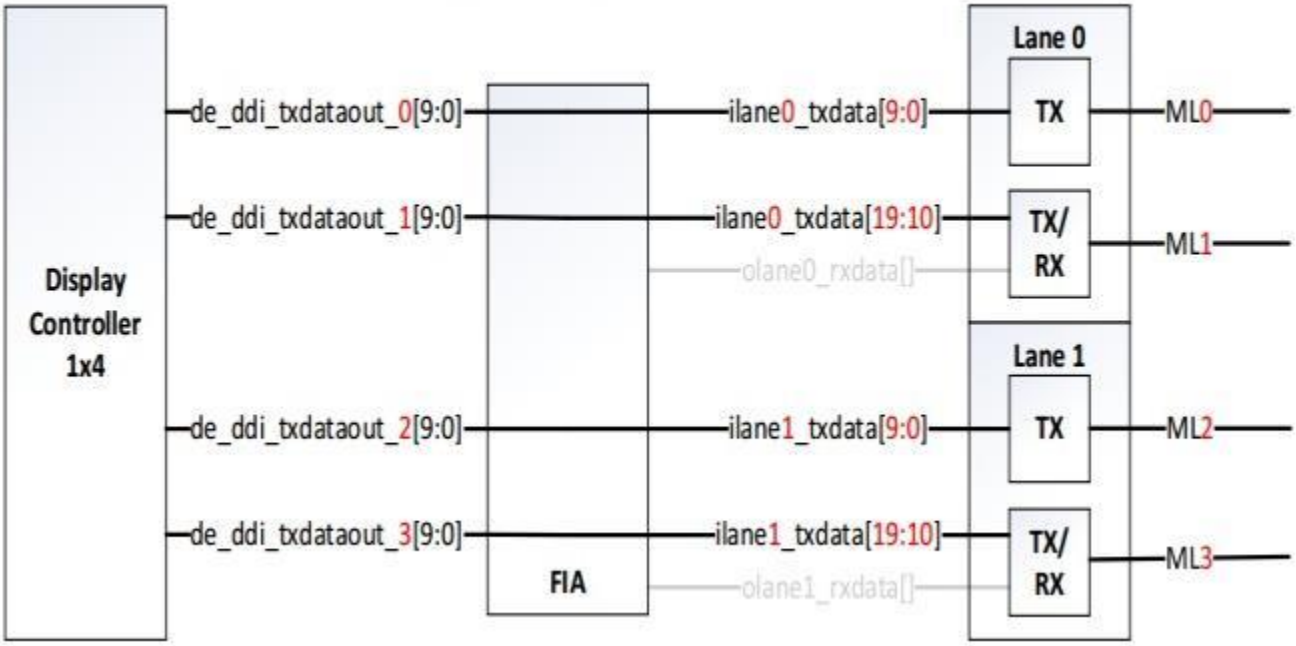
PHY registers are programmed to control voltage swing. The programming is described in the mode set sequences and DDI Buffer sections.

Pin Assignments

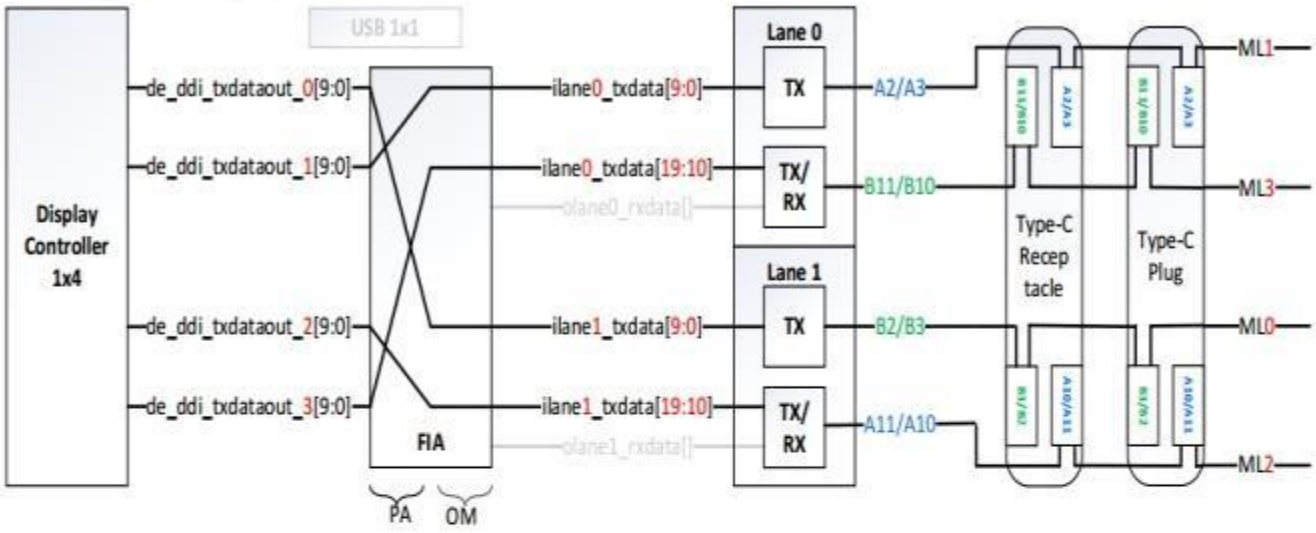
The following are the various FIA supported pin assignments, coming from the type-C specification. This shows the mapping of display data lanes through FIA to the PHY and the receiver, and the cases that limit display to 2 lanes because of using USB on the other lanes.

Please note that DE may use fewer lanes than are available, same as if it were not a Type-C connection.

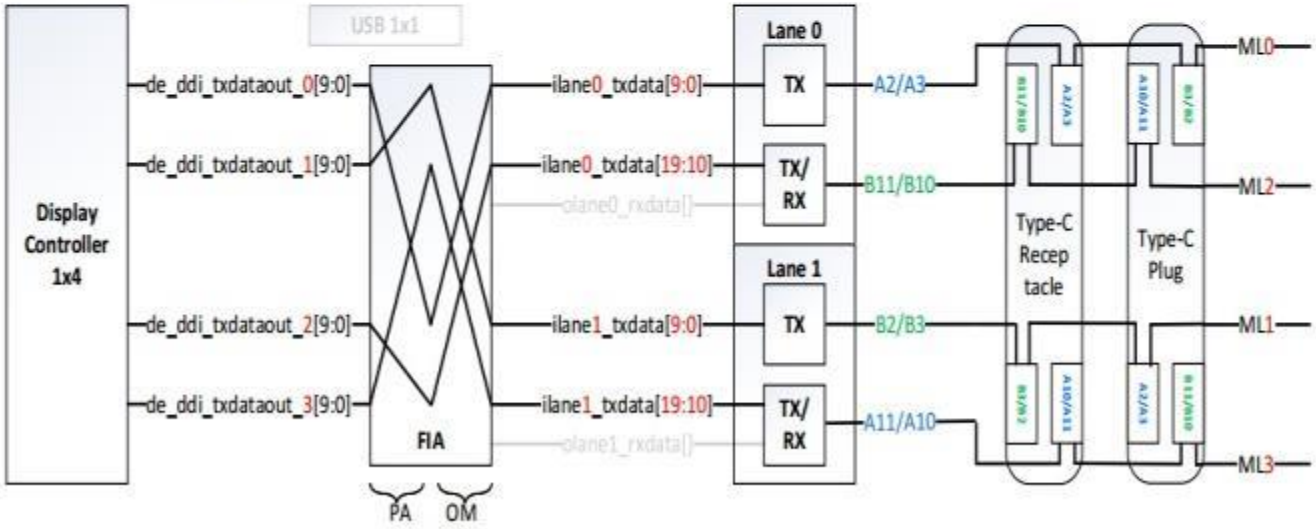
No Pin Assignment (Non Type-C DP)



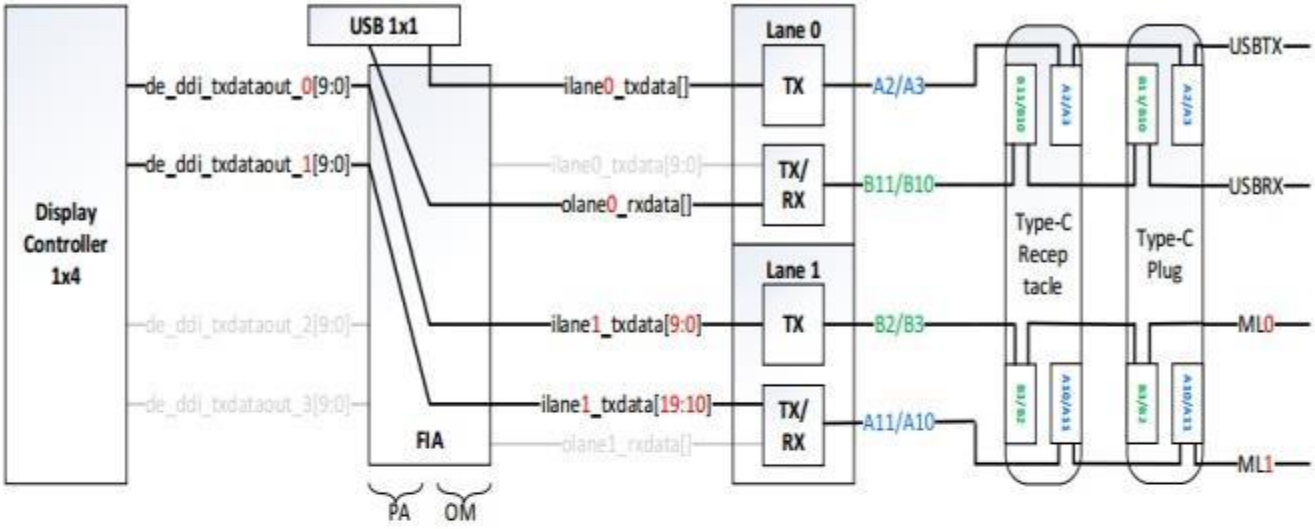
NON-flip: Pin Assignment A



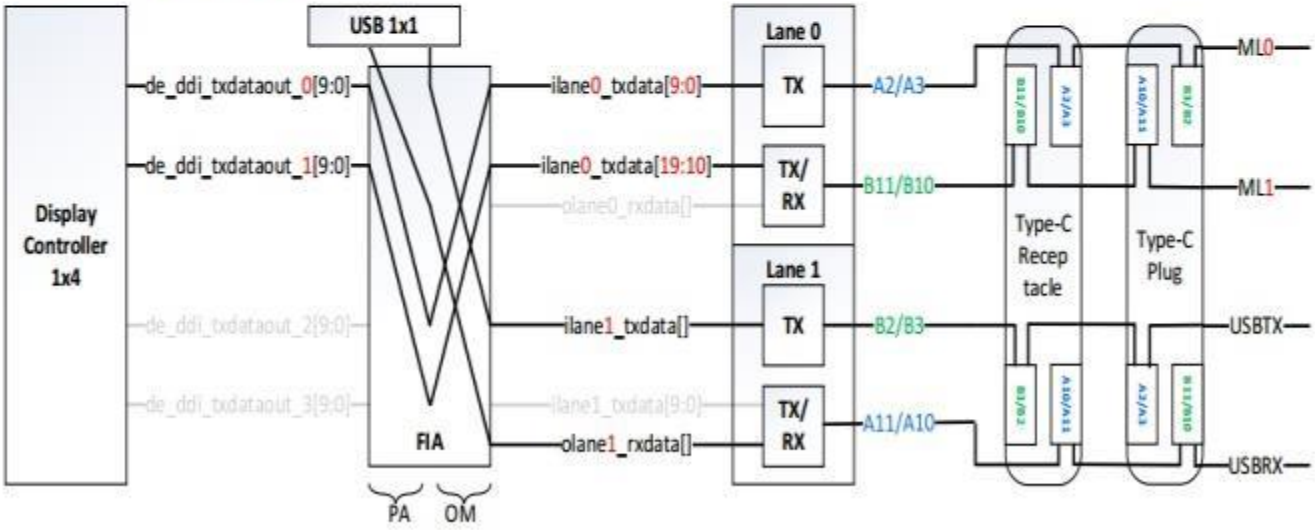
FLIP: Pin Assignment A



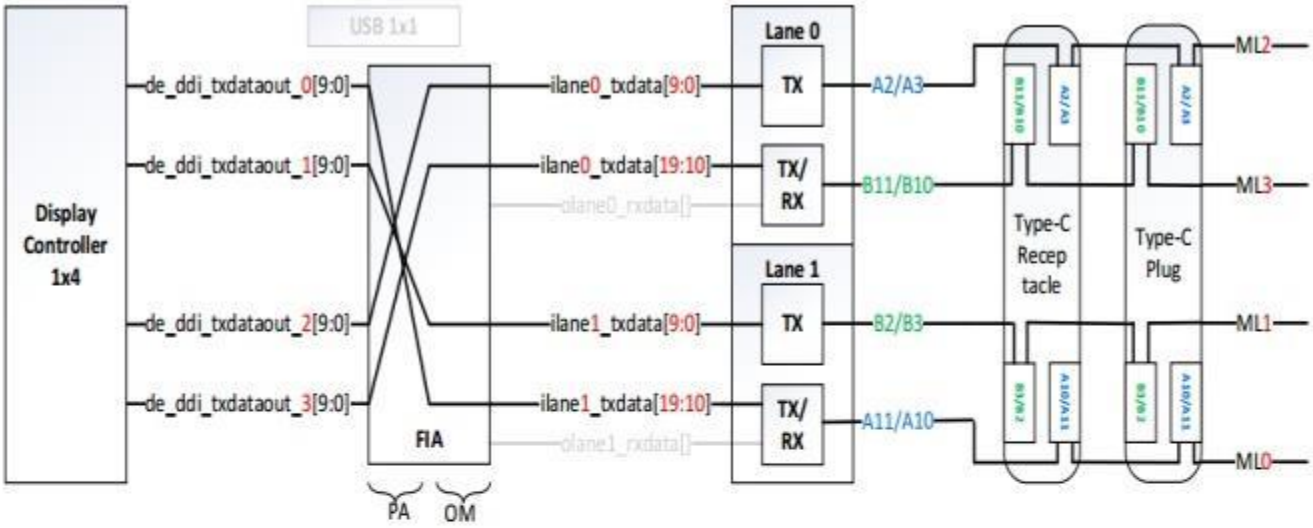
NON-flip: Pin Assignment B



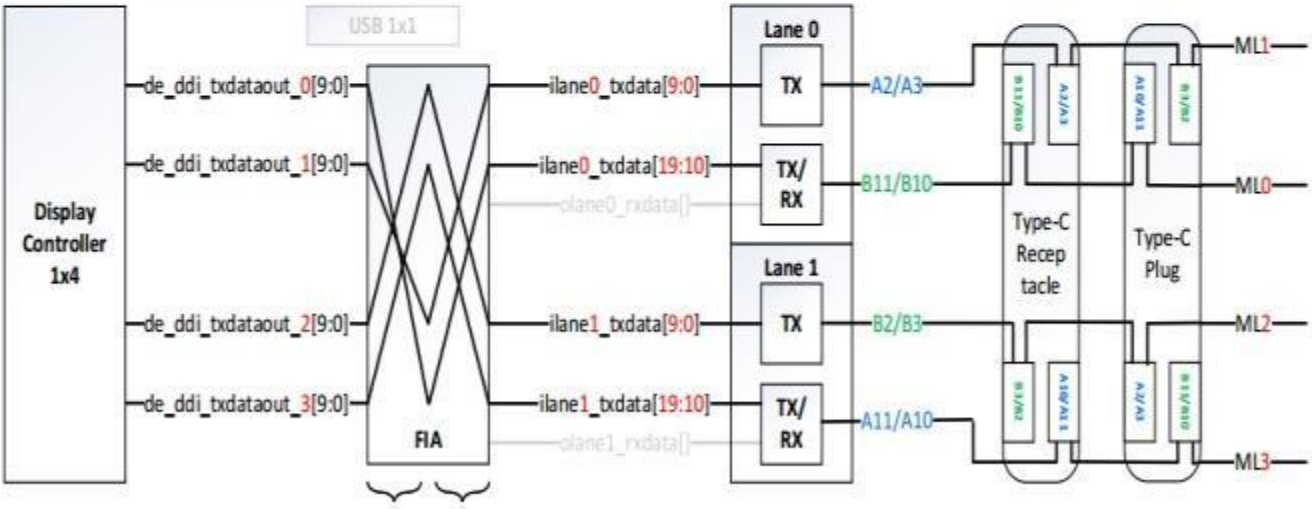
FLIP: Pin Assignment B



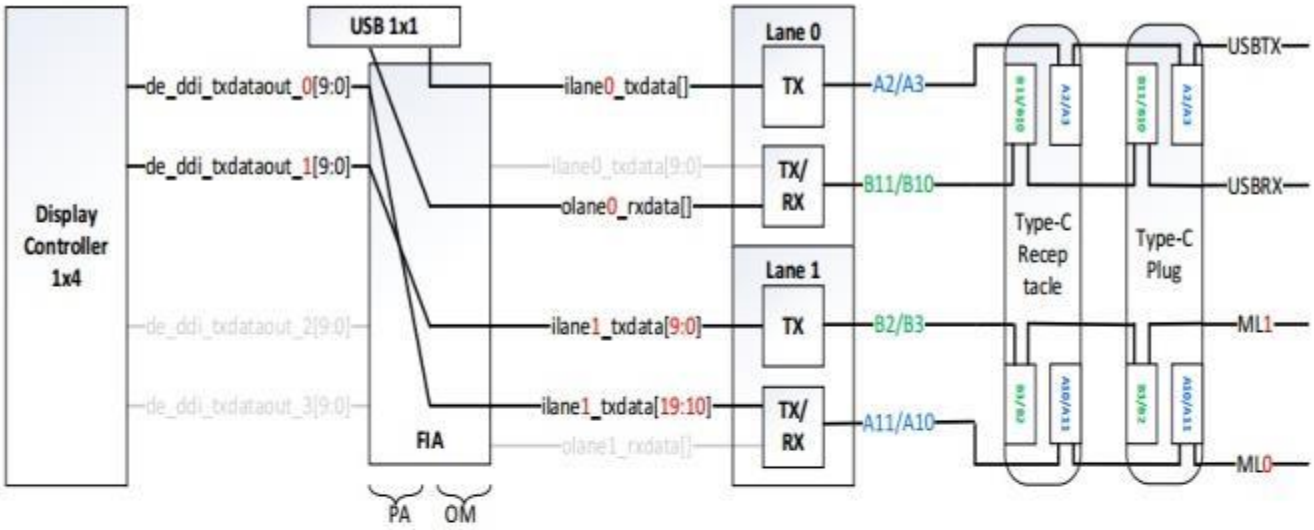
NON-flip: Pin Assignment C/E



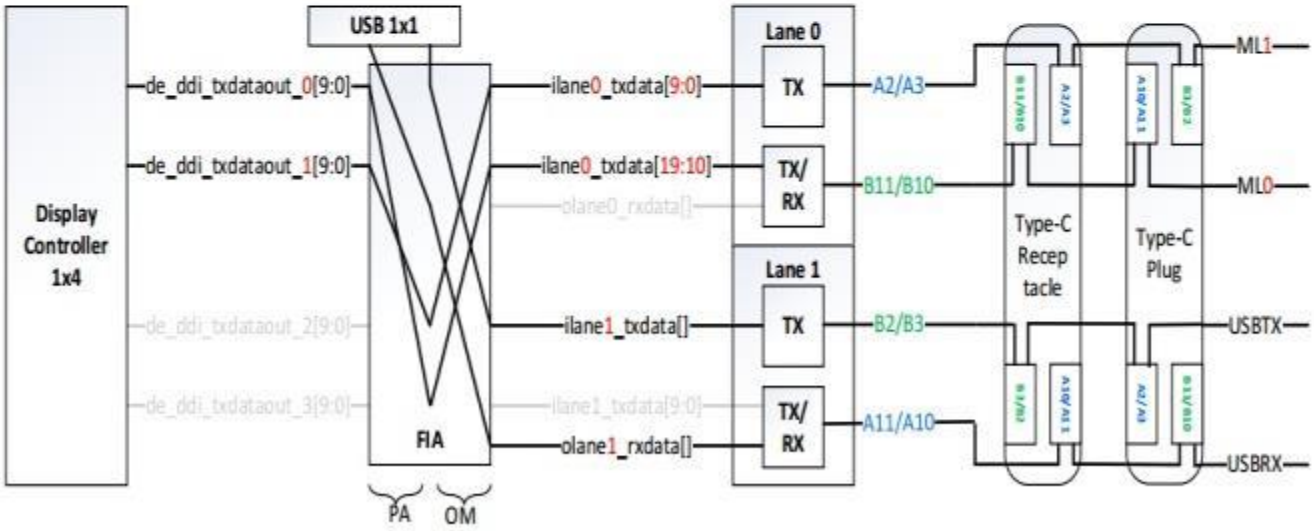
FLIP: Pin Assignment C/E



NON-flip: Pin Assignment D/F



FLIP: Pin Assignment D/F



Dekel PHY Register Access

Addressing Method

Each type-C port PHY is addressed through a 4KB aperture. Each PHY has more than 4KB of register space, so a separate index is programmed in HIP_INDEX_REG0 or HIP_INDEX_REG1, based on the port number, to set the upper 2 address bits that point the 4KB window into the full PHY register space.

Here is a programming example :

1. Assume Type C PHY #1 is allocated 16_8000-16_8FFFh.
2. Assume the Driver wants to write a Phy register 0x1234h.
3. Driver writes at (GTTMMADR+ HIP_INDEX_REG0[3:0] HIP_168_Index) = 01h
4. Driver writes at (GTTMMADR+16_8234h).
    - The write is forwarded to PHY #1 address [15:0] = 1234h

Note: Ignore any address defined within the PHY register. That is incorrect because the index is not accounted for.

PHY Base Addresses and Associated Index

Port #	PHY Base	Index Register and Field
5	0x16C000	HIP_INDEX_REG1 HIP_16C_Index
6	0x16D000	HIP_INDEX_REG1 HIP_16D_Index

MMIO Index Registers

HIP_INDEX_REG0

HIP_INDEX_REG1

PHY Registers

Register Name	PHY Building Block	PHY internal address	MMIO index value	MMIO address
DKL_PLL_DIV0	PLL2 common lane	0x2200	0x2	PHY Base + 0x200
DKL_PLL_DIV1	PLL2 common lane	0x2204	0x2	PHY Base + 0x204
DKL_PLL_LF	PLL2 common lane	0x2208	0x2	PHY Base + 0x208
DKL_SSC	PLL2 common lane	0x2210	0x2	PHY Base + 0x210
DKL_BIAS	PLL2 common lane	0x2214	0x2	PHY Base + 0x214
DKL_TDC_COLDST_BIAS	PLL2 common lane	0x2218	0x2	PHY Base + 0x218
DKL_PLL_FRAC_LOCK	PLL2 common lane	0x220C	0x2	PHY Base + 0x20C
DKL_REFCLKIN_CTL	CMN_ANA	0x212C	0x2	PHY Base + 0x12C
DKL_CLKTOP2_HSCLKCTL	CMN_ANA	0x20D4	0x2	PHY Base + 0x0D4
DKL_CLKTOP2_CORECLKCTL1	CMN_ANA	0x20D8	0x2	PHY Base + 0x0D8
DKL_CMN_UC_DW27	CMN_UC	0x236C	0x2	PHY Base + 0x36C
DKL_TX_DPCNTL0	TX Lane 0	0x02C0	0x0	PHY Base + 0x2C0
DKL_TX_DPCNTL1	TX Lane 0	0x02C4	0x0	PHY Base +

Register Name	PHY Building Block	PHY internal address	MMIO index value	MMIO address
				0x2C4
DKL_TX_DPCNTL2	TX Lane 0	0x02C8	0x0	PHY Base + 0x2C8
DKL_TX_FW_CALIB	TX Lane 0	0x02F8	0x0	PHY Base + 0x2F8
DKL_PMD_FORCE_PCS_IF_TX1	TX Lane 0	0x050C	0x0	PHY Base + 0x50C
DKL_PMD_FORCE_PCS_IF_TX1	TX/RX Lane 0	0x0104	0x0	PHY Base + 0x104
DKL_TX_PMD_LANE_SUS_LN0	TX Lane 0	0x0D00	0x0	PHY Base + 0xD00
DKL_TX_DW17	TX Lane 0	0x0DC4	0x0	PHY Base + 0xDC4
DKL_TX_DW18	TX Lane 0	0x0DC8	0x0	PHY Base + 0xDC8
DKL_TX_DPCNTL0	TX Lane 1	0x12C0	0x1	PHY Base + 0x2C0
DKL_TX_DPCNTL1	TX Lane 1	0x12C4	0x1	PHY Base + 0x2C4
DKL_TX_DPCNTL2	TX Lane 1	0x12C8	0x1	PHY Base + 0x2C8
DKL_TX_FW_CALIB	TX Lane 1	0x12F8	0x1	PHY Base + 0x2F8
DKL_PMD_FORCE_PCS_IF_TX1	TX Lane 1	0x150C	0x1	PHY Base + 0x50C
DKL_PMD_FORCE_PCS_IF_TX1	TX/RX Lane 1	0x1104	0x1	PHY Base + 0x104
DKL_TX_PMD_LANE_SUS_LN1	TX Lane 1	0x1D00	0x1	PHY Base + 0xD00
DKL_TX_DW17	TX Lane 1	0x1DC4	0x1	PHY Base + 0xDC4
DKL_TX_DW18	TX Lane 1	0x1DC8	0x1	PHY Base + 0xDC8
DKL_DP_MODE	ACU Lane 0	0x00A0	0x0	PHY Base + 0x0A0
DKL_DP_MODE	ACU Lane 1	0x10A0	0x1	PHY Base + 0x0A0

DDI AUX Channel

DDI_AUX_CTL

DDI_AUX_DATA

AUX IO Power

Each DisplayPort Aux channel has an Aux IO Power Request. If an Aux channel will not be used, it does not need to be powered up.

PSR spontaneously sends Aux transactions. If PSR is enabled on a port, then the associated Aux IO must be kept powered up.

For Type-C, Aux IO power use has extra requirements. Refer to Type-C Aux power requirements section under Type-C programming page.

AUX IO Power Enabling

1. For combo PHY (non-typeC PHY) Aux channel, the PHY must be initialized.
2. Set **PWR_WELL_CTL_AUX** Aux IO Power Request to 1b.
    - There are two sets of PWR_WELL_CTL_AUX registers for software use. It is expected that BIOS uses PWR_WELL_CTL_AUX1 and driver uses PWR_WELL_CTL_AUX2.
3. Poll for **PWR_WELL_CTL_AUX** Aux IO Power State = 1b.
    - Timeout and fail after 10 us.

AUX IO Power Disabling

1. Clear **PWR_WELL_CTL_AUX**Aux IO Power Request to 0b.

Wait for 10us. Do not poll for the power well to disable. Other clients may be keeping it enabled.

Type-C PHY Microcontroller Health

For every typeC port (static/fixed/legacy/native and DP-alternate, but not thunderbolt), after enabling aux power request and receiving aux power ack, wait for uc_health bit (Dekel PHY register DKL_CMN_UC_DW27 @BASE + 0x236c bit [15]) to assert. Timeout and fail after 10us.

AUX programming sequence

A general purpose AUX functional programming sequence is provided below.

AUX Functional Sequence

Step	Description	Register	Notes
1	Display must already be initialized.		Power well1 enabled cdclk enabled
1a	Aux power enabled		IO powered up as needed for Aux on this project
1b	Power well containing Aux logic powered up	PWR_WELL_CTL	High level power well partitioning shown in display overview diagram
2	Disable PSR1/SRD, PSR2 and GTC if they are enabled on this DDI. Disable DC5 and DC6.	DC_STATE_EN	Power wells will disable automatically for DC5 or DC6. PSR and GTC use Aux spontaneously.
3	Program AUX data registers.	DDI_AUX_DATA_*_[0-4]	
4	Program control to configure AUX and START transaction.	DDI_AUX_CTL_*	Timeout timer value must be at least 600us. To accommodate LT-tunable PHY Repeater AUX delay, AUX Reply Timeout must be programmed to the maximum value. Timer values: 0: 400 us 1: 600 us 2: 800 us 3: 4000 us START trigger: DDI_AUX_CTL_*[31]='1'
5	Wait for AUX transaction complete.		AUX Transaction complete interrupt if set OR when DDI_AUX_CTL_*[31:30] = '01'.
6	Check that receive data has no errors	DDI_AUX_CTL_*[25]	If set: write a '1' to clear this bit and skip reading AUX data registers.
7	Read AUX data register	DDI_AUX_DATA_*_[0-4]	Condition: Aux Channel Control Register Send/Busy bit is NOT asserted
8	Clear status flags	DDI_AUX_CTL_*[30]	Transaction done status

There is a field in DDI_AUX_CTL that must be programmed for the type C ports to select if the Aux transaction will go to thunderbolt

DDI FEC

Reed-Solomon code Forward Error Correction (FEC) function RS (254, 250), with a symbol size of 10 bits is capable of correcting up to two RS symbol errors per FEC block. Display controller when plugged to an FEC-capable DPRX and anticipates enabling FEC encoding sets the FEC_READY bit in the FEC_CONFIGURATION register (DPCD Address 00120h, bit 0) to 1 before initiating link training. Display controller needs to ensure completion of link training before starting FEC encoding. After link training is

complete, display controller, if it needs to enable FEC encoding, shall send an FEC_DECODE_EN sequence to indicate the start of FEC encoding. This prompts DPRX to enable FEC decoding.

When DSC is enabled, FEC shall also be enabled.

DP Support of FEC

DP Support
FEC is supported with DP SST and MST transport modes at all lane widths.

eDP Support of FEC

eDP Support	FEC without PSR	FEC with PSR1	FEC with PSR2
FEC is supported with eDP and all lane widths. PSR restrictions apply.	Yes	Yes	**No**

Global Time Code (GTC)

Global Time Code

Top Level GTC

GTC_CTL

GTC_DDA_M

GTC_DDA_N

GTC_LIVE

GTC Interrupt Bit Definition

GTC_IMR

GTC_IIR

DDI Level GTC

GTC_PORT_CTL

GTC_PORT_TX_CURR

GTC_PORT_TX_PREV

GTC_PORT_MISC

GTC Target Frequency Selection

For GTC top level logic, CDCLK is taken as an input and scaled to a "target frequency" which has a period which is an exact multiple of 0.5ns. This period is also known as the accumulator increment.

Once a target frequency + accumulator increment is selected, an M and N value can be picked and fine tuned to achieve the scaling from CDCLK to target frequency.

In order for the accumulated GTC Live value to match exactly with the real passage of time, the following must be true:

The target frequency selected must be the CLOSEST possible selection to CDCLK. This corresponds to rounding the accumulator increment to the NEAREST 0.5ns increment with respect to the period of CDCLK.

Example

	Frequency	Period
Given CDCLK	337.500 MHz	2.96 ns
1st closest target frequency/increment	333.333 MHz	3.00 ns
2nd closest target frequency/increment	400.000 MHz	2.50 ns
3rd closest target frequency/increment	285.714 MHz	3.50 ns
4th closest target frequency/increment	500.000 MHz	2.00 ns

Only using the 1st closest target frequency 333.333MHz / accumulator increment of 3.00 ns will result in the GTC Live Value correctly tracking the real passage of time.

Example Calculation Flow:

1. Find the period of CDCLK -- for 337.5 MHz = 2.96 ns.
2. Round to the nearest 0.5 ns -- 3.00 ns (this is your accumulator increment value).
3. Find the "target frequency" from the rounded period value of step 2 -- 333.333 MHz is the "target frequency".
4. Find the ratio of target frequency / CDCLK = 0.987654321.
5. Choose M and N to satisfy M / N = same ratio as step 4.

If the results are not within the 1% error tolerance, multiply the accumulator increment used by 2 (i.e., use 1/2 of the target frequency). Re-select M and N as necessary to achieve the new target frequency as shown in the example below.

Example

	Frequency	Period	with 2x accumulator increment	New Period
Given CDCLK	652.800 MHz	1.531 ns		
1st closest target frequency/increment	500.000 MHz	1.500 ns	3.000ns	333.333 MHz

Example Calculation Flow:

1. Find the period of CDCLK -- for 652.800 MHz = 1.531 ns.
2. Round to the nearest 0.5 ns -- 1.500 ns
3. Multiply the above increment by 2 (this 3.00ns is your accumulator increment value).
4. Find the "target frequency" from the rounded period value of step 3 -- 333.333 MHz is the "target frequency".
5. Find the ratio of target frequency / CDCLK = 333.333 MHz / 652.800 MHz = 0.51062.

Choose M and N to satisfy M / N = same ratio as step 5.

Display Watermark Programming

Watermark Overview

The display watermarks are used to control the display engine memory request behavior.

Description
The default settings of the watermark configuration registers will **not** allow the display engine to operate. The watermark values must be properly calculated and programmed in order to enable a display and achieve optimum power and performance. Incorrectly programmed watermark values can result in screen corruption.

The watermarks should be calculated and programmed when any of the watermark calculation inputs change. This includes planes enabling or disabling, plane source format or size changing, etc.

Besides programming the watermark registers, there are other display configuration requirements and registers that must be programmed in order for the display to operate in a low power mode, and there are memory controller configuration requirements which are not documented here.

Watermark Calculations

The display watermarks are calculated using information from the display configuration and memory latencies. The watermarks must be calculated and programmed before enabling a plane or changing a plane configuration.

YUV 420 planar surface format	Plane source bytes per pixel
NV12	1 Bpp for Y surface and 2 Bpp for UV surface. Watermark values must be calculated and programmed for Y and UV surfaces separately in their watermark registers.
P0xx	2 Bpp for Y surface and 4 Bpp for UV surface. Watermark values must be calculated and programmed for Y and UV surfaces separately in their watermark registers.

The ceiling function rounds any non-integer value up to the next greater integer. Example: ceiling[0.3]=1, ceiling[2.1]=3, ceiling[4.8]=5, ceiling[4]=4

Resolutions Requiring Combined Pipes

For resolutions requiring 2 or more pipes to be joined together inside display engine, each pipe processes only 1/<# of joined pipes> of the image. The pixel rate and horizontal total pixels seen by each pipe is 1/<# of joined pipes> of that for the full resolution of the panel.

The excess pixels added for scaling smoothly across the seam between pipes do not impact the watermark.

For example: 7680x4320 CVT1.2 RB1 pixel rate is 2089.75 MHz with horizontal total 7840. That is split across 2 pipes, so each pipe is 3840x4320 with a pixel rate of 1044.875 MHz and horizontal total 3920.

Watermark Algorithm

1. Retrieve memory latency values
    - See the Memory Values section to find the memory latency values
    - The memory values do not change after boot, so software may cache them to avoid re-reading
2. For each enabled pipe (run each time pipe configuration changes)
    A. Calculate adjusted pipe pixel rate
        I. Adjusted pipe pixel rate = pixel rate for the screen resolution
            - If there will be dynamic switching between refresh rates, either use the fastest pixel rate, or re-calculate using the current pixel rate when the refresh rate is switched. Because Plane watermark registers and Link M/N registers have different double buffer update points, software will have to program both at the beginning of vactive. Alternatively, when switching from a low to high refresh rate, program the WM values first and when switching from a high to low refresh rate program the Link M/N values first. This second approach can result in a one frame delay in the refresh rate switch.
            - If plane 90 or 270 rotation is enabled, use the rotated width and height in pixel rate calculations.
        II. If TRANS_CONF Interlaced Mode == PF-ID, adjusted pipe pixel rate = adjusted pipe pixel rate * 2
        III. If pipe scaling enabled, adjusted pipe pixel rate = adjusted pipe pixel rate * pipe down scale amount
            - See the Scaling section to find the down scale amount
    B. Program WM_LINETIME Line Time = roundup[8 * pipe horizontal total pixels / adjusted pipe pixel rate MHz]
3. For each enabled plane (run each time pipe or plane configuration changes)
    A. Calculate adjusted plane pixel rate
        I. Adjusted plane pixel rate = adjusted pipe pixel rate
        II. If plane scaling enabled, adjusted plane pixel rate = adjusted plane pixel rate * plane down scale amount
            - See the Scaling section to find the down scale amount
    B. For each valid memory latency level

Plane Bytes per pixel	Minimum Scanlines for Y Tile	
	0/180 Rotation	90/270 Rotation
1	4	16
2	4	8
4	4	4
8	4	N/A

Plane Memory format	DBuf block size
8 bits per pixel surface format + Yf tiling	256
All other tiling and surface formats	512

I. Calculate method 1
   - Method 1 = (memory latency microseconds * adjusted plane pixel rate MHz * plane source bytes per pixel / DBuf block size) +1

II. Calculate method 2
   - plane bytes per line = plane source width pixels * plane source bytes per pixel
   - Calculate plane blocks per line
     - If plane memory format is Linear
       - plane blocks per line = ceiling[plane bytes per line / DBuf block size] + 1
     - Else If plane memory format is Y tile
       - plane blocks per line = ceiling[(Minimum Scanlines for Y tile * plane bytes per line/ DBuf block size)+1]/Minimum Scanlines for Y tile
     - Else
       - plane blocks per line = ceiling[plane bytes per line / DBuf block size] + 1
   - Method 2 = ceiling[(memory latency microseconds * adjusted plane pixel rate MHz) / Pipe horizontal total number of pixels] * plane blocks per line

III. Calculate Y tile minimum
   - Y tile minimum = Minimum Scanlines for Y tile * plane blocks per line

IV. Select the watermark result
   - line time microseconds = pipe horizontal total pixels/adjusted plane pixel rate MHz
   - If plane memory format is X tile or linear
     - If (((plane source bytes per pixel * pipe horizontal total number of pixels) / DBuf block size) < 1) AND ((plane bytes per line / DBuf block size) < 1) // Special case for unrealistically small horizontal total
       - Selected Result Blocks = Method 2
     - Else If ('plane buffer allocation' is known and (plane buffer allocation / plane blocks per line) >=1)
       - Selected Result Blocks = Method 2
     - Else If (memory latency microseconds >= line time microseconds)
       - Selected Result Blocks = Method 2
     - Else
       - Selected Result Blocks = Method 1
   - Else // Y tile

- Selected Result Blocks = maximum[Method 2, Y tile minimum]
- If Pipe YUV420 Bypass // PIPE_MISC YUV420 Enable == Enable and YUV420 Mode == Bypass
  - Selected Result Blocks = maximum[Selected Result Blocks, plane blocks per line] // Minimum is 1 line when using pipe YUV420 bypass

V. Convert result to blocks and lines
- Result Blocks = ceiling[Selected Result Blocks] + 1
- Result Lines = ceiling[Selected Result Blocks / plane blocks per line]
- If Y Tiling

  If 'Result Lines' is multiple of 'Minimum Scanlines for Y tile'

  Extra Lines = Minimum Scanlines for Y tile

  Else

  Extra Lines = (Minimum Scanlines for Y tile * 2) - (Result Lines % Minimum Scanlines for Y tile)

  Minimum Display Buffer allocation Needed = ceiling[(Result Lines + Extra Lines) * plane blocks per line]

  Else

  Minimum Display Buffer allocation Needed = ceiling[Result Blocks + (Result Blocks * 0.1)]

VI. Compare against the maximum
- If (Result Blocks >= plane buffer allocation), maximum exceeded for this latency level
- Maximum Lines: 31
- If (Result Lines > Maximum Lines), maximum exceeded for this latency level
- If (Minimum Display Buffer allocation Needed >= plane buffer allocation), maximum exceeded for this latency level
- or YUV 420 Planar formats, perform the above check for both Y and UV planes.

4. For each transition watermark
   - This includes<PLANE,CUR>_WM_TRANS, which is calculated relative to memory latency level 0.

A. Calculate transition offset
   - Transition Offset Blocks = Transition minimum + Transition amount
   - See Transition Watermark section for transition minimum and transition amount

B. Calculate transition Y tile minimum
   - Transition Y tile minimum = 2 * relative memory latency level Y tiled minimum

C. Select the watermark result
   - If plane memory format is X tile or linear

- Result Blocks = Relative memory latency level Selected Result Blocks + Transition Offset Blocks
- Else // Y tile
  - Result Blocks = maximum[Relative memory latency level Selected Result Blocks, Transition Y tile minimum] + Transition Offset Blocks

D. Convert result to blocks
- Result Blocks = ceiling[Result Blocks] + 1

E. Compare against the maximum
- If (Result Blocks >= plane buffer allocation), maximum exceeded for transition watermark
- For YUV 420 Planar formats, perform the above check for both Y and UV planes.

5. Program watermark registers

A. For each latency level
- If memory latency for this level is invalid, or the maximum was exceeded for this level or any previous level, program PLANE_WM_<latency level> Enable = 0
  - **If watermark latency level 0 exceeds the maximum, the plane must not be enabled.**
- Else program PLANE_WM_<latency level> Enable = 1, Lines = Result Lines, Blocks = Result Blocks

B. For transition watermarks
- If the maximum was exceeded for the transition watermark, program the transition watermark register Enable = 0
- Else program the transition watermark register Enable = 1, Blocks = Transition Result Blocks
  - The transition watermark Lines value is ignored by hardware

C. Write the plane surface base address register to trigger update of the watermarks and other plane double buffered registers. This should be done only after all plane configuration is configured to match the new watermark values.

Transition Watermark

The transition watermark is used for Isochronous Priority Control (IPC). When IPC is enabled, plane read requests are sent at high priority until filling above the transition watermark, then the requests are sent at lower priority until dropping below the level 0 watermark. The lower priority requests allow other memory clients to have better memory access. If the transition watermark is not enabled, the plane behaves as if the transition watermark was programmed to the top of the plane buffer allocation. When IPC is disabled, all plane read requests are sent at high priority. It is allowed for one of a pair of planes supporting YUV420 together to have transition watermark disabled while the other plane has it enabled.

The transition watermark is programmed as a tunable amount above the relative memory latency level watermark. Tuning to higher values will tend to cause longer periods of high priority reads followed by

longer periods of lower priority reads. Tuning to lower values will tend to cause shorter periods of high and lower priority reads. The exact behavior depends on the memory bandwidth, display bandwidth, and other memory traffic in the system.

The transition watermark has a minimum value to ensure the demote does not happen before enough data has been read to meet the relative memory latency level watermark requirements.

Transition Minimum: 4 Blocks

IPC Enable Register Field
ARB_CTL2 Enable IPC

Scaling

A scaler (pipe or plane scaler) is down scaling when it is enabled and the scaler input size is greater than the scaler output size.

Down scaling effectively increases the pixel rate. Up scaling does not reduce the pixel rate.

For plane scaling, the scaler input size is the plane size and the output size is the scaler window size.

For pipe scaling, the scaler input size is the pipe source size and the output size is the scaler window size.

Horizontal down scale amount = maximum[1, Horizontal source size / Horizontal destination size]

Vertical down scale amount = maximum[1, Vertical source size / Vertical destination size]

Total down scale amount = Horizontal down scale amount * Vertical down scale amount

System Agent Geyserville (SAGV) Interaction with Watermarks

SAGV dynamically adjusts the system agent voltage and clock frequencies depending on power and performance requirements. The display engine access to system memory is blocked outside of display engine, without a PM handshake, during the adjustment time.

To compensate for the block, display engine must increase the latency for watermark level 0 by the SAGV block time. If there is not enough data buffer to support that increase, then software must disable SAGV. Other watermark levels do not need increase because after a package C exit SAGV will not run until after display has refilled above watermark level 0.

SAGV defaults to enabled.

See the Memory Values section to find the SAGV block time.

The latency input to the watermark calculation for each level must be greater than or equal to the lower level. The latency increase to level 0 for SAGV requires the upper levels to be adjusted to meet that requirement. Use MIN(latency for this level, latency for next lower level) to correct the latency.

Requirement before plane enabling or configuration change: Calculate watermark level 0 with level 0 latency + SAGV block time. If the result can be supported (does not exceed maximum), then the plane can tolerate SAGV, so proceed with watermark calculations for level 0 using latency increased by SAGV block time.

If the plane cannot tolerate SAGV, proceed with watermark calculations for all levels using the normal latency

values.

Disable SAGV (sequence below) if any enabled plane cannot tolerate SAGV, or any transcoder is interlaced. Else, enable SAGV.

The expectation is that SAGV can be tolerated in most non-interlaced configurations and will only need to be disabled for some high resolution, multiple display scenarios.

If one plane was already enabled and able to tolerate SAGV, then a second plane becomes enabled and cannot tolerate SAGV, causing SAGV to disable, software can recalculate and reprogram the watermarks for the first plane without the SAGV block time. That reprogramming is preferred to give more optimal watermark results, but not required.

To disable SAGV, follow the SAGV section SAGV Point Selection Runtime flow to mask off all but the one QGV point that supplies the highest bandwidth for display.

To enable SAGV, follow the SAGV section SAGV Point Selection Runtime flow to unmask all of the QGV points that supply enough bandwidth for the display configuration.

Examples

Example pixel rate adjustments:

Pixel rate for screen resolution is 130 MHz. No interlacing. Pipe scale 1920x1080 pipe source size to 1714x1120 scaler window size. Plane scale 1920x1080 plane size to 800x600 scaler window size.

Pipe horizontal down scale amount = maximum[1, 1920 / 1714] = 1.12

Pipe vertical down scale amount = maximum[1, 1080 / 1120] = 1 **// Max condition was hit**

Pipe total down scale amount = 1.12 * 1 = 1.12

Adjusted pipe pixel rate = 130 MHz * 1.12 = 145.6 MHz

Plane horizontal down scale amount = maximum[1, 1920 / 800] = 2.4

Plane vertical down scale amount = maximum[1, 1080 / 600] = 1.8

Plane total down scale amount = 2.4 * 1.8 = 4.32

Adjusted plane pixel rate = 145.6 MHz * 4.32 = 628.99 MHz

Example method, block, and line calculations:

Plane source 4 Bpp, Plane X tile, Plane source width 1920 pixels, Horizontal total 2200 pixels, Adjusted plane pixel rate 148.5 MHz, memory latency 7.5 us

Method 1 = 148.5 MHz * 4 Bpp * 7.5 us / 512 = 8.7 blocks

Plane bytes per line = 1920 pixels * 4 Bpp = 7680 Bytes/line

Plane blocks per lines = ceiling[7680 / 512] = 15 blocks

Method 2 = ceiling[(7.5 us * 148.5 MHz) / 2200 pixels] * 15 blocks = 15 blocks

Y tile minimum = 4 * 15 blocks = 60 blocks

Result Blocks = minimum[8.7 blocks, 15 blocks] = 8.7 blocks // X tile so does not use Y tile minimum

Result Blocks = ceiling[8.7 blocks] + 1 block = 10 blocks

Result Lines = ceiling[8.7 blocks / 15] = 1 lines

Memory Values

Retrieve Memory Latency Data

1. Ensure any previous GT Driver Mailbox transaction is complete.
2. Write GT Driver Mailbox Data0=0x0000_0000 (first set of latency values) and GT Driver Mailbox Data1=0x0000_0000
3. Write GT Driver Mailbox Interface Run/Busy=1, Address Control=All 0s, Command/Error Code=06h
4. Poll GT Driver Mailbox Interface for Run/Busy indication=0b and Command/Error Code=00h (success)
    - Timeout after 100 us and do not enable display planes.
5. Read GT Driver Mailbox Data0 for the first set of memory latency values
6. Write GT Driver Mailbox Data0=0x0000_0001 (second set of latency values) and GT Driver Mailbox Data1=0x0000_0000
7. Write GT Driver Mailbox Interface Run/Busy=1, Address Control=All 0s, Command/Error Code=06h
8. Poll GT Driver Mailbox Interface for Run/Busy indication=0b and Command/Error Code=00h (success)
    - Timeout after 100 us and do not enable display planes.
9. Read GT Driver Mailbox Data0 for the second set of memory latency values

Memory Latency Data Definition

First Set:

Data0 Bit	Name	Description
31:24	Level 3	Number of microseconds for level 3.
23:16	Level 2	Number of microseconds for level 2.
15:8	Level 1	Number of microseconds for level 1.
7:0	Level 0	Number of microseconds for level 0.

Second Set:

Data0 Bit	Name	Description
31:24	Level 7	Number of microseconds for level 7.
23:16	Level 6	Number of microseconds for level 6.
15:8	Level 5	Number of microseconds for level 5.

Data0 Bit	Name	Description
7:0	Level 4	Number of microseconds for level 4.

If level 1 or any higher level has a value of 0x00, that level and any higher levels are unused and invalid, so the associated watermark registers must not be enabled. Level 0 is always valid.

It is allowed to have the same value in adjacent levels.

Programming Note	
Context:	Display Watermark Programming
The mailbox response data may not account for memory read latency. If the mailbox response data for level 0 is 0us, add 3 microseconds to the result for each valid level. The previous requirement was 2us, but high density DIMMs have increased refresh times that requires a 1us increase.	

SAGV Block Time

The block time is retrieved from the GT driver pcode mailbox.

1. Ensure any previous GT Driver Mailbox transaction is complete.
2. Write GT Driver Mailbox Data0=0x0000_0000 (first set of latency values) and GT Driver Mailbox Data1=0x0000_0000
3. Write GT Driver Mailbox Interface Run/Busy=1, Address Control=All 0s, Command/Error Code=23h
4. Poll GT Driver Mailbox Interface for Run/Busy indication=0b and Command/Error Code=00h (success)
    - Timeout after 100 us and do not enable display planes.
5. Read GT Driver Mailbox Data0 for the block time in microseconds

Intel® Iris® Xe and UHD Graphics Open Source

Programmer's Reference Manual

For the 2020-2021 11th Generation Intel Xeon®, Core™, Celeron®, Pentium® Gold Processors based on the "Tiger Lake" Platform

Volume 13: General Assets

December 2021, Revision 1.0

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Table of Contents

General Assets 1

MMIO 1

Force Wake and Steering Table 1

Multicast Steering and Die Recovery 9

SW Virtualization Reserved MMIO range 11

Register Address Maps 11

Graphics Register Address Map 11

VGA and Extended VGA Register Map 11

GUC 15

GuC Introduction 15

Arming Doorbells 17

GuC Shim (GUCSHIM) Register Functions 18

GUCSHIM Registers 19

Guc DMA (GUCDMA) 20

GuC Interrupt (GUCINT) Register Functions 22

Observability 25

Observability Overview 25

GT Power-up/RC6 Exit 25

Render Engine Power-up 25

Media Engine Power-up 25

Resume From Partial GT Power Down 26

Trace 26

GPU Doorbells 43

The Concept and Usage Flow 44

Doorbells and Lock 50

Interrupts Overview: 50

GT Engine Interrupts: 52

Hardware Scheduler/MinIA SW Interface 53

Host SW Interface 53

Interrupt Aggregating Logic 54

General Assets

This is the General Assets section.

MMIO

Force Wake and Steering Table

MMIO Range Start	MMIO Range End	# Bytes	Wake Target	Replicated / Multicast ?	Replication Group Type	Inst. Count	Steering
00000000	00000AFF	2816					
00000B00	00000BFF	256	AON	Yes	SQIDI	2	subsliceid[0..1]
00000C00	00000DFF	512	AON	No	-	1	-
00000E00	00000FFF	512	AON	No	-	1	-
00001000	00001FFF	4096	AON	Yes	SQIDI	2	subsliceid[0..1]
00002000	000026FF	1792	RENDER	No	-	1	-
00002700	000027FF	256	GT	No	-	1	-
00002800	00002AFF	768	RENDER	No	-	1	-
00002B00	00002FFF	1280	GT	No	-	1	-
00003000	00003FFF	4096	RENDER	No	-	1	-
00004000	000041FF	512	GT	No	-	1	-
00004200	000043FF	512	GT	No	-	1	-
00004400	000048FF	1280	GT	No	-	1	-
00004900	00004FFF	1792					
00005000	000051FF	512					
00005200	000052FF	256	RENDER	No	-	1	-
00005300	000053FF	256	RENDER	No	-	1	-
00005400	000054FF	256					
00005500	00005FFF	2816	RENDER	No	-	1	-
00006000	00006FFF	4096	RENDER	No	-	1	-
00007000	00007FFF	4096	RENDER	No	-	1	-
00008000	000080FF	256	GT	No	-	1	-
00008100	0000813F	64	GT	No	-	1	-
00008140	0000814F	16	RENDER	No	-	1	-
00008150	0000815F	16	RENDER	Yes	DSS	6	subsliceid[0..5]
00008160	0000817F	32					
00008180	000081FF	128	AON	No	-	1	-
00008200	000082FF	256	GT	No	-	1	-
00008300	000084FF	512	RENDER	No	-	1	-
00008500	000085FF	256	GT	No	-	1	-
00008600	000086FF	256	GT	No	-	1	-
00008700	000087FF	256	GT	Yes	SQIDI	2	subsliceid[0..1]
00008800	00008FFF	2048					
00009000	000093FF	1024	GT	No	-	1	-
00009400	0000947F	128	GT	No	-	1	-
00009480	000094CF	80					

000094D0	0000951F	80	RENDER	No	-	1	-
00009520	0000955F	64	RENDER	Yes	DSS	6	subsliceid[0..5]
00009560	000095FF	160	AON	No	-	1	-
00009600	000097FF	512					
00009800	00009FFF	2048	GT	No	-	1	-
0000A000	0000AFFF	4096	GT	No	-	1	-
0000B000	0000B0FF	256	RENDER	No	-	1	-
0000B100	0000B3FF	768	RENDER	Yes	L3BANK	8	subsliceid[0..7]
0000B400	0000B47F	128	GT	No	-	1	-
0000B480	0000BFFF	2944					
0000C000	0000C7FF	2048	GT	No	-	1	-
0000C800	0000CFFF	2048	GT	No	-	1	-
0000D000	0000D3FF	1024	AON	No	-	1	-
0000D400	0000D7FF	1024	AON	No	-	1	-
0000D800	0000D8FF	256	RENDER	No	-	1	-
0000D900	0000DBFF	768	GT	No	-	1	-
0000DC00	0000DDFF	512	RENDER	No	-	1	-
0000DE00	0000DE7F	128					
0000DE80	0000DEFF	128	RENDER	Yes	DSS	6	subsliceid[0..5]
0000DF00	0000DFFF	256	RENDER	Yes	DSS	6	subsliceid[0..5]
0000E000	0000E0FF	256	RENDER	Yes	DSS	6	subsliceid[0..5]
0000E100	0000E1FF	256	RENDER	Yes	DSS	6	subsliceid[0..5]
0000E200	0000E3FF	512	RENDER	Yes	DSS	6	subsliceid[0..5]
0000E400	0000E7FF	1024	RENDER	Yes	DSS	6	subsliceid[0..5]
0000E800	0000E8FF	256	RENDER	Yes	DSS	6	subsliceid[0..5]
0000E900	0000EFFF	1792					
0000F000	0000F0FF	256	GT	No	-	1	-
0000F100	0000FFFF	3840	GT	No	-	1	-
00010000	000147FF	18432					
00014800	00014FFF	2048	RENDER	No	-	1	-
00015000	00016DFF	7680					
00016E00	00016FFF	512	RENDER	No	-	1	-
00017000	00017FFF	4096	RENDER	No	-	1	-
00018000	00019FFF	8192	RENDER	No	-	1	-
0001A000	0001BFFF	8192	RENDER	No	-	1	-
0001C000	0001DFFF	8192					
0001E000	0001FFFF	8192					
00020000	00020FFF	4096	VD0	No	-	1	-
00021000	00021FFF	4096	VD2	No	-	1	-
00022000	00022FFF	4096	GT	No	-	1	-
00023000	00023FFF	4096	GT	No	-	1	-
00024000	0002407F	128	AON	No	-	1	-
00024080	0002417F	256					
00024180	000241FF	128	GT	No	-	1	-
00024200	000249FF	2048					

00024A00	00024A7F	128	RENDER	Yes	DSS	6	subsliceid[0..5]
00024A80	000251FF	1920					
00025200	0002527F	128	GT	No	-	1	-
00025280	000252FF	128	GT	No	-	1	-
00025300	000255FF	768					
00025600	0002567F	128	VD0	No	-	1	-
00025680	000256FF	128	VD2	No	-	1	-
00025700	000259FF	768					
00025A00	00025A7F	128	VD0	No	-	1	-
00025A80	00025AFF	128	VD2	No	-	1	-
00025B00	00025FFF	1280					
00026000	00027FFF	8192					
00028000	0002FFFF	32768					
00030000	0003FFFF	65536	GT	No	-	1	-

MMIO Range Start	MMIO Range End	# Bytes	Wake Target	Replicated / Multicast ?	Replication Group Type	Inst. Count	Steering
001C0000	001C07FF	2048	VD0	No	-	1	-
001C0800	001C0FFF	2048	VD0	No	-	1	-
001C1000	001C1FFF	4096	VD0	No	-	1	-
001C2000	001C27FF	2048	VD0	No	-	1	-
001C2800	001C2AFF	768	VD0	No	-	1	-
001C2B00	001C2BFF	256	VD0	No	-	1	-
001C2C00	001C2CFF	256					
001C2D00	001C2DFF	256	VD0	No	-	1	-
001C2E00	001C3EFF	4352					
001C3F00	001C3FFF	256	VD0	No	-	1	-
001C4000	001C47FF	2048					
001C4800	001C4FFF	2048					
001C5000	001C5FFF	4096					
001C6000	001C67FF	2048					
001C6800	001C6AFF	768					
001C6B00	001C6BFF	256					
001C6C00	001C6CFF	256					
001C6D00	001C6DFF	256					
001C6E00	001C7EFF	4352					
001C7F00	001C7FFF	256					
001C8000	001C9FFF	8192	VE0	No	-	1	-
001CA000	001CA0FF	256	VE0	No	-	1	-
001CA100	001CBEFF	7680					
001CBF00	001CBFFF	256	VE0	No	-	1	-
001CC000	001CCFFF	4096	VD0	No	-	1	-

001CD000	001CDFFF	4096					
001CE000	001CEFFF	4096					
001CF000	001CFFFF	4096					
001D0000	001D07FF	2048	VD2	No	-	1	-
001D0800	001D0FFF	2048	VD2	No	-	1	-
001D1000	001D1FFF	4096	VD2	No	-	1	-
001D2000	001D27FF	2048	VD2	No	-	1	-
001D2800	001D2AFF	768	VD2	No	-	1	-
001D2B00	001D2BFF	256	VD2	No	-	1	-
001D2C00	001D2CFF	256					
001D2D00	001D2DFF	256	VD2	No	-	1	-
001D2E00	001D3EFF	4352					
001D3F00	001D3FFF	256	VD2	No	-	1	-
001D4000	001D47FF	2048					
001D4800	001D4FFF	2048					
001D5000	001D5FFF	4096					
001D6000	001D67FF	2048					
001D6800	001D6AFF	768					
001D6B00	001D6BFF	256					
001D6C00	001D6CFF	256					
001D6D00	001D6DFF	256					
001D6E00	001D7EFF	4352					
001D7F00	001D7FFF	256					
001D8000	001D9FFF	8192					
001DA000	001DA0FF	256					
001DA100	001DBEFF	7680					
001DBF00	001DBFFF	256					
001DC000	001DFFFF	16384					
001E0000	001E07FF	2048					
001E0800	001E0FFF	2048					
001E1000	001E1FFF	4096					
001E2000	001E27FF	2048					
001E2800	001E2AFF	768					
001E2B00	001E2BFF	256					
001E2C00	001E2CFF	256					
001E2D00	001E2DFF	256					
001E2E00	001E3EFF	4352					
001E3F00	001E3FFF	256					
001E4000	001E47FF	2048					
001E4800	001E4FFF	2048					
001E5000	001E5FFF	4096					

001E6000	001E67FF	2048					
001E6800	001E6AFF	768					
001E6B00	001E6BFF	256					
001E6C00	001E6CFF	256					
001E6D00	001E6DFF	256					
001E6E00	001E7EFF	4352					
001E7F00	001E7FFF	256					
001E8000	001E9FFF	8192					
001EA000	001EA0FF	256					
001EA100	001EBEFF	7680					
001EBF00	001EBFFF	256					
001EC000	001EFFFF	16384					
001F0000	001F07FF	2048					
001F0800	001F0FFF	2048					
001F1000	001F1FFF	4096					
001F2000	001F27FF	2048					
001F2800	001F2AFF	768					
001F2B00	001F2BFF	256					
001F2C00	001F2CFF	256					
001F2D00	001F2DFF	256					
001F2E00	001F3EFF	4352					
001F3F00	001F3FFF	256					
001F4000	001F47FF	2048					
001F4800	001F4FFF	2048					
001F5000	001F5FFF	4096					
001F6000	001F67FF	2048					
001F6800	001F6AFF	768					
001F6B00	001F6BFF	256					
001F6C00	001F6CFF	256					
001F6D00	001F6DFF	256					
001F6E00	001F7EFF	4352					
001F7F00	001F7FFF	256					
001F8000	001F9FFF	8192					
001FA000	001FA0FF	256					
001FA100	001FBEFF	7680					
001FBF00	001FBFFF	256					
001FC000	001FFFFF	16384					
00200000	0023FFFF	262144					

- The Steering Control Registers reside at the following locations:
    - MGSR access point (access initiated by agent outside of GT):

#	Steering Reg Addr	Description
1	0xFD0	Access steering towards MCFG endpoints only.
2	0xFD4	Access steering towards MDRB endpoints only
3	0xFD8	Access steering towards SF endpoints only
4	0xFDC	Access steering towards all other endpoints (all but above)

- GuC access point:

#	Steering Reg Addr	Description
1	0xC060	Access steering towards all GT endpoints

- CS access point:

#	Steering Reg Addr	Description
1	0x20CC	Access steering towards all GT endpoints

- **Note:** All Steering Control Registers contain the following fields:

Field	Description
multicast	1: Access will be multicast to all replicated endpoints: • *WRITE* op cycles go to all endpoint instances; sliceid[]/subsliceid[] fields ignored. • *READ* op cycles go to all endpoint instances, and responses are returned from all instances; The MsgCh selects single instance's response as the final read return, based on sliceid[]/subsliceid[] fields. 0: Access will be steered using sliceid[] and subsliceid[] fields below: • Both *WRITE* and *READ* cycles go to a single instance of an endpoint, based on sliceid[]/subsliceid[] steering. Default: 1 Note: The multicast field has no impact for a non-replicated target.
sliceid[]	Default: 0
subsliceid[]	Default: 0

- The following Replication Group Types exist for multicast MMIO endpoints:

Replication Group Type	Description / Notes
SQIDI	• 2 instances • subsliceid: 0..1 • all instances are always present.
DSS	• LP has max 6 DSS • subsliceid: 0..5 • Terminated/disabled when the corresponding dss_enable bit is '0'
L3BANK	• 8 instances • subsliceid 0..7 to access • Terminated/disabled when corresponding fuse_gt_l3disable bit is 'disable'

- Fuse reflections (how to tell when an endpoint is disabled):

Fuse	Register reflection
fuse_gt_dssen[5:0]	0x913C[5:0]
fuse_gt_l3_disable[3:0]	0x9118[7:0] (fuse is replicated into [3:0] and [7:4])

Note: MsgCh termination also occurs when the domains are powered down. (i.e., not necessarily because the domain is disabled/fused off.) If reading/writing the registers is needed, then force-wake of the domain is required. Force-wake is not required for shadow register accesses coming through MGSR.

- The following table captures the force-wake and corresponding acknowledgment register locations for the various domains:

Domain	Driver ForceWake Req	Driver ForceWake Ack	GuC ForceWake Req	GuC ForceWake Status	Comment
AON	NA	NA	NA	NA	Registers sit outside of the C6 boundary. No ForceWake required.
GT	0xA188	0x00130044	NA	NA	
Render	0xA278	0x0D84	0xA27C	0xA2A0[1]	
VDBOX0	0xA540	0x0D50	0xA274[0]	0xA2A0[0]	

VDBOX1	0xA544	0x0D54	0xA274[1]	0xA2A0[0]	
VDBOX2	0xA548	0x0D58	0xA274[2]	0xA2A0[2]	
VDBOX3	0xA54C	0x0D5C	0xA274[3]	0xA2A0[2]	As available in the product
VDBOX4	0xA550	0x0D60	0xA274[4]	0xA2A0[3]	As available in the product
VDBOX5	0xA554	0x0D64	0xA274[5]	0xA2A0[3]	As available in the product
VDBOX6	0xA558	0x0D68	0xA274[6]	0xA2A0[4]	As available in the product
VDBOX7	0xA55C	0x0D6C	0xA274[7]	0xA2A0[4]	As available in the product
VEBOX0	0xA560	0x0D70	0xA274[8]	0xA2A0[0]	As available in the product
VEBOX1	0xA564	0x0D74	0xA274[9]	0xA2A0[2]	As available in the product
VEBOX2	0xA568	0x0D78	0xA274[10]	0xA2A0[3]	As available in the product
VEBOX3	0xA56C	0x0D7C	0xA274[11]	0xA2A0[4]	As available in the product

- Miscellaneous Notes:
- The MsgCh network has termination points, where cycles to endpoints that are disabled (fused-off, powered off, etc…) are gracefully completed. The termination node on the network will sink P cycles, and return dummy completions for NP cycles, on behalf of the disabled endpoints.
- Access requirements to registers that are part of GTMMADDR but not listed in the GT MMIO map table is defined elsewhere. This descriptions in this document only cover GT range (GT MMIO map xls.)

Multicast Steering and Die Recovery

Some units in GT are replicated multiple times in the design, each with their own register storage local to that instance.

- In some cases, each replica/instance gets its own MMIO address range of offsets – for example, the multiple CCS command streamers, multiple VDBox/VEBox instances. For those, direct register access targets the only instance of that registers. The programming model described on this page is moot for those cases where each register has unique address.
- In other cases, the multiple instances of the unit use the same MMIO address on message channel. For these cases, the message channel provides additional capabilities to address the instances for read/write operations in either multicast (targeting all instances) or unicast modes (target specific instance) via a set of "steering registers" which can be configured to direct the access as desired. The steering registers have 3 fields: Multicast/Unicast, Sliceid, Subsliceid.
    - Multicast write access - write goes to all instances; sliceid/subsliceid fields are ignored
    - Multicast read access – read goes to all instances and all instances generate read response; message channel selects single instance's response as the final read return (based on the steering register slice/subslice fields)
    - Unicast write access – write goes to only the instance specified in the steering register
    - Unicast read access – read goes to only the instance specified in the steering register
    - In some replicated units, all of the replicated instances always "enabled" from a message channel perspective (never fused off/separately power gated) and thus all instances are always accessible if the containing power well is on (e.g. if GT is out of RC6)
    - In some replicated units, there are die recovery/fuse down modes where some instances are fused off/disabled. For the latter, GT also contains MMIO registers which allow SW to detect which instances are fused as enabled/disabled (generally 1-hot). When this fuse down case applies, message channel is aware of the fusing and provides automatic termination of cycles toward disabled instances (writes get dropped with dummy NP completion if NP write; reads get dummy completion with 0 read return value from that instance). The fuse mirror register provides a mechanism for SW to know which instances are valid and to program the steering register toward enabled instances when needed – see comments below.

General rules:

- Some of these replicated registers are control registers which are generally expected to be all programmed with the same value – for these, writes should generally be multicast and reads can target any enabled instance (since all instances should contain the same value from prior multicast write).

- Some replicated registers are status registers and are expected to have different values as part of normal usage (for example, INSTDONE registers related to Sampler, Slice common; TDL thread

status, etc). For these typical usage model would be to either iterate over all enabled instances or select specific single instance to target.

- If an instance is disabled (access terminated on message channel via the fuse info above or if containing power well is power gated), reads from that instance will return 0s and writes are silently dropped. Since the default for the steering registers is multicast read with sliceid=subsliceid=0, the default hardware behavior is to return data from instance that corresponds with sliceid/subsliceid = 0. If that instance is disabled, message channel will return a dummy response (0). In order to get correct/valid value the steering registers must be used to access a valid instance.
    - Note that a common usage model is for SW/FW to initializing specific bits in control register by reading the current/default value, then modifying the value in memory (set/clear few bits), and then write the result back.
    - For these cases, SW must ensure that it uses the steering registers to steer to an enabled instance when performing the initial read.
- When performing engine and power context save restore, GT hardware is aware of the fuses and internally targets reads for context save toward the first enabled instance.
- In cases where steering registers are being programmed, caution must be exercised to ensure that there is no race condition/concurrent access between two different initiators using a given steering register. SW must protect against concurrent access by multiple threads to any given steering register. System level flows must also guard against concurrent access by Firmware (CSC/FSP FW, Punit pCode) and driver tools to any given steering register.
    - Multicast is the hardware default. If an agent sets a steering register to unicast mode, they should generally set it back to multicast after completion.
    - In some projects there are separate steering registers listed are intended to allow for some degree of concurrency between different usages targeting different destinations in GT by replication group.
        - MGSR uses the MMIO offset requested in the inbound cycle to select which steering register to use for routing.
        - MGSR uses SAI policy registers to identify sources as “IA” (low privilege cfg_src on message channel) vs “HW” (high privilege – includes trusted firmware such as CSC/FSP, Pcode)
        - See project specific documentation for the list of steering registers and their intended use.

SW Virtualization Reserved MMIO range

The MMIO address range from 0x178000 thru 0x178FFF is reserved for communication between a VMM and the GPU Driver executing on a Virtual Machine.

HW does not actually implement anything within this range. Instead, in a SW Virtualized environment, if a VM driver issues a read to this MMIO address range, the VMM will trap that access, and provide whatever data it wishes to pass to the VM driver. In a non-SW-Virtualizated environment (including an SR-IOV Virtualized environment), reads will return zeros, like any other unimplemented MMIO address. Writes to this range are always ignored.

It is important that no "real" HW MMIO register be defined within this range, as it would be inaccessable in a SW-virtualized environment.

Register Address Maps

Graphics Register Address Map

This chapter provides address maps of the graphics controllers I/O and memory-mapped registers. Individual register bit field descriptions are provided in the following chapters. PCI configuration address maps and register bit descriptions are provided in the following chapter.

VGA and Extended VGA Register Map

For I/O locations, the value in the address column represents the register I/O address. For memory mapped locations, this address is an offset from the base address programmed in the MMADR register.

VGA and Extended VGA I/O and Memory Register Map

Address	Register Name (Read)	Register Name (Write)
2D Registers		
3B0h-3B3h	Reserved	Reserved
3B4h	VGA CRTC Index (CRX) (monochrome)	VGA CRTC Index (CRX) (monochrome)
3B5h	VGA CRTC Data (monochrome)	VGA CRTC Data (monochrome)
3B6h-3B9h	Reserved	Reserved
3Bah	VGA Status Register (ST01)	VGA Feature Control Register (FCR)
3BBh-3BFh	Reserved	Reserved
3C0h	VGA Attribute Controller Index (ARX)	VGA Attribute Controller Index (ARX)/ VGA Attribute Controller Data (alternating writes select ARX or write ARxx Data)
3C1h	VGA Attribute Controller Data (read ARxx data)	Reserved
3C2h	VGA Feature Read Register (ST00)	VGA Miscellaneous Output Register (MSR)

Address	Register Name (Read)	Register Name (Write)
3C3h	Reserved	Reserved
3C4h	VGA Sequencer Index (SRX)	VGA Sequencer Index (SRX)
3C5h	VGA Sequencer Data (SRxx)	VGA Sequencer Data (SRxx)
3C6h	VGA Color Palette Mask (DACMASK)	VGA Color Palette Mask (DACMASK)
3C7h	VGA Color Palette State (DACSTATE)	VGA Color Palette Read Mode Index (DACRX)
3C8h	VGA Color Palette Write Mode Index (DACWX)	VGA Color Palette Write Mode Index (DACWX)
3C9h	VGA Color Palette Data (DACDATA)	VGA Color Palette Data (DACDATA)
3CAh	VGA Feature Control Register (FCR)	Reserved
3CBh	Reserved	Reserved
3CCh	VGA Miscellaneous Output Register (MSR)	Reserved
3CDh	Reserved	Reserved
3CEh	VGA Graphics Controller Index (GRX)	VGA Graphics Controller Index (GRX)
3CFh	VGA Graphics Controller Data (GRxx)	VGA Graphics Controller Data (GRxx)
3D0h-3D1h	Reserved	Reserved
2D Registers		
3D4h	VGA CRTC Index (CRX)	VGA CRTC Index (CRX)
3D5h	VGA CRTC Data (CRxx)	VGA CRTC Data (CRxx)
System Configuration Registers		
3D6h	GFX/2D Configurations Extensions Index (XRX)	GFX/2D Configurations Extensions Index (XRX)
3D7h	GFX/2D Configurations Extensions Data (XRxx)	GFX/2D Configurations Extensions Data (XRxx)
2D Registers		
3D8h-3D9h	Reserved	Reserved
3DAh	VGA Status Register (ST01)	VGA Feature Control Register (FCR)
3DBh-3DFh	Reserved	Reserved

Indirect VGA and Extended VGA Register Indices

The registers listed in this section are indirectly accessed by programming an index value into the appropriate SRX, GRX, ARX, or CRX register. The index and data register address locations are listed in the previous section. Additional details concerning the indirect access mechanism are provided in the *VGA and Extended VGA Register Description* Chapter (see SRxx, GRxx, ARxx or CRxx sections).

2D Sequence Registers (3C4h / 3C5h)

Index	Sym	Description
00h	SR00	Sequencer Reset
01h	SR01	Clocking Mode
02h	SR02	Plane / Map Mask
03h	SR03	Character Font
04h	SR04	Memory Mode
07h	SR07	Horizontal Character Counter Reset

2D Graphics Controller Registers (3CEh / 3CFh)

Index	Sym	Register Name
00h	GR00	Set / Reset
01h	GR01	Enable Set / Reset
02h	GR02	Color Compare
03h	GR03	Data Rotate
04h	GR04	Read Plane Select
05h	GR05	Graphics Mode
06h	GR06	Miscellaneous
07h	GR07	Color Don't Care
08h	GR08	Bit Mask
10h	GR10	Address Mapping
11h	GR11	Page Selector
18h	GR18	Software Flags

2D Attribute Controller Registers (3C0h / 3C1h)

Index	Sym	Register Name
00h	AR00	Palette Register 0
01h	AR01	Palette Register 1
02h	AR02	Palette Register 2
03h	AR03	Palette Register 3
04h	AR04	Palette Register 4
05h	AR05	Palette Register 5
06h	AR06	Palette Register 6
07h	AR07	Palette Register 7
08h	AR08	Palette Register 8
09h	AR09	Palette Register 9
0Ah	AR0A	Palette Register A
0Bh	AR0B	Palette Register B

Index	Sym	Register Name
0Ch	AR0C	Palette Register C
0Dh	AR0D	Palette Register D
0Eh	AR0E	Palette Register E
0Fh	AR0F	Palette Register F
10h	AR10	Mode Control
11h	AR11	Overscan Color
12h	AR12	Memory Plane Enable
13h	AR13	Horizontal Pixel Panning
14h	AR14	Color Select

2D CRT Controller Registers (3B4h / 3D4h / 3B5h / 3D5h)

Index	Sym	Register Name
00h	CR00	Horizontal Total
01h	CR01	Horizontal Display Enable End
02h	CR02	Horizontal Blanking Start
03h	CR03	Horizontal Blanking End
04h	CR04	Horizontal Sync Start
05h	CR05	Horizontal Sync End
06h	CR06	Vertical Total
07h	CR07	Overflow
08h	CR08	Preset Row Scan
09h	CR09	Maximum Scan Line
0Ah	CR0A	Text Cursor Start
0Bh	CR0B	Text Cursor End
0Ch	CR0C	Start Address High
0Dh	CR0D	Start Address Low
0Eh	CR0E	Text Cursor Location High
0Fh	CR0F	Text Cursor Location Low
10h	CR10	Vertical Sync Start
11h	CR11	Vertical Sync End
12h	CR12	Vertical Display Enable End
13h	CR13	Offset
14h	CR14	Underline Location
15h	CR15	Vertical Blanking Start
16h	CR16	Vertical Blanking End
17h	CR17	CRT Mode
18h	CR18	Line Compare

Index	Sym	Register Name
22h	CR22	Memory Read Latch Data

GUC

GUC is part of the System Interfaces Volume.

GuC Introduction

GuC is an embedded micro-controller in the graphics sub-system that is designed to perform graphics workload scheduling on the various graphics parallel engines. In this scheduling model, host software submits work through one of the 256 graphics doorbells and this invokes the micro-kernel running on the GuC core to perform the scheduling operation on the appropriate graphics engine.

Scheduling operations include determining which workload to run next, submitting a workload to a command streamer, pre-empting existing workloads running on an engine, monitoring progress and notifying host SW when work is done. To perform these actions, the GuC requires access to a wide range of assets within the graphics subsystem. The GuC has access to the entire graphics device MMIO register space to allow it to schedule work on any graphics engine.

The code that runs on the GuC is provided by the graphics driver (KMD) during the boot-up and graphics initialization phase. Code provided by the driver is copied from graphics memory and authenticated before execution.

From a functional perspective, the GuC sub-system has the following blocks:

- A Shim block that provides an interface between the micro-controller and rest of the graphics assets.
- An interrupt block that aggregates all the notifications coming from various graphics engines and communicates them to the GuC micro-controller for action. The interrupt block supports (programmable) prioritized delivery of events.
- A DMA engine to allow efficient copy of large blocks of data between memory and internal SRAM. During GuC initialization phase, this DMA engine is available to the host SW to load the GuC micro-kernel. Once the micro-kernel is successfully loaded into GuC, the access to the DMA engine is restricted to the code running on the GuC.
- It also has additional infrastructure to receive notification that are required for scheduling (semaphores from engines, page faults/faults-cleared from Memory interface, etc)
- A GuC power management unit that determines when all the GuC components are idle and supports the power management protocol with the Power Management unit.

Once code is loaded successfully, the primary method of communication with GuC is through the workload doorbells and a GuC/host interrupt mechanism. GuC automatically saves and restores its code image across RC6 power states, so no host intervention is required during these power transitions.

Terminology

Description	Software Use	Must Be Implemented As
Read/Write, R/W	This bit can be read or written.	
Reserved	Do not assume a value for these bits. Writes have no effect.	Writes are ignored. Reads return zero.
Reserved: must be zero, MBZ	Software must always write a zero to these bits. This allows new features to be added using these bits that will be disabled when using old software and as the default case.	Writes are ignored. Reads return zero. Maybe be connected as Read/Write in future projects.
Reserved: PBC, software must preserve contents	Software must write the original value back to this bit. This allows new features to be added using these bits.	Read only or test mode Read/Write.
Read Only	This bit is read only. The read value is determined by hardware. Writes to this bit have no effect.	According to each specific bit. The bit value is determined by hardware and not affected by register writes to the actual bit.
Read/Clear, Read/Write Clear	This bit can be read. Writes to it with a one cause the bit to clear.	Hardware events cause the bit to be set and the bit is cleared on a write operation where the corresponding bit has a one for a value.
Double Buffered	Write when desired. Read gives the unbuffered value (written value) unless specified otherwise. Written values will update to take effect after a certain point. Some have a specific arming sequence where a write to another register is required before the update can take place. This is used to ensure atomic updates of several registers.	Two stages of registers used. First stage is written into and used for readback (unless specified otherwise). First stage value is transferred into second stage at the update point. Second stage value is used to control hardware. Arm/disarm flag for specific arming sequences.

Arming Doorbells

As indicated in the Workload submission section, doorbell rings signal request for work to be submitted to hardware.

Doorbells need to be configured (armed) before they can be rung. Following sections describe the sequence.

Arming Memory Based Doorbells

Doorbells (As described in the GPU doorbells section in the "Memory View" chapter) are used to submit work to the hardware. Each doorbell monitors a memory address (cacheline) that detects a write to the cacheline and generates an interrupt to the MinIA core.

A distributed doorbells infrastructure is provided. Distributed doorbell units save doorbells in parallel so there are some performance benefits if SW distributes doorbells evenly across Doorbell Units. Hence the infrastructure allows SW to explicitly pick the IDI (ring interface) on which the doorbell needs to be installed.

The memory address programmed into the doorbell register needs to be Physical address. Since addresses are hashed on IDI (physical address based hashing), this address needs to have specific attributes to land on a specific IDI. To enable this Graphics KMD picks the doorbell Page Address, GuC Shim HW computes the cacheline address within the page.

When virtualization is enabled, the memory address provided by the Graphics Kernel Mode Driver needs to be translated from Guest Physical Address(GPA) to Host Physical Address(HPA) before getting programmed into the doorbell register. To conform to the security requirements of virtualization where only Virtual Machine Manager is aware of the HPA values, the GPA --> HPA translation is automatically done by GuC Shim HW as described below. The Guc Shim computes the cacheline address based on the HPA of the doorbell page address and this is returned to the KMD so that it has a fully formed Doorbell cacheline address.

Note that the hashing algorithm is different depending on how many LLC/CBO slices are present in the part. GT receives a fuse value at manufacturing that indicates which hashing rule should be used. That fuse value is not communicated directly to GUC hardware; instead, the Graphics Kernel Mode Driver is required to copy information about the hashing mode fuse from a register in the GT perimeter (SNOOP_FILTER_Q_STATUS - offset 0xB00[30]) to the GUC shim (SHIM_CTRL_GUC0_REG - offset 0xC068[10]). Driver must do this prior to arming any doorbells. Note that this value is retained across RC6 cycles but is lost on FLR, warm/cold reset, etc. Thus driver must program this value on every driver start/load as well as after any driver initiated FLR. Stated differently, any time driver loads GUC firmware, it should also perform this B00[30]-> C068[10] copy operation.

Doorbell Arming Sequence

- The memory page address to be monitored is picked by the Graphics Kernel Mode Driver and provided to the MinIA FW.
- Graphics KMD communicates the doorbell details to MinIA FW: Doorbell memory Page-address to be monitored, IDI#, Doorbell ID

- MinIA FW computes Doorbell MMIO
- MinIA FW programs the memory address to be monitored to the doorbell range: 0x1000 - 0x17FF range.
- GuC Shim HW intercepts the Doorbell MMIO targeted cycles to provide a GPA --> HPA translation and installs the translated address into the doorbell. Using the returned HPA, HW also computes the appropriate cacheline address that is returned to the FW for propagaton back up to the KMD. Detailed HW sequence is described below.
- MinIA FW monitors register 0xC090 and 0xC094 waiting for HW to signal completion of the doorbell install step and then checks status to determine if this was completed successfully.

GuC Shim (GUCSHIM) Register Functions

The GuC Shim provides the interface between GuC and the rest of GT. It is comprised of the various status registers that communicate the current state of the GuC, the infrastructure to setup the address space for GuC operation and interface with message channel.

Context:	The following table provides a view of the GuC address space

Address Top	Address Bottom	Space	Description
0xFFFF_FFFF	0xFFE0_0000	Graphics MMIO	2 MB off the top of the 4GB space
0xFFDF_FFFF	0xFEE0_1000	Hole	
0xFEE0_0FFF	0xFEE0_0000	LAPIC	4KB that houses the Local APIC registers. The accesses to this region are redirected to the LAPIC before they get to the shim decoder.
0xFEDF_FFFF	WOPCM_TOP	DRAM - Graphics Memory	Section should be decoded as: **Upper bound:** 4GB - 2MB(Gfx mmio) - 16MB **Lower bound:** 80KB Lower bound accounts for 16KB + 64KB
WOPCM_TOP-1	0x0006_8000	DRAM - WOPCM	Write Only Protected Content Memory (WO-PCM) This allows for code to straddle SRAM and memory (as described later)
0x0006_7FFF	0x0000_8000	SRAM space	384 KB SRAM This gets loaded with the GuC micro-kernel. The GuC may also use portion of the SRAM for its data, stack, and other required components.
0x0000_7FFF	0x0000_0000	Boot ROM	32 KB of BootROM for Initialization and authentication code that the GuC first jumps to is located here.

Context:	MinuteIA L1 Cacheability

By default, the Shim uses the following rules for caching in the MinuteIA L1 cache.

- 0xFEE0_0000 - 0xFFFF_FFFF: Un-Cached (Covers the 2MB Gfx address space + 16MB hole)
- 0x0000_0000 - 0xFEDF_FFFF: Cached (Everything below)
- By default, MinIA treats all WOPCM cycles as WB while Gfx-GTT cycles are WT. If C064[16] is set, all cycles (below xFFE0_0000) from MinIA are WB.

Graphics driver can create an Un-cached region window in the lower range described above, by programming the Non_Cacheable_Region_Base and Non_Cacheable_Region_Limit registers.

When using the GuC DMA engine to load the HuC uKernel, the status can be obtained by reading:

- GuC's BOOT_HASH_CHK register (0xc010 bit 8) to see if a HuC uKernel loading had been attempted, and
- HuC's HUC_STATUS2 register (0xd3b0 bit 7) to check whether or not the HuC uKernel was successfully loaded.

GUCSHIM Registers

Register
GUC_STATUS - Global MicroController Status
JMP_DEST - Jump Location
MIA_FORCE_FENCE - Minute IA Force Fence
MIA_INV_TLB - Minute IA Force TLB Invalidate
UOS_FULL_HASH_LO - uOS Full Hash Low
UOS_FULL_HASH_HI - uOS Full Hash High
SOFT_SCRATCH - Soft Scratch
UOS_RSA_SCRATCH - RSA for uOS/Soft Scratch

Guc DMA (GUCDMA)

The DMA engine allows the MinuteIA core to move data back and forth efficiently from the various memory segments listed below. Note that the DMA engine supports more than current required usage models. Memory segments supported:

- Global GTT mapped memory
- Per Process GTT mapped memory
- WOPCM
- SRAM

The MinIA is a 32 bit engine so it cannot generate an address greater than 4 GB. Thus any data that the MinIA core has to access must be located <4 GB in the graphics address space. The graphics address generated by the MinIA core goes through the regular graphics page table walk to derive the physical address that can be above 4 GB.

The DMA engine supports the full 48 bit addressing so it can be used by the MinIA core to get to the address regions above 4 GB. The MinIA core programs this DMA engine through registers that are mapped into the Gfx MMIO.

The DMA Registers

GUCDMA uses two 64-bit registers (4 DWord registers) to indicate the 48-bit Source and Destination addresses along with fetch type indication etc. Bit 1 of the DMA Control Register (described later) pins DMA Address Register 0 to Source addressing or Destination, and vice versa for DMA Address Register 1. By default, DMA Address Register 0 is assigned to Source addressing and DMA Address Register 1 to Destination addressing.

Register
DMA_ADDR_0_LOW - DMA Address Register 0 Low
DMA_ADDR_0_HIGH - DMA Address Register 0 High
DMA_ADDR_1_LOW - DMA Address Register 1 Low
DMA_ADDR_1_HIGH - DMA Address Register 1 High
DMA_COPY_SIZE - DMA Copy Size
DMA_CFG - DMA Configuration
DMA_CTRL - DMA Control

Programming Note	
Context:	The DMA Registers

Notes:

- The DMA engine can be deactivated by setting the Disable-GuC fuse. If this fuse is set, the DMA engine is rendered inoperable, so it cannot be used to load a GuC uOS or move any data. On a product that intends to use GuC, this fuse shall be zero.
- The lower 6 bits of *addressing* of both Source and Destination addresses must be same. There is no provision for barrel rotation across byte locations, during a DMA Transfer.
- The following restrictions shall be followed for placement of uOS and uApps:
  - uOS and uApps are always located on a 64-byte aligned address.
- uApps are not automatically loaded into SRAM by HW. uKernel must explicitly copy a uApp into SRAM from WOPCM when using it.
- Once the ukernel and the uApps have been loaded successfully into the WOPCM area, the HW shall not allow DMA operation to overwrite them.
- Before programming the DMA engine to access memory in the Per Process GTT address space, GuC SW must setup the PPGTT by programming registers: These registers are located in the GUC_PM unit (offsets: 0xC3B8 - 0xC3F0):
  - CTXT_INFO
  - PDP0, PDP1, PDP2, PDP3
  - PPGTT_ENABLE
- The GuC DMA engine also provides support for loading the HuC micro-kernel. To load a third party HuC ukernel, the third party GuC ukernel must be loaded first. An authenticated GuC ukernel can then be invoked to load a HuC ukernel. (If a third party HuC ukernel is loaded first, there is no way to clear the ME_DATA registers - thus locking out the ability to load a third party GuC).
- GuC DMA HW checks for the following illegal cases and rejects the DMA invocation (DMA will not happen):

Illegal Case
GuC WOPCM Base & GuC WOPCM Size is not programmed (for copy to/from WOPCM) 0xC050 and 0xC340
DMA copy into GuC WOPCM that does not fit into the GuC WOPCM
DMA copy into SRAM that falls off the SRAM edge (except for uKernel copy)
DMA size is set to 0

GuC Interrupt (GUCINT) Register Functions

This section discusses the register functions for GuC Interrupt. Registers in this section are:

Functionality
Interrupt Group Registers
Doorbell Group Registers
Engine Interrupt Regiters
GuC Timer Registers
GuC DMA Interrupt Registers
GuC Host Registers

Interrupt Group Registers

GUCINT uses two registers to map all incoming interrupts to the 256-bit interrupt vector sent to the LAPIC. Internal to the LAPIC, the 256-bit interrupt vector is organized as 16 groups of 16 vectors. GUCINT maps each group of interrupts to a single group in the vector sent to LAPIC. Note that the higher the group number, the higher the priority of the group of interrupts to which it is assigned.

Interrupt Group Registers
INTR_GROUP_1 - Interrupt Group 1
INTR_GROUP_0 - Interrupt Group 0

The following interrupt groups are supported:

Interrupt Group
Doorbells (counts as 8 groups)
Semaphores
Engines
IOMMU
DMA/TIMER/FLR
HOST

Each group of interrupts needs 4 bits to map to the correct group in the 256-bit vector.

Interrupt Group 0 Register maps the 8 groups of Doorbells. **Interrupt Group 1 Register** maps the remaining 5 groups listed above.

Programming Note	
Context:	Interrupt Group Registers
• All groups must be unique. Only Doorbell groups may share a value with another doorbell group. • The group value cannot be 4'h1. That is illegal. This restriction applies to both GROUP registers. • Group values cannot dynamically change from one nonzero value to a different nonzero value.	

Doorbell Group Registers

There is a Doorbell Control Register in GUCint, which has a single doorbell_rung bit for each one of the eight Doorbell Registers located in GTI (1900 - 191C). GTI sets this GuC Register bit when the corresponding Doorbell register GTI houses, has that GTI register value going from all 0s to having any one bit set (any one of the 32 doorbells in that doorbell group gets rung).

GTI could set multiple first_doorbell_rung bits in a single message to the GuC based Doorbell Control Register (corresponding to several doorbells rung in different doorbell group registers in GTI). Once a doorbell_rung bit is set for a group in the Doorbell Control Register, it is not updated until GUCINT reads the corresponding GTI doorbell register, at which time the corresponding rung_bit is reset.

The doorbell control register also holds 1 bit (send to Mini-Core) that routes interrupts to host or Mini-Core. By default, interrupts are sent to host. This bit must be set by software (running on Mini-Core or host) to route interrupts to Mini-Core.

GUC_DB_ISR_7 - GuC Doorbell Group 7 Interrupt Status

GUC_DB_ISR_6 - GuC Doorbell Group 6 Interrupt Status

GUC_DB_ISR_5 - GuC Doorbell Group 5 Interrupt Status

GUC_DB_ISR_4 - GuC Doorbell Group 4 Interrupt Status

GUC_DB_ISR_3 - GuC Doorbell Group 3 Interrupt Status

GUC_DB_ISR_2 - GuC Doorbell Group 2 Interrupt Status

GUC_DB_ISR_1 - GuC Doorbell Group 1 Interrupt Status

GUC_DB_ISR_0 - GuC Doorbell Group 0 Interrupt Status

DOORBELL_CTRL - Doorbell Control

Engine Interrupt Registers

GUC gets Engine Event interrupts from various engines (Render, Copy, Compute, Video Decode, Video Enhancment ..).

GUCINT also gets Engine Event interrupts from OA.

The engines support a variety of interrupts that may not be interesting to GuC from a scheduling point of view. GUCINT provides an infrastructure to redirect engine interrupts to the host driver without invoking

the Mini-Core firmware. This infrastructure allows software to specify on a per engine and per interrupt granularity the interrupts that must be delivered to Mini-Core or simply forwarded to the host (bypassing the MiniCore).

Command Streamer Status Information

During execution, the Command Streamer Status is sent to the GuC.

Programming Note	
Context:	Context Status Buffer Initialization
GuC CSB FIFO's are implemented on device reset domain, its possible following GFX Reset (All engines and GuC are reset) there are unprocessed entries present in the engine CSB FIFO's. GuC FW as part of the GuC initialization flow must ensure the engine CSB FIFO's are drained and empty before scheduling contexts to the engines.	

CSB Read Port

The following RO registers are for use by GuC FW or host. SW must read twice to obtain a single CSB entry: the first read returns bits[31:0]; the second read returns [63:32].

CS CSB

BCS CSB

VCS CSB

VECS CSB

CSB FIFO Status Registers

The following RO registers hold the status of each Command Streamer's CSB FIFO:

CS CSB Fifo Status Register

BCS CSB Fifo Status Register

VCS CSB Fifo Status Register

VECS CSB Fifo Status Register

Guc DMA Interrupt Registers

GUC_DMA_IIR - GuC DMA Interrupt Input

The DMA generates this message interrupt at the completion of a programmed DMA transfer.

Guc Host Registers

GuC and Host(IA) communicate with each other through interrupts.

- A Host-to-GUC interrupt is generated by Host SW writing to 0xC4C8. The written data will get stored in 0xC590 and an interrupt will be generated to GuC.

- A GUC-to-Host interrupt is generated by GuC FW writing to 0xC4B8 - this generates a 16bit vector to the host (this 16b vector is shared by GuC HW and GuC FW. FW write can only set FW owned bits)

GUC_HOST_INTR_IIR - GuC Host Interrupt Interrupt Input

Observability

Observability Overview

As GFX-enabled systems and usage models have grown in complexity over time, a number of hardware features have been added to provide more insight into hardware behavior while running a commercially available operating system. This chapter documents these features with pointers to relevant sections in other chapters. Supported observability features include:

Feature
Performance counters
Internal node tracing

Note: This chapter describes the registers and instructions used to monitor GPU performance. Please review other volumes in this specification to understand the terms, functionality and details for specific Intel graphics devices.

DFD Configuration Restore

Since DFD logic does not usually add value to end user usage models and its configuration space is large (which would add latency to power management restore flows), it is typically not enabled during normal operation for optimal power & performance. Hence, additional steps are required when DFD functionality is needed in combination with system configurations where GT logic loses power/is reset. The basic strategy per scenario is detailed below.

GT Power-up/RC6 Exit

Strategy
Replicate failure without power management
Configure the DFD restore feature

Render Engine Power-up

Configure the RCS RC6 W/A batch buffer to restore render engine DFD configuration ONLY.

Media Engine Power-up

Configure the applicable media command stream W/A batch buffer to restore media engine DFD configuration ONLY.

Resume From Partial GT Power Down

For cases where SW is aware of power well state, re-apply DFD configuration.

For cases where SW is not aware of power well state, configure the per-context W/A batch buffer to apply the DFD configuration on every context load.

Trace

This section contains the following contents:

Feature
• Performance Visibility

Performance Visibility

Motivation For Hardware-Assisted Performance Visibility

As the focus on GFX performance and programmability has increased over time, the need for hardware (HW) support to rapidly identify bottlenecks in HW and efficiently tune the work sent to same has become correspondingly important. This part of the BSpec describes the HW support for Performance Visibility.

Performance Event Counting

An earlier generation introduced dedicated GFX performance counters to address key issues associated with existing chipset CHAPs counters (lack of synchronization with GFX rendering work and low sampling frequency achievable when sampling via CPU MMIO read). Furthermore, reliance on SoC assets created a cross-IP dependency that was difficult to manage well. Hence, the approach since that earlier generation has been to use dedicated counters managed by the graphics device driver for graphics performance measurement. The dedicated counter values are written to memory whenever an MI_REPORT_PERF_COUNT command is placed in the ring buffer.

While this approach eliminated much of the error associated with the previous approaches, it is still limited to sampling the counters only at the boundaries between ring commands. This inherently limited the ability of performance analysis tools to drill down into a primitive, which can contain thousands of triangles and require several hundreds of milliseconds to render. It is further worth noting that precise sampling via MI_REPORT_PERF command requires flushing the GFX pipeline before and after the work of interest. The overhead of flushing the GFX pipeline can become large if the work of interest is small, hence reducing the accuracy of the performance counter measurement. In such situations, the flush can be removed or internally triggered reporting can be used with some resulting loss of precision in which draws/dispatches are being profiled.

Additionally, Intel design and architecture teams found that the existing silicon-based performance analysis tools provided only a general idea of where a problem may exist but were not able to pin point a problem. This was generally because the counter values are integrated across a very large time period, washing out the dynamic behavior of the workload.

All OA config registers are tied to GT global reset and hence are not affected by per-engine resets (e.g. render only reset).

OA Programming Guidelines

SW utilizing OA HW is expected to monitor the overflow/lost report status for the OABUFFER and respond as appropriate for the active usage model.

In order for OA counters to increment the 'Counter Stop-Resume Mechanism' bit of the OACTXCONTROL register must be set. This requires a RCS context with this bit set be loaded, and either RCS force wake be enabled or the RCS context be left active for the duration of the window this counter is needed for.

In general, OA is effectively unable to count between the power context save that happens prior to GFX entering RC6 and the power context restore that occurs on the next RC6 exit. This limitation results from the fact that the counters themselves are power context save/restored and hence the counts that (may) have accumulated in this time window are overwritten by the saved values that are read back from the power context save area. An example of the kind of information that can be missed is the GTI traffic resulting from the power context save of OA itself. The size of this performance counting blind spot is microarchitecturally minimized as much as reasonably possible but still varies from device to device.

Legacy OACS functionality is now logically split into two functions called OAG (OA Global) and OAR (OA Render). Summary of the blocks is as follows:

OAG:

- Handles OA buffer and timer/internally-triggered sampling.
- Is unaffected by engine reset / power well status.
- Is inaccessible by non-privileged batch buffers but accessible by all command streamers / GuC / CPU.
- Implements free-running utilization counters.
- Is GT power context save/restored.
- Is only allowed to access global GTT memory.

OAR:

- Is expected to behave as a part of the render engine from a clocking/power well/reset perspective.
- Implements MI_REPORT_PERF command.
- Is render context save/restored, making all values reported by MI_REPORT_PERF per-context.
- Must be initialized to power-on default values as part of RCS golden context creation (please refer to RCS section describing golden context creation for full details) or implementation-specific undesirable behavior may occur.
- Doesn't support timer/internally-triggered sampling.
- Can be enabled/disabled independent from OAG.

Is only intended to be accessed by RCS, access from other command streamers / CPU may have implementation-specific negative side-effects.

HW Support

This section contains various reporting counters and registers for hardware support for Performance Visibility.

Performance Counter Report Formats

Counters layout for various values of select from the register:

Counters layout for various values of the "Counter Select" from the register:

Counter Select = 000

A-Cntr 10 (low dword)	A-Cntr 9 (low dword)	A-Cntr 8 (low dword)	A-Cntr 7 (low dword)	GPU_TICKS	CTX ID	TIME_STAMP	RPT_ID
A-Cntr 18 (low dword)	A-Cntr 17 (low dword)	A-Cntr 16 (low dword)	A-Cntr 15 (low dword)	A-Cntr 14 (low dword)	A-Cntr 13 (low dword)	A-Cntr 12 (low dword)	A-Cntr 11 (low dword)

Counter Select = 010

A-Cntr 10 (low dword)	A-Cntr 9 (low dword)	A-Cntr 8 (low dword)	A-Cntr 7 (low dword)	GPU_TICKS	CTX ID	TIME_STAMP	RPT_ID
A-Cntr 18 (low dword)	A-Cntr 17 (low dword)	A-Cntr 16 (low dword)	A-Cntr 15 (low dword)	A-Cntr 14 (low dword)	A-Cntr 13 (low dword)	A-Cntr 12 (low dword)	A-Cntr 11 (low dword)
B-Cntr 7	B-Cntr 6	B-Cntr 5	B-Cntr 4	B-Cntr 3	B-Cntr 2	B-Cntr 1	B-cntr 0
C-Cntr 7	C-Cntr 6	C-Cntr 5	C-Cntr 4	C-Cntr 3	C-Cntr 2	C-Cntr 1	C-Cntr 0

Counter Select = 111

C-Cntr 3	C-Cntr 2	C-Cntr 1	C-Cntr 0	GPU_TICKS	CTX ID	TIME_STAMP	RPT_ID
B-Cntr 7	B-Cntr 6	B-Cntr 5	B-Cntr 4	B-Cntr 3	B-Cntr 2	B-Cntr 1	B-Cntr 0

OAR Report Format (Counter Select = 0b101):

A-Cntr 3 (low dword)	A-Cntr 2 (low dword)	A-Cntr 1 (low dword)	A-Cntr 0 (low dword)	GPU_TICKS	CTX ID	TIME_STAMP	RPT_ID
A-Cntr 11 (low dword)	A-Cntr 10 (low dword)	A-Cntr 9 (low dword)	A-Cntr 8 (low dword)	A-Cntr 7 (low dword)	A-Cntr 6 (low dword)	A-Cntr 5 (low dword)	A-Cntr 4 (low dword)
A-Cntr 19 (low dword)	A-Cntr 18 (low dword)	A-Cntr 17 (low dword)	A-Cntr 16 (low dword)	A-Cntr 15 (low dword)	A-Cntr 14 (low dword)	A-Cntr 13 (low dword)	A-Cntr 12 (low dword)
A-Cntr 27 (low dword)	A-Cntr 26 (low dword)	A-Cntr 25 (low dword)	A-Cntr 24 (low dword)	A-Cntr 23 (low dword)	A-Cntr 22 (low dword)	A-Cntr 21 (low dword)	A-Cntr 20 (low dword)
A-Cntr 35 (low dword)	A-Cntr 34 (low dword)	A-Cntr 33 (low dword)	A-Cntr 32 (low dword)	A-Cntr 31 (low dword)	A-Cntr 30 (low dword)	A-Cntr 29 (low dword)	A-Cntr 28 (low dword)
High bytes of A31-A28	High bytes of A27-A24	High bytes of A23-A20	High bytes of A19-A16	High bytes of A15-A12	High bytes of A11-A8	High bytes of A7-A4	High bytes of A3-A0
B-Cntr 7	B-Cntr 6	B-Cntr 5	B-Cntr 4	B-Cntr 3	B-Cntr 2	B-Cntr 1	B-Cntr 0

C-Cntr 7	C-Cntr 6	C-Cntr 5	C-Cntr 4	C-Cntr 3	C-Cntr 2	C-Cntr 1	C-Cntr 0

OAG Report Format (Counter Select = 0b101)

A-Cntr 3 (low dword)	A-Cntr 2 (low dword)	A-Cntr 1 (low dword)	A-Cntr 0 (low dword)	GPU_TICKS	CTX ID	TIME_STAMP	RPT_ID
A-Cntr 11 (low dword)	A-Cntr 10 (low dword)	A-Cntr 9 (low dword)	A-Cntr 8 (low dword)	A-Cntr 7 (low dword)	A-Cntr 6 (low dword)	A-Cntr 5 (low dword)	A-Cntr 4 (low dword)
A-Cntr 19 (low dword)	A-Cntr 18 (low dword)	A-Cntr 17 (low dword)	A-Cntr 16 (low dword)	A-Cntr 15 (low dword)	A-Cntr 14 (low dword)	A-Cntr 13 (low dword)	A-Cntr 12 (low dword)
A-Cntr 27 (low dword)	A-Cntr 26 (low dword)	A-Cntr 25 (low dword)	A-Cntr 24 (low dword)	A-Cntr 23 (low dword)	A-Cntr 22 (low dword)	A-Cntr 21 (low dword)	A-Cntr 20 (low dword)
A-Cntr 35 (low dword)	A-Cntr 34 (low dword)	A-Cntr 33 (low dword)	A-Cntr 32 (low dword)	A-Cntr 31 (low dword)	A-Cntr 30 (low dword)	A-Cntr 29 (low dword)	A-Cntr 28 (low dword)
High bytes of A31-A28	High bytes of A27-A24	High bytes of A23-A20	High bytes of A19-A16	High bytes of A15-A12	High bytes of A11-A8	High bytes of A7-A4	High bytes of A3-A0
B-Cntr 7	B-Cntr 6	B-Cntr 5	B-Cntr 4	B-Cntr 3	B-Cntr 2	B-Cntr 1	B-Cntr 0
C-Cntr 7	C-Cntr 6	C-Cntr 5	C-Cntr 4	C-Cntr 3	C-Cntr 2	C-Cntr 1	C-Cntr 0

Counter Select = 111

C-Cntr 3	C-Cntr 2	C-Cntr 1	C-Cntr 0	GPU_TICKS	CTX ID	TIME_STAMP	RPT_ID
B-Cntr 7	B-Cntr 6	B-Cntr 5	B-Cntr 4	B-Cntr 3	B-Cntr 2	B-Cntr 1	B-Cntr 0

Description of RPT_ID and other important fields of the layout:

Field	Description
GPU TICKS[31:0]	GPU_TICKS is simply a free-running count of render clocks elapsed used for normalizing other counters (e.g. EU active time), it is expected that the rate that this value advances will vary with frequency and freeze (but not lose its value) when all GT clocks are gated, GT is in RC6, and so on.
Context ID[31:0]	This field carries the Context ID of the active context in render engine. [31:0]: Context ID in Execlist mode of scheduling.
TIME_STAMP[31:0]	This field provides an elapsed real-time value that can be used as a timestamp for GPU events over short periods of time. This field has the same format at TIMESTAMP register defined in Vol1C.4 Render Command Streamer BSpec.
RPT_ID[46:38]	Reserved (for future use)
RPT_ID[35:34]	Reserved (for future Tile IDs)
RPT_ID[31:0]	This field has several sub fields as defined below: 31:26 **SourceID[5:0]**

Field	Description	
		Encoded value to identify various sources like any CS or Shader unit from which the Report was requested. **Programming note:**
	25:19	**Report Reason[6:0]** Report_reason[0]: When set indicates current report is due to "Timer Triggered". Report_reason[1]: When set indicates current report is due to "Internal report trigger 1". Report_reason[2]: When set indicates current report is due to "Internal report trigger 2". Report_reason[3]: When set indicates current report is due to "Context switch". Report_reason[4]: When set indicates current report is due to "GO transition from '1' to '0' ". Report_reason[5]: When set indicates current report is due to a change in unslice/slice ratio Report_reason[6]: When set indicates current report is due to a MMIO Trigger **Programming note:**
	18	**Start Trigger Event:**This bit is multiplexed from "Start Trigger Event-1" or "Start Trigger Event-2" based on the "Internal Report Trigger-1" or "Internal Report Trigger-2" asserted in the Report Reason respectively. "Internal Report Trigger-1" is given priority over "Internal Report Trigger-2". By default Start Trigger Event-1 is outputted.
	17	**Threshold Enable:** This bit is multiplexed from "Report Trigger Threshold Enable-1" or "Report Trigger Threshold Enable-2" based on the "Internal Report Trigger-1" or "Internal Report Trigger-2" asserted in the Report Reason respectively. "Internal Report Trigger-1" is given priority over "Internal Report Trigger-2". By default "Report Trigger Threshold Enable-1" is outputted.
	16	**Timer Enabled**
	15:0	Reserved

Performance Counting Register Interface

Global Registers
OACTXID - Observation Architecture Control Context ID
OA_IMR - OA Interrupt Mask Register
OASTATUS - Observation Architecture Status Register
OAHEADPTR - Observation Architecture Head Pointer
OATAILPTR - Observation Architecture Tail Pointer
OABUFFER - Observation Architecture Buffer
OASTARTTRIG_COUNTER - Observation Architecture Start Trigger Counter
OARPTTRIG_COUNTER - Observation Architecture Report Trigger Counter
OAREPORTTRIG2 - Observation Architecture Report Trigger 2
OAREPORTTRIG6 - Observation Architecture Report Trigger 6
CEC0-0 - Customizable Event Creation 0-0
CEC1-0 - Customizable Event Creation 1-0
CEC1-1 - Customizable Event Creation 1-1
CEC2-0 - Customizable Event Creation 2-0
CEC2-1 - Customizable Event Creation 2-1
CEC3-0 - Customizable Event Creation 3-0
CEC3-1 - Customizable Event Creation 3-1
CEC4-0 - Customizable Event Creation 4-0
CEC5-0 - Customizable Event Creation 5-0
CEC5-1 - Customizable Event Creation 5-1
CEC6-0 - Customizable Event Creation 6-0
CEC6-1 - Customizable Event Creation 6-1
CEC7-0 - Customizable Event Creation 7-0
CEC7-1 - Customizable Event Creation 7-1
EU_PERF_CNT_CTL0 - Flexible EU Event Control 0
EU_PERF_CNT_CTL1 - Flexible EU Event Control 1
EU_PERF_CNT_CTL2 - Flexible EU Event Control 2
EU_PERF_CNT_CTL3 - Flexible EU Event Control 3
EU_PERF_CNT_CTL4 - Flexible EU Event Control 4
EU_PERF_CNT_CTL5 - Flexible EU Event Control 5
EU_PERF_CNT_CTL6 - Flexible EU Event Control 6

Symmetrical Registers	OAG Offset	OAR Offset
OAPERF_A0 - Aggregate Perf Counter A0		
OAPERF_A0_UPPER - Aggregate Perf Counter A0 Upper DWord		
OAPERF_A1 - Aggregate Perf Counter A1		
OAPERF_A1_UPPER - Aggregate Perf Counter A1 Upper DWord		
OAPERF_A2 - Aggregate Perf Counter 2		
OAPERF_A2_UPPER - Aggregate Perf Counter A2 Upper DWord		
OAPERF_A3 - Aggregate Perf Counter A3		
OAPERF_A3_UPPER - Aggregate Perf Counter A3 Upper DWord		
OAPERF_A4 - Aggregate Perf Counter A4		
OAPERF_A4_UPPER - Aggregate Perf Counter A4 Upper DWord		
OAPERF_A4_LOWER_FREE - Aggregate Perf Counter A4 Lower DWord Free		
OAPERF_A4_UPPER_FREE - Aggregate Perf Counter A4 Upper DWord Free		
OAPERF_A5 - Aggregate Perf Counter A5		
OAPERF_A5_UPPER - Aggregate Perf Counter A5 Upper DWord		
OAPERF_A6 - Aggregate Perf Counter A6		
OAPERF_A6_UPPER - Aggregate Perf Counter A6 Upper DWord		
OAPERF_A6_LOWER_FREE - Aggregate Perf Counter A6 Lower DWord Free		
OAPERF_A6_UPPER_FREE - Aggregate Perf Counter A6 Upper DWord Free		
OAPERF_A7 - Aggregate Perf Counter A7		
OAPERF_A7_- Upper Aggregate Perf Counter A7 Upper DWord		
OAPERF_A8 - Aggregate Perf Counter A8		
OAPERF_A8_UPPER - Aggregate Perf Counter A8 Upper DWord		
OAPERF_A9 - Aggregate Perf Counter A9		
OAPERF_A9_UPPER - Aggregate Perf Counter A9 Upper DWord		
OAPERF_A10 - Aggregate Perf Counter A10		
OAPERF_A10_UPPER - Aggregate Perf Counter A10 Upper DWord		
OAPERF_A11 - Aggregate Perf Counter A11		
OAPERF_A11_UPPER - Aggregate Perf Counter A11 Upper DWord		
OAPERF_A12 - Aggregate Perf Counter A12		
OAPERF_A12_UPPER - Aggregate Perf Counter A12 Upper DWord		
OAPERF_A13 - Aggregate Perf Counter A13		
OAPERF_A13_UPPER - Aggregate Perf Counter A13 Upper DWord		
OAPERF_A14 - Aggregate Perf Counter A14		
OAPERF_A14_UPPER - Aggregate Perf Counter A14 Upper DWord		
OAPERF_A15 - Aggregate Perf Counter A15		
OAPERF_A15_UPPER - Aggregate Perf Counter A15 Upper DWord		

Symmetrical Registers	OAG Offset	OAR Offset
OAPERF_A16 - Aggregate Perf Counter A16		
OAPERF_A16_UPPER - Aggregate Perf Counter A16 Upper DWord		
OAPERF_A17 - Aggregate Perf Counter A17		
OAPERF_A17_UPPER - Aggregate Perf Counter A17 Upper DWord		
OAPERF_A18 - Aggregate Perf Counter A18		
OAPERF_A18_UPPER - Aggregate Perf Counter A18 Upper DWord		
OAPERF_A19 - Aggregate Perf Counter A19		
OAPERF_A19_UPPER - Aggregate Perf Counter A19 Upper DWord		
OAPERF_A19_LOWER_FREE - Aggregate Perf Counter A19 Lower DWord Free		
OAPERF_A19_UPPER_FREE - Aggregate Perf Counter A19 Upper DWord Free		
OAPERF_A20 - Aggregate Perf Counter A20		
OAPERF_A20_UPPER - Aggregate Perf Counter A20 Upper DWord		
OAPERF_A20_UPPER_FREE - Aggregate Perf Counter A20 Upper DWord Free		
OAPERF_A20_LOWER_FREE - Aggregate Perf Counter A20 Lower DWord Free		
OAPERF_A21 - Aggregate Perf Counter A21		
OAPERF_A21_UPPER - Aggregate Perf Counter A21 Upper DWord		
OAPERF_A22 - Aggregate Perf Counter A22		
OAPERF_A22_UPPER - Aggregate Perf Counter A22 Upper DWord		
OAPERF_A23 - Aggregate Perf Counter A23		
OAPERF_A23_UPPER - Aggregate Perf Counter A23 Upper DWord		
OAPERF_A24 - Aggregate Perf Counter A24		
OAPERF_A24_UPPER - Aggregate Perf Counter A24 Upper DWord		
OAPERF_A25 - Aggregate Perf Counter A25		
OAPERF_A25_UPPER - Aggregate Perf Counter A25 Upper DWord		
OAPERF_A26 - Aggregate Perf Counter A26		
OAPERF_A26_UPPER - Aggregate Perf Counter A26 Upper DWord		
OAPERF_A27 - Aggregate Perf Counter A27		
OAPERF_A27_UPPER - Aggregate Perf Counter A27 Upper DWord		
OAPERF_A28 - Aggregate Perf Counter A28		
OAPERF_A28_UPPER - Aggregate Perf Counter A28 Upper DWord		
OAPERF_A29 - Aggregate Perf Counter A29		
OAPERF_A29_UPPER - Aggregate Perf Counter A29 Upper DWord		
OAPERF_A30 - Aggregate Perf Counter A30		
OAPERF_A30_UPPER - Aggregate Perf Counter A30 Upper DWord		
OAPERF_A31 - Aggregate_Perf_Counter_A31		
OAPERF_A31_UPPER - Aggregate Perf Counter A31 Upper DWord		
OAPERF_A32 - Aggregate_Perf_Counter_A32		

Symmetrical Registers	OAG Offset	OAR Offset
OAPERF_A33 - Aggregate_Perf_Counter_A33		
OAPERF_A34 - Aggregate_Perf_Counter_A34		
OAPERF_A35 - Aggregate_Perf_Counter_A35		
GPU_TICKS - GPU_Ticks_Counter		

OA Interrupt Control Registers

The Interrupt Control Registers listed below all share the same bit definition. The bit definition is as follows:

Bit	Description
31:29	**Reserved. MBZ:** These bits may be assigned to interrupts on future products/steppings.
28	**Performance Monitoring Buffer Half-Full Interrupt:** For internal trigger (timer based) reporting, if the report buffer crosses the half full limit, this interrupt is generated.
27:0	**Reserved: MBZ (These bits must be never set by OA, these bit could be allocated to some other unit)**

- **WDBoxOAInterrupt Vector**
- IMR
- Bit Definition for Interrupt Control Registers

Performance Counter Reporting

When either the MI_REPORT_PERF_COUNT command is received or the internal report trigger logic fires, a snapshot of the performance counter values is written to memory. The format used by HW for such reports is selected using the Counter Select field within the register. The organization and number of report formats vary per project and are detailed in Here.

Details of Start Trigger Behavior

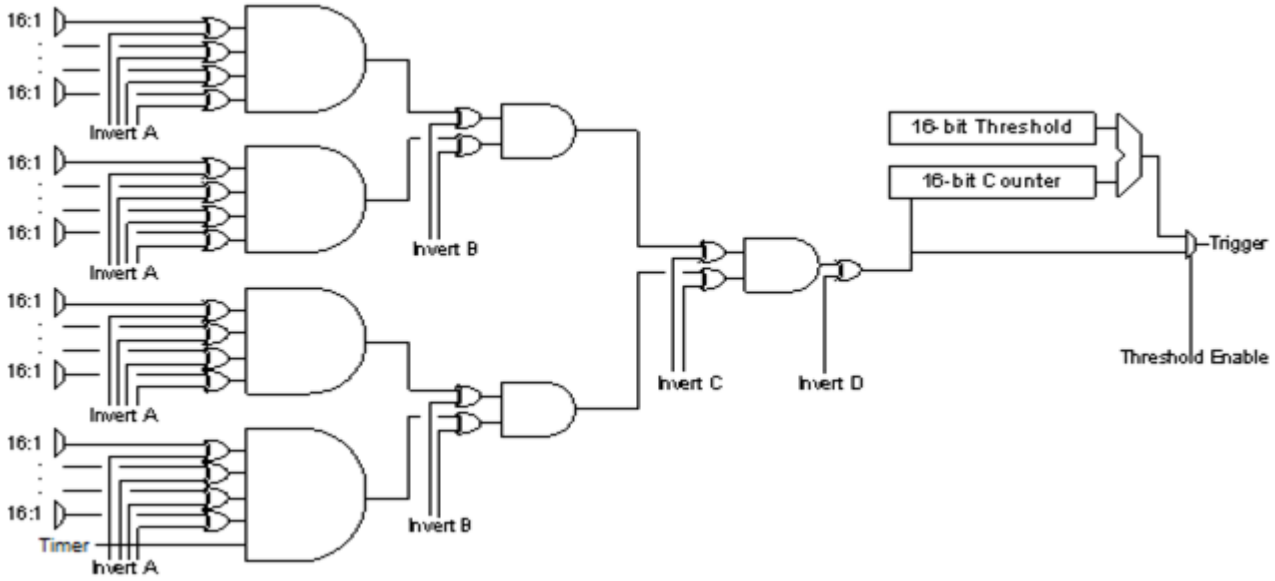
- All counters not explicitly defined as free-running will advance after the start trigger conditions are met.
- Counting will continue after the start trigger has fired until OA is disabled or device is reset.
- Multiple start triggering blocks (where implemented) are OR'd together in order to allow specification of multiple trigger conditions.
- Bit 18 in the report format reflects whether the start trigger has fired or not.

While architectural intent was that Start Trigger logic would control all qualified counter types (A/B/C), there is a long-standing implementation bug whereby start trigger logic only affects B/C counters.

Configuration of Trigger Logic

OA contains logic to control when performance counter values are reported to memory. This functionality is controlled using the OA report trigger and OA start trigger registers. More detailed

register descriptions are included in the Hardware Programming interface. The block diagram below illustrates the logic these registers control.



Note that counters which are 40 bits wide are split in the report format into low DWORD and high byte chunks for simplicity of HW implementation as well as SW-friendly alignment of report data. The performance counter read logically done before writing out report data for these 40-bit counters is guaranteed to be an atomic operation, the counter data is simply swizzled as it is being packed into the report.

Context Switch Triggered Reports

A context load/switch on RCS will cause a performance counter snapshot to be written to memory at the next location in the OA circular report buffer using the perf counter format selected in OACONTROL (). This functionality can be leveraged when preemption is enabled to re-construct the contribution of a specific context to a performance counter delta, requires SW to consider both the delta reported by MI_REPORT_PERF and the reports that may have been issued to OABUFFER by intervening contexts.

A context load/switch on RCS will cause a performance counter snapshot to be written to memory at the next location in the OA circular report buffer using the perf counter format selected in **OARCONTROL**. This functionality can be leveraged when preemption is enabled to re-construct the contribution of a specific context to a performance counter delta, requires SW to consider both the delta reported by MI_REPORT_PERF and the reports that may have been issued to OABUFFER by intervening contexts.

Frequency Change Triggered Reports

A GFX frequency change will cause a performance counter snapshot to be written to memory at the next location in the OA circular report buffer using the perf counter format selected in OACONTROL (). Please note that a change back to the same frequency can occur and that such changes will still cause a performance counter report to occur.

A GFX frequency change will cause a performance counter snapshot to be written to memory at the next location in the OA circular report buffer using the perf counter format selected in **OARCONTROL**. Please

note that a change back to the same frequency can occur and that such changes will still cause a performance counter report to occur.

Aggregating Counters

The table below described the desired high-level functionality from each of the aggregating counters.

Note that there is no counter of 2x2s sent to pixel shader, this is based on the assumption that the pixel shader invocation pipeline statistics counter increments for partially lit 2x2s as well and hence does not require a duplicate performance counter.

Please also note that some of the information provided by A-counters is useful for GFX/system load-balancing and is hence made available via free-running counters which do not require initial setup and count irrespective of OA enable/disable or freeze.

Counter #	Event	Description
A0	GPU Busy	GPU is not idle (includes all GPU engines).
A1	# of Vertex Shader Threads Dispatched	Count of VS fused threads dispatched to EUs
A2	# of Hull Shader Threads Dispatched	Count of HS fused threads dispatched to EUs
A3	# of Domain Shader Threads Dispatched	Count of DS fused threads dispatched to EUs
A4	# of GPGPU Threads Dispatched	Count of GPGPU fused threads dispatched to EUs. Available on both qualified and free-running counters.
A5	# of Geometry Shader Threads Dispatched	Count of GS fused threads dispatched to EUs
A6	# of Pixel Shader Threads Dispatched	Count of PS fused threads dispatched to EUs. Available on both qualified and free-running counters.
A7	Aggregating EU counter 0	User-defined (details in Flexible EU Event Counters section)
A8	Aggregating EU counter 1	User-defined (details in Flexible EU Event Counters section)
A9	Aggregating EU counter 2	User-defined (details in Flexible EU Event Counters section)
A10	Aggregating EU counter 3	User-defined (details in Flexible EU Event Counters section)
A11	Aggregating EU counter 4	User-defined (details in Flexible EU Event Counters section)
A12	Aggregating EU counter 5	User-defined (details in Flexible EU Event Counters section)
A13	Aggregating EU counter 6	User-defined (details in Flexible EU Event Counters section)
A14	Aggregating EU counter 7	User-defined (details in Flexible EU Event Counters section)
A15	Aggregating EU counter 8	User-defined (details in Flexible EU Event Counters section
A16	Aggregating EU counter 9	User-defined (details in Flexible EU Event Counters section)
A17	Aggregating EU counter 10	User-defined (details in Flexible EU Event Counters section)
A18	Aggregating EU counter 11	User-defined (details in Flexible EU Event Counters section)
A19	Aggregating EU counter 12	Available on both qualified and free-running counters

Counter #	Event	Description
		User-defined (details in Flexible EU Event Counters section)
A20	Aggregating EU counter 13	Available on both qualified and free-running counters User-defined (details in Flexible EU Event Counters section)
A21	2x2s Rasterized	Count of the number of samples of 2x2 pixel blocks generated from the input geometry before any pixel-level tests have been applied. (Please note that 2x2s may be in terms of pixels or in terms of samples depending on project but are consistent between A21-A27.)
A22	2x2s Failing Fast pre-PS Tests	Count of the number of samples failing fast "early" (i.e. before pixel shader execution) tests (counted at 2x2 granularity). (Please note that 2x2s may be in terms of pixels or in terms of samples depending on project but are consistent between A21-A27.)
A23	2x2s Failing Slow pre-PS Tests	Count of the number of samples of failing slow "early" (i.e. before pixel shader execution) tests (counted at 2x2 granularity). (Please note that 2x2s may be in terms of pixels or in terms of samples depending on project but are consistent between A21-A27.)
A24	2x2s Killed in PS	Number of samples entirely killed in the pixel shader as a result of explicit instructions in the kernel (counted in 2x2 granularity). (Please note that 2x2s may be in terms of pixels or in terms of samples depending on project but are consistent between A21-A27.)
A25	2x2s Failing post-PS Tests "POSTPS_DEPTH_STENCIL_ALPHA_FAIL"	Number of samples that entirely fail "late" tests (i.e. tests that can only be performed after pixel shader execution). Counted at 2x2 granularity. (Please note that 2x2s may be in terms of pixels or in terms of samples depending on project but are consistent between A21-A27.)
A26	2x2s Written To Render Target	Number of samples that are written to render target.(counted at 2x2 granularity). MRT case will report multiple writes per 2x2 processed by the pixel shader. (Please note that 2x2s may be in terms of pixels or in terms of samples depending on project but are consistent between A21-A27.) Please note that this counter will not advance if a render target update does not occur and that pixel masking operations performed by the fixed function HW or shader may not be reflected in counters A22-A25 which only track their specific defined operations. This can lead to an apparent discrepancy between A21 vs. A22-A25 vs. A26/A27.
A27	Blended 2x2s Written to Render Target	Number of samples of blendable that are written to render target.(counted at 2x2 granularity). MRT case will report multiple writes per 2x2 processed by the pixel shader. (Please note that 2x2s may be in terms of pixels or in terms of samples depending on project but are consistent between A21-A27.) Please note that this counter will not advance if a render target update does not occur and that pixel masking operations performed by the fixed function HW or shader may not be reflected in counters A22-A25 which only track their specific defined operations. This can lead to an apparent discrepancy between A21 vs. A22-A25 vs. A26/A27.

Counter #	Event	Description
A28	2x2s Requested from Sampler	Aggregated total 2x2 texel blocks requested from all EUs to all instances of sampler logic.
A29	Sampler L1 Misses	Aggregated misses from all sampler L1 caches. Please note that the number of L1 accesses varies with requested filtering mode and in other implementation specific ways. Hence it is not possible in general to draw a direct relationship between A28 and A29. However, a high number of sampler L1 misses relative to texel 2x2s requested frequently degrades sampler performance.
A30	SLM Reads	Total read requests from an EU to SLM (including reads generated by atomic operations).
A31	SLM Writes	Total write requests from an EU to SLM (including writes generated by atomic operations).
A34	Atomic Accesses	Aggregated total atomic accesses from all EUs. This counter increments on atomic accesses to both SLM and URB. **Workaround** SLM atomics are not included in prior releases by this OA event (only global memory atomics are counted), a workaround using B/C counters is possible.
A35	Barrier Messages	Aggregated total kernel barrier messages from all Eus (one per thread in barrier).

SPM Counters

Counter #	Event	Description
SPM0	EU Stall	Event reflects the condition where an EU is not idle but also not processing an ISA instruction. Each increment of the event reflects 4 clocks where a single EU met this condition.
SPM1	EU IPC	Event counts the number of ISA FPU/EM instructions that GT processes. It comprehends cases where multiple instructions are processed in a single clock. Each increment of this event reflects 8 instructions processed.
SPM2	Threads loaded	Event counts the total number of threads that have been fully loaded onto an EU in a given clock. This event DOES NOT include the time where the thread header is being sent to the EU. The per-clock increment is added to an accumulator each clock, a single increment of this event reflects that the accumulator has reached 8.
SPM3	EU Not Idle	Event reflects the condition where an EU is not idle. Each increment of the event reflects 8 clocks where a single EU met this condition.

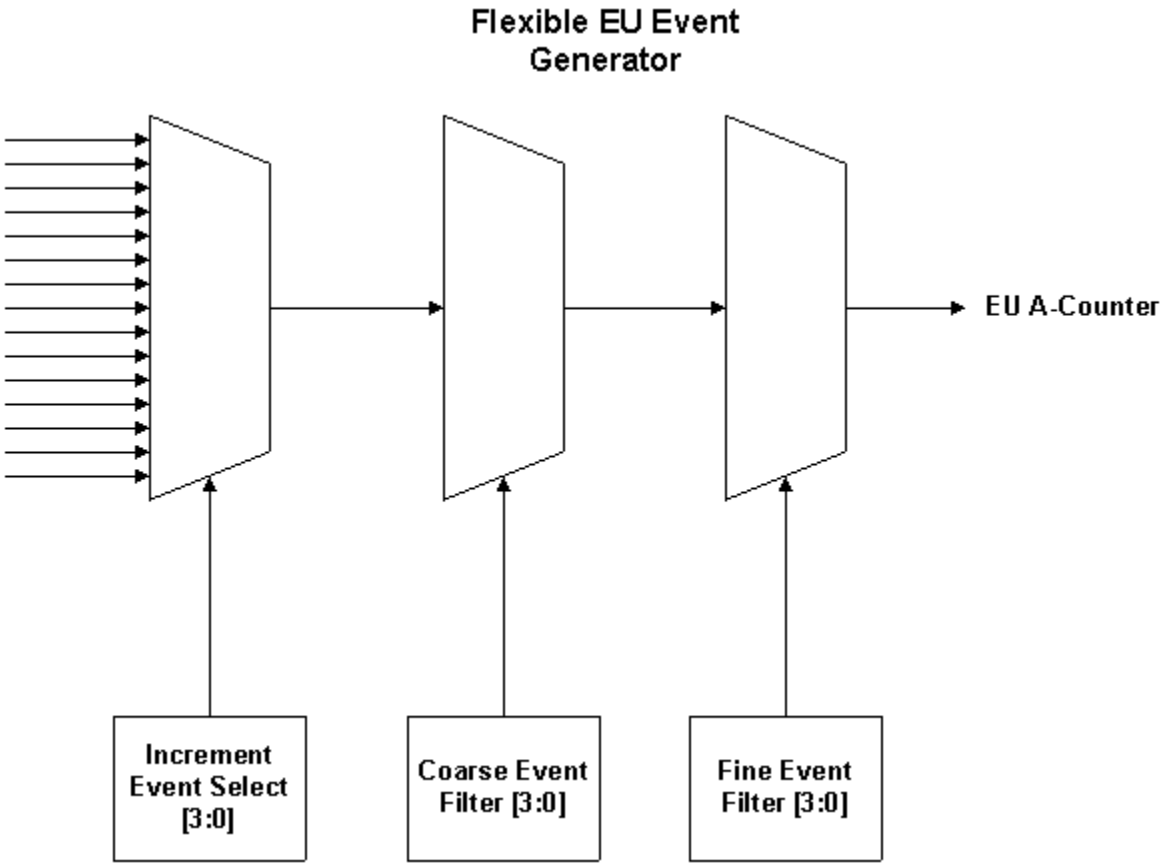
Counter #	Event	Description
SPM4	Sampler Not Idle	Event counts sampler activity.
SPM5	EU Stalled & Sampler Not Idle	Event reflects the condition where the EU has sent a request(s) to sampler and all threads on the EU are stalled. Please note that the EU could be stalled for reasons other than sampler as well. Each increment of the event reflects 8 clocks where a single EU met this condition.

Flexible EU Event Counters

Since EU performance events are most interesting in many cases when aggregated across all EUs and many interesting EU performance events are limited to certain APIs (e.g. hull shader kernel stats only applicable when running a DX11+ workload).

The following block diagram shows the high-level flow that generates each flexible EU event.

Note that no support is provided for differences between flexible EU event programming between EUs because the resulting output from each EU is eventually merged into a single OA counter anyway.



Supported Increment Events

Increment Event	Encoding	Notes
EU_INST_EXECUTED_ALU0_ALL	0b00000	Signal that is high on every EU clock where the EU FPU pipeline is actively executing an ISA instruction.
EU_INST_EXECUTED_ALU1_ALL	0b00001	Signal that is high on every EU clock where the EU EM pipeline is actively executing an ISA instruction.
		Only fine event filters 0b0000, 0b0111, 0b1000, 0b1001, and 0b1010 are supported with this increment event.
EU_INST_EXECUTED_SEND_ALL	0b00010	Number of instructions executed on SEND Pipe. Only fine event filters 0b0000,0b0101, 0b0111, 0b1000, 0b1001, and 0b1010 are supported with this increment event.
EU_PIPE_ALU0_AND_ALU1_ACTIVE_CYCLES	0b00011	Signal that is high on every EU clock where the EU FPU and EM pipelines are both actively executing an ISA instruction. Only coarse event filters 0b0000, 0b0111, and 0b1000 are supported with this increment event. Only fine event filters 0b0000, 0b0111, 0b1000, 0b1001, and 0b1010 are supported with this increment event.
EU_ACTIVE_CYCLES	0b00100	Number of occurrences of signal that is high on every EU clock where at least one EU pipeline is actively executing an ISA instruction. All coarse event filters are supported. Only fine event filters 0b0000,0b0101, 0b0111, 0b1000, 0b1001, and 0b1010 are supported with this increment event.
EU_STALL_CYCLES	0b00101	Number of occurrences of signal that is high on every EU clock where at least one thread is loaded but no EU pipeline is actively executing an ISA instruction. All coarse event filters are supported. Only fine event filters 0b0000, 0b0111, 0b1000, 0b1001, and 0b1010 are supported with this increment event.
EU_THREADS_OCCUPANCY_ALL	0b01000	Number of Thread slots occupied. Accumulated every clock. Implies an accumulator which increases every EU clock by the number of loaded threads, signal pulses high for one clock when the accumulator exceeds a multiple of the number of thread slots (e.g. for a 8-thread EU, signal pulses high every clock where the increment causes a 3-bit accumulator to overflow). Only coarse event filters 0b0000, 0b0111, and 0b1000 are supported with this increment event. Only fine event filters 0b0000, 0b0111, 0b1000, 0b1001, and 0b1010 are supported with this increment event.
	0b01111	Expected HW default, allows logic to be power-optimized.

Supported Coarse Event Filters

Coarse Event Filter	Encoding	Notes
No mask	0b0000	Never masks increment event.
VS Thread Filter	0b0001	For increment events 0b00000/0b00001/0b00010, masks increment events unless the FFID which dispatched the currently executing thread equals FFID of VS.
		For increment events 0b00100/0b00101, masks increment event unless at least one of the loaded threads was dispatched by VS.
HS Thread Filter	0b0010	For increment events 0b00000/0b00001/0b00010, masks increment event unless the FFID which dispatched the currently executing thread equals FFID of HS.
		For increment events 0b00100/0b00101, masks increment event unless at least one of the loaded threads was dispatched by HS
DS Thread Filter	0b0011	For increment events 0b00000/0b00001/0b00010, masks increment event unless the FFID which dispatched the currently executing thread equals FFID of DS.
		For increment events 0b00100/0b00101, masks increment event unless at least one of the loaded threads was dispatched by DS.
GS Thread Filter	0b0100	For increment events 0b00000/0b00001/0b00010, masks increment event unless the FFID which dispatched the currently executing thread equals FFID of GS.
		For increment events 0b00100/0b00101, masks increment event unless at least one of the loaded threads was dispatched by GS.
PS Thread Filter	0b0101	For increment events 0b00000/0b00001/0b00010, masks increment event unless the FFID which dispatched the currently executing thread equals FFID of PS.
		For increment events 0b00100/0b00101, masks increment event unless at least one of the loaded threads was dispatched by PS.
TS Thread Filter	0b0110	For increment events 0b00000/0b00001/0b00010, masks increment event unless the FFID which dispatched the currently executing thread equals FFID of TS.
		For increment events 0b00100/0b00101, masks increment event unless at least one of the loaded threads was dispatched by TS.
Row = 0	0b0111	Masks increment event unless the row ID for this EU is 0 (control register is in TDL so only have to check within quarter-slice).

Fine Event Filters

Fine Event Filter	Encoding	Notes
None	0b0000	Never mask increment event.
Cycles where hybrid instructions are being executed	0b0001	Masks increment event unless the instruction(s) being executed on the pipeline(s) selected by the increment event are hybrid instructions.
Cycles where ternary instructions are being executed	0b0010	Masks increment event unless the instruction(s) being executed on the pipeline(s) selected by the increment event are ternary instructions.
Cycles where binary instructions are being executed	0b0011	Masks increment event unless the instruction(s) being executed on the pipeline(s) selected by the increment event are binary instructions.
Cycles where mov instructions are being executed	0b0100	Masks increment event unless the instruction(s) being executed on the pipeline(s) selected by the increment event are mov instructions.
Cycles where sends start being executed	0b0101	Masks increment event unless the instruction(s) being executed on the pipeline(s) selected by the increment event are send start of dispatch. Note that if this fine event filter is used in combination with increment events not related to the EU send pipeline (e.g. FPU0 active), the associated flexible event counter will increment in an implementation-specific manner.
EU# = 0b00	0b0111	Masks increment event unless the EU number for this EU is 0b00.
EU# = 0b01	0b1000	Masks increment event unless the EU number for this EU is 0b01.
EU# = 0b10	0b1001	Masks increment event unless the EU number for this EU is 0b10.
EU# = 0b11	0b1010	Masks increment event unless the EU number for this EU is 0b11.

Flexible EU Event Config Registers

EU_PERF_CNT_CTL0 - Flexible EU Event Control 0

EU_PERF_CNT_CTL1 - Flexible EU Event Control 1

EU_PERF_CNT_CTL2 - Flexible EU Event Control 2

EU_PERF_CNT_CTL3 - Flexible EU Event Control 3

EU_PERF_CNT_CTL4 - Flexible EU Event Control 4

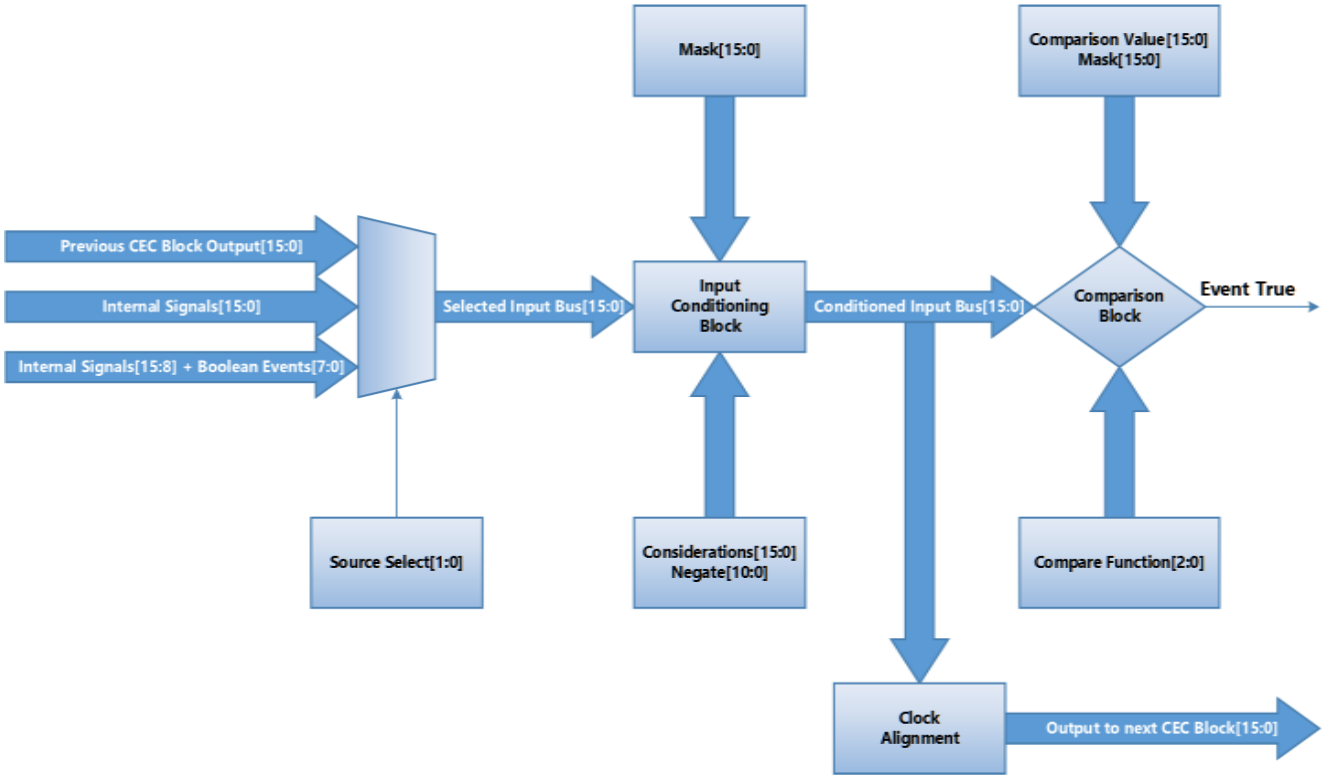
EU_PERF_CNT_CTL5 - Flexible EU Event Control 5

EU_PERF_CNT_CTL6 - Flexible EU Event Control 6

Custom Event Counters

Also known as B-counters, the events counted in these counters are defined from Boolean combinations of input signals using the custom event creation logic built into OA.

The following diagram(s) illustrate(s) the structure used to create a custom event. Every B-counter has such a block.



MI_REPORT_PERF_COUNT

MI_REPORT_PERF_COUNT

GPU Doorbells

On current platforms, graphics workload is submitted to the graphics hardware through multiple SW layers: Application, user mode driver (UMD), Graphics Runtime, kernel mode driver (KMD) and then finally down to hardware. This submission path incurs the penalty of traversing the various SW layers and then a ring3-to-ring0 switch when the kernel mode driver finally submits the workload by writing to some graphics register. With the advent of the GPGPU computing, applications like to use the parallelism provided by the graphics hardware for graphics and non-graphics workloads. However, due to the large latency introduced by the legacy submission mechanism, an App has to determine if the computation cycles required for a workload exceeds a threshold to justify incurring the latency that will be incurred to get this workload running on the graphics hardware.

The proposed GFX doorbell solution provides a mechanism for submission of workload to the graphics hardware by a ring3 application - without the penalty of ring transition for each workload submission. The structure has the following components:

- Assignment of a work queue page by Kernel Mode driver into an Apps address space
- Infrastructure in GTI/BGF to detect submission of work into a N work queue pages; GTI detects a write to a specific cacheline in the doorbell page.

- Message from GTI/BGF to a micro-controller in the graphics engine indicating new work has arrived in a queue.
- Microcontroller inspects queue and determines the specific graphics engine to submit the workload to (render engine,...).

Essentially, the hardware will support monitoring of 256 cache-line addresses that are on separate 4KB pages. Each GPGPU App/thread negotiates with KMD on the address of this 4KB page (in app's space) as a "doorbell" page. This negotiation happens during the app's call to the driver (KMD) for open GPGPU. The page is assigned for doorbell operation which is basically a way to MONITOR accesses to the specified cacheline within the doorbell page.

This portion of the document details the interactions needed in GT BGF Doorbell Block as well as the HW pieces.

The Concept and Usage Flow

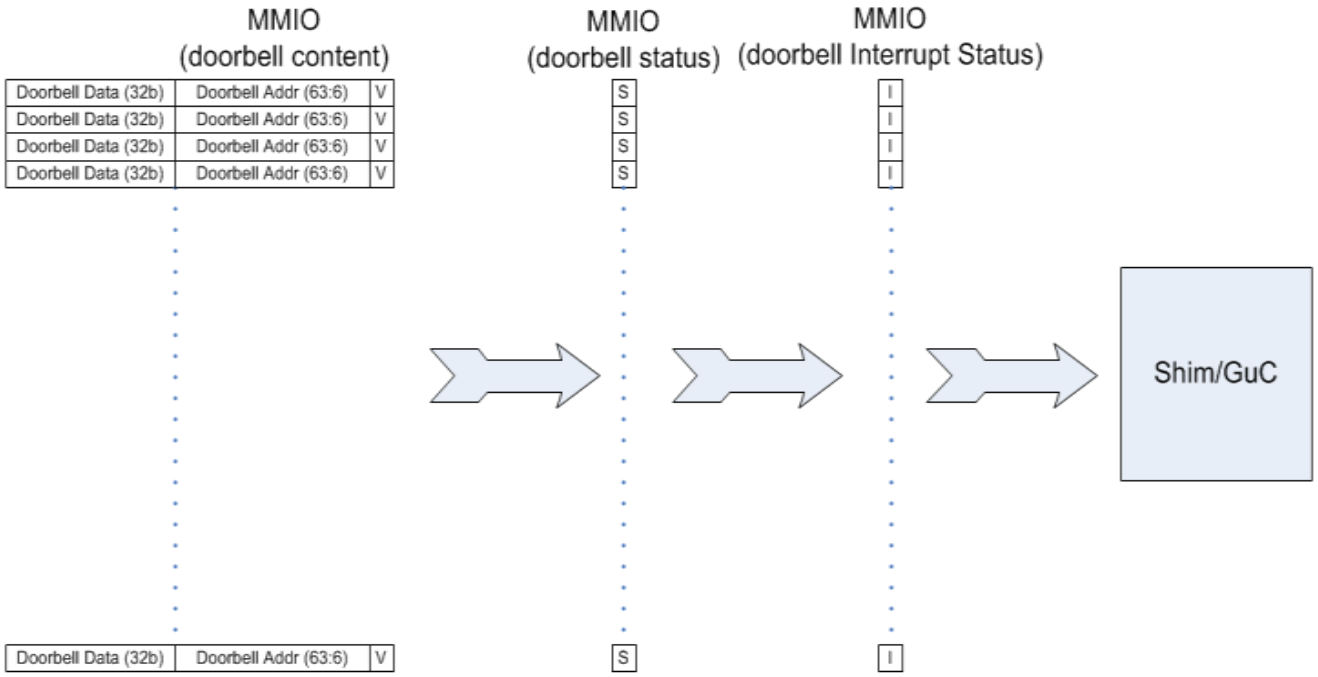
The interactions between the doorbell hardware and software flows described as viewed from the doorbell controller:

Doorbell Structures

Doorbell structures for reporting have been separated into three logical blocks in the hardware. First stage is where doorbell address is stored along with Valid bit. Once Valid is set (written by KMD/GuC), doorbell block will start monitoring and store the cookie value (DW[0] of the doorbell line).

Once a doorbell is triggered corresponding Status bit is set. Eventually status is propagated to interrupt status bit, meanwhile multiple triggers on doorbell will be collapsed within the status bits.

A read from interrupt status stage to doorbell status stage moves the bits to interrupt status register and clears the doorbell status register.



All stages are exposed via MMIO:

Doorbell Address. Read/Write support for the driver, but thru GuC/Shim (no direct updates).

Valid. Read/Write support however there are restrictions which hardware needs to ensure:

- 0=>1 transition only from GuC via message channel (not allowed for any other message channel client and memory reads of related doorbell does not change the doorbell value from 0=>1.
- 1=>0 transition is only from memory read that doorbell block does when it needs to acquire the doorbell cookie.

Hardware has to guarantee proper clients make the transitions.

Doorbell Status. Read Only (updates are HW managed).

Doorbell Interrupt Status. Read/Write - it is set by hardware and cleared by driver/GuC.

Doorbell Page and Cacheline

As part of the "*openGPGPU*" call between the App and GFX driver a doorbell page is assigned. The page is allocated within the WB space (requirement) and pinned to prevent any faulting while exchanging semaphores (requirement). The doorbell monitoring hardware operates on each doorbell at a 64B granularity.

App and KMD negotiates which cacheline (64Byte) entity within the doorbell page should be picked as doorbell line. Even though this could be simply the first 64B of the page, it is highly recommended to pick a rotating 64B entity to maximize the hardware performance. The suggested algorithm should be:

Doorbell CL# = Bell# mod 64

This simply means Bell0,64,128,192 goes to CL0 of the page, Bell1,65,129,193 goes to CL1 of the page, Bell2,66,130,194 goes to CL2 of the page, and so on.

Only the first 2 DWords of the Doorbell CL is relevant, the rest is not used by the doorbell hardware. Same is also applicable for the rest of page which could be utilized for some other means.

1st DWord (4 bytes) contains 1-bit for doorbell status: Active vs. Inactive:

- Active - Bit0=1: Doorbell is Active for Application.
- Inactive - Bit0=0: Doorbell is suspended for the Application, hence Application should re-negotiate the doorbell.

Note that as part of the negotiation a different cacheline can be picked within the doorbell page, different than what was originally agreed between GFX Driver and Application.

GFX driver control the initial setting of bit[0] of DW[0] to enable and disable doorbell for a given application. However HW setup of doorbell can be done only via GuC and disabling in HW is self-detected via snooping driver's clearing of DW0[0].

2nd DWord (4 bytes) carries a cookie value. It is reset to "0h" by the GFX driver prior to being assigned and acquired and incremented by the App. Note that value "0h" is a reserved value for cookie which App should never write. A roll-over increment should go from "FFFF_FFFFh" to "0000_0001h". Details on how cookie and flag value should be processed by application and hardware are given in later sections.

Work Queue and DoorBell Creation

App calls the driver for "OpenGPU()API", UMD calls the KMD and KMD sets up the queues. KMD maps the queue addresses to application space. KMD assigns a doorbell if one available:

- **If available:** Doorbell page gets allocated and assigned to Applications space. Doorbell cacheline is decided and QW0 is initialized as:
  - DW0 = "0000_0001h"
  - DW1 = "0000_0000h"
- **If not-available:** Driver has to victimize doorbell from an existing application and wait for it to clear from HW before assigning to new application requesting one.

KMD assigns a unique ID (i.e. Context ID) which is a 32b value to define the context. KMD allocates HW context memory for "State" - up to 3MB.

Note that the SW flows here are a high level description; the actual flows may contain further steps and will be detailed in SAS.

KMD communicates the contextID as a handle to memory (work queue) to GuC. If there is a doorbell assignment, GuC also receives the doorbell number as well as the physical address of the cacheline decided.

GuC has to ensure there are no pending doorbell events before updating the doorbell register for use. To ensure this, GuC should read "doorbell status vector". If there are any pending events to the same doorbell that is about to be assigned, GuC expedites the service of the matching vector and waits for

status to clear. If status is cleared, GuC needs to check the "doorbell interrupt status" register in the GuC/Shim to ensure there are no pending interrupts. Once both stages are clear, GuC can proceed with re-assignment of the doorbell.

The doorbell address is GPA. Rest of the flow is between the GuC Shim/GAM and Doorbell control block:

- GuC the address (GPA) of the doorbell page/cacheline. Given the address is GPA, a 2nd level page table translation is required before the monitor address can be written to doorbell block.
- GuC sends the following message pair to GAM asking for a GPA to HPA translation.
- Doorbell Address programming has to be routed thru GuC/Shim - driver (KMD) is not allowed to update the doorbell addresses directly.

Doorbell Flows

Doorbell Controller has a single primary flow of acquiring the ownership of a doorbell cacheline and detecting and reporting doorbell rings. Besides the main flow, there is various side flows to handle power states.

There are two cases where the main (ownership) flow can be invoked:

1. VALID bit in one of the doorbell registers transitions from 0=>1 indicating a new assignment of doorbell. This is considered initial flow.
2. A snoop from LLC/Cbo removing the ownership of a doorbell cacheline that was already being monitored. This is considered subsequent flow.

Both flows are handled same, as they overlap whether this is the initial ownership flow or subsequent case.

Doorbell controller collects the results of the RFO where data gets processed:

- If DW0[0] - flag - is seen as "0", that means s/w already victimized the doorbell and in the process of re-assigning it. However it is possible that application had a chance to sneak in work at the very last opportunity. And the doorbell register is still valid.
- If DW0[0] - flag - is seen as "1", the doorbell is still active.

Both cases are handled similarly in the doorbell controller.

- The value of cookie is checked:
  - If cookie is "0000_0000h", this is the initialized value of the cookie by GFX driver. Application did not use the doorbell yet. Doorbell block initializes the cookie value stored in h/w and does not ring the bell.
  - If cookie is same as what was stored in the cookie register this means either we are still at the initialized value or the ownership was lost due to LLC victimization and there is no RING event.
  - If cookie value is different than the previous value and it is not all "0's", this is a VALID RING event.
- In case of a valid RING event,

    - Doorbell controller sets up the corresponding bit flag in the "Active Doorbell Status Vector".
    - If in RC6, generate a wake-event to GT-Shim pm block.
        - Doorbell Block has to be aware of the RC6 status and need to be able to communicate the wake up request to Shim-PM block.
        - The message to Shim has to be hold-off until we wake up GT.
    - If not in RC6 or woken up from RC6, generate a message to GuC unless the 32b group of active vector is non-Zero. Meaning there has been already a message to Guc and GuC did not have a chance to read the corresponding "Active Doorbell Vector" piece. The communication between the doorbell monitoring hardware and GuC is handled via internal messaging.
- Once a group's message is sent (via message channel) subsequent rings to the same group does not cause a message to GuC. The only time a message is sent if the previous value of the doorbell vector for the corresponding group was "0000_0000h".
- At some point, GuC will read the vector from the doorbell block to see which doorbells have rung. Same read event should clear the corresponding doorbell vector register allowing any subsequent rings to generate a new message back to GuC.

Assigning Doorbells

Doorbells assignments are straight forward - see doorbell creation section

In addition s/w shall not assign the same cacheline to more than one doorbell. Such assignments will create unexpected behavior when the doorbell is rung.

Removing Doorbells

Doorbells are floating hardware resources, and they can be temporarily attached to Apps. GFX Driver can remove an application's doorbell without the need to notify or negotiate with that App. The removal process described requires KMD to poll doorbell status locations in HW to ensure doorbell is freed before re-assigning. And re-assignment process is done making sure all previous interrupts from this doorbell are processed.

An application losing a doorbell simply needs to request one as it realizes that its doorbell is lost. This is considered as a doorbell virtualization process where GFX driver and GuC can move a limited number of doorbells between a practically unbounded (2^31) number of applications.

The removal processes for doorbells are managed by the driver (KMD). However, it is completely asynchronous to what applications are doing with the doorbells and where HW is in processing these doorbells. The following flow is used to de-activate a doorbell.

As part of the de-activation:

1. GFX driver decides which doorbell to victimize and clears the DW0[0] of the doorbell cacheline in WB space. This is an indication to the GPU doorbell controller that the corresponding doorbell is removed. The doorbell block in GPU will see the update from IA core as part of the snoop monitoring (see later sections on how application checks the flag via compare&exchange).

From this point on, Application will not be able to ring that doorbell, and it needs to request a new one once it realizes its doorbell is lost.

2. The GFX driver clearing DW0[0] of the doorbell is seen by the GFX doorbell hardware as de-activation. However, there is a possibility that the App had a chance to use it to increment the cookie just before doorbell got de-activated. The doorbell block compares the cookie value new vs old to make a decision to set the interrupt status bit or not.

   From this point on we know that the particular doorbell cannot be used and HW will not match incoming snoops even if there may be snoops to the line (Valid bit is managed by the HW when getting cleared).

3. After GFX driver clears the DW0[0] of the doorbell that is getting victimized, it polls the doorbell register VALID bit in HW which indicates when HW got to see the victimized doorbell and cleared the VALID bit state. This also ensures that all previous updates of the victim doorbell are now registered in the HW.
4. Once the GFX driver sees the VALID bit cleared in the doorbell register, it is free to re-allocate the same doorbell for another APP. Note that assignment of the doorbell requires GuC to clear all previous usages of the previous owner of that doorbell.

Now the doorbell is ready to be re-assigned to any application.

Application Flow

The application use of doorbell is described in this document for completeness and to give the design owners what to expect.

Application can start using the doorbell once agreed with the GFX driver, from this point the main part of the flow is to be able cache the value of the cookie (if initial assignment application knows cookie is "0" and next value is "1"). The flow relies on the fact that application simply increments the cookie by "1" to indicate a doorbell event. And application has the value of cookie and the next value that it wants to update to. To understand how the rest of the sequence works, one needs to understand the inner works of the LOCK_CMPXCHG8B.

1. Assumption thread knows the value of the cookie - if this is the first time using it, it knows KMD initialized to "0". Thread prepares ECX = 1
2. Thread does compare exchange 8B (LOCK_CMPXCHG8B)

a. Outcome#1: **Success** - thread gets "ZF == 1", rings the doorbell successfully and resumes execution

b. Outcome#2: **Fail:** Bell is present (EAX =1) but EDX is not the expected value - *means some other thread is sharing the same bell.*

   i. Load ECX with EDX, and INC ECX - *prepare the next value for destination*

   ii. And loop to compare exchange (step#2) - *at this point this is a RACE to the bell between multiple threads. The thread that succeeds updates the bell*

c. Outcome#3: **Fail** : Bell is not there (EAX = 0) - ask for a bell and loop to step#1

In both cases algorithm is self correcting. For the case of one doorbell to one application thread, Outcome#2 is never possible given the application thread exactly knows what the value of the cookie is (it is the only thread writing to it). Outcome#3 is possible if GFX driver decides to remove the doorbell.

For the case of one doorbell to many application threads, each application thread has an idea of what the value of the cookie could be. If this is the first time the thread is writing the cookie value, it assumes "0"; if not the first time, it assumes whatever the value it wrote last time. In this case Outcome#2 becomes possible given that other threads are doing LOCK_CMPXCHG8B on the same data structure as well. However the good part of the LOCK_CMPXCHG8B is that, operation is atomic and a fail case (i.e. mismatch of the compare) returns the most up to date value from memory and does not change the contents of the memory. For multiple application-thread case, all threads are going to be racing to the flag while making successful updates and any doorbell ring is going to get uC to check and process all application-threads' work queues.

Atomic operation, "CMPXCHNG8B", is a required instruction to use if multiple application threads are mapped to same doorbell or same doorbell is virtualized via KMD between multiple applications. The latter part refers to a case where KMD removes a doorbell without letting the application know about it. If there is no multiple thread or virtualization case for doorbells, simple memory operations are suffice to operate with doorbells.

Doorbells and Lock

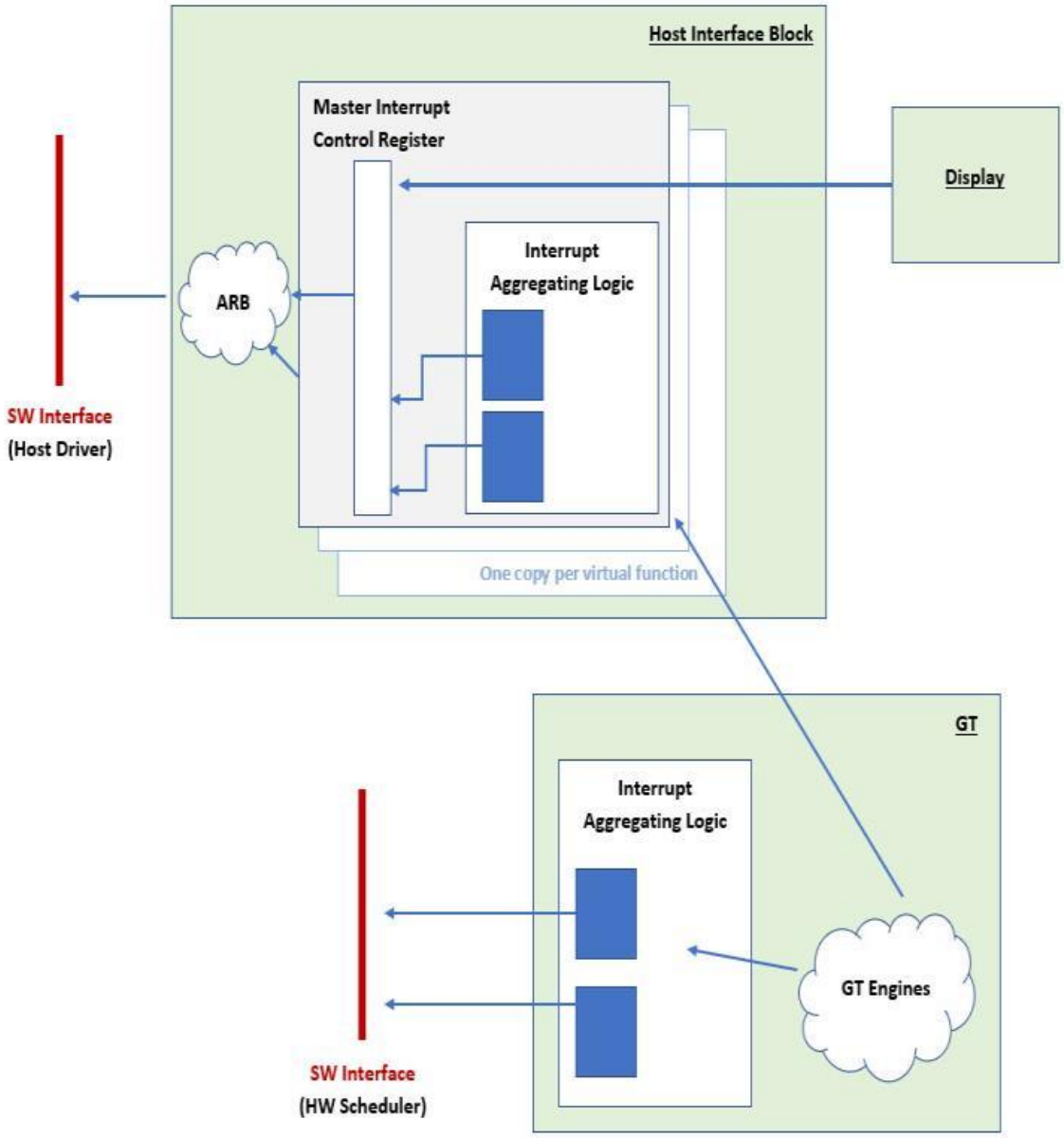
The fact that doorbell block is capable of generating accesses to IDI, it will have to be involved in LOCK process. The impact has to be looked in two different operational modes:

Interrupts Overview:

The Graphics device is comprised of a number of independent engines that can be invoked to execute workloads. Engines communicate status primarily through interrupts. The Graphics device supports two models of scheduling and handling of interrupts:

- Host SW schedules and manages all interrupts
- Scheduling and related interrupts are managed by hardware scheduler (MinIA micro-controller) and host SW manages interrupts not related to scheduling.

The hardware can be configured to work in either of these models. HW scheduling is the preferred mode because it provides best utilization of resources. The figure below shows the high level overview of the interrupt infrastructure.



The interrupt infrastructure is designed to support both of these models. Each engine is allocated a set of interrupt bits that it can set to report events (the number of bits allotted to each engine varies -- most engines are allocated 16bits, some engines which have more events are allocated 32bits). Interrupt messages sent by engines result in interrupt bits being recorded in MMIO registers and an interrupt being generated to the servicing agent (MinIA scheduler or Host SW). The interrupt handler determines the source of the interrupt (by reading registers) and then processes the interrupts. Processing interrupts involves reading the interrupt status register, performing the operations for handling the interrupt and indicating completion of handling by writing to registers (clear).

When using the HW scheduler, the scheduling related interrupts are directed to the MinIA scheduler.

GT Engine Interrupts:

Within GT, engines are categorized into different engine classes and instances. An engine class is used to differentiate between engines that perform different functions (Copy, Render, VideoDecode, VideoEncode, etc). A product may have a number of instances of a specific engine class e.g.: GT2 has 2 instances of VD, GT3 has 4 instances of VD, etc. The following table lists various engine classes as well as instances within each class.

Engine Class	Engine Instance Name	ClassID[2:0]	InstanceID[5:0]
Render	RCS	0	0
Video Decode	VCS0-N	1	0-N
Video Enhancement Engine	VECS0-N/2	2	0-N/2
Copy Engine	BCS	3	0
Other	GuC	4	0
	GTPM	4	1
	WDOAPerf	4	2
	SCTRG	4	3
	KCR	4	4
	Gunit	4	5
	CSME	4	6
Compute Engine	CCS0-N	5	0-N
Reserved		6-7	

Each engine reports up to 16 interrupts to interrupt handling logic. Source identification data is included in interrupt messages to interrupt aggregating logic, i.e. when reporting an interrupt to either host or graphics firmware, the generating engine must identify itself. 16 bits of identification is sent along with interrupt data, and comprises Engine Class ID, Instance ID and Virtual Function Number. Interrupt bit definition varies per engine class, these are listed in the Bspec in the Global/ section.

Format of interrupt message:

Bit Fied	Purpose
[31:30]	Reserved
[29:27]	VF ID
[26]	Reserved
[25:20]	Instance ID

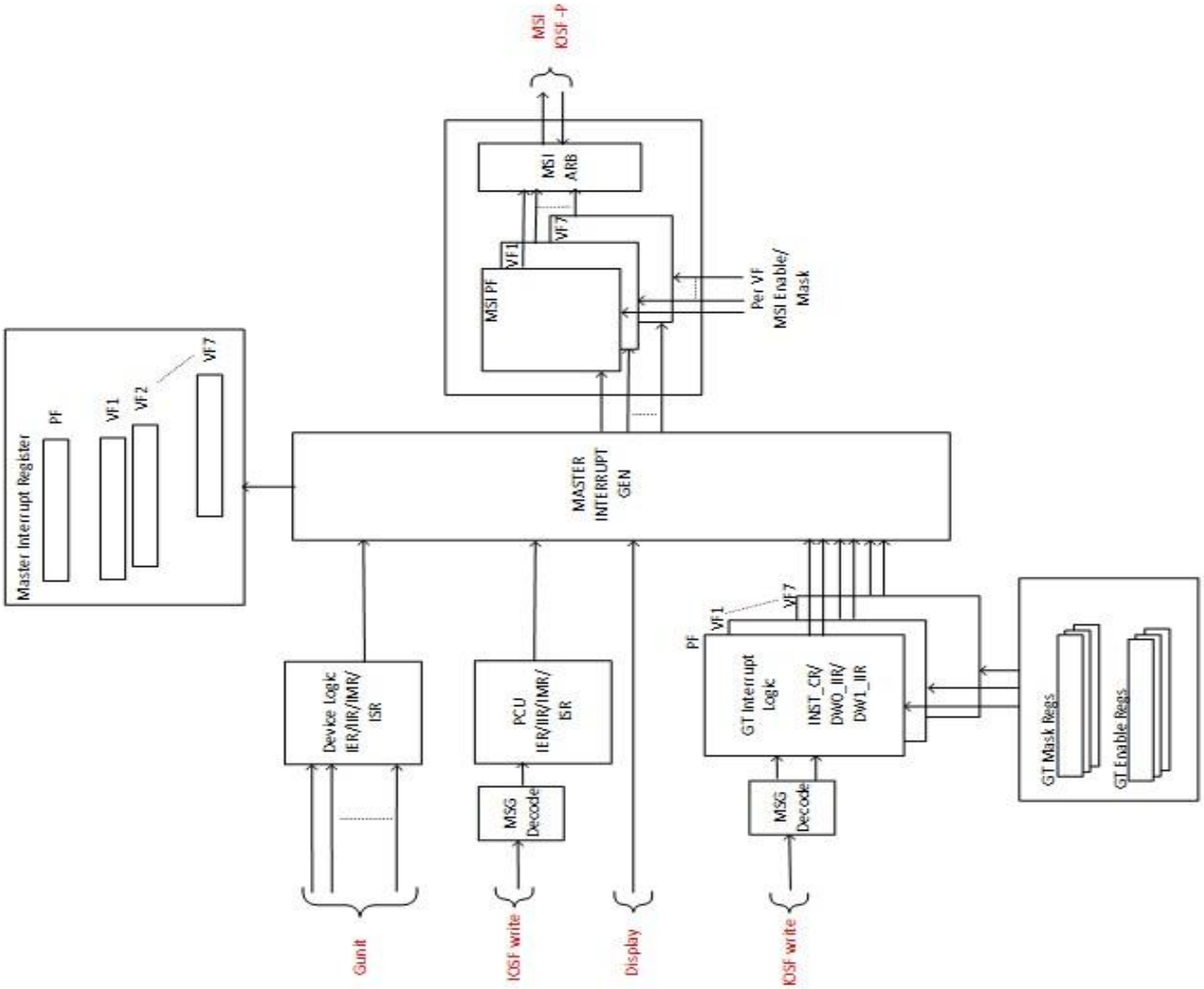
[19]	Reserved
[18:16]	Engine Class ID
[15:0]	Interrupt data

Hardware Scheduler/MinIA SW Interface

Graphics interrupts to scheduling firmware are delivered as two unique vector values. Each vector accounts for 32 graphics engines. Firmware processes each of two groups of graphics engines independently.

Service routines are independent for the two interrupt vectors presented to the MinIA firmware.

Host SW Interface



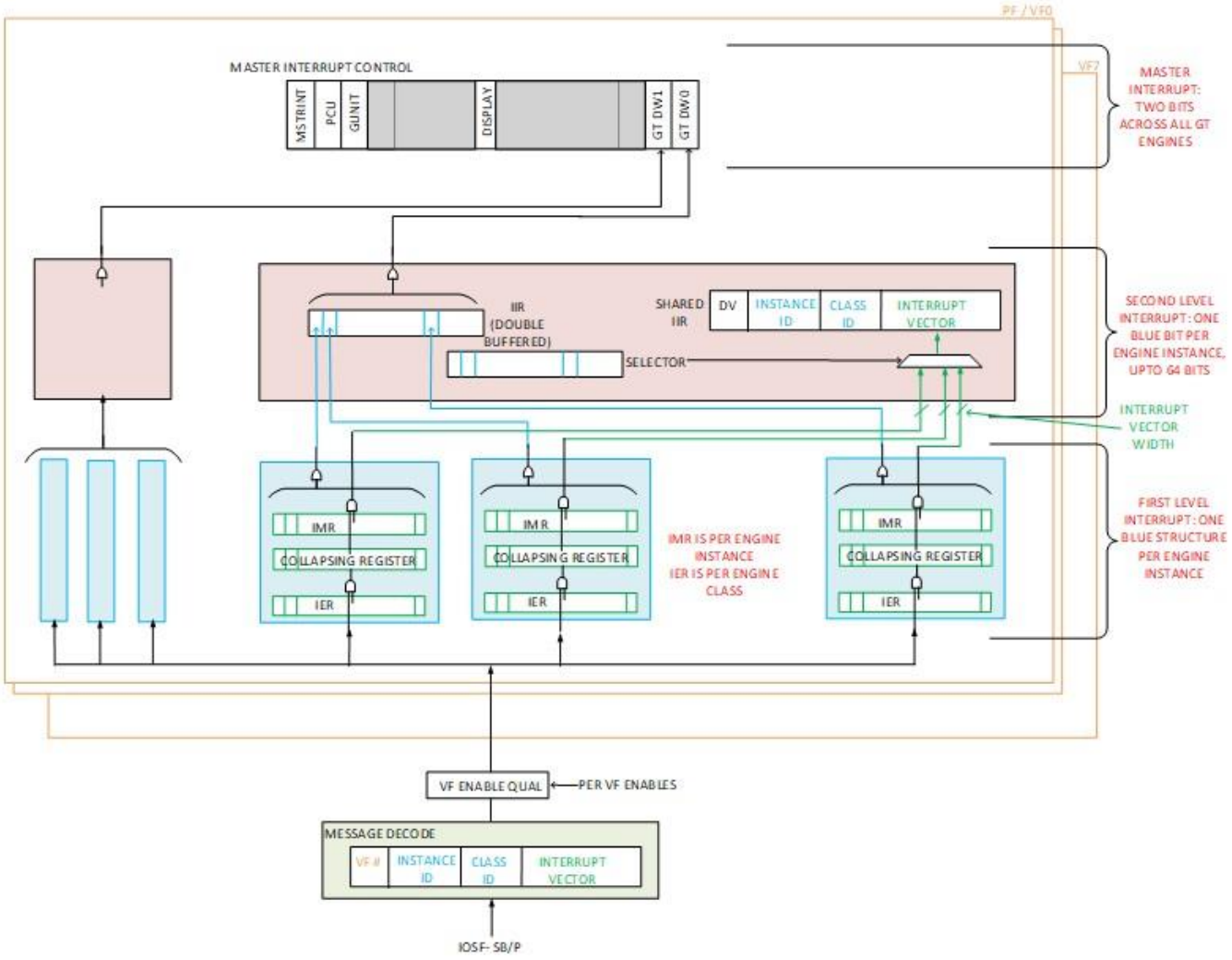
Interrupts to Host are delivered via a Primary Interrupt Control Register. Graphics interrupts use 2 bits in the Primary Interrupt Control Register. In addition, interrupt events from Display are also represented in the Primary Interrupt Control Register. Multiple copies of Primary Interrupt Control Register exist, one for every virtual machine in the system.

Interrupt bits in the Primary Interrupt Control Register are Read-Only bits, and are level indications that a second level interrupt is present (As seen earlier, second level interrupts per client are OR-ed together. When the second level IIR is cleared, the bit represented will be 0.). An interrupt is sent to driver whenever bits are set in the Primary Interrupt Control Register and the Enable bit is also set.

As a result of this interrupt, SW first resets the Primary Control Enable bit. SW then reads the Primary Interrupt Control register into a local variable, and works off this local variable to service interrupts. Once all lower level interrupts have been serviced, SW writes the Primary Interrupt Control register to set the Primary Control Enable bit.

Interrupt Aggregating Logic

A hierarchical interrupt status infrastructure is provided to efficiently determine the source of the interrupt. The first level of interrupts is generated by GT Engines. Interrupt handling logic accumulates these interrupts from the various engines, and organizes it as a single bit per engine in a second level. 32 bits of second level interrupts are OR-ed together to generate a DW-level interrupt event for up to 32 engines. Two such events are used to provide support for up to 64 GT engines. When communicating with the MinIA, these events are mapped to two unique interrupt vectors in the MinIA LAPIC. When communicating with host driver, these events form two bits of the Primary Interrupt Control Register as marked in the picture.



First Level Interrupt Bits:

When an interrupt event comes into the interrupt handling logic, it is AND-ed with a per-Engine Enable register (IER). Only enabled events make forward progress. Disabled events are simply dropped by the interrupt handling logic. [Note that multiple instances of the same engine type (except those in the 'Other' Engine Class) share the same Enable register.]

Enabled interrupts are logged in a per-instance, non-SW readable Collapsing Register. These events are AND-ed with (the inverse of) a per-Instance Mask Register (IMR). Only unmasked events make forward progress. Masked events remain in the per-Instance Collapsing Register until they are unmasked. [Note that every instance (even of the same engine type) has its own Mask Register.]

Unmasked events in the per-Instance Collapsing Register are OR-ed together to produce a single second level interrupt event.

Second Level Interrupt Bits:

Second level interrupt events are stored in a double buffered IIR structure. A snapshot of events is taken when SW reads the IIR. From the time of read to the time of SW completely clearing the second-level

IIR (to indicate end of service), all incoming interrupts are logged in a secondary storage structure. This guarantees that the record of interrupts SW is servicing will not change while under service.

Bits in the second-level IIR are OR-ed together to generate a DW-level event. The IIR is cleared by writing 1s. If events exist in the secondary storage at the time that the IIR is completely cleared, a second DW-level event will be generated.

Shared IIR, Selector:

Shared IIR and Selector registers are used when SW is in the process of handling reported interrupts. As a result of a GT interrupt (DW-level interrupt), SW reads the second-level IIR register. The read provides an indication of engines needing service. SW must then service engines one at a time by writing a one-hot selection into the Selector Register.

When a selection is made by writing the Selector, interrupt handling logic presents all the unmasked interrupt bits (first level interrupt events) for the selected engine in the Shared IIR, and sets the Data-Valid bit (MSB). SW can then read the Shared IIR and take action for the reported events. SW must clear the Shared IIR by writing 1 to the Data-Valid bit to indicate end of service for the selected engine. This clearing of the Shared IIR Data-Valid bit clears both the Shared IIR as well as the Selector. Note that the Selector data must be one-hot. Selector must not have a bit set that is not set in the second-level IIR at the time of SW read.

SW then repeats the above steps for each bit set in the second-level IIR. Multiple rounds of Selector write-Shared IIR clear may be required to service a DW level interrupt a single time.

Second-level IIR bits are cleared only after individual engines are serviced via the Selector write -Shared IIR clear routine. This clearing can be done after each iteration through the Selector write-Shared IIR clear routine (i.e. one second-level bit cleared after each iteration), or all at once after all engines have been serviced. Second-level IIR bits must not be cleared without first servicing that engine's interrupts via the Selector and Shared IIR registers.

Enable and Mask Registers:

Interrupt aggregating logic includes Enable registers(IER) per Engine Class. Different instances of the same engine class use the same Enable register, except for engines in the 'Other' class. Each instance in the 'Other' class has its own Enable register.

Interrupt aggregating logic also includes Mask registers (IMR). Each engine instance, even within the same Engine Class, has a unique Mask Register.

Enables for Engine classes at the two software interfaces are typically complements of each other.

Intel® Iris® Xe and UHD Graphics Open Source

Programmer's Reference Manual

For the 2020-2021 11th Generation Intel Xeon®, Core™, Celeron®, Pentium® Gold Processors based on the "Tiger Lake" Platform

Volume 14: Workarounds

December 2021, Revision 1.0

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Table of Contents

H81 Workarounds 1
UP3_UP4_H35 Workarounds 34

H81 Workarounds

impact	title	bspec_wa_details	sku_impact
	Audio 8K1port - For certain VDSC bpp settings, hblank asserts before hblank_early, leading to a bad audio state	WA details can be found at: Display Engine > North Display Engine Registers > Audio > Audio Programming Sequence under "Audio Hblank Early Sequence"	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	HDC L3 write moves forward for a L1 cacheable write when Sampler is stalling and can result in RAW hazard	DW-1 Bit-13 of State Compute Mode register (field name: Disable L1 Invalidate for non-L1-cacheable Writes) must be set to 0 by driver.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Invalid occlusion query results with "Pixel Shader Does not write to RT" bit	When Pixel Shader Kills Pixel is set, SW must perform a dummy render target write from the shader and not set this bit, so that Occlusion Query is correct.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang	GFXPERF: Shadow of Mordor: frame 253 hang in WW40d model	Set register bit (7018h) bit 13 = 1 when depth buffer is D16_UNORM	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
other	PSS X-prop issue in quad_valid when we see an unlit poly on the back of a chg marker with no SIMD modes enabled by the programmer	It is unknown if detected X-prop issue can generate Si failures. To avoid any possible issues, set at least one simd enable in 3dstate_ps (e.g. 16 pixel dispatch enable). If no pixel shader is valid, clear 3dstate_ps_extra "pixel shader valid"	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	OVR causes a Page fault when running out of free pages in PTBR PAGE POOL	The driver has to map 1 page of dummy resource to address PTBR_PAGE_POOL_BASE_ADDRESS + (0xFFFF * 4KB).	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
other	GRF source swap feature for SIMD16 with Src0 scalar and bundle conflict between Src1/Src2 is causing the GRF read issue.	WA: Driver must set E4F4[14]=1 to disable early read/Src Swap.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
	Default BCredits on MBUS insufficient to meet required display bandwidth	Issue: Default BCredits on MBUS insufficient to meet required display bandwidth WA: Display MBUS_DBOX_CTL* registers should be programmed with BCredit value of 12 (e.g. 7003C[12:8] = 0xC). Note that there are multiple instances of this register, one for each display pipe (A, B, C, D).. All instances should be programmed to the same value.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang	Coarse Pixel Shading - hang can occur in color pipe if CPS Aware color pipe optimization is enabled	Disable CPS Aware color pipe by setting register bit: 0x07304 Bit[9].	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
other	Coarse Pixel Shading - perf issue with floating point render targets if CPS Aware color pipe optimization is enabled	Disable CPS Aware color pipe by setting register bit. 0x07304 Bit[9]	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Coarse Pixel Shading - corruption can occur with R11G11B10_FLOAT render target if CPS Aware color pipe optimization is enabled	Disable CPS Aware color pipe by setting register bit 0x07304 Bit[9].	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Coarse Pixel Shading - data corruption can occur if CPS Aware color pipe performance optimization enabled	Disable CPS Aware color pipe by setting register bit: 0x07304 Bit[9].	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption, hang	Coarse Pixel Shading - hang or data corruption can occur with 16X MSAA if CPS aware color pipe optimization is enabled	Disable CPS Aware color pipe by setting register bit: 0x07304 Bit[9]	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
other	DPT should send VRR enable indicator to DCPR even while Push mode is enabled.	Package C2 increase when VRR is enabled with push mode. When enabling VRR, before setting TRANS_VRR_CTL VRR Enable, program GT-driver Pcode mailbox with	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
		command 0x11 and data low bit 0 = 1 to inform pcode that VRR is enabled. When disabling VRR, after clearing TRANS_VRR_CTL VRR Enable, program GT-driver Pcode mailbox with command 0x11 and data low bit 0 = 0 to inform pcode that VRR is disabled.	
data_corruption	Underrun when FBC is compressing with odd plane size and first segment is only 3 lines	FBC causes screen corruption when plane size is odd for vertical and horizontal. Set 0x43224 bit 14 to 1 before enabling FBC. It is okay to leave it set when FBC is disabled.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Coarse Pixel shading Data corruption due to dropping CP Subspan with Alpha2Coverage if CPS aware color pipe optimization is enabled	Disable CPS Aware color pipe by setting register bit: 0x07304 Bit[9].	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	VP9 VDEnc encode: segmentation within 64x64 block picks wrong segment id	Program same stream-in segmentation id for all four 32x32 blocks of SB64.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
other	RCS/POCS/CCS/BCS: Reserved fields in "Instdone" Registers are tied to "0" instead of "1"	Software must ignore the Reserved Fields in the INSTDONE register.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Data Corruption with Coarse Pixel Shading + Dual Source Blend + Dual SIMD8 pixel shader dispatch	CPS cannot be enabled alongside Dual SIMD8 Dispatch and Dual Source Blend	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
performance	Sampler cache can be thrashed in certain cases involving texture arrays resulting in low performance	added a programming note to the Render Surface State BXML saying the Array bit should not be set unless the depth of the arrayed surface is > 1.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
data_corruption	MI_ATOMIC uses wrong address for atomic operation in RCS.	MI_ATOMIC command when programmed with “Inline Data” field set to “0” must have “Dword Length” field of the command set to “9h” and must have Dword3..10 programmed with data as 0x0.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	WGF11RenderTargets failure	Issue: CPQ can be optimally pipelined from DAPRSS to the color pipe in two phases, one for fill and one for blend instead of breaking down the blend CPs into PQs. Due to bug in DAPRSS, when using R11G11B10_FLOAT format, looks like RTL uses blend data instead of fill data during fill phase. WA: Disable CPS aware color pipe by programming register bit for Common Slice Register3 (0x7304) bit 9 to 1.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang	PipeControl with Depth Flush enable can result in hang	“PIPE_CONTROL with Depth stall Enable bit must be set with any PIPE_CONTROL with Depth Flush Enable bit set “	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang	Register based invalidations for a given engine don't indicate completion if that engine is in a power domain that is powered down	SW need to always send an OA invalidation following any render /compute or media TLB register based invalidation. The sequence from driver/SW should be: (when issuing any register based invalidation) 1) issue a mmio write to any render/compute/media Inval 2) issue a mmio write to OA Inval.register (0xCEEC) 3) Now poll for respective invalidation completion	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
data_corruption	Corruption with FBC and plane enable/disable	Corruption with FBC around plane 1A enabling. In the Frame Buffer Compression programming sequence "Display Plane Enabling with FBC" add a wait for vblank between plane enabling step 1 and FBC enabling step 2.	<table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table>
data_corruption, hang	Certain Non-Pipelined State commands on RCS should work in PipeSelect compute, but don't because of FFDOP clk gating	Listed commands below are the non-pipeline state commands that may get programmed when PIPELINE_SELECT is set to Media/GPGPU in RenderCS. Due to known HW issue when these commands are executed in Media/GPGPU mode of operation, the new state may not get latched by the destination unit and stale value will prevail. In order to WA this issue SW must temporarily change the PIPELINE_SELECT mode to 3D prior to programming of these command and following that shift it back to the original mode of operation to Media/GPGPU. Since all the listed commands are non-pipelined and hence flush caused due to pipeline mode change must not cause performance issues. • STATE_BASE_ADDRESS • STATE_COMPUTE_MODE • 3DSTATE_BINDING_TABLE_POOL_ALLOC Example: Programming with No WA. PIPELINE_SELECT – GPGPU MEDIA_VFE_STATE MEDIA_INTERFACE_DESCRIPTOR_LOAD GPGPU_WALKER 3DSTATE_BINDING_TABLE_POOL_ALLOC MEDIA_VFE_STATE MEDIA_INTERFACE_DESCRIPTOR_LOAD GPGPU_WALKER …….. Programming with WA. PIPELINE_SELECT – GPGPU MEDIA_VFE_STATE	<table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table>

<table>
<tr><th>impact</th><th>title</th><th>bspec_wa_details</th><th>sku_impact</th></tr>
<tr><td></td><td></td><td>MEDIA_INTERFACE_DESCRIPTOR_LOAD
GPGPU_WALKER PIPELINE_SELECT – 3D
3DSTATE_BINDING_TABLE_POOL_ALLOC
PIPELINE_SELECT – GPGPU
MEDIA_VFE_STATE
MEDIA_INTERFACE_DESCRIPTOR_LOAD
GPGPU_WALKER</td><td></td></tr>
<tr><td>performance</td><td>[non-RCS] Preempt delay counter should not be reset on heqt restore abort</td><td>Scheduler must check the Context ID on ACTIVE to IDLE switch to make sure which element was preempted even if it is not the last element of the prior submission.</td><td><table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
<tr><td>hang</td><td>[nonRCS] CS should stop making new DMA req once decided to go to RDOP</td><td>Disable RDOP on Semaphore Wait and Wait for event using register bit. OR Disable Pre-Parser around MI_SEMAPHORE_WAIT and MI_WAIT_FOR_EVENT command using MI_ARB_ON_OFF.</td><td><table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
<tr><td></td><td>Plane with Source keying enabled on format "P010" not going transparent based on color channel selection</td><td>Source keying with source planes in the pixel formats “P010”, “P012”, “P016”, “RGB64 Unit” is not supported;</td><td><table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
<tr><td>hang</td><td>RCS is not waking up fixed function clock when specific 3d related bits are programmed in pipecontrol in compute mode</td><td>SW WA to program PIPE_CONTROl with RT Flush and CS Stall prior to PIPE_SELECT to Compute. This will be revisited while implementing dove tailing to wake FFDOP and issue flush to both 3D and compute Pipe</td><td><table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
<tr><td>power</td><td>RCU should ignore(reset) Media Sampler DOP status of engine which is idle</td><td>In Dual Context Mode of operation, a context can get executed on an engine and switch out with Media Sampler DOP Clock Gate Disabled (can be on Render Engine or Compute Engine). In such a scenario the corresponding engine keeps the Media Sampler DOP Clock Gate Disabled until further a context gets submitted resetting the state to Media Sampler DOP Clock</td><td><table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
</table>

impact	title	bspec_wa_details	sku_impact
		Gate Enabled or both the engines go Idle. This will lead to ineffective DOP Clock Gate of Media Sampler. This may happen under following circumstances: • SW didn't submit the workload exercising Media Sampler bracketed between PIPELINE_SELECT with Media Sampler DOP Clock Gate Disable and Enable respectively in a single dispatch. OR • Media Sampler Workload got preempted before PIPELINE_SELECT with Media Sampler DOP Clock Gate Enable is executed. SW may avoid the inefficient Media Sampler DOP Clock Gate Enable by avoiding above mentioned scenarios, i.e • Make workloads accessing Media Sampler non-preemptable and ensure they are bracketed between PIPELINE_SELECT with Media Sampler DOP Clock Gate Disable and Enable respectively. Or • Following a context switch status of Active to Idle for a Media Sampler workload from and engine and while other engine is busy, SW must submit a context (dummy no real workload) to the former to reset the Media Sampler DOP Clock Gate to be Enabled.	
hang	Semi pipelined flush not backpressuring when stencil buffer state is enabling thread dispatch resulting in hang	Issue: Semi pipelined flush not backpressuring when stencil buffer state is enabling thread dispatch. Workaround: An additional pipe control with post-sync = store dword operation would be required.(w/a is to have an additional pipe control after the stencil state whenever the surface state bits of this state is changing).	sku: ALL; stepping_impacted: b0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
hang	Semaphore_signal with post sync enable does not send the correct signal data to GUC	Due to known HW issue, SW must not set “Post-Sync Operation” field for MI_SEMAPHORE_SIGNAL command	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
other	Display software needs to configure SSC enable in a new PLL register	DPLL SSC enable is not correctly hooked up to DPLL_CFGCR0 SSC Enable field. WA: Use DPLL_SSC sscen field to enable SSC instead of DPLL_CFGCR0 SSC enable field.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption, hang	while loop cases causing issues in jeu fused mask	Disable Structured Control Flow by setting EnableVISAStructurizer.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	HDC: HDCTLB tdl_mode bits incorrectly decoded for hdctlbl3arb	DW-1 Bit-13 and Bit-12 of State Compute Mode register (bitfield names: Coherent access L1 Cache Disable, Disable L1 Invalidate for non-L1-cacheable Writes) must be set to 0 by driver. Coherent access L1 cacheability can be still controlled by MOCS value.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
performance	AMFS : Multi Eval perf test is having traffic only on 3 TSL ports instead of all 6 ports during the 2nd half of the run	0x7300[6] should be set to 1	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption, other	Color pipe incorrectly counts unlit pixels in some cases when Coarse Pixel Shading is used with CPS aware color pipe optimization enabled	Disable CPS Aware color pipe by setting register bit: 0x07304 Bit[9]	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
	DARBFunit early clock gating leading to underrun	Disable clock gating for DARBFunit. Set register offset 0x46530 bit 27 (DARBF Gating Dis) to 1 before first enabling display planes or cursors and keep set. No need to clear after disabling planes	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
security	Accumulator is not currently cleared with GRF clear exposing its content to new context.	Clear ACC register before EOT send mov(16) acc0.0:f 0x0:f	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang	3DMARK_NIGHTRAID_DX12 - CS not done on PIPE_CONTROL	WA/Mitigation: · Register bit to disable PC deref enhancement 0xe4f4[8]	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
other	Register reads to 0x6604 is incorrect	SW is required to only write 0x6604 as the read will not return the correct value if doing a read-modify-write. The default value for this register is zero for all fields and there are no bit masks. Updating this register requires SW to know the previous written value to retain previous programming.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang	Multicontext preemption tests hang with sampler, sc & hdc not done	Disable GSYNC.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
performance	LNCF MOCS settings are cleared on soft reset of RCS/POCS/CCS	Upon render reset, the driver needs to reprogram LNCFCMOCS0 to LNCFCMOCS31. Programming note: WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
power	AMFS Evaluate via Compute CS hangs if FFDOP clk gating is enabled	1. if compute shaders do evaluate, SW must program register 0x20ec[1] to 1 2. Shaders must not do Evaluate, in VF (Virtual Function) mode.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	EU: goto instruction with uniform predicate in CS SIMD32 kernel does not work as expected	ITo workaround this, a kernel change is proposed. Since hardware is able to turn off channels at goto but unable to change fuse mask correctly, combine the channel enable register with dispatch mask and use it to predicate NoMask instructions. Kernel with workaround looks like below. To ensure the predicate mask has all channels enabled, we can specify the 'any' modifier with the size of the JEU instruction execution size. (W) mov(1) r107.0:uw	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
		sr0.4:uw //load the dispatch mask into a temp register. (~f0.0) goto (16|M0) ELSE_UNSTRUCT ELSE_UNSTRUCT or (16|M0) r21.0<1>:uw r21.0<1;1,0>:uw 0x8:uw (W) and(1) f0.0:uw ce0.0:uw r107:uw //and the ce mask and dispatch mask loaded into r107. (W&f0.0.any16h) add (16|M0) r23.0<1>:uw r23.0<1;1,0>:uw 0x0001:uw //predicate the NoMask instruction. 'any' modifier with 16h specified because JEU execution size is 16. goto (16|M0) ELSE_UNSTRUCT END_IF_UNSTRUCT ELSE_UNSTRUCT: join (16|M0) END_IF_UNSTRUCT or (16|M0) r21.0<1>:uw r21.0<1;1,0>:uw 0x10:uw (W) and(1) f0.0:uw ce0.0:uw r107:uw //and the ce mask and dispatch mask again, before every NoMask instruction. If channel enables haven't changed, then once before the first NoMask instruction. (W&f0.0.any16h) add (16|M0) r23.0<1>:uw r23.0<1;1,0>:uw 0x0100:uw //predicate the NoMask instruction. END_IF_UNSTRUCT: join (16|M0) POST_END_IF_UNSTRUCT POST_END_IF_UNSTRUCT: This workaround is needed for all NoMask instructions inside branching instruction blocks where EUs can diverge. Exceptions: if the NoMask instruction execution size is greater than the JEU block executions size like below, an additional instruction is required to ensure flag is written for the upper channels to use. The 'any modifier will not be required in this case.' //JEU block execution size of 16 nop //do || (W) and(1) f0.0:uw ce0.0:uw r107:uw (W) and(32) (ne)f0.0 f0.0:uw 0xffff:uw //execution size same as NoMask instruction size. Immediate value as wide as the jeu block execution size. (W&f0.0) add(32) ..... | While(16) //JEU block execution size of 4 nop //do || (W) and(1) f0.0:uw ce0.0:uw r107:uw (W) and(16) (ne)f0.0 f0.0:uw 0xf:uw //execution size same as NoMask instruction execution size which is 16. Immediate value as wide as jeu block execution size which is 4. (W&f0.0) add(16) ..... (f1.0) while(4)	

impact	title	bspec_wa_details	sku_impact
data_corruption	3DMark IceStorm/IceStormExtreme Demo - corruptions	To avoid sporadic corruptions “Set 0x7010[9] when Depth Buffer Surface Format is D16_UNORM , surface type is not NULL & 1X_MSAA”	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang	pipe3D: register bit flexing : TDL is blocking deref when 8th bit of tdl chicken	Issue: When the push constant deref pipelining is disabled, it can result in some performance drop (0.5% to 1% for certain workloads). WA: Disable Push constant Buffer E48C[9]=1	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	A64 scatter messages incorrectly dispatched to same address if Addr[47:32] differ in a msg among simd lanes	IGC W/A is to avoid such A64 scatter messages by adding a loop around each A64 vector load/store so that on each iteration only lanes with identical high 32-bit addresses will execute.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	TRTT Aliased Buffers Data Mismatch - Possible race condition between Mem Wr and HDC Flush	A "HDC fence" message must be inserted before the EoT of a compute, 3D or a pixel shader thread, if there is any HDC memory write requests from the thread. [L3 cache flush from the fence message is NOT needed].	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
other	CONDDBG indx matching is for same register only	Issue: If a data pattern is detected in the thread dispatch data, the conditional debug feature should set breakpoint. The conditional debug feature can catch it on multiple data patterns, but it will only match if those data patterns are on the same data phase (which corresponds to a grf register). WA: CONDDBG indx matching is for same register only.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
other	MI_SET_PREDICATE with # of slices does not work in the condition that the slices are not contiguious	As part of the Dual Context feature the predication evaluation based on the number of slices is deprecated from MI_SET_PREDICATE. This functionality was deprecated previously.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
other	Command Streamer not sending flush to VF and SVG after Fence during PipeControl sequence of commands causing hang	In set shader mode 3DSTATE_CONSTANT_* needs to be programmed before BTP_* At CS RTL boundary, this is the order of commands 1. Constant cycle on MCR 2. Fence command 3. BTP on MCR At SVG RTL boundary, this is the order of commands seen because of MCR delay 1. Fence 2. Constant Cycle on MCR 3. BTP on MCR At fence, although fence is a non pipeline state, CS is optimizing the flush and NOT sending the flush.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang	VFURB dropping data in some scenarios involving 256 bit element format	Component packing of vertex elements associated with 256-bit surface formats is not supported due to a HW bug. WA: All components of vertex elements associated with 256-bit surface formats MUST be enabled.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
other	Input Coverage = INNER is incorrectly ANDing sample masks	Issue: While designing CPS and depth coverage mode for input coverage for conservative rasterization, implementation changed. This was noticed especially as input coverage mode = INNER started ANDing sample mask to conservative rasterization mask. This resulted in a mis-match write to the spec. WA: Have PS	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa

<table>
<tr><th>impact</th><th>title</th><th>bspec_wa_details</th><th>sku_impact</th></tr>
<tr><td></td><td></td><td>compiler logically OR input coverage mask to infer if a pixel is fully covered when INPUT_COVERAGE_MASK_MODE = INNER</td><td></td></tr>
<tr><td>data_corruption</td><td>HDC RTL does not support 16-bit typed atomics</td><td>SW must not generate 16bit Typed atomic messages. Also, there must not be API support for 16bit Typed Atomics</td><td><table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
<tr><td>data_corruption</td><td>Blitter RAW hazard between blits</td><td>"For two sequential fast copy blits when the source of the second blit is the destination of the first blit or they overlap a Flush must be inserted between the two blits (there can be one or more Fast Color blt between those two fast copy blits)."</td><td><table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
<tr><td>hang</td><td>EU instructions: Indirect address access with Acc destination doesn't work correctly on fused EU pair</td><td>WA: Shader compiler should not generate EU instruction that has both indirect addressing and Acc destination. Indirect addressing can be used with non-Acc destinations; Acc destination can be used in cases other than indirect addressing.</td><td><table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
<tr><td>other</td><td>Atomic operation does not work on compressed data</td><td>Driver must make sure there is not atomic operation done on compressed data. For DX API, this means compression will be disabled for any SINT/UINT surfaces. For OCL, compression is allowed on untyped surfaces. But it is the responsibility of the driver to check kernels for any atomic operations, and resolve the surfaces that could be accessed by atomic, before the kernel launch.</td><td><table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
</table>

impact	title	bspec_wa_details	sku_impact
data_corruption	MMIO remapping feature in command streamer MI_* register access functions doesn't work for certain offsets	WA Name: SelectiveMMIORemapEnable "MMIO Remap Enable" can be enabled only for the "Register Offsets" mentioned in the "MMIO remap table" of a given engine on which the MI commands accessing the MMIO registers are getting executed.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang	EU hang can occur if regular send instructions are followed by URB atomic	WaName: WaResolveDepBeforeAtomics When multiple sends to low priority bus (obus) are present before an Atomic chain of sends to high priority bus (sbus), MA switches grants to high priority bus after the first low priority grant and never goes back to grant the remaining low priority requests. WA: If Atomic chain ends with EOT then resolve all SBID dependencies before the Atomic chain of instructions (sync.allrd), else if Atomic chain does not end with EOT then resolve all SBID dependencies present within the Atomic chain before starting the chain.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang	POSH/PTBR workloads can hang if varying tile counts within a tile pass and preemption happens	WA Name: PoshPreemptionTilePassInfoCmd "Tile Count" value programmed must be same in the 3DSTATE_PTBR_TILE_PASS_INFO command programmed for "Start of Tile Pass" and "End of Tile Pass".	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
other	HW default polarity for Sampler Small PL is "disabled" - not optimal for power	Issue: To ensure optimal power in 3D Sampler. WA:Enable bit 15 of E18C.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
hang	OVR Issue where initialize that follows the restart is not deferred causing an invalid page to be allotted for storing the tokens	OVR Issue if pocs_ovr_restart is asserted within 256 clks after the ctx restore is done. WA: The WA could be to do a page pool size mmio write with a value of 0 followed by 256 noops before any page pool restart.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Incorrect blue channel value when sampling from R32G32_FLOAT surface with border texture addressing mode	Issue: When sampling from an R32G32_FLOAT surface with border texture addressing mode, there is an issue where the blue channel value is missing. WA: Set the shader channel select to 1.0 (instead of 0) for the missing blue channel.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
other	3D Tiled-YF surface corruption in MIP tail LODs because of X-adjacent RCC cacheline composition	WaSetMipTailStartLODLargertoSurface LOD RCC cacheline is composed of X-adjacent 64B fragments instead of memory adjacent. This causes a single 128B cacheline to straddle multiple LODs inside the TYF MIPtail for 3D surfaces (beyond a certain slot number) , leading to corruption when CCS is enabled for these LODs and RT is later bound as texture. WA: If RENDER_SURFACE_STATE.Surface Type = 3D and RENDER_SURFACE_STATE.Auxiliary Surface Mode != AUX_NONE and RENDER_SURFACE_STATE.Tiled ResourceMode is TYF or TYS, Set the value of RENDER_SURFACE_STATE.Mip Tail Start LOD to a mip that larger than those present in the surface (i.e. 15)	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
performance	Perf: 3DMark11 Shadowmap : TDS dual dispatch issue	mmio offset 6604h bits 23:16 must be set to 4h	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
hang	HS Hang & TDG mismatches when dual_instance_enable is zero AND HS is handle limited.	Restricting the min number of input handles to 256+128 (?) and output handles to 8 when instancing is enabled	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang	DualContext : During CSB update GAM will not have dualcontext information causing issue	Program GAM 0xCE90 Register's Dual Context Mode bits whenever RCU mode control reg 0x14800 is programmed. (same value of bit0 with mask).	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	3D: AMFS: pipe3d : Same virtual address is being sent/used by amfs for different LODs, resulting in dropping of some LODs resulting in corruption	BitField: Procedural Texture 11: Name: Procedural Texture Description This bit, when set, indicates that the associated surface is a procedural texture which is used for AMFS. This bit can be ENABLED for the following surface types: SURFTYPE_2D arrayed / non-arrayed, SURFTYPE_3D non-arrayed, SURFTYPE_CUBE arrayed/ non arrayed, and surftype = NULL. This bit can be set for the pixel formats that are supported has typed UAVs as per the DX spec. Therefore, writes from only HDC are supported to Procedural Textures. This bit cannot be ENABLED for the following surface types: SURFTYPE_3D arrayed, SURFTYPE_BUFFER Description This bit cannot be ENABLED for SURFTYPE_SCRATCH. ProgrammingNote This bit cannot be set when surface walk (tiling mode) is legacy Y This bit cannot be set when Tiled Resource Mode = TileYS and LOD >= MIP tail LOD	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
hang	Cs - Gam Deadlock after Root Entry Not Present Fault	WA Name: NoResetReadynessHandShake SW must not do Reset Readiness Handshake as part of the reset recovery on an CAT error.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Spec clarification: Z Clear Color Location	There was a hole in the definition for Clear value for the case of D24X8 depth surfaces. Added a programming note in RENDER_SURFACE_STATE as well as in Clear Color section describing the need to write the converted value to the lower 16B.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	[DAPRSS] Color: DaprSsDaprSc.ss_phase0.cpq_mask.sample_mask Mismatch	Disable CPS Aware color pipe by setting register bit. 0x07304 Bit[9]	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	[DAPRSS] DAPRSS Sending Blend CData Encoding For Fill CPQ	Disable CPS Aware color pipe by setting register bit. 0x07304 Bit[9].	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
other	PSDunit is dropping MSB of the blend state pointer from SD FIFO	Limit the Blend State Pointer to < 2G	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	[3D-WHCK] wgf11resourceaccess workload fail	Before fast clearing any resource, SW must partially resolve the resource i.e. corresponding CCS for the resource MUST NOT be in CLEAR state	sku: ALL; stepping_impacted: a0; stepping_fixed: b0; wa_status: driver_temporary_wa
data_corruption	[GT1] [DAPRSS] Data Corruption on R10G10B10_FLOAT_A2_UNORM After Blend2Fill	See the Errata on Pre-Blend Color Clamping	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	[GT1] [DAPRSS] Repcol with R10G10B10_FLOAT_A2_UNORM Not Properly Down-converted	De-feature Repcol Messages	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa

<table>
<tr><th>impact</th><th>title</th><th>bspec_wa_details</th><th>sku_impact</th></tr>
<tr><td></td><td>PCH display clock remains active when it shouldn't; impact to power and sleep state residency</td><td>Display driver should set and clear register offset 0xC2000 bit #7 as last step in programming south display registers in preparation for entering S0ix state, or set 0xC2000 bit #7 on S0ix entry and clear it on S0ix exit.</td><td><table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
<tr><td>data_corruption</td><td>[AMFS] SW Workarounds for AMFS flush</td><td>1.A pipe control flush with "AMFS flush Enable" set and "DC flush enable set" must be sent down the pipe before a context switch, when compute shaders do evaluate. 2. if compute shader does evaluates, and SW needs to flush the AMFS pipe, it has to first send a pipecontrol flush to the compute pipe and then switch to 3D pipe before sending a pipecontrol with "Command Streamer Stall Enable", AMFS flush Enable, and DC flush enable set on it 3. If compute shaders do evaluate, disable premption, until AMFS data is flushed out of all the caches. 4. All shaders that perform evaluates must send a Cache Flush message to the sampler with a non-zero read-length after all evaluates are issued and before End-Of-Thread 5. Compute shaders run on a CCS context must not issue AMFS evaluates. All AMFS evaluates must run in an RCS context</td><td><table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
</table>

impact	title	bspec_wa_details	sku_impact
hang	Handle block deref size is part of 3dstate_sf & is non-privileged register bit	Driver will have to correctly program bits [30:29] on every 3dstate_SF programming (driver would have to reprogram this field with all the rest of the fields disabled prior to 3DPRIMITIVE command.) SF Body: [bits 30:29]	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
other	Read/write access to OAG registers blocked for non-priv batch buffers from RCS/POCS/CCS; required for certain performance instrumentation cases to work	WA: READ/WRITE ACCESS to OAG Registers 1. Software must use the Force_To_Non_Priv registers to enable Read/WRITE access to the below register offsets RCS: 0xD920 - 0xD93F and 0xDA10 - 0xDA27 (2 ranges) POCS: 0xD920 - 0xD93F and 0xDA10 - 0xDA27 (2 ranges) CCS: 0xD920 - 0xD93F and 0xDA10 - 0xDA27 (2 ranges)	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Page Faults: Write access to page marked as read only results in write being dropped, but fault may not be reported	Errata: WR permission faults may not be reported for write access to Read Only pages. SW can choose not to use read only pages OR just live with the fact that write accesses can be silently dropped without permission fault reporting.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang	Hangs can occur if using constant cache invalidate command with RCS+CCS concurrency	If the intention of “constant cache invalidate” is to invalidate the L1 cache (which can cache constants), use “HDC pipeline flush” instead of Constant Cache invalidate command. Some units bypass the L3 cache when they access memory - CS, MediaFF and Guc. When data sharing (e.g. semaphore) between these units and a shader is needed, the L3 cache may need to be	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa

<table>
<tr><th>impact</th><th>title</th><th>bspec_wa_details</th><th>sku_impact</th></tr>
<tr><td></td><td></td><td>invalidated using a pipe-control CS command with a "Const cache Invalidate" set. In these cases, the w/a should be to set the "State $ invalidate" in the pipecontrol command, in addition to the "HDC pipeline flush". Setting "state $ invalidate" will also invalidate the RO section (including constants) of L3 cache. So, WA should be applied to GT1 all steppings.</td><td></td></tr>
<tr><td>other</td><td>Vertex fetch unit can fetch past end of vertex buffer resulting in page faults</td><td>Add one extra page to the vertex buffer when in sequential mode.</td><td><table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
<tr><td>data_corruption</td><td>*CS: sometimes ctx time stamp register doesn't get restored to value from the engine context image on context switch</td><td>The below workaround must be used to overcome the ctx timestamp issue 1. For BCS/VCS/VECS: -- In the Per-Context WABB (workaround batch buffer) Software must program 3 back to back LRM (MI_LOAD_REGISTER_MEM) commands with - For RCS/CCS -- In the Indirect Context Pointer, Software must program 3 back to back LRM (MI_LOAD_REGISTER_MEM) commands with Dw0[19] = 1, Register Address = CTX_TIMESTAMP and Memory Address = LRCA + 108Ch. 2. The first two MI_LOAD_REGISTER_MEM commands must have Dw0 bit 21 = 1 3. The third MI_LOAD_REGISTER_MEM command must have Dw0 bit 21 = 0 4. All three commands must have “Add CS MMIO Start Offset” Dw0[19] = 1 to enable auto addition of CS MMIO Start Offset. For Example in case of RCS, if LRCA for a given context is DEADh the below commands must be programmed in the per-context workaround batch buffer. 1.</td><td><table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
</table>

<table>
<tr><th>impact</th><th>title</th><th>bspec_wa_details</th><th>sku_impact</th></tr>
<tr><td></td><td></td><td>MI_LOAD_REGISTER_MEM (dw0[19] = 1, dw0[21]= 1, REGISTER ADDR = 3a8h, Memory Address = DEADh + 108Ch 2. MI_LOAD_REGISTER_MEM (dw0[19] = 1, dw0[21]= 1, REGISTER ADDR = 3a8h, Memory Address = DEADh + 108Ch 3. MI_LOAD_REGISTER_MEM (dw0[19] = 1, dw0[21]= 0, REGISTER ADDR = 3a8h, Memory Address = DEADh + 108Ch</td><td></td></tr>
<tr><td></td><td>dupunit not generating line_pop indication for plane with minimum size</td><td>plane horizontal minimum size in PLANE_SIZE register need to be increased according to the following: 8bpp: 18 16bpp: 10 32bpp,yuv212,yuv216: 6 64bpp: 4 NV12: 20 P010,P012,P016: 12</td><td><table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
<tr><td>data_corruption, performance</td><td>Sampler power context is not saved/restored</td><td>Issue: Sampler Power Context save operation doesn't work correctly. This means that if driver writes to any of the following offsets (E100, E180, E184, E188, E18C, E190, E194) with non-default values will not be persisted across Render power gating/RC6. There are some cases we already know of where driver is expected/required to write non-default values for correct functional operation and best performance: All E18C[0] E18C[15] In addition to these, more cases may be identified later where driver wants/needs to program these registers with non-default values and needs to have that programming be restored after render/RC6 power gating. Workaround: KMD to configure RC6 WA BB for RCS (CTX_WA_PTR) if not already enabled; allocate buffer to contain the commands and ensure it is pinned in GGTT. In the RC6 WA BB, include LRI command that writes to any offsets which require non-default values. More specifically, if KMD programs any of the 7 offsets identified above during driver boot and/or after engine reset, those same offset/value pairs must also include that offset/value in an LRI command in the RC6 WA BB for RCS. Note that all 7 of these offsets are masked registers (upper 16b mask; lower 16b value)- driver only needs to enable the mask bits</td><td><table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
</table>

<table>
<tr><th>impact</th><th>title</th><th>bspec_wa_details</th><th>sku_impact</th></tr>
<tr><td></td><td></td><td>for the specific bits it wants to program to non-default value (e.g. for the value for E18C could be 0x1001_1001 (e.g. mask bits 31 & 16 set to allow the values in bit 15 & 0 to take).</td><td></td></tr>
<tr><td>other</td><td>Multicontext: rsi message retrieves inst_Base address from RCS for both contexts</td><td>When dual context or dual queue (e.g. async compute) is enabled, SW cannot rely on the RSI message for getting the instruction base address due to this bug. If needed, driver can pass the instruction base address to the kernel as a kernel argument</td><td><table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
<tr><td>data_corruption</td><td>Disable DFR</td><td>SW must disable DFR, (permanent work around); that by setting DFRRATIOEN9550[9] -> 1)</td><td><table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
<tr><td>data_corruption</td><td>DAPRSS Clamping NaN Inconsistently</td><td>Errata: If Pre-Blend Source Only Clamp is enabled and Clamp Range is set to COLORCLAMP_UNORM, hardware will not clamp FLOAT render targets to 0.</td><td><table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
<tr><td>other</td><td>MPEG2 & AVC Encode: As part of encode operation, CONDITIONAL_BATCH_BUFFER_END command fetches Compare data (related to Panic mode/QP) and pushes to hw engine for subsequent frame; sends wrong data if the comparison data was in upper 4 QWORD of cacheline</td><td>VCS WA: To sample image status from read return data during MI_CONDTIONAL_BATCH_BUFFER_END command The conditional Batch buffer End should have address such that it will always have expected data (image status data) in lower half cacheline . The MI_STORE_REGISTER_MEM have the same address condition for register 08b4 and 08b8 (image status data) so that the mem write is for lower half CL. EX. MI_STORE_REGISTER_MEM 0000_08b4 lw_address up_address MI_STORE_REGISTER_MEM 0000_08b8 lw_address+4 up_address MI_CONDTIONAL_BATCH_BUFFER_EN</td><td><table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
</table>

impact	title	bspec_wa_details	sku_impact
		D Data lw_address up_address	
hang	Coarse Pixel Shading: Hang can occur with CPS Aware color pipe optimization enabled: CPQ sequence sent with no state in case where SubspanValid=true but SubspanValid=false	Disable CPS Aware color pipe by setting register bit. 0x07304 Bit[9]	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang,security	3DState programming on RCS while in PIPELINE_SELECT= GPGPU mode can cause system hang due to FFDOP clock gating	Kernel driver should disable FF DOP clk gating via masked write to 20EC[1] = 1.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Corruption may occur with the surface formats B5G5R5X1_UNORM and B5G5R5X1_UNORM_SRGB if Color Blend is enabled	Errata: Corruption may occur with the surface formats B5G5R5X1_UNORM and B5G5R5X1_UNORM_SRGB if Color Blend is enabled.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang	Hull Shader Control and Header Fifo in TRG going out of sync results in hang	Please insert 3D State HS before every 3D primitive that has HS enabled	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	PLANE_CC_VAL not getting updated immediately on async flip	Display async flips will not update the clear color value at the right point. The potential workarounds: WA1: KMD must convert async flip to sync flip upon clear color change. WA2: UMD must do partial resolve upon color clear change before submitting the flip to Display, KMD keeps async as async flip.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang	Remove PM Req with unblock/memup + fill support -- SAGV enhancement not working as expected	SAGV fill timeout. Set 0x46434 bits 24 ,25, 26, and 27 to 1 at display initialization .	sku: ALL; stepping_impacted: b0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
data_corruption	Unexpected ResInfo results with LOD out of bounds.	When doing a resinfo message SW needs to check if any of the LOD values in an aligned 4 channel group is different: channel 0-3, 5-7, 8-11 , etc. If any of them are different It must sequence the resinfo message so that there is at most one unique LOD per valid channel in each 4 pixel group.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang	HW default value for fusedEU timeout for thread dispatch can hang HS / DS	The GS Timer Bits [31:24] in the GangTimer Register [MMIO: 0x6604] should be set to 0xE0 (224 decimal)	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang	With pixel scoreboard disabled, PSS is creating an extra thread with no slotquads loaded when it sees an FC64 8x8 with a different topology have an overlapping X/Y with two already committed partial threads	When SIMD32 is enabled, do not disable pixel scoreboard. In other words, 3DSTATE_PS Bitgroup5[21] = 0 when 3DSTATE_PS Bitgroup5[2] = 1	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang	SVSM: Dual Context - Invalidate hang	SW must ensure pipeline is IDLE prior HW or SW executing a state cache invalidation. There are two possible cases SW or HW may cause this to happen: 1) Scheduler must ensure that CCS and RCS are not running in parallel. CCS could invalidate the state cache while RCS is executing and visa-versa. 2) SW must insert a PIPE_CONTROL with CS stall prior to any PIPE_CONTROL with "State Cache Invalidate Enable" bit. Any PIPE_CONTROL with "State Cache Invalidate Enable" bit set will do an invalidation of the state cache prior to flushing the pipe while sampler is active.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
performance	HDC issues an uncacheable 'clear' color read when compression is enabled, using MOCS#0 instead of MOCS#3	No w/a is needed for functionality. For performance w/a: KMD should set MOCS[0] as "L3 cacheable". Mocs[0] is usually reserved.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	CS power context save/restore doesn't work properly for 0x20E4[2:1]	Driver must program FF_SLICE_CS_CHICKEN2 register 20e4[2:1] - with required preemption granularity along with the corresponding mask bits as part of WABB during every power context restore.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang	Hang can occur on VS UAV write when TE-DOP clk gating is enabled	Set Tessellation DOP Gating Disable via bit [19] in the ThreadMode Register [0x020A0]. eg: 0x020A0[19]=0x1	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
other	PSD is indicating the first payload phase as null for PSD_REG_P_BARY_PLANE phase	Corruption can exist in Fused SIMD16 threads if R68-R71 is the first phase after R1. This scenario might happen if experimenting with "remove BC" kernel. Enable any phase from R3-R67 to prevent the issue.	sku: ALL; stepping_impacted: b0; wa_status: driver_permanent_wa
data_corruption	During Object-Level preemption and an odd number of objects VF does no change the Topology correctly in the Ctx Restore	Multiple WAs are proposed for this issue. Details of them are captured below in "workaround_details section". Due to perf regression of disabling object-level preemption per topo, a blanket disable can be used instead. Disable Object Preemption Set 0x2580[0] = 0 or 0x20ec[0]. It is derived from the condition in RTL - object_preempt_en = 0x20e0[14] ? 0x2580[0] : 0x20ec[0];	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	dx10_sdksamples_sc-default-effect-pools-msaa-2_win-skl_main - triangular corruptions	Set Tessellation DOP Gating Disable via bit [19] in the ThreadMode Register [0x020A0]. eg: 0x020A0[19]=0x1	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
data_corruption	Clock gating issue results in rendering corruption	Set Tessellation DOP Gating Disable via bit [19] in the ThreadMode Register [0x020A0]. eg: 0x020A0[19]=0x1	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
other	disp_reg_addr going to X (PSD_REG_ERR) instead of R67 phase	Corruption can exist in dual-simd8 threads if R66-R71 is the first phase after R1. This scenario might happen if experimenting with "remove BC" kernel. Enable any phase from R3-R65 to prevent the issue.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang	SVG RTL doesn't correctly handle Push Constant buffer with length 0 when buffer address bit 5 is set; Results in render hang	Issue: SVG RTL not zeroing out address bit 5 when the Push Constant buffer length is 0. This is causing additional derefs to be generated. WA: Two options WA1 Program the Push constant buffer address in the Push constant command to be cacheline aligned i.e. make sure bit 5 of the address is set to 0, if any of the 4 push constant buffer length is programmed to be 0 for that constant buffer address. If the above WA is difficult to do, then please do this more generic WA WA2 Program the Push constant buffer address to be always cacheline aligned irrespective of buffer length i.e. make sure bit 5 of the address is set to 0 always in PC command programming.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Corruption in viewmask token coming into CL for POSH enabled workloads when TE DOP is disabled	Disable TEDOP Clock Gating with register bit 20A0 bit 19 set to 1 at boot + Disable POSH for draw calls with PRIM Replication OR PRIM ID enabled	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
data_corruption	Corruption on 3D engine writes to media compressible render target due to incorrect memory cycle type used for read operations when RHWO optimization is enabled	0x7010[14] needs to be set for all media compressed render targets	<table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table>
data_corruption	3DSTATE_CONSTANT_ALL command not processed correctly in certain cases	1. The easiest W/A for the S/W is to use 3DSTATE_CONST command for individual shader instead of 3DSTATE_CONST_ALL COMMNAD. 2. To W/A this issue and to still use 3DSTATE_CONST_ALL command and not lose out on perf, we have to restrict the “Pointer to constant Buffer” filed to always have the address bits [12:8] as zero. Note this is just restricting the start address and CS can still prefetch CL as mentioned in size field. 3. If this address bits (Pointer to constant Buffer[12:8]) needs to be used, then only for those address range we can switch to shader specific push constant commands and rest address can still use 3DSTATE_CONST_ALL.	<table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table>
other	Driver writes to SVL register offsets sometimes don't work correctly due to FFDOP clk gating	Disable FF DOP clk gating when accessing registers in SVL unit (range 0x7000-0x7FFC). This could be done: EITHER on a per access basis - save current 20EC[1] polarity, masked write 20EC[1]=1 to disable, write SVL register, masked write to 20EC[1] to restore original polarity. OR statically disable FFDOP clk gating all the time via 20EC[1]=1 or 9424[2]=0 from driver boot. FFDOP is already being	<table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table>

impact	title	bspec_wa_details	sku_impact
		required to be applied all the time as security workaround for another issue (hard hang if non-priv BB sends 3D STATE command while pipeline_select is in GPGPU mode). As such, the simpler static w/a (option B, specifically 20EC[1] version) is preferred for simplicity/consistency.	
	Underrun can occur in certain cases when FBC is enabled	For non-modulo 4 plane size(including plane size + yoffset), disable FBC when scanline is Vactive -10	sku: ALL; stepping_impacted: c0; wa_status: driver_permanent_wa
hang	CSB data in hw status page may be stale when read out by SW (memory ordering for CS write vs engine interrupt delivery)	SW on processing an CSB interrupt requiring to process more than one CSB entry, SW must introduce a delay of 30us between CSB fetch and processing. OR SW must process on chip CSB present in CS through MMIO reads.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Display underrun can occur on cursor plane if WM0 is used without WM1	Bug in the register unit which results in WM1 register used when only WM0 is enabled on cursor. A similar bug was fixed in the planes in 11p5, but Cursor was missed. Software workaround is when only WM0 enabled on cursor, copy contents of CUR_WM_0[30:0] (exclude the enable bit) into CUR_WM_1[30:0]	sku: ALL; stepping_impacted: b0; wa_status: driver_permanent_wa
hang	PSS flush done does not comprehend PSD state change, it only comprehends all PS threads completed.	WA: Insert a csstall after every 10 draws. Performance impact of this w/a on select DX9 workloads has been found to be negligible.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
other	PCH display HPD IRQ is not detected with default filter value	WA: Program 0xC7204 (PP_CONTROL) bit #0 to '1' to enable workaround and clear to disable it. Driver shall enable this WA when external display is connected and remove WA when display is unplugged or before going into sleep to allow CS entry. Driver shall not enable WA when eDP is connected.	<table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table>
hang	Battlefield 4 + AA causing hang in MTunit	Fast Clear must not be used on 8- nor 16-bit-per-sample, MSAA color surfaces (e.g. B5G6R5, R8G8, R16, R8, etc. MSFMT_MSS surfaces), unless the following is true . . . 1. Surface Format is R8G8_UNORM, R16_UNORM, or R16_FLOAT. 2. Surface Width is a multiple of 8. 3. Surface Height is a multiple of 4. 4. Either . . . A. Surface is Tile64 (which is always the case for MSFMT_MSS on Tile64 platforms). B. Surface is TileYF or TileYS. C. Surface Horizontal Alignment is either HALIGN_8 or HALIGN_16. When implementing SW WA for this bug . . . If a surface meets 1+2+3 but not A/B, please also create the surface as HALIGN_8.	<table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table>

impact	title	bspec_wa_details	sku_impact
hang	Blank screen seen with 4 MST Displays	Issue: Blank screen seen with 4 MST Displays WA: If MST master is being enabled, clear DP VC Payload Bit before start of MST enable sequence and set is as part of regular MST enable sequence. If MST slave is being added to the MST primary transcoder, keep the VC Payload allocate bit of slave stream set throughout the MST slave enable sequence. Clear DP VC Payload Bit only for MST/DP2.0 case before Wait for ACT Sent Status Handshake during Disable Sequence Keep DP VC Payload Bit ON as part of HDMI/DVI Enable/Disable Sequence. When enabling non-MST cases (eDP/DP-SST/HDMI/DVI), MstTransportSelect in TRANS_DDI_FUNC_CTL must be programmed to match the assigned pipe.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
other	Depth stats (occlusion query) gives wrong results when using Render Target Independent Rasterization (STATE_RASTER::ForcedSampleCount != NUMRASTSAMPLES_0) and no pixel shader bound	When 3DSTATE_RASTER::ForcedSampleCount != NUMRASTSAMPLES_0, SW should program a dummy pixel shader in case occlusion query is required.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	3DMark - Firestrike - corruption in OOTB run	WA: Program maximum of 1536 handles for GS.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
other	Display junk and underrun on Pipe A while playing video using MTA with PSR2 enabled.	Issue: Display junk and underrun on Pipe A while playing video using MTA/ while launching edge browser/ opening folders/ interacting with windows icons and taskbar. WA: Set bit 0x46430[23]=0x1 whenever delayed Vblank is used.	<table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>b0</td><td>driver_permanent_wa</td></tr></table>
other	[MPEG2] Panic mode issue	Software must ensure the “Compare Address” programmed in MI_CONDITIONAL_BATCH_BUFFER_END command for the Compare Data Qword in memory is always within the first 256b of a cacheline (i.e address bit[5] must be ‘0’). MI_STORE_REGISTER_MEM have the same address condition for register image status mask (08b4)and image status data(08b8) so that the mem write is always within the first 256b of a cacheline	<table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table>
data_corruption	LRR Cmd addr 2360 not correctly remapped	Software must use only MI_LOAD_REGISTER_IMM to program 0x2360 register	<table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table>
hang	OVR does not send init_abort to POCS when it runs out of free pages	WA: “ Only in the POSH pipeline, add N NOPs after 3DSTATE_PTBR_TILE_PASS_INFO with end of tile bit set (N = 5 * Num of PTBR Tiles programmed)“	<table><tr><th>sku</th><th>stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>b0</td><td>driver_permanent_wa</td></tr></table>

impact	title	bspec_wa_details	sku_impact
data_corruption	Read data corruption due to delayed Writes with State Access --	WA: Driver can enable HDC L1 cacheability for "read-only" buffers only. Setting a "read-write" buffer as L1 cacheable can corrupt memory data. L1 cacheability is set by programming MOCS[6:1] = [48, 59] (in decimal).	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang	Hang due to deadlock created by RHWO scenario with RHWO optimization enabled.	WA: Disable RHWO by setting 0x7010[14] by default except during resolve pass.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
	BW Buddy CTL Register has incorrect default value for TLB Request timeout	Program BW_BUDDY_CTL0 and BW_BUDDY_CTL1 "TLB Request Timer" field to 8h.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	3D Surface Type Height\Width Restricted to 2047 in render_surface_state	WA: Max Height and Width of a 3D Surface Type is 2047.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
hang	Coarse Pixel Shading: DAPRSS incorrectly sending CPQ with No Pixels Lit, can causing hang/incorrect rendering when CPS Aware color pipe optimization enabled	Disable CPS Aware color pipe by setting register bit: 0x07304 Bit[9].	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
other	Display combo PHY DPLL and thunderbolt PLL fractional divider error	Display DPLL and TBT PLL fractional divider value is shifted when reference is 38.4 MHz, giving slightly incorrect frequencies. Workaround when reference is 38.4 MHz, divide by 2 the value programmed into registers DPLL*_CFGCR0 and TBTPLL_CFGCR0 field DCO Fraction. Example, original DCO Fraction value of 0x7000h must be divided to 0x3800h.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
performance	Edp panel will flicker when system idle at desktop with specific background picture	WA: The driver needs to program the FBC_STRIDE (0x43228) and enable the override stride once. The override stride should be programmed with : Compressed buffer seg stride (in CLs) = ceiling[(at least plane width in pixels * 4 * 4) / (64 * compression limit factor)] + 1 If the CFB size computed by: CFB size (in bytes) = Compressed buffer seg stride * Ceiling(MIN(FBC compressed vertical limit/4, plane vertical source size/4)) * 64, will not fit into the memory allocated to FBC, then driver will need to use a more aggressive compression limit factor.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
security	MI_FORCE_WAKEUP and engine reset happen at almost same time, then hang can occur	Prior to doing a reset, SW/FW must ensure command streamer is stopped. Setting both the ring stop and preparser enable bit in the below registers will cause the command streamer to halt. Note preparser is only enabled for RCS and CCS command streamers but bit exists in all CS's. MI_MODE set bit 8. GFX_MODE set bit 10.	sku: ALL; stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Panel Flicker after press F11 or Alt+Tab switch tasks under system.	Corruption seen when FBC is first enabled. After setting the FBC enable, wait for the next start of vblank, then write the plane 1A surface address register.	sku: ALL; stepping_impacted: b0; wa_status: driver_permanent_wa

UP3_UP4_H35 Workarounds

impact	title	bspec_wa_details	sku_impact
other	Command Streamer not sending flush to VF and SVG after Fence during PipeControl sequence of commands causing hang	In set shader mode 3DSTATE_CONSTANT_* needs to be programmed before BTP_* At CS RTL boundary, this is the order of commands 1. Constant cycle on MCR 2. Fence command 3. BTP on MCR At SVG RTL boundary, this is the order of commands seen because of MCR delay 1. Fence 2. Constant Cycle on MCR 3. BTP on MCR At fence, although fence is a non pipeline state, CS is optimizing the flush and NOT sending the flush.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	VFURB dropping data in some scenarios involving 256 bit element format	Issue: Component packing of vertex elements associated with 256-bit surface formats is not supported due to a HW bug. WA: All components of vertex elements associated with 256-bit surface formats MUST be enabled.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
other	Input Coverage = INNER is incorrectly ANDing sample masks	Issue: While designing CPS and depth coverage mode for input coverage for conservative rasterization, implementation changed. This was noticed especially as input coverage mode = INNER started ANDing sample mask to conservative rasterization mask. This resulted in a mis-match write to the spec. WA: Have PS compiler logically OR input coverage mask to infer if a pixel is fully covered when INPUT_COVERAGE_MASK_MODE = INNER	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Blitter RAW hazard between blits	"For two sequential fast copy blits when the source of the second blit is the destination of the first blit or they overlap a Flush must be inserted between the two blits (there can be one or more Fast Color blt between those two fast copy blits)."	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	EU instructions: Indirect address access with Acc destination doesn't work correctly on fused EU pair	WA: Shader compiler should not generate EU instruction that has both indirect addressing and Acc destination. Indirect addressing can be used with non-Acc destinations; Acc destination can be used in cases other than indirect addressing.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
data_corruption	Sel Denorm Failure in Mixed Mode	WaDenormFlushWithRoundUp When half-float denormals are disabled (i.e. flushed to zero) and Rounding mode is set to "Rnd towards +Inf", output denormals gets flushed to zero during float to half-float conversion. WA: Compiler must not generate instructions with the above combination.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	EU hang can occur if regular send instructions are followed by URB atomic	WaName: WaResolveDepBeforeAtomics When multiple sends to low priority bus (obus) are present before an Atomic chain of sends to high priority bus (sbus), MA switches grants to high priority bus after the first low priority grant and never goes back to grant the remaining low priority requests. WA: If Atomic chain ends with EOT then resolve all SBID dependencies before the Atomic chain of instructions (sync.allrd), else if Atomic chain does not end with EOT then resolve all SBID dependencies present within the Atomic chain before starting the chain.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	POSH/PTBR workloads can hang if varying tile counts within a tile pass and preemption happens	WA Name: PoshPreemptionTilePassInfoCmd "Tile Count" value programmed must be same in the 3DSTATE_PTBR_TILE_PASS_INFO command programmed for "Start of Tile Pass" and "End of Tile Pass".	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
	Audio 8K1port - For certain VDSC bpp settings, hblank asserts before hblank_early, leading to a bad audio state	WA details can be found at: Display Engine > North Display Engine Registers > Audio > Audio Programming Sequence under "Audio Hblank Early Sequence"	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
other	HW default polarity for Sampler Small PL is "disabled" - not optimal for power	Issue: To ensure optimal power in 3D Sampler. WA:Enable bit 15 of E18C.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	OVR Issue where initialize that follows the restart is not deferred causing an invalid page to be allotted for storing the tokens	OVR Issue if pocs_ovr_restart is asserted within 256 clks after the ctx restore is done. WA: The WA could be to do a page pool size mmio write with a value of 0 followed by 256 noops before any page pool restart.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
data_corruption	Incorrect blue channel value when sampling from R32G32_FLOAT surface with border texture addressing mode	Issue: When sampling from an R32G32_FLOAT surface with border texture addressing mode, there is an issue where the blue channel value is missing. WA: Set the shader channel select to 1.0 (instead of 0) for the missing blue channel.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption,hang	While loop cases causing issues in jeu fused mask	Issue: One EU executes while loop sequence, other EU breaks out. However, due to NoMask after endif, both EUs end up executing mov and send. JEU Fused Mask not correct in HW. WA: Disable Structured Control Flow by setting EnableVISAStructurizer.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	HDC: HDCTLB tdl_mode bits incorrectly decoded for hdctlbl3arb	DW-1 Bit-13 and Bit-12 of State Compute Mode register (bitfield names: Coherent access L1 Cache Disable, Disable L1 Invalidate for non-L1-cacheable Writes) must be set to 0 by driver. Coherent access L1 cacheability can be still controlled by MOCS value.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	HDC L3 write moves forward for a L1 cacheable write when Sampler is stalling and can result in RAW hazard	DW-1 Bit-13 of State Compute Mode register (field name: Disable L1 Invalidate for non-L1-cacheable Writes) must be set to 0 by driver.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	Media compression issue: Issue during Macroblock processing during error concealment can result in page faults/engine soft hang	Use the first valid reference (or the closest reference if POC is available to detect) from reference list if available to fill all unused reference frame address regardless coding type (I, P or B) to prevent potential page fault. If valid reference is not available from reference list, use decode output surface for dummy reference if MMCD is disabled, otherwise make an intermediate allocation as dummy reference. Correspondent reference index needs to be programmed as frame.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	Semi pipelined flush not backpressuring when stencil buffer state is enabling thread dispatch resulting in hang	Issue: Semi pipelined flush not backpressuring when stencil buffer state is enabling thread dispatch. Workaround: An additional pipe control with post-sync = store dword operation would be required.(w/a is to have an additional pipe control after the stencil state whenever the surface state bits of this state is changing).	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
other	PSS X-prop issue in quad_valid when we see an unlit poly on the back of a chg marker with no SIMD modes enabled by the programmer	It is unknown if detected X-prop issue can generate Si failures. To avoid any possible issues, set at least one simd enable in 3dstate_ps (e.g. 16 pixel dispatch enable). If no pixel shader is valid, clear 3dstate_ps_extra "pixel shader valid"	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	OVR causes a Page fault when running out of free pages in PTBR PAGE POOL	The driver has to map 1 page of dummy resource to address PTBR_PAGE_POOL_BASE_ADDRESS + (0xFFFF * 4KB).	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
	Default BCredits on MBUS insufficient to meet required display bandwidth	Issue: Default BCredits on MBUS insufficient to meet required display bandwidth WA: Display MBUS_DBOX_CTL* registers should be programmed with BCredit value of 12 (e.g. 7003C[12:8] = 0xC). Note that there are multiple instances of this register, one for each display pipe (A, B, C, D).. All instances should be programmed to the same value.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	Coarse Pixel Shading - hang can occur in color pipe if CPS Aware color pipe optimization is enabled	Issue: Hang can occur in color pipe if CPS Aware color pipe optimization is enabled. WA: Register bit for Common Slice Register3 (0x7304) bit 9 can be set to disable CPS Aware Color Pipe.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
other	Coarse Pixel Shading - perf issue with floating point render targets if CPS Aware color pipe optimization is enabled	Issue: In CPS enabled cases, some extra cycle in daprss to daprsc. WA:Disable CPS Aware color pipe by setting register bit. 0x07304 Bit[9]	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Coarse Pixel Shading - corruption can occur with R11G11B10_FLOAT render target if CPS Aware color pipe optimization is enabled	Issue: If CPs within CPQ have different blend enables, the CPQ can be optimally pipelined from DAPRSS to the color pipe in two phases, one for fill and one for blend instead of breaking down the blend CPs into PQs. WA: Disable CPS Aware color pipe by setting register bit. 0x07304 Bit[9]	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Coarse Pixel Shading - data corruption can occur if CPS Aware color pipe performance optimization enabled	Issue: If CPs within CPQ have different blend enables, the CPQ can be optimally pipelined from DAPRSS to the color pipe in two phases, one for fill and one for blend instead of breaking down the blend CPs into PQs. WA: Disable CPS Aware color pipe by setting register bit. 0x07304 Bit[9]	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
other	DPT should send VRR enable indicator to DCPR even while Push mode is enabled.	Package C2 increase when VRR is enabled with push mode. When enabling VRR, before setting TRANS_VRR_CTL VRR Enable, program GT-driver Pcode mailbox with command 0x11 and data low bit 0 = 1 to inform pcode that VRR is enabled. When disabling VRR, after clearing TRANS_VRR_CTL VRR Enable, program GT-driver Pcode mailbox with command 0x11 and data low bit 0 = 0 to inform pcode that VRR is disabled.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Underrun when FBC is compressing with odd plane size and first segment is only 3 lines	FBC causes screen corruption when plane size is odd for vertical and horizontal. Set 0x43224 bit 14 to 1 before enabling FBC. It is okay to leave it set when FBC is disabled.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Coarse Pixel shading Data corruption due to dropping CP Subspan with Alpha2Coverage if CPS aware color pipe optimization is enabled	Disable CPS Aware color pipe by setting register bit. 0x07304 Bit[9]	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	VP9 VDEnc encode: segmentation within 64x64 block picks wrong segment id	Program same stream-in segmentation id for all four 32x32 blocks of SB64.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
other	RCS/POCS/CCS/BCS: Reserved fields in "Instdone" Registers are tied to "0" instead of "1"	Software must ignore the Reserved Fields in the INSTDONE register.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Data Corruption with Coarse Pixel Shading + Dual Source Blend + Dual SIMD8 pixel shader dispatch	CPS cannot be enabled alongside Dual SIMD8 Dispatch and Dual Source Blend	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	Register based invalidations for a given engine don't indicate completion if that engine is in a power domain that is powered down	SW need to always send an OA invalidation following any render /compute or media TLB register based invalidation. The sequence from driver/SW should be: (when issuing any register based invalidation) 1) issue a mmio write to any render/compute/media Inval 2) issue a mmio write to OA Inval.register (0xCEEC) 3) Now poll for respective invalidation completion	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
data_corruption	MI_ATOMIC uses wrong address for atomic operation in RCS.	MI_ATOMIC command when programmed with “Inline Data” field set to “0” must have “Dword Length” field of the command set to “9h” and must have Dword3..10 programmed with data as 0x0.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	PipeControl with Depth Flush enable can result in hang	“PIPE_CONTROL with Depth stall Enable bit must be set with any PIPE_CONTROL with Depth Flush Enable bit set “	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Blend HW incorrectly uses color clamp range in cases where blend=enabled, Preblend source only clamp= disabled	Driver should always program Color Clamp Range Based on Table in Pre-Blend Color Clamping.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Corruption with FBC and plane enable/disable	Corruption with FBC around plane 1A enabling. In the Frame Buffer Compression programming sequence "Display Plane Enabling with FBC" add a wait for vblank between plane enabling step 1 and FBC enabling step 2.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
other	Multicontext: rsi message retrieves inst_Base address from RCS for both contexts	When dual context or dual queue (e.g. async compute) is enabled, SW cannot rely on the RSI message for getting the instruction base address due to this bug. If needed, driver can pass the instruction base address to the kernel as an kernel argument	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	WM dropping transactions on AMFS_TXT_PTR-only state change	AMFS 3-pass failure. If only AMFS State is programmed, it gets dropped. It will not be a problem if another state like WM state is programmed. Workaround: An additional pipe control with post-sync = store dword operation would be required when programming the AMFS_TXT_PTR state.	sku: ALL; Stepping_impacted: b0; wa_status: driver_permanent_wa
data_corruption	AV1 decode corruption on due to non-deterministic state on exit from reset/power gating	For every AV1 batch buffer, do a force reset/flush on the AV1 pipeline prior to running an Inter workload	sku: ALL; Stepping_impacted: b0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
other	3D Tiled-YF surface corruption in MIP tail LODs because of X-adjacent RCC cacheline composition	WaSetMipTailStartLODLargertoSurfaceLOD RCC cacheline is composed of X-adjacent 64B fragments instead of memory adjacent. This causes a single 128B cacheline to straddle multiple LODs inside the TYF MIPtail for 3D surfaces (beyond a certain slot number) , leading to corruption when CCS is enabled for these LODs and RT is later bound as texture. WA: If RENDER_SURFACE_STATE.Surface Type = 3D and RENDER_SURFACE_STATE.Auxiliary Surface Mode != AUX_NONE and RENDER_SURFACE_STATE.Tiled ResourceMode is TYF or TYS, Set the value of RENDER_SURFACE_STATE.Mip Tail Start LOD to a mip that larger than those present in the surface (i.e. 15)	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
performance	Perf: 3DMark11 Shadowmap : TDS dual dispatch issue	mmio offset 6604h bits 23:16 must be set to 4h	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	HS Hang & TDG mismatches when dual_instance_enable is zero AND HS is handle limited.	Restricting the min number of input handles to 256+128 (?) and output handles to 8 when instancing is enabled	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	3D: AMFS: pipe3d : Same virtual address is being sent/used by amfs for different LODs, resulting in dropping of some LODs resulting in corruption	BitField: Procedural Texture 11: Name: Procedural Texture Description This bit, when set, indicates that the associated surface is a procedural texture which is used for AMFS. This bit can be ENABLED for the following surface types: SURFTYPE_2D arrayed / non-arrayed, SURFTYPE_3D non-arrayed, SURFTYPE_CUBE arrayed/ non arrayed, and surftype = NULL. This bit can be set for the pixel formats that are supported has typed UAVs as per the DX spec. Therefore, writes from only HDC are supported to Procedural Textures. This bit cannot be ENABLED for the following surface types: SURFTYPE_3D arrayed, SURFTYPE_BUFFER Description This bit cannot be ENABLED for SURFTYPE_SCRATCH. ProgrammingNote This bit cannot be set when surface walk (tiling mode) is legacy Y This bit cannot be set when Tiled Resource Mode = TileYS and LOD >= MIP tail LOD	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
hang	Emulation: PTBR tests are hanging with WMFE and IZ	S.W Workaround There is a potential software workaround for the issue by doing these 2 steps 1) setting the force thread dispatch enable(bits 20:19) in the 3dstate_WM_body state to be set to Force_OFF (value of 1) along with the first WM_HZ_OP state cycle 2) The second WM_HZ_OP state which is required by programming sequencing to complete the HZ_OP operation can reprogram the 3dstate_WM_body to set to NORMAL(value of 0).	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
	Plane with Souce Window keying enabled on format "P010" not going transparent based on color channel selection	Source keying with source planes in the pixel formats “P010”, “P012”, “P016”, “RGB64 Unit” is not supported;	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Invalid occlusion query results with "Pixel Shader Does not write to RT" bit	When Pixel Shader Kills Pixel is set, SW must perform a dummy render target write from the shader and not set this bit, so that Occlusion Query is correct.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
performance	AMFS : Multi Eval perf test is having traffic only on 3 TSL ports instead of all 6 ports during the 2nd half of the run	Issue: AMFS not sending TS EOT to TDC causing it not properly load balance and utilize idle EUs in the system. Multi Eval perf test is having traffic only on 3 TSL ports instead of all 6 ports during the 2nd half of the run. WA: 0x7300[6] should be set to 1.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
other	Test CoarsePixelShading_817250 failing with FATAL_ERROR at GT due to X-propagation from RCC unit	Disable CPS Aware color pipe by setting register bit. 0x07304 Bit[9]	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa

<table>
<tr><th>impact</th><th>title</th><th>bspec_wa_details</th><th>sku_impact</th></tr>
<tr><td>power</td><td>RCU should ignore(reset) Media Sampler DOP status of engine which is idle</td><td>In Dual Context Mode of operation, a context can get executed on an engine and switch out with Media Sampler DOP Clock Gate Disabled (can be on Render Engine or Compute Engine). In such a scenario the corresponding engine keeps the Media Sampler DOP Clock Gate Disabled until further a context gets submitted resetting the state to Media Sampler DOP Clock Gate Enabled or both the engines go Idle. This will lead to ineffective DOP Clock Gate of Media Sampler. This may happen under following circumstances: • SW didn't submit the workload exercising Media Sampler bracketed between PIPELINE_SELECT with Media Sampler DOP Clock Gate Disable and Enable respectively in a single dispatch. OR • Media Sampler Workload got preempted before PIPELINE_SELECT with Media Sampler DOP Clock Gate Enable is executed. SW may avoid the inefficient Media Sampler DOP Clock Gate Enable by avoiding above mentioned scenarios, i.e • Make workloads accessing Media Sampler non-preemptable and ensure they are bracketed between PIPELINE_SELECT with Media Sampler DOP Clock Gate Disable and Enable respectively. Or • Following a context switch status of Active to Idle for a Media Sampler workload from and engine and while other engine is busy, SW must submit a context (dummy no real workload) to the former to reset the Media Sampler DOP Clock Gate to be Enabled.</td><td><table><tr><th>sku</th><th>Stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
<tr><td>data_corruption</td><td>GRF source swap feature for SIMD16 with Src0 scalar and bundle conflict between Src1/Src2 is causing the GRF read issue.</td><td>WA: Driver must set E4F4[14]=1 to disable early read/Src Swap.</td><td><table><tr><th>sku</th><th>Stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
<tr><td>hang</td><td>RCS is not waking up fixed function clock when specific 3d related bits are programmed in pipecontrol in compute mode</td><td>SW WA to program PIPE_CONTROl with RT Flush and CS Stall prior to PIPE_SELECT to Compute.</td><td><table><tr><th>sku</th><th>Stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
</table>

impact	title	bspec_wa_details	sku_impact
data_corruption	Vs-CL Edge Flag mismatch - revert fix	Disable component packing when edgeflag is enabled.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	*CS does active to idle transition in certain timing cases with pending lite restore and subsequent preempt with other context AND HW preemption delay enabled	Recommendation 1: Scheduler when detects a pending pre-emption and receives Active2Idle should make sure which elements are pending. Scheduler must check the Context ID on ACTIVE to IDLE switch to make sure which element was preempted even if it is not the last element of the prior submission. Recommendation 2: Pre-emption Delay as part of the SW Scheduler instead enabled in HW.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	B0: Multicontext preemption tests hang with sampler, sc & hdc not done	Disable GSYNC.	sku: ALL; Stepping_impacted: b0; wa_status: driver_permanent_wa

<table>
<tr><th>impact</th><th>title</th><th>bspec_wa_details</th><th>sku_impact</th></tr>
<tr>
<td>data_corruption,hang</td>
<td>Certain Non-Pipelined State commands on RCS should work in PipeSelect compute, but don't because of FFDOP clk gating</td>
<td>Listed commands below are the non-pipeline state commands that may get programmed when PIPELINE_SELECT is set to Media/GPGPU in RenderCS. Due to known HW issue when these commands are executed in Media/GPGPU mode of operation, the new state may not get latched by the destination unit and stale value will prevail. In order to WA this issue SW must temporarily change the PIPELINE_SELECT mode to 3D prior to programming of these command and following that shift it back to the original mode of operation to Media/GPGPU. Since all the listed commands are non-pipelined and hence flush caused due to pipeline mode change must not cause performance issues. • STATE_BASE_ADDRESS • STATE_COMPUTE_MODE • 3DSTATE_BINDING_TABLE_POOL_ALLOC Example: Programming with No WA. PIPELINE_SELECT – GPGPU MEDIA_VFE_STATE MEDIA_INTERFACE_DESCRIPTOR_LOAD GPGPU_WALKER 3DSTATE_BINDING_TABLE_POOL_ALLOC MEDIA_VFE_STATE MEDIA_INTERFACE_DESCRIPTOR_LOAD GPGPU_WALKER …….. Programming with WA. PIPELINE_SELECT – GPGPU MEDIA_VFE_STATE MEDIA_INTERFACE_DESCRIPTOR_LOAD GPGPU_WALKER PIPELINE_SELECT – 3D 3DSTATE_BINDING_TABLE_POOL_ALLOC PIPELINE_SELECT – GPGPU MEDIA_VFE_STATE MEDIA_INTERFACE_DESCRIPTOR_LOAD GPGPU_WALKER</td>
<td>
<table>
<tr><th>sku</th><th>Stepping_impacted</th><th>wa_status</th></tr>
<tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr>
</table>
</td>
</tr>
</table>

impact	title	bspec_wa_details	sku_impact
data_corruption	[AMFS] SW Workarounds for AMFS flush	1.A pipe control flush with "AMFS flush Enable" set and "DC flush enable set" must be sent down the pipe before a context switch, when compute shaders do evaluate. 2. if compute shader does evaluates, and SW needs to flush the AMFS pipe, it has to first send a pipecontrol flush to the compute pipe and then switch to 3D pipe before sending a pipecontrol with "Command Streamer Stall Enable", AMFS flush Enable, and DC flush enable set on it 3. If compute shaders do evaluate, disable premption, until AMFS data is flushed out of all the caches. 4. All shaders that perform evaluates must send a Cache Flush message to the sampler with a non-zero read-length after all evaluates are issued and before End-Of-Thread 5. Compute shaders run on a CCS context must not issue AMFS evaluates. All AMFS evaluates must run in an RCS context	sku: ALL; Stepping_impacted: b0; wa_status: driver_permanent_wa
data_corruption	LRR Cmd addr 2360 not correctly remapped	Software must use only MI_LOAD_REGISTER_IMM to program 0x2360 register	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	Hang if using VEBox for GEC+3DLut and concurrent SFC scaling	This usage was not initially planned as there was no pre-si validation done. However, this was enabled on Si directly and the issue was found for a corner case schmoo. The usage did not involve SFC, instead AVS was used. This was changed to SFC. Hence, the workaround would involve reverting back to AVS usage too.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	Handle block deref size is part of 3dstate_sf & is non-privileged register bit	Driver will have to correctly program bits [30:29] on every 3dstate_SF programming (driver would have to reprogram this field with all the rest of the fields disabled prior to 3DPRIMITIVE command.) SF [bits 30:29]	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
performance	LNCF MOCS settings are cleared on soft reset of RCS/POCS/CCS	WAReprogramMOCS: Upon render reset, the driver needs to reprogram LNCFCMOCS0 to LNCFCMOCS31 Programming note: WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
other	Register reads to 0x6604 is incorrect	SW is required to only write 0x6604 as the read will not return the correct value if doing a read-modify-write. The default value for this register is zero for all fields and there are no bit masks. Updating this register requires SW to know the previous written value to retain previous programming.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	[B0] *CS Changes: CS should stop making new DMA req once decided to go to RDOP	Issue: When Semaphore/Wait for event does not get satisfied, power management logic in CS might decide to Initiate Idle clock gating flows. WA: Disable RDOP for all Wait For Events like MI_SEMAPHORE_WAIT MI_WAIT_FOR_EVENT_2 MI_WAIT_FOR_EVENT.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	*CS Runlist fix for Use HW pointer reloading completed context	Ongoing execution of contexts in the hardware for a given engine can be stopped and make engine go idle by writing “Preempt to Idle” to the EXECLIST_CONTROL register. Following the “Preempt to Idle” flow, resume can be issued by writing “Load” with “Use HW Element Pointer” to the EXECLIST_CONTROL register. Due to known Hardware issue, “Load” with “Use HW Element Pointer” is not functional, hence following a “Preempt to Idle” flow, SW must do a fresh submission to the “Execlist Submit Queue” by submitting the required contexts to be submit queue followed by a “Load” to the EXECLIST_CONTROL register.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
other	imagestatus based on upper DW of half CL data	Software must ensure the “Compare Address” programmed in MI_CONDITIONAL_BATCH_BUFFER_END command for the Compare Data Qword in memory is always within the first 256b of a cacheline (i.e address bit[5] must be ‘0’). MI_STORE_REGISTER_MEM have the same address condition for register image status mask (08b4)and image status data(08b8) so that the mem write is always within the first 256b of a cacheline	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
other	Vertex fetch unit can fetch past end of vertex buffer resulting in page faults	WA: Add one extra page to the vertex buffer when in sequential mode.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
performance	Sampler cache can be thrashed in certain cases involving texture arrays resulting in low performance	added a programming note to the Render Surface State BXML saying the Array bit should not be set unless the depth of the arrayed surface is > 1.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
other	Read/write access to OAG registers blocked for non-priv batch buffers from RCS/POCS/CCS; required for certain performance instrumentation cases to work	1. Software must use the Force_To_Non_Priv registers to enable Read/WRITE access to the below register offsets RCS: 0xD920 - 0xD93F and 0xDA10 - 0xDA27 (2 ranges) POCS: 0xD920 - 0xD93F and 0xDA10 - 0xDA27 (2 ranges) CCS: 0xD920 - 0xD93F and 0xDA10 - 0xDA27 (2 ranges)	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
other	MPEG2 & AVC Encode: As part of encode operation, CONDITIONAL_BATCH_BUFFER_END command fetches Compare data (related to Panic mode/QP) and pushes to hw engine for subsequent frame; sends wrong data if the comparison data was in upper 4 QWORD of cacheline	Software must ensure the “Compare Address” programmed in MI_CONDITIONAL_BATCH_BUFFER_END command for the Compare Data Qword in memory is always within the first 256b of a cacheline (i.e address bit[5] must be ‘0’). MI_STORE_REGISTER_MEM have the same address condition for register image status mask (08b4)and image status data(08b8) so that the mem write is always within the first 256b of a cacheline	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	[SVM][B0 Revisit] Cs - Gam Deadlock after Root Entry Not Present Fault	WA Name: NoResetReadynessHandShake SW must not do Reset Readiness Handshake as part of the reset recovery on a CAT error.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
other	Display software needs to configure SSC enable in a new PLL register	DPLL SSC enable is not correctly hooked up to DPLL_CFGCR0 SSC Enable field. WA: Use DPLL_SSC sscen field to enable SSC instead of DPLL_CFGCR0 SSC enable field.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Spec clarification: Z Clear Color Location	There was a hole in the BSPEC definition for Clear value for the case of D24X8 depth surfaces. Added a programming note to BSPEC in RENDER_SURFACE_STATE as well as in Clear Color section describing the need to write the converted value to the lower 16B. Also, this programming note is removed by HSD 397398 which is HW Managed Z Clear.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
	DARBFunit early clock gating leading to underrun	Disable clock gating for DARBFunit. Set register offset 0x46530 bit 27 (DARBF Gating Dis) to 1 before first enabling display planes or cursors and keep set. No need to clear after disabling planes	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
security	[SECURITY] Accumulator is not currently cleared with GRF clear exposing its content to new context.	Clear ACC register before EOT send mov(16) acc0.0:f 0x0:f	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Corruption is seen on the top part of the Edge browser during Netflix AVC/HEVC playback at 4K resolution.	Resolve Compressed buffers prior to submission on Render Pipe for Protected Render scenarios.	sku: ALL; Stepping_impacted: a0; wa_status: driver_temporary_wa
power	AMFS Evaluate via Compute CS hangs if FFDOP clk gating is enabled	1. if compute shaders do evaluate, SW must program register 0x20ec[1] to 1 2. Shaders must not do Evaluate, in VF (Virtual Function) mode.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
data_corruption	EU: goto instruction with uniform predicate in CS SIMD32 kernel does not work as expected	ITo workaround this, a kernel change is proposed. Since hardware is able to turn off channels at goto but unable to change fuse mask correctly, combine the channel enable register with dispatch mask and use it to predicate NoMask instructions. Kernel with workaround looks like below. To ensure the predicate mask has all channels enabled, we can specify the 'any' modifier with the size of the JEU instruction execution size. (W) mov(1) r107.0:uw sr0.4:uw //load the dispatch mask into a temp register. (~f0.0) goto (16|M0) ELSE_UNSTRUCT ELSE_UNSTRUCT or (16|M0) r21.0<1>:uw r21.0<1;1,0>:uw 0x8:uw (W) and(1) f0.0:uw ce0.0:uw r107:uw //and the ce mask and dispatch mask loaded into r107. (W&f0.0.any16h) add (16|M0) r23.0<1>:uw r23.0<1;1,0>:uw 0x0001:uw //predicate the NoMask instruction. 'any' modifier with 16h specified because JEU execution size is 16. goto (16|M0) ELSE_UNSTRUCT END_IF_UNSTRUCT ELSE_UNSTRUCT: join (16|M0) END_IF_UNSTRUCT or (16|M0) r21.0<1>:uw r21.0<1;1,0>:uw 0x10:uw (W) and(1) f0.0:uw ce0.0:uw r107:uw //and the ce mask and dispatch mask again, before every NoMask instruction. If channel enables haven't changed, then once before the first NoMask instruction. (W&f0.0.any16h) add (16|M0) r23.0<1>:uw r23.0<1;1,0>:uw 0x0100:uw //predicate the NoMask instruction. END_IF_UNSTRUCT: join (16|M0) POST_END_IF_UNSTRUCT POST_END_IF_UNSTRUCT: This workaround is needed for all NoMask instructions inside branching instruction blocks where EUs can diverge. Exceptions: if the NoMask instruction execution size is greater than the JEU block executions size like below, an additional instruction is required to ensure flag is written for the upper channels to use. The 'any modifier will not be required in this case.' //JEU block execution size of 16 nop //do || (W) and(1) f0.0:uw ce0.0:uw r107:uw (W) and(32) (ne)f0.0 f0.0:uw 0xffff:uw //execution size same as NoMask instruction size. Immediate value as wide as the jeu block execution size. (W&f0.0) add(32) ..... | While(16) //JEU block execution size of 4 nop //do || (W) and(1) f0.0:uw ce0.0:uw r107:uw (W) and(16) (ne)f0.0 f0.0:uw 0xf:uw //execution size same as NoMask instruction execution size which is 16. Immediate value as wide as jeu block execution size which is 4. (W&f0.0) add(16) ..... (f1.0) while(4)	sku | Stepping_impacted | wa_status ALL | a0 | driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
data_corruption	[DX11][Corruption] 3DMark IceStorm/IceStormExtreme Demo - corruptions	To avoid sporadic corruptions “Set 0x7010[9] when Depth Buffer Surface Format is D16_UNORM , surface type is not NULL & 1X_MSAA”	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	TRTT Aliased Buffers Data Mismatch - Possible race condition between Mem Wr and HDC Flush	A "HDC fence" message must be inserted before the EoT of a compute, 3D or a pixel shader thread, if there is any HDC memory write requests from the thread. [L3 cache flush from the fence message is NOT needed].	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	[DAPRSS] Color: DaprSsDaprSc.ss_phase0.cpq_mask.sample_mask Mismatch	Disable CPS Aware color pipe by setting register bit. 0x07304 Bit[9]	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	[DAPRSS] DAPRSS Sending Blend CData Encoding For Fill CPQ	Disable CPS Aware color pipe by setting register bit: 0x07304 Bit[9]	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
other	PSDunit is dropping MSB of the blend state pointer from SD FIFO	Limit the Blend State Pointer to < 2G	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	[DAPRSS] Data Corruption on R10G10B10_FLOAT_A2_UNORM After Blend2Fill	See the Errata on Pre-Blend Color Clamping	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Unexpected ResInfo results with LOD out of bounds.	When doing a resinfo message SW needs to check if any of the LOD values in an aligned 4 channel group is different: channel 0-3, 5-7, 8-11 , etc. If any of them are different It must sequence the resinfo message so that there is at most one unique LOD per valid channel in each 4 pixel group.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	[DAPRSS] Repcol with R10G10B10_FLOAT_A2_UNORM Not Properly Down-converted	Defeature Repcol Messages	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
	AV1 ALN LR temp flops need to be reloaded at top of new tile	AV1 decoder will put all the tiles programming into single batch buffer (frame based) [instead of 1 tile per batch buffer]	sku: ALL; Stepping_impacted: b0; wa_status: driver_permanent_wa
	PCH display clock remains active when it shouldn't; impact to power and sleep state residency	Display driver should set and clear register offset 0xC2000 bit #7 as last step in programming south display registers in preparation for entering S0ix state, or set 0xC2000 bit #7 on S0ix entry and clear it on S0ix exit.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa

<table>
<tr><th>impact</th><th>title</th><th>bspec_wa_details</th><th>sku_impact</th></tr>
<tr><td>hang</td><td>Hangs can occur if using constant cache invalidate command [original HSD]</td><td>If the intention of “constant cache invalidate” is to invalidate the L1 cache (which can cache constants), use “HDC pipeline flush” instead of Constant Cache invalidate command. Some units bypass the L3 cache when they access memory - CS, MediaFF and Guc. When data sharing (e.g. semaphore) between these units and a shader is needed, the L3 cache may need to be invalidated using a pipe-control CS command with a "Const cache Invalidate" set. In these cases, the w/a should be to set the "State $ invalidate" in the pipecontrol command, in addition to the "HDC pipeline flush". Setting "state $ invalidate" will also invalidate the RO section (including constants) of L3 cache.</td><td><table><tr><th>sku</th><th>Stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
<tr><td></td><td>dupunit not generating line_pop indication for plane with minimum size</td><td>plane horizontal minimum size in PLANE_SIZE register need to be increased according to the following: 8bpp: 18 16bpp: 10 32bpp,yuv212,yuv216: 6 64bpp: 4 NV12: 20 P010,P012,P016: 12</td><td><table><tr><th>sku</th><th>Stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table></td></tr>
</table>

impact	title	bspec_wa_details	sku_impact
data_corruption, performance	Sampler power context is not saved/restored	Issue: Sampler Power Context save operation doesn't work correctly. This means that if driver writes to any of the following offsets (E100, E180, E184, E188, E18C, E190, E194) with non-default values will not be persisted across Render power gating/RC6. There are some cases we already know of where driver is expected/required to write non-default values for correct functional operation and best performance: All E18C[0]: E18C[15] In addition to these, more cases may be identified later where driver wants/needs to program these registers with non-default values and needs to have that programming be restored after render/RC6 power gating. Workaround: KMD to configure RC6 WA BB for RCS (CTX_WA_PTR) if not already enabled; allocate buffer to contain the commands and ensure it is pinned in GGTT. In the RC6 WA BB, include LRI command that writes to any offsets which require non-default values. More specifically, if KMD programs any of the 7 offsets identified above during driver boot and/or after engine reset, those same offset/value pairs must also include that offset/value in an LRI command in the RC6 WA BB for RCS. Note that all 7 of these offsets are masked registers (upper 16b mask; lower 16b value)- driver only needs to enable the mask bits for the specific bits it wants to program to non-default value (e.g. for the value for E18C could be 0x1001_1001 (e.g. mask bits 31 & 16 set to allow the values in bit 15 & 0 to take).	<table><tr><th>sku</th><th>Stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table>
data_corruption	DAPRSS Clamping NaN Inconsistently	Errata: If Pre-Blend Source Only Clamp is enabled and Clamp Range is set to COLORCLAMP_UNORM, hardware will not clamp FLOAT render targets to 0.	<table><tr><th>sku</th><th>Stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table>
hang	Coarse Pixel Shading: Hang can occur with CPS Aware color pipe optimization enabled: CPQ sequence sent with no state in case where SubspanValid=true but SubspanValid=false	Disable CPS Aware color pipe by setting register bit: 0x07304 Bit[9]	<table><tr><th>sku</th><th>Stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table>

impact	title	bspec_wa_details	sku_impact
data_corruption	PLANE_CC_VAL not getting updated immediately on async flip	Display async flips will not update the clear color value at the right point. The potential workarounds: WA1: KMD must convert async flip to sync flip upon clear color change. WA2: UMD must do partial resolve upon color clear change before submitting the flip to Display, KMD keeps async as async flip.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	B0+ Remove PM Req with unblock/memup + fill support -- SAGV enhancement not working as expected	SAGV fill timeout. Set 0x46434 bits 24 ,25, 26, and 27 to 1 at display initialization.	sku: ALL; Stepping_impacted: b0; wa_status: driver_permanent_wa
hang	HWM unit doesn't check for ack response from downstream unit (backpressure) on tile boundaries, results in hang	Real Tile Scale Decoder insert below commands after every HCP_BSD_OBJECT: (Tile boundary) MFX_WAIT (with MFX_Sync_Control_Flag=1) VD_PIPELINE_FLUSH (with HEVC flush + VDcmd flush + HEVC done=1)	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	Hang on VECS reset due to clk gating issue in IECP	WA: Disable IECP clkgating by writing to 0x1C3F10[22]=1 and 0x1D3F10[22]=1	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	During Object-Level preemption and an odd number of objects VF does not change the Topology correctly in the Ctx Restore	Multiple WAs are proposed for this issue. Details of them are captured below in "workaround_details section". Due to perf regression of disabling object-level preemption per topo, a blanket disable can be used instead. Disable Object Preemption Set 0x2580[0] = 0 or 0x20ec[0]. It is derived from the condition in RTL - object_preempt_en = 0x20e0[14] ? 0x2580[0] : 0x20ec[0];	sku: ALL; Stepping_impacted: b0; wa_status: driver_permanent_wa
data_corruption	[B0]dx10_sdksamples_sc-default-effect-pools-msaa-2_win-skl_main - triangular corruptions	Set Tessellation DOP Gating Disable via bit [19] in the ThreadMode Register [0x020A0]. eg: 0x020A0[19]=0x1	sku: ALL; Stepping_impacted: b0; wa_status: driver_permanent_wa
data_corruption	Clock gating issue results in rendering corruption	Set Tessellation DOP Gating Disable via bit [19] in the ThreadMode Register [0x020A0]. eg: 0x020A0[19]=0x1	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
hang	HCP + SFC reset doesn't work correctly	This bug would affect the VDBOX-SFC reset sequence. We need to VE-SFC forced lock to get around this issue. Here are the steps 1. .Check MFX-SFC usage 2. If (MFX-SFC usage is 1) { a. Issue a MFX-SFC forced lock b. Wait for MFX-SFC forced lock ack c. Check the MFX-SFC usage bit d. If (MFX-SFC usage bit is 1) Reset VDBOX and SFC else Reset VDBOX f. Release the force lock MFX-SFC } 3. else (check HCP-SFC usage). 4. if(HCP+SFC usage is 1) 1. Issue a VE-SFC forced lock 2. Wait for SFC forced lock ack 3. Check the VE-SFC usage bit 4. If (VE-SFC usage bit is 1) Reset VDBOX else Reset VDBOX and SFC 5. Release the force lock VE-SFC. else Reset VDBOX	<table><tr><th>sku</th><th>Stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table>
data_corruption	3DSTATE_CONSTANT_ALL command not processed correctly in certain cases	1. The easiest W/A for the S/W is to use 3DSTATE_CONST command for individual shader instead of 3DSTATE_CONST_ALL COMMNAD. 2. To W/A this issue and to still use 3DSTATE_CONST_ALL command and not lose out on perf, we have to restrict the “Pointer to constant Buffer” filed to always have the address bits [12:8] as zero. Note this is just restricting the start address and CS can still prefetch CL as mentioned in size field. 3. If this address bits (Pointer to constant Buffer[12:8]) needs to be used, then only for those address range we can switch to shader specific push constant commands and rest address can still use 3DSTATE_CONST_ALL.	<table><tr><th>sku</th><th>Stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table>
data_corruption	[Cayucos] Checkerboard background on text input and composition rendering across multiple apps.	0x7010[14] needs to be set for all media compressed render targets	<table><tr><th>sku</th><th>Stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table>

impact	title	bspec_wa_details	sku_impact
other	Driver writes to SVL register offsets sometimes don't work correctly due to FFDOP clk gating	Disable FF DOP clk gating when accessing registers in SVL unit (range 0x7000-0x7FFC). This could be done: EITHER on a per access basis - save current 20EC[1] polarity, masked write 20EC[1]=1 to disable, write SVL register, masked write to 20EC[1] to restore original polarity. OR statically disable FFDOP clk gating all the time via 20EC[1]=1 or 9424[2]=0 from driver boot. FFDOP is already being required to be applied all the time as security workaround for another issue (hard hang if non-priv BB sends 3D STATE command while pipeline_select is in GPGPU mode). As such, the simpler static w/a (option B, specifically 20EC[1] version) is preferred for simplicity/consistency.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
	Underrun can occur in certain cases when FBC is enabled	For non-modulo 4 plane size(including plane size + yoffset), disable FBC when scanline is Vactive -10	sku: ALL; Stepping_impacted: c0; wa_status: driver_permanent_wa
data_corruption	Display underrun can occur on cursor plane if WM0 is used without WM1	Bug in the register unit which results in WM1 register used when only WM0 is enabled on cursor. A similar bug was fixed in the planes in 11p5, but Cursor was missed. Software workaround is when only WM0 enabled on cursor, copy contents of CUR_WM_0[30:0] (exclude the enable bit) into CUR_WM_1[30:0]	sku: ALL; Stepping_impacted: b0; wa_status: driver_permanent_wa
other	Depth stats (occlusion query) gives wrong results when using Render Target Independent Rasterization (STATE_RASTER::ForcedSampleCount != NUMRASTSAMPLES_0) and no pixel shader bound	When 3DSTATE_RASTER::ForcedSampleCount != NUMRASTSAMPLES_0, SW should program a dummy pixel shader in case occlusion query is required.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	PSS flush done does not comprehend PSD state change, it only comprehends all PS threads completed.	WA: Insert a csstall after every 10 draws. Performance impact of this w/a on select DX9 workloads has been found to be negligible.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
hang	Battlefield 4 + AA causing hang in MTunit	Fast Clear must not be used on 8- nor 16-bit-per-sample, MSAA color surfaces (e.g. B5G6R5, R8G8, R16, R8, etc. MSFMT_MSS surfaces), unless the following is true . . . 1. Surface Format is R8G8_UNORM, R16_UNORM, or R16_FLOAT. 2. Surface Width is a multiple of 8. 3. Surface Height is a multiple of 4. 4. Either . . . A. Surface is Tile64 (which is always the case for MSFMT_MSS on Tile64 platforms). B. Surface is TileYF or TileYS (which I don't think Windows or Mesa UMD's are currently using, apart from D3D Sparse Resources). C. Surface Horizontal Alignment is either HALIGN_8 or HALIGN_16. When implementing SW WA for this bug . . . If a surface meets 1+2+3 but not A/B, please also create the surface as HALIGN_8.	sku: ALL; Stepping_impacted: b0; wa_status: driver_permanent_wa
other	PCH display HPD IRQ is not detected with default filter value	WA: Program 0xC7204 (PP_CONTROL) bit #0 to '1' to enable workaround and clear to disable it. Driver shall enable this WA when external display is connected and remove WA when display is unplugged or before going into sleep to allow CS entry. Driver shall not enable WA when eDP is connected.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	[MF][20h1] Blank screen seen with 4 MST Displays	Issue: Blank screen seen with 4 MST Displays WA: If MST primary is being enabled, clear DP VC Payload Bit before start of MST enable sequence and set is as part of regular MST enable sequence. If MST secondary is being added to the MST primary transcoder, keep the VC Payload allocate bit of secondary stream set throughout the MST secondary enable sequence. Clear DP VC Payload Bit only for MST/DP2.0 case before Wait for ACT Sent Status Handshake during Disable Sequence Keep DP VC Payload Bit ON as part of HDMI/DVI Enable/Disable Sequence. When enabling non-MST cases (eDP/DP-SST/HDMI/DVI), MstTransportSelect in TRANS_DDI_FUNC_CTL must be programmed to match the assigned pipe.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	TE DOP disable with idle flush enabled causes CS/CL/SVG hang	Disable IDLE flush during boot with the below sequence. Disable idlemsg via 2050[0], poll for CS FSM csbase+2AC[3:0] = 0 to show idle program TE DOP reenable idlemsg via 2050[0]	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
other	[21H2][Surface Lucca]:Display junk and underrun on Pipe A while playing video using MTA with PSR2 enabled.	Issue: Display junk and underrun on Pipe A while playing video using MTA/ while launching edge browser/ opening folders/ interacting with windows icons and taskbar. WA: Set bit 0x46430[23]=0x1 whenever delayed Vblank is used.	<table><tr><th>sku</th><th>Stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>b0</td><td>driver_permanent_wa</td></tr></table>
hang	Semaphore_signal with post sync enable does not send the correct signal data to GUC	Due to known HW issue, SW must not set “Post-Sync Operation” field for MI_SEMAPHORE_SIGNAL command	<table><tr><th>sku</th><th>Stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr></table>
data_corruption	[RCS/CCS] Sometimes ctx time stamp register doesn't get restored to value from the engine context image on context switch	The below workaround must be used to overcome the ctx timestamp issue 1. For BCS/VCS/VECS: -- In the Per-Context WABB (workaround batch buffer) Software must program 3 back-to-back LRM (MI_LOAD_REGISTER_MEM) commands with - For RCS/CCS -- In the Indirect Context Pointer, Software must program 3 back to back LRM (MI_LOAD_REGISTER_MEM) commands with Dw0[19] = 1, Register Address = CTX_TIMESTAMP and Memory Address = LRCA + 108Ch. 2. The first two MI_LOAD_REGISTER_MEM commands must have Dw0 bit 21 = 1 3. The third MI_LOAD_REGISTER_MEM command must have Dw0 bit 21 = 0 4. All three commands must have “Add CS MMIO Start Offset” Dw0[19] = 1 to enable auto addition of CS MMIO Start Offset. For Example, in case of RCS, if LRCA for a given context is DEADh the below commands must be programmed in the per-context workaround batch buffer. 1. MI_LOAD_REGISTER_MEM (dw0[19] = 1, dw0[21]= 1, REGISTER ADDR = 3a8h, Memory Address = DEADh + 108Ch 2. MI_LOAD_REGISTER_MEM (dw0[19] = 1, dw0[21]= 1, REGISTER ADDR = 3a8h, Memory Address = DEADh + 108Ch 3. MI_LOAD_REGISTER_MEM (dw0[19] = 1, dw0[21]= 0, REGISTER ADDR = 3a8h, Memory Address = DEADh + 108Ch	<table><tr><th>sku</th><th>Stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>b0</td><td>driver_permanent_wa</td></tr></table>

impact	title	bspec_wa_details	sku_impact
data_corruption	Media Compression : counter overflow leads to premature flush done reporting - can result in corruption due to dirty cachelines not getting evicted when high read latency occurs	Issue: Media compression block can have a counter overflow issue in certain long memory latency scenarios that leads to premature flush and some dirty cachelines don't get evicted. Workaround: At the end of VDBox/VEBox batch buffers which involve access to media compressed buffers, SW must insert an extra MI_FLUSH_DW command and specify an address that is different from the compressed allocation (can be compressed or uncompressed).	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang,security	3DState programming on RCS while in PIPELINE_SELECT= GPGPU mode can cause system hang due to FFDOP clock gating.	Kernel driver should disable FF DOP clk gating via masked write to 20EC[1] = 1.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
performance	HDC issues an uncacheable 'clear' color read when compression is enabled, using MOCS#0 instead of MOCS#3	No w/a is needed for functionality. For performance w/a: KMD should set MOCS[0] as "L3 cacheable". Mocs[0] is usually reserved.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	HW default value for fusedEU timeout for thread dispatch can hang HS / DS	The GS Timer Bits [31:24] in the GangTimer Register [MMIO: 0x6604] should be set to 0xE0 (224 decimal)	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Incorrect decoding of DW67 in MFX_PIPE_BUF_ADDR state.	Slice size streamout buffer address should be programmed as zero and disable Slice size streamout feature (Slice Stats Streamout Enable in MFX_AVC_IMG state to zero) till the HW bug is fixed. SW has two methods of generating slice size for the frame. Method 1: At the end of each slice, read MFC Bitstream Byte Count register and store in SLICE_SIZE_BUFFER, increment SLICE_SIZE_BUFFER address by 4 bytes. Method 2: SW can parse the bitstream and determine each individual slice size.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	CS power context save/restore doesn't work properly for 0x20E4[2:1]	Driver must program register 20e4[2:1] - with required preemption granularity along with the corresponding mask bits as part of WABB during every power context restore.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
hang	SVG RTL doesn't correctly handle Push Constant buffer with length 0 when buffer address bit 5 is set; Results in render hang	Issue: SVG RTL not zeroing out address bit 5 when the Push Constant buffer length is 0. This is causing additional derefs to be generated. WA: Two options WA1 Program the Push constant buffer address in the Push constant command to be cacheline aligned i.e. make sure bit 5 of the address is set to 0, if any of the 4 push constant buffer length is programmed to be 0 for that constant buffer address. If the above WA is difficult to do, then please do this more generic WA WA2 Program the Push constant buffer address to be always cacheline aligned irrespective of buffer length i.e. make sure bit 5 of the address is set to 0 always in PC command programming.	sku: ALL; Stepping_impacted: b0; wa_status: driver_permanent_wa
data_corruption	Corruption in viewmask token coming into CL for POSH enabled workloads when TE DOP is disabled	Disable TEDOP Clock Gating with register bit 20A0 bit 19 set to 1 at boot + Disable POSH for draw calls with PRIM Replication OR PRIM ID enabled	sku: ALL; Stepping_impacted: b0; wa_status: driver_permanent_wa

<table>
<tr><th>impact</th><th>title</th><th>bspec_wa_details</th><th>sku_impact</th></tr>
<tr><td>data_corruption</td><td>[BCS/VCS/VECS/POCS] Sometimes ctx time stamp register doesn't get restored to value from the engine context image on context switch</td><td>The below workaround must be used to overcome the ctx timestamp issue 1. For BCS/VCS/VECS: -- In the Per-Context WABB (workaround batch buffer) Software must program 3 back-to-back LRM (MI_LOAD_REGISTER_MEM) commands with - For RCS/CCS -- In the Indirect Context Pointer, Software must program 3 back to back LRM (MI_LOAD_REGISTER_MEM) commands with Dw0[19] = 1, Register Address = CTX_TIMESTAMP and Memory Address = LRCA + 108Ch. 2. The first two MI_LOAD_REGISTER_MEM commands must have Dw0 bit 21 = 1 3. The third MI_LOAD_REGISTER_MEM command must have Dw0 bit 21 = 0 4. All three commands must have “Add CS MMIO Start Offset” Dw0[19] = 1 to enable auto addition of CS MMIO Start Offset. For Example in case of RCS, if LRCA for a given context is DEADh the below commands must be programmed in the per-context workaround batch buffer. 1. MI_LOAD_REGISTER_MEM (dw0[19] = 1, dw0[21]= 1, REGISTER ADDR = 3a8h, Memory Address = DEADh + 108Ch 2. MI_LOAD_REGISTER_MEM (dw0[19] = 1, dw0[21]= 1, REGISTER ADDR = 3a8h, Memory Address = DEADh + 108Ch 3. MI_LOAD_REGISTER_MEM (dw0[19] = 1, dw0[21]= 0, REGISTER ADDR = 3a8h, Memory Address = DEADh + 108Ch</td><td>
<table>
<tr><th>sku</th><th>Stepping_impacted</th><th>wa_status</th></tr>
<tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr>
</table>
</td></tr>
<tr><td>data_corruption</td><td>[CS RTL] Timestamp Reporting in RCS compute mode</td><td>WA: - In compute Mode of operation for RCS, Driver must not program PIPE_CONTROL with Post_Sync “Write Timestamp” and Protected Memory Enable/Disable bits set in the same command. - Driver must instead use a separate Pipe_Control command to perform Post-Sync with “Write Timestamp” And another pipe_control command for protection memory enable/disable bits.</td><td>
<table>
<tr><th>sku</th><th>Stepping impacted</th><th>wa_status</th></tr>
<tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr>
</table>
</td></tr>
<tr><td>data_corruption</td><td>Read data corruption due to delayed Writes with State Access -- Sporadic failures in dEQP-VK.subgroups.basic.graphics.subgroupbarrier test</td><td>WA: Driver can enable HDC L1 cacheability for "read-only" buffers only. Setting a "read-write" buffer as L1 cacheable can corrupt memory data. L1 cacheability is set by programming MOCS[6:1] = [48, 59] (in decimal).</td><td>
<table>
<tr><th>sku</th><th>Stepping_impacted</th><th>wa_status</th></tr>
<tr><td>ALL</td><td>a0</td><td>driver_permanent_wa</td></tr>
</table>
</td></tr>
</table>

impact	title	bspec_wa_details	sku_impact
hang	Hang due to deadlock created by RHWO scenario with RHWO optimization enabled.	WA: Disable RHWO by setting 0x7010[14] by default except during resolve pass.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Diagonal error propagation for vertical intra refresh on H264 VDEnc	The solution is to disable all prediction modes that uses reference values from not refreshed area. Those are modes 3,7 for 4x4 and modes 0, 2, 3, 4, 5, 7 for 8x8 (due to filtering). In the driver code it looks like: AvcIntra4X4ModeMask = 0x88 AvcIntra8X8ModeMask = 0xBD	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
	BW Buddy CTL Register has incorrect default value for TLB Request timeout	Program BW_BUDDY_CTL0 and BW_BUDDY_CTL1 "TLB Request Timer" field to 8h.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Page Faults: Write access to page marked as read only results in write being dropped, but fault may not be reported	Errata: WR permission faults may not be reported for write access to Read Only pages. SW can choose not to use read only pages OR just live with the fact that write accesses can be silently dropped without permission fault reporting.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	Coarse Pixel Shading: DAPRSS incorrectly sending CPQ with No Pixels Lit, can causing hang/incorrect rendering when CPS Aware color pipe optimization enabled	Disable CPS Aware color pipe by setting register bit. 0x07304 Bit[9]	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
other	PSD RTL bug caught through UVM: disp_reg_addr going to X (PSD_REG_ERR) instead of R67 phase	Corruption can exist in dual-simd8 threads if if R66-R71 is the first phase after R1. This scenario might happen if experimenting with "remove BC" kernel. Enable any phase from R3-R65 to prevent the issue.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Corruption may occur with the surface formats B5G5R5X1_UNORM and B5G5R5X1_UNORM_SRGB if Color Blend is enabled	Errata: Corruption may occur with the surface formats B5G5R5X1_UNORM and B5G5R5X1_UNORM_SRGB if Color Blend is enabled.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	Hull Shader Control and Header Fifo in TRG going out of sync results in hang	Please insert 3D State HS before every 3D primitive that has HS enabled	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
other	Display combo PHY DPLL and thunderbolt PLL fractional divider error	Display DPLL and TBT PLL fractional divider value is shifted when reference is 38.4 MHz, giving slightly incorrect frequencies. Workaround when reference is 38.4 MHz, divide by 2 the value programmed into registers DPLL*_CFGCR0 and TBTPLL_CFGCR0 field DCO Fraction. Example, original DCO Fraction value of 0x7000h must be divided to 0x3800h.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	With pixel scoreboard disabled, PSS is creating an extra thread with no slotquads loaded when it sees an FC64 8x8 with a different topology have an overlapping X/Y with two already committed partial threads	When SIMD32 is enabled, do not disable pixel scoreboard. In other words, 3DSTATE_PS Bitgroup5[21] = 0 when 3DSTATE_PS Bitgroup5[2] = 1	sku: ALL; Stepping_impacted: b0; wa_status: driver_permanent_wa
hang	Hang can occur on VS UAV write when TE-DOP clk gating is enabled	Set Tessellation DOP Gating Disable via bit [19] in the ThreadMode Register [0x020A0]. eg: 0x020A0[19]=0x1	sku: ALL; Stepping_impacted: b0; wa_status: driver_permanent_wa
other	PSD is indicating the first payload phase as null for PSD_REG_P_BARY_PLANE phase	Ensure that for a PS thread dispatch (3DSTATE_PS_EXTRA[31]), when any one of the bits of 3DSTATE_PS_EXTRA[21:19] is set (requesting Z, W, BARY P/NP planes), we must have at least one of the following bits set - 3DSTATE_WM_BODY BitGroup0 [16:11] (Bary Interpolation Modes) 3DSTATE_PS_EXTRA[24] : Pixel Shader uses source Z 3DSTATE_PS_EXTRA[23]: Pixel Shader uses source W 3DSTATE_PS_EXTRA[18]: Pixel Shader uses Subsample Offsets 3DSTATE_PS_EXTRA[1:0]: Pixel Shader uses Coverage Mask 3DSTATE_PS BitGroup5 [4]: Position X/Y Offset	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	Media compression: Decode output writes sometimes sends data as uncompressed but doesn't properly update tile compression status to match, resulting in corruption when data is consumed later	Compression Control Surface should be cleared for destination buffers at the start of the batch buffer for MFX codecs.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
hang	CSB data in hw status page may be stale when read out by SW (memory ordering for CS write vs engine interrupt delivery)	Driver initializes CSB data[0...11] with -1 during GPU initialization; - When driver receives the interrupt, it will try to read out the value of every new CSB data, if the value is -1, driver will reread it continuously in 50us until the value is not equal to -1; - After getting the value of every new CSB data, driver will write -1 back into current CSB data offset position; - If the valid value could not be read out in 50us, one warning will be given;	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
performance	dp panel will flicker when system idle at desktop with specific background picture	WA: The driver needs to program the FBC_STRIDE (0x43228) and enable the override stride once. The override stride should be programmed with : Compressed buffer seg stride (in CLs) = ceiling[(at least plane width in pixels * 4 * 4) / (64 * compression limit factor)] + 1 If the CFB size computed by: CFB size (in bytes) = Compressed buffer seg stride * Ceiling(MIN(FBC compressed vertical limit/4, plane vertical source size/4)) * 64, will not fit into the memory allocated to FBC, then driver will need to use a more aggressive compression limit factor.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
security	MI_FORCE_WAKEUP and engine reset happen at almost same time, then hang can occur	Prior to doing a reset, SW/FW must ensure command streamer is stopped. Setting both the ring stop and preparser enable bit in the below registers will cause the command streamer to halt. Note preparser is only enabled for RCS and CCS command streamers but bit exists in all CS's. MI_MODE set bit 8 GFX_MODE set bit 10	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
data_corruption	3DMark - Firestrike - corruption in OOTB run	WA: Program maximum of 1536 handles for GS.	sku: ALL; Stepping_impacted: a0; wa_status: driver_permanent_wa
hang	Panel Flicker after press F11 or Alt+Tab switch tasks under system.	Corruption seen when FBC is first enabled. After setting the FBC enable, wait for the next start of vblank, then write the plane 1A surface address register.	sku: ALL; Stepping_impacted: b0; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
data_corruption	Page Fault when small number of pixels using Sampler Feedback are rendered	Any kernel that contains AMFS evaluate (WriteSamplerFeedback) operations must issue two sampler cache flush messages after all evaluate operations are sent and before the kernel EOT message. The first sampler cache flush message must have a zero-length return. This is used to signal EOT to the AMFS unit. The second sampler cache flush message must have a non-zero-length return. This is used to block the kernel EOT until all AMFS operations are flushed out of the sampler. Failure to do both sampler cache flush messages can result in HW hangs and/or spurious page faults.	<table><tr><th>sku</th><th>Stepping_impacted</th><th>wa_status</th></tr><tr><td>ALL</td><td>b0</td><td>driver_permanent_wa</td></tr></table>

Intel® Iris® Xe and UHD Graphics Open Source

Programmer's Reference Manual

For the 2020-2021 11th Generation Intel Xeon®, Core™, Celeron®, Pentium® Gold Processors based on the "Tiger Lake" Platform

Volume 2a: Command Reference: Instructions

October 2023, Revision 2.0

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Revision History

Revision	Description
1.0	Initial Release
2.0	The following sections were revised: • Updated PIPE_CONTROL instructions

intel

Table of Contents

3DPRIMITIVE 1
3DSTATE_3D_MODE 7
3DSTATE_AA_LINE_PARAMETERS 10
3DSTATE_AMFS 12
3DSTATE_BINDING_TABLE_POINTERS_DS 13
3DSTATE_BINDING_TABLE_POINTERS_GS 14
3DSTATE_BINDING_TABLE_POINTERS_HS 15
3DSTATE_BINDING_TABLE_POINTERS_PS 16
3DSTATE_BINDING_TABLE_POINTERS_VS 17
3DSTATE_BINDING_TABLE_POOL_ALLOC 18
3DSTATE_BLEND_STATE_POINTERS 20
3DSTATE_CC_STATE_POINTERS 21
3DSTATE_CHROMA_KEY 22
3DSTATE_CLEAR_PARAMS 24
3DSTATE_CLIP 25
3DSTATE_CONSTANT_ALL 26
3DSTATE_CONSTANT_DS 29
3DSTATE_CONSTANT_GS 31
3DSTATE_CONSTANT_HS 33
3DSTATE_CONSTANT_PS 35
3DSTATE_CONSTANT_TS_POINTER 37
3DSTATE_CONSTANT_VS 38
3DSTATE_CPS_POINTERS 40
3DSTATE_DEPTH_BOUNDS 41
3DSTATE_DEPTH_BUFFER 42
3DSTATE_DRAWING_RECTANGLE 51
3DSTATE_DS 54
3DSTATE_GS 55
3DSTATE_HIER_DEPTH_BUFFER 56
3DSTATE_HS 57
3DSTATE_INDEX_BUFFER 58
3DSTATE_LINE_STIPPLE 59

3DSTATE_MULTISAMPLE 61
3DSTATE_POLY_STIPPLE_OFFSET 62
3DSTATE_POLY_STIPPLE_PATTERN 64
3DSTATE_PRIMITIVE_REPLICATION 65
3DSTATE_PS_BLEND 66
3DSTATE_PS_EXTRA 67
3DSTATE_PS 68
3DSTATE_PTBR_FREE_LIST_BASE_ADDRESS 69
3DSTATE_PTBR_MARKER 71
3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS 72
3DSTATE_PTBR_RENDER_LIST_BASE_ADDRESS 74
3DSTATE_PTBR_TILE_PASS_INFO 76
3DSTATE_PTBR_TILE_SELECT 80
3DSTATE_PUSH_CONSTANT_ALLOC_DS 81
3DSTATE_PUSH_CONSTANT_ALLOC_GS 83
3DSTATE_PUSH_CONSTANT_ALLOC_HS 85
3DSTATE_PUSH_CONSTANT_ALLOC_PS 87
3DSTATE_PUSH_CONSTANT_ALLOC_VS 89
3DSTATE_RASTER 91
3DSTATE_SAMPLE_MASK 93
3DSTATE_SAMPLE_PATTERN 94
3DSTATE_SAMPLER_STATE_POINTERS_DS 104
3DSTATE_SAMPLER_STATE_POINTERS_GS 105
3DSTATE_SAMPLER_STATE_POINTERS_HS 106
3DSTATE_SAMPLER_STATE_POINTERS_PS 107
3DSTATE_SAMPLER_STATE_POINTERS_VS 108
3DSTATE_SBE 109
3DSTATE_SBE_SWIZ 110
3DSTATE_SCISSOR_STATE_POINTERS 111
3DSTATE_SF 112
3DSTATE_SLICE_TABLE_STATE_POINTERS 114
3DSTATE_SO_BUFFER_INDEX_0 115
3DSTATE_SO_BUFFER_INDEX_1 116
3DSTATE_SO_BUFFER_INDEX_2 117

3DSTATE_SO_BUFFER_INDEX_3 118
3DSTATE_SO_BUFFER 119
3DSTATE_SO_DECL_LIST 122
3DSTATE_STENCIL_BUFFER 125
3DSTATE_STREAMOUT 126
3DSTATE_SUBSLICE_HASH_TABLE 127
3DSTATE_TE 129
3DSTATE_URB_ALLOC_DS 130
3DSTATE_URB_ALLOC_GS 132
3DSTATE_URB_ALLOC_HS 134
3DSTATE_URB_ALLOC_VS 136
3DSTATE_URB_DS 138
3DSTATE_URB_GS 139
3DSTATE_URB_HS 140
3DSTATE_URB_VS 141
3DSTATE_VERTEX_BUFFERS 143
3DSTATE_VERTEX_ELEMENTS 145
3DSTATE_VF_COMPONENT_PACKING 147
3DSTATE_VF_INSTANCING 149
3DSTATE_VF 150
3DSTATE_VF_SGVS_2 153
3DSTATE_VF_SGVS 155
3DSTATE_VF_STATISTICS 157
3DSTATE_VF_TOPOLOGY 158
3DSTATE_VIEWPORT_STATE_POINTERS_CC 159
3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP 160
3DSTATE_VS 161
3DSTATE_WM_CHROMAKEY 162
3DSTATE_WM_DEPTH_STENCIL 163
3DSTATE_WM_HZ_OP 165
3DSTATE_WM 167
A64 Byte Scattered Read MSD 168
A64 Byte Scattered Write MSD 169
A64 Dword Scattered Read MSD 171

A64 Dword Scattered Write MSD 173
A64 Dword Untyped Atomic Float with Return Data Operation MSD 175
A64 Dword Untyped Atomic Float Write Only Operation MSD 177
A64 Dword Untyped Atomic Integer with Return Data Operation MSD 179
A64 Dword Untyped Atomic Integer Write Only Operation MSD 181
A64 Hword Block Read MSD 183
A64 Hword Block Write MSD 185
A64 Oword Aligned Block Read MSD 187
A64 Oword Aligned Block Write MSD 189
A64 Oword Block Read MSD 191
A64 Oword Block Write MSD 193
A64 Qword Scattered Read MSD 195
A64 Qword Scattered Write MSD 197
A64 Qword Untyped Atomic Integer with Return Data Operation MSD 199
A64 Qword Untyped Atomic Integer Write Only Operation MSD 201
A64 Untyped Surface Read MSD 203
A64 Untyped Surface Write MSD 204
A64 Word Untyped Atomic Float with Return Data Operation MSD 205
A64 Word Untyped Atomic Float Write Only Operation MSD 207
A64 Word Untyped Atomic Integer with Return Data Operation MSD 209
A64 Word Untyped Atomic Integer Write Only Operation MSD 211
Addition 213
Addition with Carry 218
Arithmetic Shift Right 223
Average 228
AVP_BSD_OBJECT 233
AVP_IND_OBJ_BASE_ADDR_STATE 235
AVP_INLOOP_FILTER_STATE 237
AVP_INTER_PRED_STATE 251
AVP_PIC_STATE 258
AVP_PIPE_BUF_ADDR_STATE 290
AVP_PIPE_MODE_SELECT 301
AVP_SEGMENT_STATE 305
AVP_SURFACE_STATE 310

AVP_TILE_CODING 316
AVP_VD_CONTROL_STATE 326
Barrier 327
Bit Field Extract 328
Bit Field Insert 1 334
Bit Field Insert 2 339
Bit Field Reverse 344
Branch Converging 349
Branch Diverging 353
Break 357
Byte Scattered Read MSD 361
Byte Scattered Write MSD 363
Call 365
Call Absolute 369
Compare 373
Compare NaN 378
Conditional Select 384
Constant Cache Dword Scattered Read MSD 389
Constant Cache Oword Aligned Block Read MSD 391
Constant Cache Oword Block Read MSD 393
Continue 395
Count Bits Set 399
Dot Product 4 Accumulate 404
Dword Atomic Counter with Return Data Operation MSD 410
Dword Atomic Counter Write Only Operation MSD 412
Dword Scattered Read MSD 414
Dword Scattered Write MSD 416
Dword Typed Atomic Integer with Return Data Operation MSD 418
Dword Typed Atomic Integer Write Only Operation MSD 420
Dword Untyped Atomic Float with Return Data Operation MSD 422
Dword Untyped Atomic Float Write Only Operation MSD 424
Dword Untyped Atomic Integer with Return Data Operation MSD 426
Dword Untyped Atomic Integer Write Only Operation MSD 428
Else 430

End If 434
Extended Math Function 438
Find First Bit from LSB Side 445
Find First Bit from MSB Side 450
Fraction 455
Goto 459
GPGPU_CSR_BASE_ADDRESS 463
GPGPU_WALKER 465
Half Precision HI8DS Render Target Write MSD 469
Half Precision LO8DS Render Target Write MSD 472
Half Precision REP16 Render Target Write MSD 475
Half Precision SIMD8 Render Target Write MSD 478
Half Precision SIMD16 Render Target Write MSD 481
Halt 484
HCP_BSD_OBJECT 488
HCP_FQM_STATE 490
HCP_IND_OBJ_BASE_ADDR_STATE 493
HCP_PAK_INSERT_OBJECT 497
HCP_PAK_OBJECT 502
HCP_PALETTE_INITIALIZER_STATE 505
HCP_PIC_STATE 507
HCP_PIPE_BUF_ADDR_STATE 539
HCP_PIPE_MODE_SELECT 547
HCP_QM_STATE 554
HCP_REF_IDX_STATE 557
HCP_SFC_LOCK 559
HCP_SFC_STATE 561
HCP_SLICE_STATE 562
HCP_SURFACE_STATE 578
HCP_TILE_CODING 583
HCP_TILE_STATE 590
HCP_VP9_PAK_OBJECT 592
HCP_VP9_PIC_STATE 595
HCP_VP9_SEGMENT_STATE 617

HCP_WEIGHTOFFSET_STATE 622
HEVC_SFC_AVS_CHROMA_Coeff_Table 624
HEVC_SFC_AVS_LUMA_Coeff_Table 625
HEVC_SFC_AVS_STATE 626
HEVC_SFC_FRAME_START 627
HEVC_SFC_IEF_STATE 628
HEVC_VP9_RDOQ_STATE 629
HI8DS Render Target Write MSD 631
If 634
Illegal 638
Join 639
Jump Indexed 643
Leading Zero Detection 647
LO8DS Render Target Write MSD 652
Logic And 655
Logic Not 660
Logic Or 665
Logic Xor 670
MEDIA_CURBE_LOAD 675
MEDIA_INTERFACE_DESCRIPTOR_LOAD 677
MEDIA_OBJECT_GRPID 679
MEDIA_OBJECT 682
MEDIA_OBJECT_WALKER 686
MEDIA_STATE_FLUSH 693
MEDIA_VFE_STATE 695
Media Block Read MSD 700
Media Block Write MSD 701
Media Transpose Read MSD 702
Memory Fence MSD 703
MFC_AVC_PAK_OBJECT 705
MFC_JPEG_HUFF_TABLE_STATE 707
MFC_JPEG_SCAN_OBJECT 709
MFC_MPEG2_PAK_OBJECT 712
MFC_MPEG2_SLICEGROUP_STATE 714

MFD_AVC_BSD_OBJECT 722
MFD_AVC_DPB_STATE 724
MFD_AVC_PICID_STATE 727
MFD_AVC_SLICEADDR 729
MFD_IT_OBJECT 732
MFD_JPEG_BSD_OBJECT 736
MFD_MPEG2_BSD_OBJECT 739
MFD_VC1_BSD_OBJECT 741
MFD_VC1_LONG_PIC_STATE 744
MFD_VC1_SHORT_PIC_STATE 757
MFD_VP8_BSD_OBJECT 766
MFX_AVC_DIRECTMODE_STATE 773
MFX_AVC_IMG_STATE 775
MFX_AVC_REF_IDX_STATE 795
MFX_AVC_SLICE_STATE 798
MFX_AVC_WEIGHTOFFSET_STATE 811
MFX_BSP_BUF_BASE_ADDR_STATE 814
MFX_DBK_OBJECT 820
MFX_FQM_STATE 827
MFX_IND_OBJ_BASE_ADDR_STATE 829
MFX_JPEG_HUFF_TABLE_STATE 833
MFX_JPEG_PIC_STATE 835
MFX_MPEG_TS_CONTROL command 842
MFX_MPEG2_PIC_STATE 844
MFX_PAK_INSERT_OBJECT 858
MFX_PIPE_BUF_ADDR_STATE 862
MFX_PIPE_MODE_SELECT 884
MFX_QM_STATE 889
MFX_STATE_POINTER 891
MFX_STITCH_OBJECT 893
MFX_SURFACE_STATE 896
MFX_VC1_DIRECTMODE_STATE 904
MFX_VC1_PRED_PIPE_STATE 906
MFX_VP8_BSP_BUF_BASE_ADDR_STATE 912

MFX_VP8_Encoder_CFG....915
MFX_VP8_PAK_OBJECT....927
MFX_VP8_PIC_STATE....929
MFX_WAIT....954
MI_ARB_CHECK....955
MI_ARB_ON_OFF....957
MI_ATOMIC....959
MI_BATCH_BUFFER_END....965
MI_BATCH_BUFFER_START....966
MI_CONDITIONAL_BATCH_BUFFER_END....973
MI_COPY_MEM_MEM....977
MI_COPY_MEM_MEM....979
MI_COPY_MEM_MEM....981
MI_COPY_MEM_MEM....983
MI_DISPLAY_FLIP....985
MI_FLUSH_DW....990
MI_FLUSH_DW....993
MI_FLUSH_DW....996
MI_FORCE_WAKEUP....999
MI_LOAD_REGISTER_IMM....1002
MI_LOAD_REGISTER_MEM....1006
MI_LOAD_REGISTER_REG....1009
MI_LOAD_SCAN_LINES_EXCL....1013
MI_LOAD_SCAN_LINES_EXCL....1015
MI_LOAD_SCAN_LINES_INCL....1017
MI_LOAD_SCAN_LINES_INCL....1019
MI_MATH....1021
MI_MATH....1022
MI_MATH....1023
MI_MATH....1024
MI_NOOP....1025
MI_NOOP....1026
MI_NOOP....1027
MI_NOOP....1028

MI_PREDICATE 1029
MI_REPORT_HEAD 1031
MI_REPORT_HEAD 1032
MI_REPORT_HEAD 1033
MI_REPORT_HEAD 1034
MI_REPORT_PERF_COUNT 1035
MI_RS_STORE_DATA_IMM 1037
MI_SEMAPHORE_SIGNAL 1039
MI_SEMAPHORE_WAIT 1041
MI_SET_CONTEXT 1046
MI_SET_PREDICATE 1049
MI_STORE_DATA_IMM 1051
MI_STORE_DATA_INDEX 1054
MI_STORE_REGISTER_MEM 1056
MI_SUSPEND_FLUSH 1059
MI_SUSPEND_FLUSH 1060
MI_SUSPEND_FLUSH 1061
MI_SUSPEND_FLUSH 1062
MI_TOPOLOGY_FILTER 1063
MI_UPDATE_GTT 1064
MI_USER_INTERRUPT 1066
MI_USER_INTERRUPT 1067
MI_USER_INTERRUPT 1068
MI_USER_INTERRUPT 1069
MI_WAIT_FOR_EVENT 1070
MI_WAIT_FOR_EVENT_2 1074
Monitor Event 1077
Monitor No Event 1078
Move 1079
Move Indexed 1084
Multiply 1089
Multiply Accumulate 1095
Multiply Accumulate High 1100
Multiply Add 1106

No Operation **1112**
Oword Aligned Block Read MSD **1114**
Oword Block Read MSD **1116**
Oword Block Write MSD **1118**
PIPE_CONTROL **1120**
PIPELINE_SELECT **1131**
Read Surface Info MSD **1134**
REP16 Render Target Write MSD **1135**
Reserved Instruction0 ..... **1138**
Reserved Instruction1 ..... **1139**
Reserved Instruction2 ..... **1140**
Reserved Instruction3 ..... **1141**
Reserved Instruction4 ..... **1142**
Reserved Instruction5 ..... **1143**
Reserved Instruction6 ..... **1144**
Reserved Instruction7 ..... **1145**
Reserved Instruction8 ..... **1146**
Reserved Instruction9 ..... **1147**
Reserved Instruction10 ..... **1148**
Reserved Instruction11 ..... **1149**
Reserved Instruction12 ..... **1150**
Reserved Instruction13 ..... **1151**
Reserved Instruction14 ..... **1152**
Reserved Instruction15 ..... **1153**
Reserved Instruction16 ..... **1154**
Reserved Instruction17 ..... **1155**
Reserved Instruction18 ..... **1156**
Reserved Instruction19 ..... **1157**
Reserved Instruction20 ..... **1158**
Reserved Instruction21 ..... **1159**
Reserved Instruction22 ..... **1160**
Reserved Instruction23 ..... **1161**
Reserved Instruction24 ..... **1162**
Reserved Instruction25 ..... **1163**

Reserved Instruction26 ..... **1164**
Return **1165**
Rotate Left **1169**
Rotate Right **1174**
Round Down **1179**
Round to Nearest or Even **1184**
Round to Zero **1189**
Round Up **1194**
Sampler Cache Media Block Read MSD **1199**
Sampler Cache Oword Aligned Block Read MSD **1201**
Scratch Block Read MSD **1203**
Scratch Block Write MSD **1205**
Select **1207**
Send Message **1213**
Send Message Conditional **1221**
SFC_AVS_CHROMA_Coeff_Table **1227**
SFC_AVS_LUMA_Coeff_Table **1228**
SFC_AVS_STATE **1229**
SFC_FRAME_START **1230**
SFC_IEF_STATE **1231**
SFC_LOCK **1232**
SFC_STATE **1234**
Shift Left **1235**
Shift Right **1240**
Signal Event **1245**
SIMD8 Render Target Read MSD **1246**
SIMD8 Render Target Write MSD **1248**
SIMD16 Render Target Read MSD **1251**
SIMD16 Render Target Write MSD **1253**
STATE_BASE_ADDRESS **1256**
STATE_COMPUTE_MODE **1266**
STATE_SIP **1269**
Subtraction with Borrow **1271**
Synchronize **1276**

Typed Surface Read MSD **1281**
Typed Surface Write MSD **1283**
Untyped Surface Read MSD **1285**
Untyped Surface Write MSD **1287**
URB Dword Read MSD **1289**
URB Dword Write MSD **1291**
URB Masked Dword Write MSD **1293**
VD_CONTROL_STATE **1295**
VD_PIPELINE_FLUSH **1296**
VDENC_CONTROL_STATE **1298**
VDENC_DS_REF_SURFACE_STATE **1299**
VDENC_PIPE_BUF_ADDR_STATE **1300**
VDENC_PIPE_MODE_SELECT **1304**
VDENC_REF_SURFACE_STATE **1307**
VDENC_SRC_SURFACE_STATE **1308**
VDENC_WALKER_STATE **1310**
VEBOX_STATE **1318**
VEBOX_SURFACE_STATE **1328**
VEBOX_TILING_CONVERT **1337**
Wait for Event **1339**
While **1340**
Word Atomic Counter with Return Data Operation MSD **1344**
Word Atomic Counter Write Only Operation MSD **1346**
Word Typed Atomic Integer with Return Data Operation MSD **1348**
Word Typed Atomic Integer Write Only Operation MSD **1350**
Word Untyped Atomic Float with Return Data Operation MSD **1352**
Word Untyped Atomic Float Write Only Operation MSD **1354**
Word Untyped Atomic Integer with Return Data Operation MSD **1356**
Word Untyped Atomic Integer Write Only Operation MSD **1358**
XY_BLOCK_COPY_BLT **1360**
XY_COLOR_BLT **1365**
XY_FAST_COLOR_BLT **1367**
XY_FAST_COPY_BLT **1371**
XY_FULL_BLT **1374**

XY_FULL_IMMEDIATE_PATTERN_BLT 1377
XY_FULL_MONO_PATTERN_BLT 1380
XY_FULL_MONO_PATTERN_MONO_SRC_BLT 1384
XY_FULL_MONO_SRC_BLT 1387
XY_FULL_MONO_SRC_IMMEDIATE_PATTERN_BLT 1390
XY_MONO_PAT_BLT 1393
XY_MONO_PAT_FIXED_BLT 1396
XY_MONO_SRC_COPY_BLT 1399
XY_MONO_SRC_COPY_IMMEDIATE_BLT 1402
XY_PAT_BLT_IMMEDIATE 1405
XY_PAT_BLT 1408
XY_PAT_CHROMA_BLT_IMMEDIATE 1411
XY_PAT_CHROMA_BLT 1414
XY_PIXEL_BLT 1417
XY_SCANLINES_BLT 1418
XY_SETUP_BLT 1419
XY_SETUP_CLIP_BLT 1422
XY_SETUP_MONO_PATTERN_SL_BLT 1423
XY_SRC_COPY_BLT 1426
XY_SRC_COPY_CHROMA_BLT 1429
XY_TEXT_BLT 1432
XY_TEXT_IMMEDIATE_BLT 1434

3DPRIMITIVE

3DPRIMITIVE

Source: RenderCS

Length Bias: 2

The 3DPRIMITIVE command is used to submit 3D primitives to be processed by the 3D pipeline. Typically the processing results in rendering pixel data into the render targets, but this is not required.
The parameters passed in this command are forwarded to the Vertex Fetch function. The Vertex Fetch function will use this information to generate vertex data structures and store them in the URB. These vertices are then passed down the 3D pipeline.

Programming Notes

If the threads spawned by this command are required to observe memory writes performed by threads spawned from a previous command, software must precede this command with a command that performs a (preferably pipelined) memory flush (e.g., 3D_PIPECONTROL).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 3h 3DPRIMITIVE Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 0h 3DPRIMITIVE Format: OpCode
	15	**Reserved**
	14	**Reserved** Access: RO Format: MBZ
	13	**Reserved** Access: RO Format: MBZ
	12	**POSH Enable** POSH functionality is enabled for this 3DPRIMITIVE command when this bit is set. When executed from the POCS command stream, the 3DPRIMITIVE command is processed and visibility information is recorded. When subsequently executed from the RCS command stream, the previously-recorded visibility information is used to optimize the

processing for this 3DPRIMITIVE command.
Setting this bit will also (temporarily -- for the 3DPRIMITIVE command) inhibit modification of the following statistics counters when the 3DPRIMITIVE command is executed from the RCS command stream, even if these statistics counters are enabled via VF/VS state. This is done to prevent RCS from double-counting statistics that have already been counted by POCS.

- IA_VERTICES_COUNT
- IA_PRIMITIVES_COUNT
- VS_INVOCATION_COUNT

When this bit is clear, this 3DPRIMITIVE command is ignored by the POCS command stream, and when executed in the RCS command stream the command is processed without use of visibility information (as none will have been recorded by the POCS command stream) and with statistics counted as specified by VF/VS state.

Programming Notes

This bit must be clear (POSH disabled) when any one of the following conditions are met:

- A topology type other than: 3DPRIM_TRILIST, 3DPRIM_TRISTRIP, 3DPRIM_TRISTRIP_REVERSE
- UAV Coherency Required and/or Accesses UAV set in 3DSTATE_VS
- Tessellation enabled
- Geometry Shader enabled
- StreamOut enabled

For each 3DPRIMITIVE command that SW submits to the POCS command stream with POSH enabled, SW shall also submit an identical 3DPRIMITIVE command with POSH enabled to the RCS command stream, in the same sequence as submitted to the POCS command stream. In addition to the parameters passed in the 3DPRIMITIVE command fields, the following states shall also match:

- 3DSTATE_VF: States controlling cut index processing must match
- For RANDOM indexing, the Index Buffer states must match

11 **Extended Parameters Present**

Format:	Boolean

If true, three additional DWords containing XP0, XP1 and XP2 parameters are included in this command. Depending on the setting of Indirect Parameter Enable, XP0-2 parameters (sourced either from the inline XP0-2 DWords or indirectly from the 3DPRIM_XP0-2 registers) are passed to the VF stage as possible sources for VF SGV insertion.
If false, XP0-2 DWords are not included in this command and instead XP0-2 values default to zero if enabled as sources in VF SGV insertion.

10 **Indirect Parameter Enable**

Format:	Enable

If set, the values in DW 2-5 are ignored and replaced by the current values of the corresponding 3DPRIM_xxx MMIO registers:

- 3DPRIM_VERTEX_COUNT (instead of DW2: VertexCountPerInstance)
- 3DPRIM_START_VERTEX (instead of DW3: StartVertexLocation)
- 3DPRIM_INSTANCE_COUNT (instead of DW4: InstanceCount)
- 3DPRIM_START_INSTANCE (instead of DW5: StartInstanceLocation)
- 3DPRIM_BASE_VERTEX (instead of DW6: BaseVertexLocation)

Indirect Parameter Enable and End Offset Enable shall not be ENABLED at the same time, or behavior is UNDEFINED.

If set and Extended Parameters Present is true, the current contents of the 3DPRIM_XP0-2 MMIO registers are passed to the VF stage as possible sources for VF SGV insertion and the Extended Parameter 0-2 values included in this command are ignored.

9 **UAV Coherency Required**

Format:	U1

SW will be required to set this bit if there is the possibility of sharing a UAV from a previous 3DPRIMITVE command. If set, this command may cause a flush due to UAV coherency requirements. If none of the shaders have UAV access enabled, then this bit is ignored.

8 **Predicate Enable**

Format:	Enable

If set, this command is executed (or not) depending on the current value of the MI Predicate internal state bit. This command is ignored only if PredicateEnable is set and the Predicate state bit is 0.

7:0 **DWord Length**

Format:	=n

Value	Name	Programming Notes	Exists If
5h	5		[Extended Parameter Enable] == 'false'
8h	8 **[Default]**	When Extended Parameter Enable is true, three Dwords containing Extended Parameter 0-2 shall be included in this command and the DWord Length field set to account for these additional DWords.	[Extended Parameter Enable] == 'true'

1 31:10 **Reserved**

Access:	RO

		Format: MBZ
	9	**End Offset Enable** Format: Enable If set, the Vertex Count Per Instance field is IGNORED, and the 3DPRIM_END_OFFSET register is used to indirectly specify the vertex count by defining the amount of valid data in VB0. The following restrictions apply: • VB0 must be enabled for use • VertexAccessType = SEQUENTIAL • Start Vertex Location = 0 • Start Instance Location = 0 • Base Vertex Location = 0 Vertices are output until EndOffset is reached or exceeded in VB0. If EndOffset is reached or exceeded within the data associated with a vertex, that vertex is considered incomplete and will not be output. Partial objects will be discarded (as is normally done). If clear, End Offset is ignored. Indirect Parameter Enable and End Offset Enable must not be ENABLED at the same time, or behavior is UNDEFINED.
	8	**Vertex Access Type** This field specifies how data held in vertex buffers marked as VERTEXDATA is accessed by Vertex Fetch. Value / Name / Description: 0h / SEQUENTIAL / VERTEXDATA buffers are accessed sequentially Required if End Offset Enable is ENABLED. 1h / RANDOM / VERTEXDATA buffers are accessed randomly via an index obtained from the Index Buffer.
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**Primitive Topology Type** Format: 3D_Prim_Topo_Type This field specifies the topology type of 3D primitive generated by this command. Note that a single primitive topology (list/strip/fan/etc.) can contain a number of basic objects (lines, triangles, etc.). This field is ignored. The topology type is specified via the 3DSTATE_VF_TOPOLOGY command.
2	31:0	**Vertex Count Per Instance** Format: U32

3DPRIMITIVE

		This field specifies how many vertices are to be generated for each instance of the primitive topology. If End Offset Enable is clear: Format = U32 count of vertices Range = [0, 2^32-1] (upper limit probably constrained by VB size)Ignored if End Offset Enable or Indirect Parameter Enable is ENABLED. **Programming Notes** • This per-instance value should specify a valid number of vertices for the primitive topology type. E.g., for 3DPRIM_TRILIST_ADJ, this field should specify a multiple of 6 vertices. However, in cases where too few or too many vertices are provided, the unused vertices will be silently discarded by the pipeline. • A 0 value is this field effectively makes the command a 'no-operation'.
3	31:0	**Start Vertex Location** Format: U32 This field specifies the "starting vertex" for each instance. This allows skipping over part of the vertices in a buffer if, for example, a previous 3DPRIMITIVE command had already drawn the primitives associated with the earlier entries. For SEQUENTIAL access, this field specifies, for each instance, a starting structure index into the vertex buffers For RANDOM access, this field specifies, for each instance, a starting index into the Index Buffer. **Programming Notes** • Access of any data outside of the valid extent of a vertex or index buffer will return the value 0 (i.e., appears as if the data stored at the invalid location was 0). • Must be set to 0 if End Offset Enable is ENABLED. • Ignored if Indirect Parameter Enable is ENABLED
4	31:0	**Instance Count** Format: U32 This field specifies the number of instances by which the primitive topology is to be regenerated. A value of 0 indicates "no instances" (no-op operation). A value of 1 effectively specifies "non-instanced" operation, though vertex buffers will still be used to provide instance data, if so programmed. Ignored if Indirect Parameter Enable is ENABLED. **Value** | **Name** [0, FFFFFFFFh] |
5	31:0	**Start Instance Location** Format: U32 This field specifies the "starting instance" for the command as an initial structure index into Vertex Buffers for vertex elements with InstancingEnable set. Subsequent instances will access sequential instance data structures, as controlled by the Instance Data Step Rate.

		Programming Notes • Access of any data outside of the valid extent of a vertex or index buffer will return the value 0 (i.e., appears as if the data stored at the invalid location was 0). • Must be set to 0 if End Offset Enable is ENABLED. • Ignored if Indirect Parameter Enable is ENABLED.
6	31:0	**Base Vertex Location** Format: S31 This field specifies a signed bias to be added to values read from the index buffer. This allows the same index buffer values to access different vertex data for different commands. This field applies only to RANDOM access mode. This field is ignored for SEQUENTIAL access mode, where there Start Vertex Location can be used to specify different regions in the vertex buffers. **Programming Notes** • Access of any data outside of the valid extent of a vertex or index buffer will return the value 0 (i.e., appears as if the data stored at the invalid location was 0). • Must be set to 0 if End Offset Enable is ENABLED. • Ignored if Indirect Parameter Enable is ENABLED.
7 **Exists if:** [Extended Parameters Present] == TRUE	31:0	**Extended Parameter 0** Format: U32 If Indirect Parameter Enable is not set, this field specifies a U32 XP0 parameter value to be passed to the VF stage as a possible source for SGV insertion. If Indirect Parameter Enable is set, this field is ignored.
8 **Exists if:** [Extended Parameters Present] == TRUE	31:0	**Extended Parameter 1** Format: U32 If Indirect Parameter Enable is not set, this field specifies a U32 XP1 parameter value to be passed to the VF stage as a possible source for SGV insertion. If Indirect Parameter Enable is set, this field is ignored.
9 **Exists if:** [Extended Parameters Present] == TRUE	31:0	**Extended Parameter 2** Format: U32 If Indirect Parameter Enable is not set, this field specifies a U32 XP2 parameter value to be passed to the VF stage as a possible source for SGV insertion. If Indirect Parameter Enable is set, this field is ignored.

3DSTATE_3D_MODE

3DSTATE_3D_MODE

Source: RenderCS

Length Bias: 2

This command is general 3D programming state that can be shared from the top to bottom of the pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h GFXPIPE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 1Eh 3DSTATE_3D_MODE Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 1h Excludes DWord (0,1) Format: =n
1	31:16	**Mask Bits 1** Format: Enable[16] This field is the mask bits for the state bits below. This is a bit wise mask where the bit number-16 is the value of the corresponding bit being masked in the same data word. For example, if you want to update state for bits 3:2 then bits 19:18 must be set.
	15:11	**Reserved** Access: RO Format: MBZ
	10	**Reserved** Access: RO Format: MBZ

3DSTATE_3D_MODE

9 Float Blend Optimization Disable

Value	Name	Description
0h	**[Default]**	Enables blend optimization for floating point RTs.
1h		Disables blend optimization for floating point RTs.

8 Reserved

Access:	RO
Format:	MBZ

7 Reserved

Access:	RO
Format:	MBZ

6 Slice Hashing Table Enable

Format:	Enable

This field enables the use of indirect state **SLICE_HASH_TABLE** programmed via **3DSTATE_SLICE_HASH_STATE_POINTER**.

Value	Name	Description
0h	Disable **[Default]**	Slice Hashing is based on default lookup tables with the following function (X+Y) % total_enabled_subslices % enabled_slices
1h	Enable	Slice Hashing is via SLICE_HASH_TABLE from 3DSTATE_SLICE_HASH_STATE_POINTER[total_enabled_subslices-4]

5 Subslice Hashing Table Enable

Format:	Enable

This field enables the use of the Subslice Hashing Mode table programmed via **3DSTATE_SUBSLICE_HASH_TABLE**.

Value	Name	Description
0h	Disable **[Default]**	Subslice Hashing is computed
1h	Enable	Subslice Hashing is via 3DSTATE_SUBSLICE_HASH_TABLE

4 3D Scoreboard Hashing Mode

Value	Name	Description
0	Scoreboard address calculation optimized for Subslice Hashing Mode **[Default]**	Before scoreboard address calculations one address bit will be removed to increase the efficiency of the scoreboard.
1	Scoreboard address calculation not optimized	All address bits used when calculating scoreboard address.

Programming Notes
When **Subslice Hashing Table Enable** and table entries do not contain a checkerboard pattern for each subslice, then optimizing for **Subslice Hashing Mode** may result in reduced performance due to increased false dependencies.

3DSTATE_3D_MODE

3:2 Subslice Hashing Mode

This field is not used by the hardware. Hardware performs 16x16 hashing only.

1:0 Cross Slice Hashing Mode

Format:	U2

Value	Name	Description	Programming Notes
0h	Normal Mode **[Default]**	num_slices > 1 : 16x16 Hashing enabled num_slices == 1: No cross slice hashing	
1h	Cross Slice Hashing Disable	Disables the cross slice hashing	
2h	Reserved	Reserved	
3h	32X32 hashing	32X32 pixel hashing across slices	This setting must be used when sub-slice hashing mode is 16x16 and num_slices > 1

Programming Notes

Cross slice hashing disable must be used when num_slices > 1, normal mode of operation when num_slices > 1 will be to use either 16x16 Hashing or 32x32 Hashing.
A stalling flush is required before changing the value of this field. This is to make sure the entire pipeline is drained.

2 31:16 Mask Bits 2

Format:	Enable[16]

This field is the mask bits for the state bits below. This is a bit wise mask where the bit number-16 is the value of the corresponding bit being masked in the same data word. For example, if you want to update state for bits 6:0 then bits 22:16 must be set.

15:10 Reserved

Access:	RO
Format:	MBZ

9:7 Reserved

Access:	RO
Format:	MBZ

6:0 AMFS MOCS

Format:	MEMORY_OBJECT_CONTROL_STATE

Programming Notes

SW must explicitly program AMFS_FLUSH using PIPE_CONTROL command prior to changing this field. SW must also explicitly flush DC cache flush if the setting of this field is changing from cacheable to non-cacheable in L3.

3DSTATE_AA_LINE_PARAMETERS

3DSTATE_AA_LINE_PARAMETERS

Source: RenderCS

Length Bias: 2

The 3DSTATE_AA_LINE_PARAMS command is used to specify the slope and bias terms used in the improved alpha coverage computation (specifically for DX WHQL compliance). Note that in these devices the coverage values passed to PS threads are full U0.8 values, versus where U0.4 values are passed.

Workaround

Workaround : This command must be followed by a PIPE_CONTROL with CS Stall bit set.,

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 0Ah 3DSTATE_AA_LINE_PARAMETERS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**Dword Length** Default Value: 1h Excludes Dword (0,1) Format: =n
1	31:24	**AA Point Coverage Bias** Format: U0.8 This field specifies the bias term to be used in the aa coverage computation for edges 0 and 3.
	23:16	**AA Coverage Bias** Format: U0.8 This field specifies the bias term to be used in the aa coverage computation for edges 0 and 3.
	15:8	**AA Point Coverage Slope** Format: U0.8 This field specifies the slope term to be used in the aa coverage computation for edges 0 and 3.

3DSTATE_AA_LINE_PARAMETERS		
		If this field is zero, the Windower will revert to legacy aa line coverage computation (though still output expanded U0.8 coverage values).
	7:0	**AA Coverage Slope** Format: U0.8 This field specifies the slope term to be used in the aa coverage computation for edges 0 and 3. If this field is zero, the Windower will revert to legacy aa line coverage computation (though still output expanded U0.8 coverage values).
2	31:24	**AA Point Coverage EndCap Bias** Format: U0.8 This field specifies the bias term to be used in the aa coverage computation for edges 1 and 2.
	23:16	**AA Coverage EndCap Bias** Format: U0.8 This field specifies the bias term to be used in the aa coverage computation for edges 1 and 2.
	15:8	**AA Point Coverage EndCap Slope** Format: U0.8 This field specifies the slope term to be used in the aa coverage computation for edges 1 and 2.
	7:0	**AA Coverage EndCap Slope** Format: U0.8 This field specifies the slope term to be used in the aa coverage computation for edges 1 and 2.

3DSTATE_AMFS

3DSTATE_AMFS		
Source:	BSpec	
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: Opcode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: Opcode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: Opcode
	23:16	**3D Command Sub Opcode** Default Value: 6Fh 3DSTATE_AMFS Format: Opcode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Format: =n n = Total Length -2 Value: 0h — Name: Excludes DWord (0,1)
1	31:0	**AMFS State Body** Format: 3DSTATE_AMFS_BODY

3DSTATE_BINDING_TABLE_POINTERS_DS

3DSTATE_BINDING_TABLE_POINTERS_DS

Source: RenderCS

Length Bias: 2

The 3DSTATE_BINDING_TABLE_POINTERS_DS command is used to define the location of fixed functions' BINDING_TABLE_STATE. Only some of the fixed functions utilize binding tables.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 28h 3DSTATE_BINDING_TABLE_POINTERS_DS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**Binding Table Pointers State Body** Format: 3DSTATE_BINDING_TABLE_POINTERS_BODY

3DSTATE_BINDING_TABLE_POINTERS_GS

3DSTATE_BINDING_TABLE_POINTERS_GS

Source: RenderCS

Length Bias: 2

The 3DSTATE_BINDING_TABLE_POINTERS_GS command is used to define the location of fixed functions' BINDING_TABLE_STATE. Only some of the fixed functions utilize binding tables.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 29h 3DSTATE_BINDING_TABLE_POINTERS_GS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**Binding Table Pointers State Body** Format: 3DSTATE_BINDING_TABLE_POINTERS_BODY

3DSTATE_BINDING_TABLE_POINTERS_HS

3DSTATE_BINDING_TABLE_POINTERS_HS

Source: RenderCS

Length Bias: 2

The 3DSTATE_BINDING_TABLE_POINTERS_HS command is used to define the location of fixed functions' BINDING_TABLE_STATE. Only some of the fixed functions utilize binding tables.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 27h 3DSTATE_BINDING_TABLE_POINTERS_HS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**Binding Table Pointers State Body** Format: 3DSTATE_BINDING_TABLE_POINTERS_BODY

3DSTATE_BINDING_TABLE_POINTERS_PS

3DSTATE_BINDING_TABLE_POINTERS_PS

Source: RenderCS

Length Bias: 2

The 3DSTATE_BINDING_TABLE_POINTERS_PS command is used to define the location of fixed functions' BINDING_TABLE_STATE. Only some of the fixed functions utilize binding tables.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 2Ah 3DSTATE_BINDING_TABLE_POINTERS_PS Format: OpCode
	15	**POSH-Enabled Render State Optimization Enable**
	14:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**Binding Table Pointers State Body** Format: 3DSTATE_BINDING_TABLE_POINTERS_BODY

3DSTATE_BINDING_TABLE_POINTERS_VS

3DSTATE_BINDING_TABLE_POINTERS_VS

Source: RenderCS

Length Bias: 2

The 3DSTATE_BINDING_TABLE_POINTERS_VS command is used to define the location of fixed functions' BINDING_TABLE_STATE. Only some of the fixed functions utilize binding tables.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 26h 3DSTATE_BINDING_TABLE_POINTERS_VS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**Binding Table Pointers State Body** Format: 3DSTATE_BINDING_TABLE_POINTERS_BODY

3DSTATE_BINDING_TABLE_POOL_ALLOC

3DSTATE_BINDING_TABLE_POOL_ALLOC

Source: RenderCS, ComputeCS

Length Bias: 2

This command is to program the base address and size of the binding table pool. The address to fetch the binding table is based on the Binding Table Pool Base Address and the binding table pointer if the Binding Table Pool is enabled. Otherwise the binding table pointer is an offset from the Surface Base Address.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 19h 3DSTATE_BINDING_TABLE_POOL_ALLOC Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 2h DWORD_COUNT_n Format: =n
1..2	63:12	**Binding Table Pool Base Address** Format: VIRTUAL_ADDR[63:12] Specifies the 4K-byte aligned base address for Binding Table Pool. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	11	**Binding Table Pool Enable** Format: U1 When this bit is set it enables HW generation of binding tables. When this bit is cleared it disables HW generation of binding tables. **Programming Notes** If the Binding Table Pool Enable is set when resource streamer is disabled, the engine will fetch binding table entries from the Binding Table Pool Address. In this case, driver must ensure the

		pool is populated and the binding table is coherent prior to engine fetching binding table entries. The format of the binding table must match the same format output by the resource streamer hardware.
	10:7	**Reserved** Access: RO Format: MBZ
	6:0	**Surface Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface. **Programming Notes** Bit 0 is not programmable and is always zero.
3	31:12	**Binding Table Pool Buffer Size** Format: U20 This field specifies the size of the buffer in 4K pages. Any access which straddle or go past the end of the buffer will return 0. Value / Name / Description [0,1048575] / / 0 / No Valid Data / There is no valid data in the buffer **Restriction** Programming size of zero is illegal in the case that the pool is enabled.
	11:0	**Reserved** Access: RO Format: MBZ

3DSTATE_BLEND_STATE_POINTERS

3DSTATE_BLEND_STATE_POINTERS		
Source: RenderCS Length Bias: 2		
The 3DSTATE_BLEND_STATE_POINTERS command is used to set up the pointers to the color calculator state.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 24h 3DSTATE_BLEND_STATE_POINTERS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**Blend State Pointer State Body** Format: 3DSTATE_BLEND_STATE_POINTERS_BODY

3DSTATE_CC_STATE_POINTERS

3DSTATE_CC_STATE_POINTERS		
Source: RenderCS Length Bias: 2		
The 3DSTATE_CC_STATE_POINTERS command is used to set up the pointers to the color calculator state.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 0Eh 3DSTATE_CC_STATE_POINTERS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**CC State Pointers Body** Format: 3DSTATE_CC_STATE_POINTERS_BODY

3DSTATE_CHROMA_KEY

3DSTATE_CHROMA_KEY

Source: RenderCS, ComputeCS

Length Bias: 2

The 3DSTATE_CHROMA_KEY instruction is used to program texture color/chroma-key key values. A table containing four set of values is supported. The ChromaKey Index sampler state variable is used to select which table entry is associated with the map. Texture chromakey functions are enabled and controlled via use of the ChromaKey Enable texture sampler state variable. Texture Color Key (keying on a paletted texture index) is not supported.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: Opcode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: Opcode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: Opcode
	23:16	**3D Command Sub Opcode** Default Value: 04h 3DSTATE_CHROMA_KEY Format: Opcode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:30	**ChromaKey Table Index** Format: U2 Selects which entry in the ChromaKey table is to be loaded Value: [0,3] — Name:
	29:0	**Reserved** Access: RO Format: MBZ

3DSTATE_CHROMA_KEY

2	31:0	**ChromaKey Low Value** This field specifies the "low" (minimum) value of the chroma key range. Texel samples are considered "matching the key" if each component of the texel falls within the (inclusive) chroma range. See ChromaKey High Value for further format, programming info.
3	31:0	**ChromaKey High Value** This field specifies the "high" (maximum) value of the chroma key range. Texel samples are considered "matching the key" if each component of the texel falls within the (inclusive) chroma range.

Programming Notes

ChromaKey values are specified using 8-bit channels. When using surface formats with less than 8 bits per channel, the device will expand channels by replicating the required number of MSBs into the LSBs of each channel. Software must account for this conversion when it programs Chromakey Low/High Values (e.g., by performing the same replication).

For channels that do not exist in the actual surface (e.g., Alpha channel for non-ARGB maps), software must explicitly program full range high/low values (High=FFh, Low=0h for formats using unsigned chroma key values, High=7Fh, Low=FFh for formats using sign magnitude chroma key values) in order to effectively remove the comparison of that field from the ChromaKey function.

For channels in SNORM format in the surface format, the value in the high/low value for that channel is interpreted in sign magnitude format. Negative zero value is not supported (use positive zero instead). For channels with mixed UNORM/SNORM formats (i.e. R5G5_SNORM_B6_UNORM), the ChromaKey is programmed as if all channels are SNORM.

YUV ChromaKey will use an interpolated chrominance value from the map for comparison to the chroma key values for those texels without chrominance due to downsampling. The chrominance value used is the average of values to the left and right of the texel in question.

It is UNDEFINED to program any component of the ChromaKey High Value to be less than the corresponding component of ChromaKey Low Value.

Format = interpreted according to associated texel format "class":

Only the surface formats listed as supported for chroma key in the surface formats table can be used with this feature. Use of any other surface format with chroma key enabled is UNDEFINED.

Surface Format	31:24	23:15	16:8	7:0
ARGB and BC (DXT) formats	A	R	G	B
YCrCb formats	A	Cr	Y	Cb

3DSTATE_CLEAR_PARAMS

3DSTATE_CLEAR_PARAMS

Source: RenderCS

Length Bias: 2

This command defines the depth clear value delivered as a pipelined state command. However, the state change pipelining isn't completely transparent (see restriction below).

HW will internally manage the draining pipe and flushing of the caches when this command is issued. The PIPE_CONTROL restrictions are removed.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 04h 3DSTATE_CLEAR_PARAMS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**Dword Length** Default Value: 1h Excludes Dword (0,1) Format: =n
1..2	63:0	**Clear Params State Body** Format: 3DSTATE_CLEAR_PARAMS_BODY

3DSTATE_CLIP

3DSTATE_CLIP

Source: RenderCS

Length Bias: 2

Restriction
When exected in POCS command stream, this programs the state for CLR stage of the POCS pipeline

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 12h 3DSTATE_CLIP Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 02h Excludes DWord (0,1) Format: =n Total Length - 2
1..3	95:0	**Clip State Body** Format: 3DSTATE_CLIP_BODY

3DSTATE_CONSTANT_ALL

3DSTATE_CONSTANT_ALL

Source: RenderCS, PositionCS

Length Bias: 2

This instruction species pointers and sizes of data to be fetched from memory and loaded as part of the shaders thread payload.

Programming Notes

The programming of enabled buffers is in ascending order where if buffer zero is programmed it will be the first pointer programmed and parsed. Any buffers being omitted must not be programmed.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 6Dh 3DSTATE_CONSTANT_ALL Format: OpCode
	15	**POSH-Enabled Render State Optimization Enable** This bit allows a state command to participate in the POSH-Enabled Render State Optimization function. That function skips over selected commands deemed unnecessary due to the subsequent POSH-Enabled 3DPRIMITIVE command being completely culled by the POSH pipeline. Software shall ensure that the following conditions are met as a prerequisite to setting this bit: 1. The state command is only executed by the Render pipeline. (The bit must be cleared if the state command is executed by the POSH pipeline.) 2. The very next 3DPRIMITIVE command has POSH-Enabled specified. 3. The state programmed is only used by the very next 3DPRIMITIVE command. I.e., following the very next 3DPRIMITIVE command, the relevant state is reprogrammed prior to a subsequent 3DPRIMITIVE command. For improved performance, Software should set this bit if these conditions are met, however Software is not required to do so.

3DSTATE_CONSTANT_ALL

	14:13	**Reserved** Access: RO Format: MBZ
	12:8	**Shader Mask** This bit specifies if the updated pointers are valid for specific shaders. Value / Name: xxxx1b – Vertex Shader Update Enable xxx1xb – Hull Shader Update Enable xx1xxb – Domain Shader Update Enable x1xxxb – Geometry Shader Update Enable 1xxxxb – Pixel Shader Update Enable
	7:0	**DWord Length** Format: =n n = 2b (where b = # of pointers included) Value / Name: 8 – DWORD_COUNT_n **[Default]** [0,8] – 0-4 Pointers
1	31	**Update Mode** If set, pointers that are not valid will retain their value. If clear, then all pointers not valid will be cleared with zero address and zero size. If pointer is valid, then the value of this field is a don't care and the value programmed is always loaded. Value / Name: 0 – Clear pointer and size of non-valid pointers 1 – Retain value of non-valid pointers
	30:20	**Reserved** Access: RO Format: MBZ
	19:16	**Pointer Buffer Mask** This bit specifies which pointers are valid in this command. Value / Name: xxx1b – Buffer 0 Valid xx1xb – Buffer 1 Valid x1xxb – Buffer 2 Valid 1xxxb – Buffer 3 Valid
	15:7	**Reserved** Access: RO Format: MBZ

3DSTATE_CONSTANT_ALL		
	6:0	**Constant Buffer Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for all constant buffers defined in this command.
2..n	255:0	**Constant All Data** Format: 3DSTATE_CONSTANT_ALL_DATA

3DSTATE_CONSTANT_DS

3DSTATE_CONSTANT_DS

Source: RenderCS

Length Bias: 2

This command sets pointers to the push constants for the DS unit. The constant data pointed to by this command is loaded into the DS unit's push constant buffer (PCB).

Workaround

The driver must ensure the following case does not occur without a flush to the 3D engine: 3DSTATE_CONSTANT_* with buffer 3 read length equal to zero committed followed by a 3DSTATE_CONSTANT_* with buffer 0 read length not equal to zero committed. Possible ways to avoid this condition include:

- always force buffer 3 to have a non-zero read length
- always force buffer 0 to a zero read length

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 1Ah 3DSTATE_CONSTANT_DS Format: OpCode
	15	**Reserved** Access: RO Format: MBZ
	14:8	**Constant Buffer Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for all constant buffers defined in this command.
	7:0	**DWord Length** Default Value: 9h Excludes DWord (0,1) Format: =n

3DSTATE_CONSTANT_DS		
1..10	319:0	**Constant Body** Format: **3DSTATE_CONSTANT(Body)** See the 3DSTATE_CONSTANT(Body) format for the shared portion of the 3DSTATE_CONSTANT command for VS, HS, DS, and GS.

3DSTATE_CONSTANT_GS

3DSTATE_CONSTANT_GS

Source: RenderCS

Length Bias: 2

This command sets pointers to the push constants for the GS unit. The constant data pointed to by this command will be loaded into the GS unit's push constant buffer (PCB).

Workaround

The driver must ensure the following case does not occur without a flush to the 3D engine: 3DSTATE_CONSTANT_* with buffer 3 read length equal to zero committed followed by a 3DSTATE_CONSTANT_* with buffer 0 read length not equal to zero committed. Possible ways to avoid this condition include:

- always force buffer 3 to have a non-zero read length
- always force buffer 0 to a zero read length

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 16h 3DSTATE_CONSTANT_GS Format: OpCode
	15	**Reserved** Access: RO Format: MBZ
	14:8	**Constant Buffer Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for all constant buffers defined in this command.
	7:0	**DWord Length** Default Value: 9h Excludes DWord (0,1) Format: =n

3DSTATE_CONSTANT_GS		
1..10	319:0	**Constant Body** Format: **3DSTATE_CONSTANT(Body)** Following table is the shared portion of the 3DSTATE_CONSTANT command for VS, HS, DS, and GS

3DSTATE_CONSTANT_HS

3DSTATE_CONSTANT_HS

Source: RenderCS

Length Bias: 2

This command sets pointers to the push constants for the HS unit. The constant data pointed to by this command is loaded into the HS unit's push constant buffer (PCB).

Workaround

The driver must ensure the following case does not occur without a flush to the 3D engine: 3DSTATE_CONSTANT_* with buffer 3 read length equal to zero committed followed by a 3DSTATE_CONSTANT_* with buffer 0 read length not equal to zero committed. Possible ways to avoid this condition include:

- always force buffer 3 to have a non-zero read length
- always force buffer 0 to a zero read length

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 19h 3DSTATE_CONSTANT_HS Format: OpCode
	15	**Reserved** Access: RO Format: MBZ
	14:8	**Constant Buffer Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for all constant buffers defined in this command.
	7:0	**DWord Length** Default Value: 9h Excludes DWord (0,1) Format: =n

3DSTATE_CONSTANT_HS		
1..10	319:0	**Constant Body** Format: **3DSTATE_CONSTANT(Body)** Following table is the shared portion of the 3DSTATE_CONSTANT command for VS, HS, DS, and GS

3DSTATE_CONSTANT_PS

3DSTATE_CONSTANT_PS

Source: RenderCS

Length Bias: 2

This command sets pointers to the push constants for the PS unit. The constant data pointed to by this command is loaded into the PS unit's push constant buffer (PCB).

Programming Notes

A 3DSTATE_GATHER_PS command must be dispatched along with any 3DSTATE_CONSTANT_PS command when the Gather Pool is enabled within a batch buffer.

The 3DSTATE_CONSTANT_* command is not committed to the shader unit until the corresponding (same shader) 3DSTATE_BINDING_TABLE_POINTER_* command is parsed. For example, the 3DSTATE_CONSTANT_VS command will not fetch the constant buffers from memory and make available to the shader until the 3DSTATE_BINDING_TABLE_POINTERS_VS is parsed by the render command streamer. In case of multiple 3DSTATE_CONSTANT_VS programmed prior to 3DSTATE_BINDING_TABLE_POINTER_VS, only the most recently programmed 3DSTATE_CONSTANT_VS will be committed.
SW must ensure any uncommitted 3DSTATE_CONSTANT_* commands programmed get explicitly committed prior to enabling or disabling the legacy mode.
On usage model of enabling legacy mode is when Resource Streamer is not enabled.

Workaround

The driver must ensure the following case does not occur without a flush to the 3D engine: 3DSTATE_CONSTANT_* with buffer 3 read length equal to zero committed followed by a 3DSTATE_CONSTANT_* with buffer 0 read length not equal to zero committed. Possible ways to avoid this condition include:

- always force buffer 3 to have a non-zero read length
- always force buffer 0 to a zero read length

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 17h 3DSTATE_CONSTANT_PS

3DSTATE_CONSTANT_PS

		Format: OpCode
	15	**Reserved** Access: RO Format: MBZ
	14:8	**Constant Buffer Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for all constant buffers defined in this command.
	7:0	**Dword Length** Default Value: 9h Excludes DWord (0,1) Format: =n
1..10	319:0	**Constant Body** Format: 3DSTATE_CONSTANT(Body) Following table is the shared portion of the 3DSTATE_CONSTANT command for VS, HS, DS, and GS

3DSTATE_CONSTANT_TS_POINTER

3DSTATE_CONSTANT_TS_POINTER

Source: RenderCS

Length Bias: 2

This command sets pointers to the Texel Shaders push constants for AMFS unit. The constant data pointed to by this command is stored in memory, not in push constant buffer (PCB).
The 3DSTATE_CONSTANT_TS_POINTER command gets committed to the shader on parsing 3DPRIMTIVE command or on encountering an explicit/implicit flush (Texel Shaders do not use binding table or Resource Streamer).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 73h 3DSTATE_CONSTANT_TS_POINTER Format: OpCode
	15	**Reserved** Access: RO Format: MBZ
	14:8	**Constant Buffer Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for constant buffers defined in this command.
	7:0	**DWord Length** Default Value: 1h Excludes DWord (0,1) Format: =n n = Total Length -2
1..2	63:0	**Constant TS Pointer State Body** Format: 3DSTATE_CONSTANT_TS_POINTER_BODY

3DSTATE_CONSTANT_VS

3DSTATE_CONSTANT_VS

Source: RenderCS

Length Bias: 2

This command sets pointers to the push constants for VS unit. The constant data pointed to by this command is loaded into the VS unit's push constant buffer (PCB).

Programming Notes

When executed in the POCS command stream, this command programs state for the VSR stage of the POCS pipeline.

Workaround

The driver must ensure the following case does not occur without a flush to the 3D engine: 3DSTATE_CONSTANT_* with buffer 3 read length equal to zero committed followed by a 3DSTATE_CONSTANT_* with buffer 0 read length not equal to zero committed. Possible ways to avoid this condition include:

- always force buffer 3 to have a non-zero read length
- always force buffer 0 to a zero read length

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 15h 3DSTATE_CONSTANT_VS Format: OpCode
	15	**Reserved** Access: RO Format: MBZ
	14:8	**Constant Buffer Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for all constant buffers defined in this command.

<table>
<tr><th colspan="3">3DSTATE_CONSTANT_VS</th></tr>
<tr><td></td><td>7:0</td><td>DWord Length<table><tr><td>Default Value:</td><td>9h Excludes DWord (0,1)</td></tr><tr><td>Format:</td><td>=n</td></tr></table></td></tr>
<tr><td>1..10</td><td>319:0</td><td>Constant Body<table><tr><td>Format:</td><td>3DSTATE_CONSTANT(Body)</td></tr></table>Following table is the shared portion of the 3DSTATE_CONSTANT command for VS, HS, DS, and GS</td></tr>
</table>

3DSTATE_CPS_POINTERS

3DSTATE_CPS_POINTERS		
Source: RenderCS Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: Opcode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: Opcode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 22h 3DSTATE_CPS_POINTERS Format: OpCode
	15:0	**DWord Length** Default Value: 0h Format: =n
1	31:0	**CPS Pointers State Body** Format: 3DSTATE_CPS_POINTERS_BODY

3DSTATE_DEPTH_BOUNDS

3DSTATE_DEPTH_BOUNDS

Source: BSpec

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 03h GFXPIPE Format: Opcode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: Opcode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: Opcode
	23:16	**3D Command Sub Opcode** Default Value: 71h 3DSTATE_DEPTH_BOUNDS Format: Opcode
	15	**Depth Bounds Test Enable Modify Disable** Format: Disable When this bit is set, the following fields will be ignored: • Depth Bounds Test Enable
	14	**Depth Bounds Test Value Modify Disable** Format: Disable When this bit is set, the following fields will be ignored: • Depth Bounds Test Min Value • Depth Bounds Test Max Value
	13:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 02h Excludes DWord (0,1) Format: =n Total Length - 2
1..3	95:0	**Depth Bounds State Body** Format: 3DSTATE_DEPTH_BOUNDS_BODY

3DSTATE_DEPTH_BUFFER

3DSTATE_DEPTH_BUFFER

Source: RenderCS

Length Bias: 2

The depth buffer surface state is delivered as a pipelined state packet. However, the state change pipelining isn't completely transparent (see restriction below).

WM HW will internally manage the draining pipe and flushing of the caches when this command is issued. The PIPE_CONTROL restrictions are removed.

Programming Notes

Note for validation teams. If the depth surface is backdoor initialized or written to directly by the CPU, the values placed in the Depth Surface must be within the numeric range of [0.0 ... 1.0] for DirectX and may in the future include +/- max floating-point values; but not +/-Inf, DNORMs or any NaN code.
If the Depth surface is not present, SW must set the Surface Type field to SURFTYPE_NULL

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 5h 3DSTATE_DEPTH_BUFFER Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Format: =n Excludes DWord(0,1) Value: 6h — Name: Excludes Dword (0,1) **[Default]**
1	31:29	**Surface Type** Value: 0h — Name: Reserved — Description: Reserved

1h	SURFTYPE_2D	Defines a 2-dimensional map or array of maps
2h	Reserved	Reserved
3h	SURFTYPE_CUBE	Defines a cube map
4h-6h	Reserved	
7h	SURFTYPE_NULL	Defines a null surface

Programming Notes
The Surface Type of the depth buffer must be the same as the Surface Type of the 1. Render target(s) (defined in SURFACE_STATE), unless either the depth buffer or render targets are SURFTYPE_NULL 2. Stencil buffer (defined in 3DSTATE_STENCIL_BUFFER) unless either the depth buffer or Stencil buffer surf_type are SURFTYPE_NULL. If depth is enabled with 1D render target, depth surface type needs to be set to 2D surface type and height set to 1. For this case only, the Surface Type of the depth buffer can be 2D while the Surface Type of the render target(s) are 1D, representing an exception to a programming note above.

28 **Depth Write Enable**

Format:	Enable

This field enables depth writes to the depth buffer surface. Both this field and the Depth Buffer Write Enable field in DEPTH_STENCIL_STATE must be enabled in order for depth writes to occur.

27 **Null Page Coherency Enable**

Format:	Enable

This field is used for enabling NULL coherency as defined under Tiled Resources.

Value	Name
1	Enable
0	Disable **[Default]**

Programming Notes
SW must enable this bit only if Tiled Resource is enabled

26:24 **Surface Format**

Specifies the format of the depth buffer.

Value	Name
0h	Reserved
1h	D32_FLOAT
2h	Reserved
3h	D24_UNORM_X8_UINT
4h	Reserved
5h	D16_UNORM
6h-7h	Reserved

3DSTATE_DEPTH_BUFFER

23 **Corner Texel Mode**

Format:	Enable

This field, when ENABLED, indicates when a surface is using corner texel-mode for depth surface. This bit changes how the size of each MIP when calculating the offset within a surface.

Value	Name	Description
0h	Disable **[Default]**	Corner Texel mode is not enabled.
1h	Enable	Corner Texel Mode is enabled.

Programming Notes
Corner texel for the depth buffer must be the same as the Surface Type of the 1. Render target(s) (defined in SURFACE_STATE), unless either the depth buffer or render targets are SURFTYPE_NULL 2. Stencil buffer (defined in 3DSTATE_STENCIL_BUFFER) unless either the depth buffer or Stencil buffer surf_type are SURFTYPE_NULL

22 **Hierarchical Depth Buffer Enable**

Format:	Enable

If enabled, indicates that a hierarchical depth buffer is defined.

Programming Notes
If this field is enabled, the Software Tiled Rendering Mode must be NORMAL. This field must be disabled if Early Depth Test Enable is disabled OR if depth buffer surface type is NULL. This field must be disabled if surface type field is SURFTYPE_1D

21 **Depth Buffer Compression Enable**

Format:	Enable

if enabled, indicates that Depth Buffer Compression is Enabled
When thisfield is enabled, Depth Buffer must be initialized via Depth Clear (HZ_OP) when HiZ is enabled. If HiZ is disabled, Depth Buffer must be initialized via full screen primitive with Depth Write enabled and Depth Test Disabled.

Programming Notes
SW must set this bit if the Depth Control surface enable is also set. The depth surface control enable is in Bit[19] of this DWORD.

20 **Reserved**

Access:	RO
Format:	MBZ

19 **Control Surface Enable**

If set to 1, it indicates if the common control surface is present. The read and write transaction opcodes sent by RCZto the fabric are different depending on the control surface. If the control surface is not present, the reads and writes are in legacy mode. If the control surface is present, the reads and write opcodes will be either UNCOMPRESSED_TYP for uncompressible transactions (resolves) or COMPRESSED_TYP for compressible transactions.

Programming Notes
SW must set this bit to "1", if the common control surface is present in the system.

3DSTATE_DEPTH_BUFFER

	18	**Reserved** Access: RO Format: MBZ
	17:0	**Surface Pitch** Format: U18-1 For TileYF and TileYS surfaces, the range is dependent on the Cu parameter (refer to *Memory Data Formats* section for the definition of the Cu parameter depending on the case). The range in bytes is [2^{Cu}-1,262143] -> [(2^{Cu})B,256KB] = [1 tile, 256KB/(2^{Cu}) tiles]
2..3	63:0	**Surface Base Address** Format: VIRTUAL_ADDR[63:0]
4	31	**Reserved** Access: RO Format: MBZ
	30:17	**Height** Format: U14-1 This field specifies the height of the surface. If the surface is MIP-mapped, this field contains the height of the base MIP level.

Value	Name	Description
[7Fh,3FFFFh]		corresponding to [128B, 256KB] also restricted to a multiple of 128B

Programming Notes

The minimum pitch should be calculated based on Cu, Cv, W_L.
The Cu, Cv are the tile constants and $W_L$is the aligned width adjusted for MSAA.
Refer to **2D Surfaces** *to get the Cu, Cv, $W_L$values and Calculations.*
Then use this for pitch formula :
Minimum_pitch =(ceiling((W_0 pixel_size) / (1 « Cu)) *(1 « Cu)) 1 ; //$W_0$is the aligned width for the largest LOD (i.e LOD 0)*
(1 « Cu) = tile width in bytes
(1 « Cv) = tile height in lines

Programming Notes

This field specifies address of the buffer in mapped Graphics Memory. The Depth Buffer can only be mapped to Main Memory (uncached). If the surface is tiled, the base address must conform to the Per-Surface Tiling Alignment. If the buffer is linear, the surface must be 64-byte aligned.
If the buffer is linear, the surface must be 64-byte aligned.

Value	Name	Description	Exists If
[0,16383]	Legal Range	Height of surface - 1 (y/v dimension)	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_2D')
[0,16383]	Legal Range	y/v dimension	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_CUBE')

3DSTATE_DEPTH_BUFFER

Programming Notes

The Height of the depth buffer must be the same as the Height of the
1. render target(s) (defined in SURFACE_STATE), unless Surface Type is SURFTYPE_2D with Depth = 0 (non-array) and LOD = 0 (non-mip mapped) or if SURFACE_STATE_SURFTYPE is NULL.
2.Stencil buffer (defined in 3DSTATE_STENCIL_BUFFER) unless Stencil buffer surf_type is SURFTYPE_NULL

16 **Reserved**

Access:	RO
Format:	MBZ

15 **Reserved**

Access:	RO
Format:	MBZ

14:1 **Width**

Format:	U14-1

This field specifies the width of the surface. If the surface is MIP-mapped, this field specifies the width of the base MIP level. The width is specified in units of pixels.

Value	Name	Description	Exists If
[0,16383]	Legal Range	Width of surface - 1 (x/u dimension)	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_2D')
[0,16383]	Legal Range	Width of surface - 1 (x/u dimension)	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_CUBE')

Programming Notes

The Width specified by this field must be less than or equal to the surface pitch (specified in bytes via the Surface Pitch field). For cube maps, Width must be set equal to Height.
The Width of the depth buffer must be the same as the
1. Width of the render target(s) (defined in SURFACE_STATE), unless Surface Type is SURFTYPE_1D or SURFTYPE_2D with Depth = 0 (non-array) and LOD = 0 (non-mip mapped) or if SURFACE_STATE_SURFTYPE is NULL.
2.Stencil buffer (defined in 3DSTATE_STENCIL_BUFFER) unless Stencil buffer surf_type is SURFTYPE_NULL

0 **Reserved**

Access:	RO
Format:	MBZ

5 31 **Reserved**

Access:	RO
Format:	MBZ

3DSTATE_DEPTH_BUFFER

30:20 **Depth**

Format:	U11-1

This field specifies the total number of levels for a volume texture or the number of array elements allowed to be accessed starting at the Minimum Array Element for arrayed surfaces. If the volume texture is MIP-mapped, this field specifies the depth of the base MIP level.

Value	Name	Description	Exists If
[0,2047]	Legal Range	Number of array elements - 1	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_2D')
[0,0]	Legal Range	Must be zero	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_CUBE')

Programming Notes

The Depth of the depth buffer must be the same as
1. The Depth of the render target(s) (defined in SURFACE_STATE).
2.Stencil buffer (defined in 3DSTATE_STENCIL_BUFFER) unless Stencil buffer surf_type is SURFTYPE_NULL

19 **Reserved**

Access:	RO
Format:	MBZ

18:8 **Minimum Array Element**

Format:	U11

For 2D Surfaces: This field indicates the minimum array element that can be accessed as part of this surface. The delivered array index is added to this field before being used to address the surface. For Other Surfaces This field is ignored

Value	Name	Exists If
[0,2047]	SURFTYPE_2D	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_1D'|Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_2D')

Programming Notes

Minimum array element of the depth buffer must be the same as the Surface Type of the
1. Render target(s) (defined in SURFACE_STATE), unless either the depth buffer or render targets are SURFTYPE_NULL
2. Stencil buffer (defined in 3DSTATE_STENCIL_BUFFER) unless either the depth buffer or Stencil buffer surf_type are SURFTYPE_NULL

7 **Reserved**

Access:	RO
Format:	MBZ

6:0 **Depth Buffer Object Control State**

Format:	MEMORY_OBJECT_CONTROL_STATE

Specifies the memory object control state for the depth buffer.

3DSTATE_DEPTH_BUFFER

6 — 31:30 — Tiled Mode

For Depth Buffer Surfaces: This field specifies the tiled mode. For other surfaces: This field is ignored.

Value	Name	Description
0h	NONE	No tiled modes (TileYf, TileYs). Use Standard TileY
1h	TILEYF	4KB tiled mode
2h	TILEYS	64KB tiled mode
3h	Reserved	

Programming Notes
If **Tile Mode** is not set to TILEMODE_YMAJOR, this field must be set to TRMODE_NONE.

29:26 — Mip Tail Start LOD

Format:	U4

For Sampling Engine, Render Target, and Typed Surfaces: This field indicates which LOD is the first one in the MIP tail if **Tiled Mode** is not TRMODE_NONE. The MIP tail has a different layout than the rest of the surface. Refer to the *Memory Data Formats* section for more details.
For other surfaces: This field is ignored.

Programming Notes
This field must be zero if the **Surface Format** is MONO8. This field is ignored if **Tiled Mode** is TRMODE_NONE unless **Surface Type** is SURFTYPE_1D.
If Tiled Mode is not TRMODE_NONE, this field must be set to ensure that mips within the mip tail do not overlap given the storage algorithms given in the Memory Data Formats section. The following table indicates the maximum size of the mip that is set to be the Mip Tail Start LOD for various cases:

Tiling Mode	Slot Size in Bytes	8-bit Size	16-bit Size	32-bit Size
2D TIleYs 1x	32KB	(128, 256)	(128, 128)	(64, 128)
2D TileYf 1x	2KB	(32, 64)	(32, 32)	(16, 32)

25:6 — Reserved

Access:	RO
Format:	MBZ

5:0 — Reserved

Access:	RO
Format:	MBZ

7 — 31:21 — Render Target View Extent

Format:	U11-1

Value	Name	Description	Exists If
[0,2047]	Legal	Number of array	(Structure[RENDER_SURFACE_STATE][Surface

	Range	elements- 1	Type]=='SURFTYPE_2D')
[0,0]	Legal Range	Must be zero	(Structure[RENDER_SURFACE_STATE][Surface Type]=='SURFTYPE_CUBE')

Programming Notes
Render target View Extent of the depth buffer must be the same as the Surface Type of the 1. Render target(s) (defined in SURFACE_STATE), unless either the depth buffer or render targets are SURFTYPE_NULL 2. Stencil buffer (defined in 3DSTATE_STENCIL_BUFFER) unless either the depth buffer or Stencil buffer surf_type are SURFTYPE_NULL

20 **Reserved**

Access:	RO
Format:	MBZ

19:16 **Surf LOD**

Value	Name
[0-14]	

Programming Notes
Surf LOD of the depth buffer must be the same as the Surface Type of the 1. Render target(s) (defined in SURFACE_STATE), unless either the depth buffer or render targets are SURFTYPE_NULL 2. Stencil buffer (defined in 3DSTATE_STENCIL_BUFFER) unless either the depth buffer or Stencil buffer surf_type are SURFTYPE_NULL

15 **Reserved**

Access:	RO
Format:	MBZ

14:0 **Surface QPitch**

Format:	U17[16:2]

Format: QPitch[16:2]
The interpretation of this field is dependent on **Surface Type** as follows:

- SURFTYPE_2D/CUBE: distance in *rows* between array slices.

Other surface types: field is ignored

Value	Name	Description
[1h,7FFFh]		in multiples of 4 (low 2 bits missing)

Programming Notes
For 2D Surfaces: This field must be set to the same value as the Depth field. For Other Surfaces,

3DSTATE_DEPTH_BUFFER		
		This field is ignored . Refer to Alignment Unit Size for alignment sizes based on MSAA and Depth-format. Software must ensure that this field is set to a value sufficiently large that array slices in the surface do not overlap. Refer to the Memory Data Formats section for information on how surfaces are stored. TYS/TYF QPitch is valid only for 2D array surfaces and represents the tile-padded total number of texels(lines) in a single array slice. Height of each LOD: HL = AlignToTileHeight(MSAA_height_factor * (**height**»L) > 0?**height**»L : 1), where AlignToTileHeight(x) is (ceiling((x) / (1 « Cv)) *(1 « Cv)) Height of all LODs is a sum: H = H0+H1+..Hn, N is number of mip levels. If surface has MIP tail, equation stops at Hn where n=MipTailStartLOD. MipTail is single tile. QPitch is multiple of tile height (1 « Cv) and should be equal or greater H computed above.

3DSTATE_DRAWING_RECTANGLE

3DSTATE_DRAWING_RECTANGLE

Source: RenderCS

Length Bias: 2

The 3DSTATE_DRAWING_RECTANGLE command is used to set the 3D drawing rectangle and related state.

Restriction
When executed in POCS command stream, this programs the drawing rectangle for the SFR stage of the POCS pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 00h 3DSTATE_DRAWING_RECTANGLE Format: OpCode
	15:14	**Core Mode Select** Format: U2 Specifies which core this command will be considered valid and update based on the state in this command. Value / Name / Description: 0h / Legacy / Both cores are enabled and will update the state. 1h / Core 0 Enabled / State will be updated in Core 0 only 2h / Core 1 Enabled / State will be updated in Core 1 only 3h / Reserved /
	13:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) Format: =n

3DSTATE_DRAWING_RECTANGLE

1 — 31:16 — Clipped Drawing Rectangle Y Min

Format:	U16

Specifies Ymin value of (inclusive) intersection of Drawing rectangle with the Color (Destination) Buffer, used for clipping. Pixels with Y coordinates less than Ymin will be clipped out.

Value	Name
[0,16383]	Device ignores bits 31:30

Programming Notes
This value can be larger than Clipped Drawing Rectangle Y Max. If Ymin>Ymax, the clipped drawing rectangle is null, all polygons are discarded. If Ymin==Ymax, the clipped drawing rectangle is 1 pixel wide in the Y direction.

1 — 15:0 — Clipped Drawing Rectangle X Min

Format:	U16

Specifies Xmin value of (inclusive) intersection of Drawing rectangle with the Color (Destination) Buffer, used for clipping. Pixels with X coordinates less than Xmin will be clipped out.

Value	Name
[0,16383]	Device ignores bits 15:14

Programming Notes
This value can be larger than Clipped Drawing Rectangle X Max. If Xmin>Xmax, the clipped drawing rectangle is null, all polygons are discarded. If Xmin==Xmax, the clipped drawing rectangle is 1 pixel wide in the X direction.

2 — 31:16 — Clipped Drawing Rectangle Y Max

Format:	U16

Specifies Ymax value of (inclusive) intersection of Drawing rectangle with the Color (Destination) Buffer, used for clipping. Pixels with coordinates greater than Ymax will be clipped out.

Value	Name
[0,16383]	Device ignores bits 31:30

Programming Notes
This value can be less than Clipped Drawing Rectangle Y Min. If Ymax<Ymin, the clipped drawing rectangle is null, all polygons are discarded. If Ymin==Ymax, the clipped drawing rectangle is 1 pixel wide in the Y direction.

2 — 15:0 — Clipped Drawing Rectangle X Max

Format:	U16

Specifies Xmax value of (inclusive) intersection of Drawing rectangle with the Color (Destination) Buffer, used for clipping. Pixels with coordinates greater than Xmax will be clipped out.

Value	Name
[0,16383]	Device ignores bits 15:14

3DSTATE_DRAWING_RECTANGLE		
		Programming Notes This value can be less than Clipped Drawing Rectangle X Min. If Xmax<Xmin, the clipped drawing rectangle is null, all polygons are discarded. If Xmin==Xmax, the clipped drawing rectangle is 1 pixel wide in the X direction.
3	31:16	**Drawing Rectangle Origin Y** Format: S15 Range: [-16384,16383] (Bit 31 should be a sign extension) Specifies Y origin of Drawing Rectangle (in whole pixels) relative to origin of the Color Buffer, used to map incoming (Draw Rectangle-relative) vertex positions to the Color Buffer space.
	15:0	**Drawing Rectangle Origin X** Format: S15 Range: [-16384,16383] (Bit 15 should be a sign extension) Specifies X origin of Drawing Rectangle (in whole pixels) relative to origin of the Color Buffer, used to map incoming (Draw Rectangle-relative) vertex positions to the Color Buffer space.

3DSTATE_DS

3DSTATE_DS		
Source:	RenderCS	
Length Bias:	2	
The state used by DS is defined with this inline state packet		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 1Dh 3DSTATE_DS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 9h Excludes DWord (0,1) Format: =n
1..10	319:0	**DS State Body** Format: 3DSTATE_DS_BODY

3DSTATE_GS

3DSTATE_GS		
Source:	RenderCS	
Length Bias:	2	
Controls the GS stage hardware.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 11h 3DSTATE_GS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 8h Excludes DWord (0,1) Format: =n
1..9	287:0	**GS State Body** Format: 3DSTATE_GS_BODY

3DSTATE_HIER_DEPTH_BUFFER

3DSTATE_HIER_DEPTH_BUFFER

Source: RenderCS

Length Bias: 2

This command sets the surface state of the hierarchical depth buffer, delivered as a pipelined state command. However, the state change pipelining isn't completely transparent (see restriction below).

WM HW will internally manage the draining pipe and flushing of the caches when this command is issued. The PIPE_CONTROL restrictions are removed.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 07h 3DSTATE_HIER_DEPTH_BUFFER Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**Dword Length** Default Value: 3h Excludes Dword (0,1) Format: =n
1..4	127:0	**Hier Depth Buffer State Body** Format: 3DSTATE_HIER_DEPTH_BUFFER_BODY

3DSTATE_HS

3DSTATE_HS		
Source: RenderCS Length Bias: 2		
Controls the HS stage hardware.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 1Bh 3DSTATE_HS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 7 Excludes DWord (0,1) Format: =n
1..8	255:0	**HS State Body** Format: 3DSTATE_HS_BODY

3DSTATE_INDEX_BUFFER

3DSTATE_INDEX_BUFFER

Source: RenderCS

Length Bias: 2

This command is used to specify the current IB state used by the VF function. At most one IB is defined and active at any given time. NOTES: The IB must be specified before any RANDOM 3D_PRIMITIVE commands are issued It is possible to have vertex elements source completely from generated ID values and therefore not require any Index Buffer accesses. In this case, VF function will simply ignore the Index Buffer state.

Programming Notes

When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 0Ah 3DSTATE_INDEX_BUFFER Format: OpCode
	15	**Reserved** Access: RO Format: MBZ
	14:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 3h Excludes DWord (0,1) Format: =n
1..4	127:0	**Index Buffer State Body** Format: 3DSTATE_INDEX_BUFFER_BODY

3DSTATE_LINE_STIPPLE

3DSTATE_LINE_STIPPLE

Source: RenderCS

Length Bias: 2

The 3DSTATE_LINE_STIPPLE command is used to specify state variables used in the Line Stipple function.

Workaround

Workaround : This command must be followed by a PIPE_CONTROL with CS Stall bit set.,

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 08h 3DSTATE_LINE_STIPPLE Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**Dword Length** Default Value: 1h Excludes Dword (0,1) Format: =n
1	31	**Modify Enable (Current Repeat Counter, Current Stipple Index)** Format: Enable Modify enable for **Current Repeat Counter** and **Current Stipple Index** fields. **Programming Notes** It is provided only for HW-generated commands as part of context save/restore. SW must initialize the current repeat counter, current stipple count fields if it sets this bit to enable. SW must set this bit to reset the stipple count.
	30	**Reserved** Access: RO Format: MBZ

3DSTATE_LINE_STIPPLE

	29:21	**Current Repeat Counter** Format: U9 This field sets the HW-internal repeat counter state. SW must initilize it to 1 if the modify enable is set.
	20	**Reserved** Access: RO Format: MBZ
	19:16	**Current Stipple Index** Format: U4 This field sets the HW-internal stipple pattern index. SW must initialize it to 0 if the modify enable is set.
	15:0	**Line Stipple Pattern** Format: Enable[16] Specifies a pattern used to mask out bit specific pixels while rendering lines.
2	31:15	**Line Stipple Inverse Repeat Count** Format: U1.16 Range: [0.00390625, 1.0] Specifies the inverse (truncated) of the repeat count for the line stipple function.
	14:9	**Reserved** Access: RO Format: MBZ
	8:0	**Line Stipple Repeat Count** Format: U9 Specifies the repeat count for the line stipple function. Value: [1, 256] — Name:

3DSTATE_MULTISAMPLE

3DSTATE_MULTISAMPLE

Source: RenderCS

Length Bias: 2

The 3DSTATE_MULTISAMPLE command is used to specify multisample state associated with the current render target/depth buffer/stencil buffer.

Programming Notes

It is illegal to render to surfaces with multiple different values of the state fields in this command.

Restriction : When executed in the POCS command stream, this command programs the multisample state for the Raster stage of the POCS pipeline

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 0Dh 3DSTATE_MULTISAMPLE Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n
1	31:0	**Multisample State Body** Format: 3DSTATE_MULTISAMPLE_BODY

3DSTATE_POLY_STIPPLE_OFFSET

3DSTATE_POLY_STIPPLE_OFFSET

Source: RenderCS

Length Bias: 2

The 3DSTATE_POLY_STIPPLE_OFFSET command is used to specify the origin of the repeated screen-space Polygon Stipple Pattern as an X, Y offset from the Color Buffer origin.

Workaround

This command must be followed by a PIPE_CONTROL with CS Stall bit set.,

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 06h 3DSTATE_POLY_STIPPLE_OFFSET Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**Dword Length** Default Value: 0h Excludes Dword (0,1) Format: =n
1	31:13	**Reserved** Access: RO Format: MBZ
	12:8	**Polygon Stipple X Offset** Format: U5 Specifies a 5 bit x address offset in the poly stipple pattern Value: [0,31] — Name:

3DSTATE_POLY_STIPPLE_OFFSET

	7:5	**Reserved** Access: RO Format: MBZ
	4:0	**Polygon Stipple Y Offset** Format: U5 Specifies a 5 bit y address offset in the poly stipple pattern [0,31]

3DSTATE_POLY_STIPPLE_PATTERN

3DSTATE_POLY_STIPPLE_PATTERN

Source: RenderCS

Length Bias: 2

The 3DSTATE_POLY_STIPPLE_PATTERN command is used to specify the 32x32 Polygon Stipple Pattern used in the Polygon Stipple function of the WM unit.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 07h 3DSTATE_POLY_STIPPLE_PATTERN Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**Dword Length** Default Value: 1Fh Excludes Dword (0,1) Format: =n
1..32	1023:0	**Pattern Row** Format: U32[32] Specifies a pattern used by Polygon Stipple to mask out specific pixels of every 32x32 area rendered.

3DSTATE_PRIMITIVE_REPLICATION

3DSTATE_PRIMITIVE_REPLICATION		
Source:	RenderCS, PositionCS	
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 6Ch 3DSTATE_PRIMITIVE_REPLICATION Format: OpCode
	15:11	**Reserved** Access: RO Format: MBZ
	10:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 4h Excludes DWord (0,1) Format: =n
1..5	159:0	**Primitive Replication State Body** Format: 3DSTATE_PRIMITIVE_REPLICATION_BODY

3DSTATE_PS_BLEND

3DSTATE_PS_BLEND		
Source: RenderCS Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 4Dh 3DSTATE_PS_BLEND Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:0	**PS Blend State Body** Format: 3DSTATE_PS_BLEND_BODY

3DSTATE_PS_EXTRA

3DSTATE_PS_EXTRA		
Source: RenderCS Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 4fh 3DSTATE_PS_EXTRA Format: OpCode
	15	**POSH-Enabled Render State Optimization Enable 1**
	14:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:0	**PS Extra State Body** Format: 3DSTATE_PS_EXTRA_BODY

3DSTATE_PS

3DSTATE_PS		
Source: RenderCS Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 20h 3DSTATE_PS Format: OpCode
	15	**Reserved** Access: RO Format: MBZ
	14:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0Ah Excludes DWord (0,1) Format: =n
1..11	351:0	**PS State Body** Format: 3DSTATE_PS_BODY

3DSTATE_PTBR_FREE_LIST_BASE_ADDRESS

3DSTATE_PTBR_FREE_LIST_BASE_ADDRESS

Source: PositionCS

Length Bias: 2

The 3DSTATE_PTBR_FREE_LIST_BASE_ADDRESS command programmed specifies a 4KB aligned base address to 128KB (2^{16}* 2B) of contiguous memory surface called Free-List memory. This command must be programmed only for POSH pipe and must not be programmed for render pipe. This command is context specific and must be programmed for each of the POSH enabled context. This command gets context save/restored as part of the POSH context.

POSH pipe uses Free-List memory to track the free pages (page offsets from PTBR Page Pool Base Address are stored) available for recording visibility data. Pages get consumed form the Free-List as and when the visibility data is recorded by POSH pipe and pages get added to the Free-List when freed up by the render engine on rendering.

Programming Notes

The Free List Base Address must be set to a valid page if any batch buffer is enabled for the POSH pipe.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 24h 3DSTATE_PTBR_FREE_LIST_BASE_ADDRESS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 01h Excludes DWord (1,2) Format: =n
1..2	63:12	**Free List Base Address** Format: VIRTUAL_ADDR[63:12] The Free List Base Address specifies a 4KB aligned base address for a Free-List memory.

<table>
<tr><th colspan="3">3DSTATE_PTBR_FREE_LIST_BASE_ADDRESS</th></tr>
<tr><td></td><td>11:7</td><td>Reserved<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>6:0</td><td>Free List Memory Object Control State<table><tr><td>Format:</td><td>MEMORY_OBJECT_CONTROL_STATE</td></tr></table>Specifies the MOCS state for Free-List memory accesses using PTBR_FREE_LIST_BASE_ADDRESS.</td></tr>
</table>

3DSTATE_PTBR_MARKER

3DSTATE_PTBR_MARKER

Source: RenderCS

Length Bias: 2

Programming Notes

Start of Tile and End of Tile must not be set in the same command. Every Start of Tile marker programmed must have a corresponding End of Tile marker programmed.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 6Ah 3DSTATE_PTBR_MARKER Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Format: =n
1	31:0	**PTBR Marker State Body** Format: 3DSTATE_PTBR_MARKER_BODY

3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS

3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS

Source: PositionCS

Length Bias: 2

The 3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS command specifies base address of the PTBR page pool allocated to HW for recording visibility data in POSH Tile Based Rendering Mode (PTBR). The size of the PTBR Page Pool Size is programmed through the MMIO register PTBR_PAGE_POOL_SIZE_REGISTER following this command. SW must always set Page Pool Restart to enabled when programming this command.
This command must be programmed only for POSH pipe and must not be programmed for render pipe. This command is context specific and must be programmed for each of the POSH enabled context. This command gets context save/restored as part of the POSH context. HW context save/restores 3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS command with Page Pool Restart set to disabled.

Programming Notes

The Page Pool Base Address must be set to a valid page if any batch buffer is enabled for the POSH pipe.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 21h 3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 01h Excludes DWord (1,2) Format: =n
1..2	63:12	**Page Pool Base Address** Format: VIRTUAL_ADDR[63:12] Page Pool Base Address specifies the 4KB aligned base address of the contiguous graphics memory allocated to HW for recording visibility data in POSH Tile Based Rendering Mode (PTBR).

3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS

	11	**Page Pool Restart** Format: Enable • SW must always set Page Pool Restart to enabled on programming 3DSTATE_PTBR_PAGE_POOL_ADDRESS command. • SW must ensure both POSH and Render pipe are flushed and synchronized prior to programing 3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS command, this ensures there aren't any pages in use on the render side from the old PTBR page pool when this command is parsed by HW.
	10:7	**Reserved** Access: RO Format: MBZ
	6:0	**Page Pool Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the MOCS state for PTBR Page Pool memory accesses using Page Pool Base Address.
3	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Reserved** Access: RO Format: MBZ

3DSTATE_PTBR_RENDER_LIST_BASE_ADDRESS

3DSTATE_PTBR_RENDER_LIST_BASE_ADDRESS

Source: PositionCS

Length Bias: 2

The 3DSTATE_PTBR_RENDER_LIST_BASE_ADDRESS command specifies a 4KB aligned base address for storing Render-Lists in PTBR mode. Render-List is an array consisting of starting page offset's corresponding to each of the tiles visibility data. Starting page offsets stored are offsets relative to Page Pool Base Address. The number of entries (array size) in a Render-List is equivalent to the number of tiles in a Tile Pass. Each Tile Pass has a corresponding Render-List populated by POSH pipe and each entry from the Render-List is read by Render Pipe when the corresponding tile is rendered.

Programming Notes

Both POSH pipe and Render pipe use the Render List Base Address programmed in POSH pipe, SW must ensure both POSH and Render pipe are flushed and synchronized prior to programing this command.
This command must be programmed only for POSH pipe and must not be programmed for render pipe. This command is context specific and must be programmed for each of the POSH enabled context. This command gets context save/restored as part of the POSH context.

The Render List Base Address must be set to a valid page if any batch buffer is enabled for the POSH pipe.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 23h 3DSTATE_PTBR_RENDER_LIST_BASE_ADDRESS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 01h Excludes DWord (1,2) Format: =n
1..2	63:12	**Render List Base Address** Format: VIRTUAL_ADDR[63:12]

3DSTATE_PTBR_RENDER_LIST_BASE_ADDRESS		
		Render List Base Address specifies a 4KB aligned base address for storing Render-Lists in PTBR mode.
	11:7	**Reserved** Access: RO Format: MBZ
	6:0	**Render List Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control start for using the Render-List Address.

3DSTATE_PTBR_TILE_PASS_INFO

3DSTATE_PTBR_TILE_PASS_INFO

Source: RenderCS, PositionCS

Length Bias: 2

The 3DSTATE_PTBR_TILE_PASS_INFO command is used to define the required parameters for a Tile Pass in PTBR mode.

Programming Notes

This command must be programmed for both POSH pipe and render pipe. This command is context specific and must be programmed for each of the POSH enabled context. This command gets context save/restored as part of the POSH context by POSH pipe and gets context save/restored as part of the render context by render pipe.

RENDER PIPE ONLY: PIPE_CONTROL DepthStallEnable and DepthCacheFlushEnable are required before the TILE_PASS_INFO.EndofTilePass marker.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 22h 3DSTATE_PTBR_TILE_PASS_INFO Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Format: =n Value / Name 02h: End Of Tile Pass Marker Not Set **[Default]** 0h: End Of Tile Pass Marker Set **Programming Notes** On setting the End Of;Tile Pass Marker, SW must not program the Dword2 and Dword3 of the

command and must accordingly program the Dword count in the header.

1 | 31 **End of Tile Pass**

Format:	Enable

Marker for indicating End of Tile Pass in a command sequence. This bit is always save/restored as 0 in HW CONTEXT.

Value	Name	Description
0	**[Default]**	Not an indicator of End of Tile Pass in a command sequence.
1		Indicates Start of Tile Pass in a command sequence.

Programming Notes

- A Tile Pass must begin with a Start of Tile Pass marker and end with End of Tile Pass marker.
- Start of Tile Pass marker and End of Tile Pass marker must not be set in the same command
- On setting the End of Tile Pass Marker, SW must not program the Dword2 and Dword3 of the command and must accordingly program the Dword count in the header.

30 **Start of Tile Pass**

Format:	Enable

Marker for indicating Start of Tile Pass in a command sequence.

Value	Name	Description
0	**[Default]**	Not an indicator of Start of Tile Pass in a command sequence.
1		Indicates Start of Tile Pass in a command sequence.

Programming Notes

- A Tile Pass must begin with a Start of Tile Pass marker and end with End of Tile Pass marker.
- Start of Tile Pass marker and End of Tile Pass marker must not be set in the same command

29 **Tile Pass Flag Status**

Format:	Enable

This bit is used by hardware to context save/restore the Tile Pass Flag status.

- When set indicates context got switched out is in middle of a Tile Pass,
- when reset indicates context got switched out outside the Tile Pass.

Value	Name	Description
0	**[Default]**	Context got switched out outside the Tile Pass.
1		Context got switched out in middle of a Tile Pass.

3DSTATE_PTBR_TILE_PASS_INFO

		Programming Notes SW must never set this bit.
	28:10	**Reserved** Access: RO Format: MBZ
	9:0	**Tile Count** Format: U10-1 **Description** Tile Count indicates the number of tiles the render region is divided in to, a maximum of 128 tiles is supported by POSH pipe and a maximum of 1024 tiles by Render pipe. A tile count of zero is a special indication stating the whole render region is considered as one single tile of the dimensions of the drawing rectangle. A tile count of 1 indicates the whole render region is divided in to two tiles and tile count of 127 indicates the render region is divided in to 128 tiles. For a Tile Count greater then '0', Tile Rect Array Pointer provides the details for each of the tiles dimensions. **Value / Name / Description** 0h — Default **[Default]** — The whole render region is considered as one single tile of the dimensions of the drawing rectangle. 7Fh — POSH Max Value — The render region is divided in to 128 tiles. 3FFh — Render Max Value — The render region is divided in to 1024 tiles. **Programming Notes** "Tile Count" value programmed must be same in the 3DSTATE_PTBR_TILE_PASS_INFO command programmed for "Start of Tile Pass" and for "End of Tile Pass".
2	31:6	**Tile Rect Array Pointer** Format: DynamicStateOffset[31:6]TILE_RECT Specifies the 64-byte aligned offset pointing to the element zero of an array, each element is a TILE_RECT state. This offset is relative to the Dynamic State Base Address. Each TILE_RECT state is two dwords in size comprising {Y Max, X Max, Y Min, X Min} of the tile. Number of TILE_RECT states in the array is equivalent to the tile count. Example: For a Tile count of 63, 64 TILE_RECT states are programmed in consecutive memory locations constituting the array, i.e 64 consecutive qwords (qword-64bits) of data in the memory, with starting qword corresponding to the TILE_RECT state of tile0 and the last qword corresponding to tile-63.
	5:0	**Reserved** Access: RO Format: MBZ

3DSTATE_PTBR_TILE_PASS_INFO

3	31:6	**Render List Pointer** Format: RenderListOffset[31:6] Specifies the 64byte offset pointing to Render-List of the current Tile Pass. This offset is relative to PTBR_RENDER_LIST_BASE_ADDRESS. Render-List is an array consisting of starting page offset's corresponding to each of the tiles visibility data. Starting page offsets stored are 16bit offsets relative to PTBR_PAGE_POOL_BASE_ADDRESS. The number of entries (array size) in a Render-List is equivalent to the tile count of the current Tile Pass.
	5:2	**Reserved** Access: RO Format: MBZ
	1	**Reserved** Access: RO Format: MBZ

	0	**Initialize Render List**
		Format: Enable
		On parsing this command with Initialize Render List set, POSH pipe allocates a page for each of the tile from the Free-List to which respective visibility data will be outputted. The page offsets corresponding to these pages are written to the Render-List. POSH pipe doesn't parse the next command in sequence until the Render-List is populated. Each entry from the Render-List is read by Render Pipe when the corresponding tile is rendered. Example: For a Tile count of 63, POSH pipe will write to 64 consecutive Words (2bytes) to the Render-List, where in word-0 belongs to Tile-0, word1 belongs to Tile-1 word-63 belongs to Tile-63.
		This bit is ignored by RenderCS on executing 3DSTATE_PTBR_TILE_PASS_INFO command.
		Programming Notes
		SW must ensure Render-List is populated by POSH pipe before it gets read by the Render pipe. This can be achieved by using semaphores in render pipe command sequence to wait for POSH pipe to complete Render-List update.
		• SW must always set this bit to '1'.
		• SW must Initialize Render List only once in a Tile Pass.
		• SW must ensure to program the below listed commands prior to programming 3DSTATE_PTBR_TILE_PASS_INFO with "Initialize Render List bit set.
		  • 3DSTATE_PTBR_PAGE_POOL_BASE_ADDRESS
		  • 3DSTATE_PTBR_FREE_LIST
		  • 3DSTATE_PTBR_RENDER_LIST
		This bit is always save/restored as 0 in HW CONTEXT.

3DSTATE_PTBR_TILE_SELECT

3DSTATE_PTBR_TILE_SELECT

Source: RenderCS

Length Bias: 2

The 3DSTATE_PTBR_TILE_SELECT command is programmed in render engine to define the required parameters for a tile to be rendered in POSH Tile Base Rendering mode (PTBR).
This command must be programmed only for render pipe and must not be programmed for POSH pipe. This command is context specific and must be programmed for each of the POSH enabled context. This command gets context save/restored by render engine as part of the render context.

Programming Notes

SW must ensure Render-List is populated by POSH pipe before it gets read by the Render pipe.
This can be achieved by using semaphores in render pipe command sequence to wait for POSH pipe to complete Render-List update prior to programming of 3DSTATE_PTBR_TILE_SELECT command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 6Bh 3DSTATE_PTBR_TILE_SELECT Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Format: =n
1	31:0	**PTBR Tile Select State Body** Format: 3DSTATE_PTBR_TILE_SELECT_BODY

3DSTATE_PUSH_CONSTANT_ALLOC_DS

3DSTATE_PUSH_CONSTANT_ALLOC_DS

Source: RenderCS

Length Bias: 2

This command sets up the URB configuration for DS Push Constant Buffer.

Programming Notes

Programming Restriction:

- The sum of the Constant Buffer Offset and the Constant Buffer Size may not exceed the maximum value of the Constant Buffer Size.
- The sum of the constant length of the push constants must be equal or smaller than the size of the allocated space in the URB including the buffering for half cachelines. See **Push Constant URB Allocation section for more details**.
- The Domain Shader Push Constants state must be programmed prior to any state is committed into the pipeline or any preemptible command(i.e. prior to PIPE_CONTROL, 3DPRIMITIVE, 3DMESH or COMPUTE_WALKER)after programming the 3DSTATE_PUSH_CONSTANT_ALLOC_DS.

When gather at set shader is disabled, Push Constant command is committed when 3DPRIMITIVE command is parsed.
When gather at set shader is enabled, commit point on the Push Constant command is a 3DSTATE_BINDING_TABLE_POINTER command.

The 3DSTATE_BINDING_TABLE_POINTER must be reprogrammed after 3DSTATE_PUSH_CONST_ALLOC command and the reprogramming of the push constants prior to the next 3DPRIMITIVE if gather at set shader is enabled to ensure the new programming is committed.

DWord	Bit	Description
0 **Programming Notes:** Workaround : This command must be followed by a PIPE_CONTROL with CS Stall bit set.,	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 14h 3DSTATE_PUSH_CONSTANT_ALLOC_DS Format: OpCode
	15:8	**Reserved** Access: RO

3DSTATE_PUSH_CONSTANT_ALLOC_DS

		Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n
1	31:21	**Reserved** Access: RO Format: MBZ
	20:16	**Constant Buffer Offset** Format: U5 Specifies the offset of the DS constant buffer into the URB. **Value** [0,31] **Name** (0KB - 31KB) Increments of 2KB
	15:6	**Reserved** Access: RO Format: MBZ
	5:0	**Constant Buffer Size** Format: U6 Specifies the size of the DS constant buffer. This value will determine the amount of data the command stream can pre-fetch before the buffer is full. Value of zero is only valid when constants are not enabled for DS. **Value** [0,32] **Name** (0KB - 32KB) Increments of 2KB

3DSTATE_PUSH_CONSTANT_ALLOC_GS

3DSTATE_PUSH_CONSTANT_ALLOC_GS

Source: RenderCS

Length Bias: 2

This command sets up the URB configuration for GS Push Constant Buffer.

Programming Notes

- The sum of the Constant Buffer Offset and the Constant Buffer Size may not exceed the maximum value of the Constant Buffer Size.
- The sum of the constant length of the push constants must be equal or smaller than the size of the allocated space in the URB including the buffering for half cachelines. See **Push Constant URB Allocation section for more details**.
- The Geometry Shader Push Constants state must be programmed prior to any state is committed into the pipeline or any preemptible command(i.e. prior to PIPE_CONTROL, 3DPRIMITIVE, 3DMESH or COMPUTE_WALKER)after programming the 3DSTATE_PUSH_CONSTANT_ALLOC_GS.

When gather at set shader is disabled, Push Constant command is committed when 3DPRIMITIVE command is parsed.
When gather at set shader is enabled, commit point on the Push Constant command is a 3DSTATE_BINDING_TABLE_POINTER command.

The 3DSTATE_BINDING_TABLE_POINTER must be reprogrammed after 3DSTATE_PUSH_CONST_ALLOC command and the reprogramming of the push constants prior to the next 3DPRIMITIVE if gather at set shader is enabled to ensure the new programming is committed.

Workaround

Workaround : This command must be followed by a PIPE_CONTROL with CS Stall bit set.,

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 15h 3DSTATE_PUSH_CONSTANT_ALLOC_GS Format: OpCode
	15:8	**Reserved**

3DSTATE_PUSH_CONSTANT_ALLOC_GS

		Access: RO
		Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2

Value	Name	Description
0h	3DSTATE_PUSH_CONSTANT_ALLOC_GS **[Default]**	Excludes DWord (0,1)

1	31:21	**Reserved** Access: RO Format: MBZ
	20:16	**Constant Buffer Offset** Format: U5 Specifies the offset of the GS constant buffer into the URB.

Value	Name
[0,31]	(0KB - 31KB) Increments of 2KB

	15:6	**Reserved** Access: RO Format: MBZ
	5:0	**Constant Buffer Size** Format: U6 Specifies the size of the GS constant buffer. This value will determine the amount of data the command stream can pre-fetch before the buffer is full. Value of zero is only valid when constants are not enabled for GS.

Value	Name
[0,32]	(0KB - 32KB) Increments of 2KB

3DSTATE_PUSH_CONSTANT_ALLOC_HS

3DSTATE_PUSH_CONSTANT_ALLOC_HS

Source: RenderCS

Length Bias: 2

This command sets up the URB configuration for HS Push Constant Buffer.

Programming Notes

Programming Restriction:

- The sum of the Constant Buffer Offset and the Constant Buffer Size may not exceed the maximum value of the Constant Buffer Size.
- The sum of the constant length of the push constants must be equal or smaller than the size of the allocated space in the URB including the buffering for half cachelines. See **Push Constant URB Allocation section for more details**.
- The Hull Shader Push Constants state must be programmed prior to any state is committed into the pipeline or any preemptible command(i.e. prior to PIPE_CONTROL, 3DPRIMITIVE, 3DMESH or COMPUTE_WALKER)after programming the 3DSTATE_PUSH_CONSTANT_ALLOC_HS.

When gather at set shader is disabled, Push Constant command is committed when 3DPRIMITIVE command is parsed.
When gather at set shader is enabled, commit point on the Push Constant command is a 3DSTATE_BINDING_TABLE_POINTER command.

The 3DSTATE_BINDING_TABLE_POINTER must be reprogrammed after 3DSTATE_PUSH_CONST_ALLOC command and the reprogramming of the push constants prior to the next 3DPRIMITIVE if gather at set shader is enabled to ensure the new programming is committed.

Workaround

Workaround : This command must be followed by a PIPE_CONTROL with CS Stall bit set.,

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 13h 3DSTATE_PUSH_CONSTANT_ALLOC_HS Format: OpCode

3DSTATE_PUSH_CONSTANT_ALLOC_HS

	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n
1	31:21	**Reserved** Access: RO Format: MBZ
	20:16	**Constant Buffer Offset** Format: U5 Specifies the offset of the HS constant buffer into the URB. **Value** [0,31] — **Name** (0KB - 31KB) Increments of 2KB
	15:6	**Reserved** Access: RO Format: MBZ
	5:0	**Constant Buffer Size** Format: U6 Specifies the size of the HS constant buffer. This value will determine the amount of data the command stream can pre-fetch before the buffer is full. Value of zero is only valid when constants are not enabled for HS. **Value** [0,32] — **Name** (0KB - 32KB) Increments of 2KB

3DSTATE_PUSH_CONSTANT_ALLOC_PS

3DSTATE_PUSH_CONSTANT_ALLOC_PS

Source: RenderCS

Length Bias: 2

This command sets up the URB configuration for PS Push Constant Buffer.

Programming Notes

Restriction:

- The sum of the Constant Buffer Offset and the Constant Buffer Size may not exceed the maximum value of the Constant Buffer Size.
- The sum of the constant length of the push constants must be equal or smaller than the size of the allocated space in the URB including the buffering for half cachelines. See **Push Constant URB Allocation section for more details**.
- The Pixel Shader Push Constants state must be programmed prior to any state is committed into the pipeline or any preemptible command(i.e. prior to PIPE_CONTROL, 3DPRIMITIVE, 3DMESH or COMPUTE_WALKER)after programming the 3DSTATE_PUSH_CONSTANT_ALLOC_PS.

When gather at set shader is disabled, Push Constant command is committed when 3DPRIMITIVE command is parsed.
When gather at set shader is enabled, commit point on the Push Constant command is a 3DSTATE_BINDING_TABLE_POINTER command.

The 3DSTATE_BINDING_TABLE_POINTER must be reprogrammed after 3DSTATE_PUSH_CONST_ALLOC command and the reprogramming of the push constants prior to the next 3DPRIMITIVE if gather at set shader is enabled to ensure the new programming is committed.

Workaround

Workaround : This command must be followed by a PIPE_CONTROL with CS Stall bit set.,

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 16h 3DSTATE_PUSH_CONSTANT_ALLOC_PS Format: OpCode

3DSTATE_PUSH_CONSTANT_ALLOC_PS

	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**Dword Length** Default Value: 0h Excludes Dword (0,1) Format: =n
1	31:21	**Reserved** Access: RO Format: MBZ
	20:16	**Constant Buffer Offset** Format: U5 Specifies the offset of the PS constant buffer into the URB. **Value** [0,31] — **Name** (0KB - 31KB) Increments of 2KB
	15:6	**Reserved** Access: RO Format: MBZ
	5:0	**Constant Buffer Size** Format: U6 Specifies the size of the PS constant buffer. This value will determine the amount of data the command stream can pre-fetch before the buffer is full. Value of zero is only valid when constants are not enabled for PS. **Value** [0,32] — **Name** (0KB - 32KB) Increments of 2KB

3DSTATE_PUSH_CONSTANT_ALLOC_VS

3DSTATE_PUSH_CONSTANT_ALLOC_VS

Source: RenderCS

Length Bias: 2

This command sets up the URB configuration for VS Push Constant Buffer.

Programming Notes

Programming Restriction:

- The sum of the Constant Buffer Offset and the Constant Buffer Size may not exceed the maximum value of the Constant Buffer Size.
- The sum of the constant length of the push constants must be equal or smaller than the size of the allocated space in the URB including the buffering for half cachelines. See **Push Constant URB Allocation section for more details**.
- The VertexShader Push Constants state must be programmed prior to any state is committed into the pipeline or any preemptible command(i.e. prior to PIPE_CONTROL, 3DPRIMITIVE, 3DMESH or COMPUTE_WALKER)after programming the 3DSTATE_PUSH_CONSTANT_ALLOC_VS.

When executed in the POCS command stream, this command programs state for the VSR stage of the POCS pipeline. The offset & size of the VSR allocation is relative to the 32KB of URB reserved for POCS pipe Push Constant Buffers, vs. the 32KB used by the RCS pipe.

When gather at set shader is disabled, programmed constants are committed when 3DPRIMITIVE command is parsed.
When gather at set shader is enabled, commit point of the constants programmed area 3DSTATE_BINDING_TABLE_POINTER command.

The 3DSTATE_BINDING_TABLE_POINTER must be reprogrammed after 3DSTATE_PUSH_CONST_ALLOC command and the reprogramming of the push constants prior to the next 3DPRIMITIVE if gather at set shader is enabled to ensure the new programming is committed.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 12h 3DSTATE_PUSH_CONSTANT_ALLOC_VS Format: OpCode

3DSTATE_PUSH_CONSTANT_ALLOC_VS

	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n
1	31:21	**Reserved** Access: RO Format: MBZ
	20:16	**Constant Buffer Offset** Format: U5 Specifies the offset of the VS constant buffer into the URB. **Value** / **Name** [0,31] (0KB - 31KB) Increments of 2KB **Programming Notes** When executed from the POCS pipe, the offset is relative to the VSR_PUSH_CONSTANT_BASE (MMIO offset e518)region reserved for POCS pipe Push Constants.
	15:6	**Reserved** Access: RO Format: MBZ
	5:0	**Constant Buffer Size** Format: U6 Specifies the size of the VS constant buffer. This value will determine the amount of data the command stream can pre-fetch before the buffer is full. Value of zero is only valid when constants are not enabled for VS. **Value** / **Name** [0,32] (0KB - 32KB) Increments of 2KB

3DSTATE_RASTER

3DSTATE_RASTER

Source: RenderCS

Length Bias: 2

Restriction

When executed in the POCS command stream, this command programs the raster state for CLR and SFR stages of the POCS pipeline

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 50h 3DSTATE_RASTER Format: OpCode
	15:14	**Reserved** Access: RO Format: MBZ
	13	**Raster State Modify Disable** Format: Disable When this bit is set, the following fields will be ignored: • SubPixel Aligned Quad Rasterization Enable • Line/Point Conservative Rasterization Enable • Conservative Rasterization Enable • API Mode • Forced Sample Count • Force Multisampling • Smooth Point Enable • DX Multisample Rasterization Enable • DX Multisample Rasterization Mode • Global Depth Offset Enable Solid • Global Depth Offset Enable Wireframe

3DSTATE_RASTER

		• Global Depth Offset Enable Point • Antialiasing Enable • Scissor Rectangle Enable
	12	**Front Winding Modify Disable** Format: Disable When this bit is set, the following fields will be ignored: • Front Winding
	11	**Cull Mode Modify Disable** Format: Disable When this bit is set, the following fields will be ignored: • Cull Mode
	10	**Fill Mode Modify Disable** Format: Disable When this bit is set, the following fields will be ignored: • Front Face Fill Mode • Back Face Fill Mode
	9	**Viewport Z Clip Test Enable Modify Disable** Format: Disable When this bit is set, the following fields will be ignored: • Viewport Z Far Clip Test Enable • Viewport Z Near Clip Test Enable
	8	**Global Depth Offset Modify Disable** Format: Disable When this bit is set, the following fields will be ignored: • Global Depth Offset Constant • Global Depth Offset Scale • Global Depth Offset Clamp
	7:0	**DWord Length** Default Value: 03h Excludes DWord (0,1) Format: =n Total Length - 2
1..4	127:0	**Raster State Body** Format: 3DSTATE_RASTER_BODY

3DSTATE_SAMPLE_MASK

3DSTATE_SAMPLE_MASK

Source: RenderCS

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 18h 3DSTATE_SAMPLE_MASK Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**Dword Length** Default Value: 0h Excludes Dword (0,1) Format: =n
1	31:0	**Sample Mask State Body** Format: 3DSTATE_SAMPLE_MASK_BODY

3DSTATE_SAMPLE_PATTERN

3DSTATE_SAMPLE_PATTERN

Source: RenderCS

Length Bias: 2

The 3DSTATE_SAMPLE_PATTERN command is used to specify the sample offsets for all multisample sample modes. The set of offset used will be selected based on the multisample mode. This is non-pipelined state.

Programming Notes

When programming the sample offsets (for NUMSAMPLES_4 or _8 and MSRASTMODE_xxx_PATTERN), the order of the samples 0 to 3 (or 7 for 8X, or 15 for 16X) must have monotonically increasing distance from the pixel center. This is required to get the correct centroid computation in the device.

Restriction : When executed in the POCS command stream, this command programs the multisample pattern state for the CLR and SFR stage of the POCS pipeline

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 1Ch 3DSTATE_SAMPLE_PATTERN Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 7 Excludes Dword (0,1) Format: =n
1	31:28	**16x Sample3 X Offset** Format: U0.4 Subpixel X offset of Sample 3 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]

3DSTATE_SAMPLE_PATTERN

	27:24	**16x Sample3 Y Offset** Format: U0.4 Subpixel Y offset of Sample 3 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	23:20	**16x Sample2 X Offset** Format: U0.4 Subpixel X offset of Sample 2 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	19:16	**16x Sample2 Y Offset** Format: U0.4 Subpixel Y offset of Sample 2 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES_4 , _8 or _16. Range: [0,0.9375]
	15:12	**16x Sample1 X Offset** Format: U0.4 Subpixel X offset of Sample 1 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES_2, _4 , _8 or _16. Range: [0,0.9375]
	11:8	**16x Sample1 Y Offset** Format: U0.4 Subpixel Y offset of Sample 1 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	7:4	**16x Sample0 X Offset** Format: U0.4 Subpixel X offset of Sample 0 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	3:0	**16x Sample0 Y Offset** Format: U0.4 Subpixel Y offset of Sample 0 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]

3DSTATE_SAMPLE_PATTERN

2	31:28	**16x Sample7 X Offset** Format: U0.4 Subpixel X offset of Sample 7 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	27:24	**16x Sample7 Y Offset** Format: U0.4 Subpixel Y offset of Sample 7 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	23:20	**16x Sample6 X Offset** Format: U0.4 Subpixel X offset of Sample 6 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES__8 or _16. Range: [0,0.9375]
	19:16	**16x Sample6 Y Offset** Format: U0.4 Subpixel Y offset of Sample 6 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	15:12	**16x Sample5 X Offset** Format: U0.4 Subpixel X offset of Sample 5 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	11:8	**16x Sample5 Y Offset** Format: U0.4 Subpixel Y offset of Sample 5 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES__8 or _16. Range: [0,0.9375]
	7:4	**16x Sample4 X Offset** Format: U0.4 Subpixel X offset of Sample 4 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES__8 or _16. Range: [0,0.9375]

3DSTATE_SAMPLE_PATTERN		
	3:0	**16x Sample4 Y Offset** Format: U0.4 Subpixel Y offset of Sample 4 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
3	31:28	**16x Sample11 X Offset** Format: U0.4 Subpixel X offset of Sample 11 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	27:24	**16x Sample11 Y Offset** Format: U0.4 Subpixel Y offset of Sample 11 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES _16. Range: [0,0.9375]
	23:20	**16x Sample10 X Offset** Format: U0.4 Subpixel X offset of Sample 10 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	19:16	**16x Sample10 Y Offset** Format: U0.4 Subpixel Y offset of Sample 10 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES _16 Range: [0,0.9375]
	15:12	**16x Sample9 X Offset** Format: U0.4 Subpixel X offset of Sample 9 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	11:8	**16x Sample9 Y Offset** Format: U0.4 Subpixel Y offset of Sample 9 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]

3DSTATE_SAMPLE_PATTERN

	7:4	**16x Sample8 X Offset** Format: U0.4 Subpixel X offset of Sample 8 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	3:0	**16x Sample8 Y Offset** Format: U0.4 Subpixel Y offset of Sample 8 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
4	31:28	**16x Sample15 X Offset** Format: U0.4 Subpixel X offset of Sample 15 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	27:24	**16x Sample15 Y Offset** Format: U0.4 Subpixel Y offset of Sample 15 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	23:20	**16x Sample14 X Offset** Format: U0.4 Subpixel X offset of Sample 14 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES _16. Range: [0,0.9375]
	19:16	**16x Sample14 Y Offset** Format: U0.4 Subpixel Y offset of Sample 14 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES _16 Range: [0,0.9375]
	15:12	**16x Sample13 X Offset** Format: U0.4 Subpixel X offset of Sample 13 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]

3DSTATE_SAMPLE_PATTERN

	11:8	**16x Sample13 Y Offset** Format: U0.4 Subpixel Y offset of Sample 13 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
	7:4	**16x Sample12 X Offset** Format: U0.4 Subpixel X offset of Sample 12 relative to the UL pixel origin. Valid only when NUMRASTSAMPLES_16. Range: [0,0.9375]
	3:0	**16x Sample12 Y Offset** Format: U0.4 Subpixel Y offset of Sample 12 relative to the UL pixel origin for 16x mode. Range: [0,0.9375]
5	31:28	**8x Sample7 X Offset** Format: U0.4 Subpixel X offset of Sample 7 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	27:24	**8x Sample7 Y Offset** Format: U0.4 Subpixel Y offset of Sample 7 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	23:20	**8x Sample6 X Offset** Format: U0.4 Subpixel X offset of Sample 6 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	19:16	**8x Sample6 Y Offset** Format: U0.4 Subpixel Y offset of Sample 6 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]

3DSTATE_SAMPLE_PATTERN

	15:12	**8x Sample5 X Offset** Format: U0.4 Subpixel X offset of Sample 5 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	11:8	**8x Sample5 Y Offset** Format: U0.4 Subpixel Y offset of Sample 5 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	7:4	**8x Sample4 X Offset** Format: U0.4 Subpixel X offset of Sample 4 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	3:0	**8x Sample4 Y Offset** Format: U0.4 Subpixel Y offset of Sample 4 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
6	31:28	**8x Sample3 X Offset** Format: U0.4 Subpixel X offset of Sample 3 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	27:24	**8x Sample3 Y Offset** Format: U0.4 Subpixel Y offset of Sample 3 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	23:20	**8x Sample2 X Offset** Format: U0.4 Subpixel X offset of Sample 2 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	19:16	**8x Sample2 Y Offset** Format: U0.4

3DSTATE_SAMPLE_PATTERN

		Subpixel Y offset of Sample 2 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	15:12	**8x Sample1 X Offset** Format: U0.4 Subpixel X offset of Sample 1 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	11:8	**8x Sample1 Y Offset** Format: U0.4 Subpixel Y offset of Sample 1 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	7:4	**8x Sample0 X Offset** Format: U0.4 Subpixel X offset of Sample 0 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
	3:0	**8x Sample0 Y Offset** Format: U0.4 Subpixel Y offset of Sample 0 relative to the UL pixel origin for 8x mode. Range: [0,0.9375]
7	31:28	**4x Sample3 X Offset** Format: U0.4 Subpixel X offset of Sample 3 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	27:24	**4x Sample3 Y Offset** Format: U0.4 Subpixel Y offset of Sample 3 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	23:20	**4x Sample2 X Offset** Format: U0.4 Subpixel X offset of Sample 2 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]

3DSTATE_SAMPLE_PATTERN

	19:16	**4x Sample2 Y Offset** Format: U0.4 Subpixel Y offset of Sample 2 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	15:12	**4x Sample1 X Offset** Format: U0.4 Subpixel X offset of Sample 1 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	11:8	**4x Sample1 Y Offset** Format: U0.4 Subpixel Y offset of Sample 1 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	7:4	**4x Sample0 X Offset** Format: U0.4 Subpixel X offset of Sample 0 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
	3:0	**4x Sample0 Y Offset** Format: U0.4 Subpixel Y offset of Sample 0 relative to the UL pixel origin for 4x mode. Range: [0,0.9375]
8	31:24	**Reserved** Access: RO Format: MBZ
	23:20	**1x Sample0 X Offset** Format: U0.4 Subpixel X offset of Sample 0 relative to the UL pixel origin for 1x mode. Range: [0,0.9375]
	19:16	**1x Sample0 Y Offset** Format: U0.4 Subpixel Y offset of Sample 0 relative to the UL pixel origin for 1x mode. Range: [0,0.9375]

3DSTATE_SAMPLE_PATTERN

	15:12	**2x Sample1 X Offset** Format: U0.4 Subpixel X offset of Sample 1 relative to the UL pixel origin for 2x mode. Range: [0,0.9375]
	11:8	**2x Sample1 Y Offset** Format: U0.4 Subpixel Y offset of Sample 1 relative to the UL pixel origin for 2x mode. Range: [0,0.9375]
	7:4	**2x Sample0 X Offset** Format: U0.4 Subpixel X offset of Sample 0 relative to the UL pixel origin for 2x mode. Range: [0,0.9375]
	3:0	**2x Sample0 Y Offset** Format: U0.4 Subpixel Y offset of Sample 0 relative to the UL pixel origin for 2x mode. Range: [0,0.9375]

3DSTATE_SAMPLER_STATE_POINTERS_DS

3DSTATE_SAMPLER_STATE_POINTERS_DS

Source: RenderCS

Length Bias: 2

The 3DSTATE_SAMPLER_STATE_POINTERS_DS command is used to define the location of DS SAMPLER_STATE table. Only some of the fixed functions utilize sampler state tables.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 2Dh 3DSTATE_SAMPLER_STATE_POINTERS_DS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**Sampler State Pointers State Body** Format: 3DSTATE_SAMPLER_STATE_POINTERS_BODY

3DSTATE_SAMPLER_STATE_POINTERS_GS

3DSTATE_SAMPLER_STATE_POINTERS_GS

Source: RenderCS

Length Bias: 2

The 3DSTATE_SAMPLER_STATE_POINTERS_GS command is used to define the location of GS SAMPLER_STATE table. Only some of the fixed functions utilize sampler state tables.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 2Eh 3DSTATE_SAMPLER_STATE_POINTERS_GS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**Sampler State Pointers State Body** Format: 3DSTATE_SAMPLER_STATE_POINTERS_BODY

3DSTATE_SAMPLER_STATE_POINTERS_HS

3DSTATE_SAMPLER_STATE_POINTERS_HS

Source: RenderCS

Length Bias: 2

The 3DSTATE_SAMPLER_STATE_POINTERS_HS command is used to define the location of HS SAMPLER_STATE table. Only some of the fixed functions utilize sampler state tables.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 2Ch 3DSTATE_SAMPLER_STATE_POINTERS_HS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**Sampler State Pointers State Body** Format: 3DSTATE_SAMPLER_STATE_POINTERS_BODY

3DSTATE_SAMPLER_STATE_POINTERS_PS

3DSTATE_SAMPLER_STATE_POINTERS_PS

Source: RenderCS

Length Bias: 2

The 3DSTATE_SAMPLER_STATE_POINTERS_PS command is used to define the location of PS SAMPLER_STATE table. Only some of the fixed functions utilize sampler state tables.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 2Fh 3DSTATE_SAMPLER_STATE_POINTERS_PS Format: OpCode
	15	**POSH-Enabled Render State Optimization Enable 1**
	14:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**Sampler State Pointers State Body** Format: 3DSTATE_SAMPLER_STATE_POINTERS_BODY

3DSTATE_SAMPLER_STATE_POINTERS_VS

3DSTATE_SAMPLER_STATE_POINTERS_VS

Source: RenderCS

Length Bias: 2

The 3DSTATE_SAMPLER_STATE_POINTERS_VS command is used to define the location of VS SAMPLER_STATE table. Only some of the fixed functions utilize sampler state tables.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 2Bh 3DSTATE_SAMPLER_STATE_POINTERS_VS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**Sampler State Pointers State Body** Format: 3DSTATE_SAMPLER_STATE_POINTERS_BODY

3DSTATE_SBE

3DSTATE_SBE		
Source: RenderCS Length Bias: 2		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 1Fh 3DSTATE_SBE Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 04h Excludes DWord (0,1) Format: =n Total Length - 2
1..5	159:0	**SBE State Body** Format: 3DSTATE_SBE_BODY

3DSTATE_SBE_SWIZ

3DSTATE_SBE_SWIZ		
Source:	RenderCS	
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 51h 3DSTATE_SBE_SWIZ Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 9h Excludes DWord (0,1) Format: =n Total Length - 2
1..10	319:0	**SBE SWIZ State Body** Format: 3DSTATE_SBE_SWIZ_BODY

3DSTATE_SCISSOR_STATE_POINTERS

3DSTATE_SCISSOR_STATE_POINTERS

Source: RenderCS

Length Bias: 2

The 3DSTATE_SCISSOR_STATE_POINTERS command is used to define the location of the indirect SCISSOR_RECT state.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 0Fh 3DSTATE_SCISSOR_STATE_POINTERS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**Scissor State Pointers Body** Format: 3DSTATE_SCISSOR_STATE_POINTERS_BODY

3DSTATE_SF

3DSTATE_SF

Source: RenderCS

Length Bias: 2

Restriction

When executed in the POCS command stream, this command programs the state for the SFR stage of the POCS pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 13h 3DSTATE_SF Format: OpCode
	15:11	**Reserved** Access: RO Format: MBZ
	10	**SF State Modify Disable** Format: Disable When this bit is set, the following fields will be ignored: • Legacy Global Depth Bias Enable • Statistics Enable • Viewport Transform Enable • Fast Scissor Clip Disable • Line End Cap Antialiasing Region Width • Zero Pixel Triangle Filter Disable • 2x2 Pixel Triangle Filter Disable • Last Pixel Enable • Triangle Strip/List Provoking Vertex Select • Line Strip/List Provoking Vertex Select • Triangle Fan Provoking Vertex Select

<table>
<tr><th colspan="3">3DSTATE_SF</th></tr>
<tr><td></td><td></td><td>

AA Line Distance Mode
Smooth Point Enable
Vertex Sub Pixel Precision Select
Point Width Source

</td></tr>
<tr><td></td><td>9</td><td>Line Width Modify Disable
<table>
<tr><td>Format:</td><td>Disable</td></tr>
</table>
When this bit is set, the following fields will be ignored:

Line Width

</td></tr>
<tr><td></td><td>8</td><td>Point Width Modify Disable
<table>
<tr><td>Format:</td><td>Disable</td></tr>
</table>
When this bit is set, the following fields will be ignored:

Point Width

</td></tr>
<tr><td></td><td>7:0</td><td>DWord Length
<table>
<tr><td>Default Value:</td><td>2h Excludes DWord (0,1)</td></tr>
<tr><td>Format:</td><td>=n</td></tr>
</table>
Total Length - 2
</td></tr>
<tr><td>1..3</td><td>95:0</td><td>SF State Body
<table>
<tr><td>Format:</td><td>3DSTATE_SF_BODY</td></tr>
</table>
</td></tr>
</table>

3DSTATE_SLICE_TABLE_STATE_POINTERS

3DSTATE_SLICE_TABLE_STATE_POINTERS

Source: RenderCS

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: Opcode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: Opcode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 20h 3DSTATE_SLICE_TABLE_STATE_POINTERS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n
1	31:6	**Slice Hash Table State Pointer** Format: DynamicStateOffset[31:6]SLICE_HASH_TABLE Specifies the 64-byte aligned offset of the SLICE_HASH_TABLE. This offset is relative to the **Dynamic State Base Address**.
	5:1	**Reserved** Access: RO Format: MBZ
	0	**Slice Hash State Pointer Valid** Format: Enable This bit, if set, indicates that the SLICE_HASH_TABLE pointer has changed and new state needs to be fetched. Value / Name 0h Disable 1h Enable

3DSTATE_SO_BUFFER_INDEX_0

3DSTATE_SO_BUFFER_INDEX_0

Source: RenderCS

Length Bias: 2

This command is to program the SO buffer addresses and Attributes for Index 0.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 60h 3DSTATE_SO_BUFFER_INDEX_0 Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 6h Excludes DWord (0,1) Format: =n Total Length - 2
1..7	223:0	**SO Buffer Index State Body** Format: 3DSTATE_SO_BUFFER_INDEX_BODY

3DSTATE_SO_BUFFER_INDEX_1

3DSTATE_SO_BUFFER_INDEX_1

Source: RenderCS

Length Bias: 2

This command is to program the SO buffer addresses and Attributes for Index 1.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 61h 3DSTATE_SO_BUFFER_INDEX_1 Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 6h Excludes DWord (0,1) Format: =n Total Length - 2
1..7	223:0	**SO Buffer Index State Body** Format: 3DSTATE_SO_BUFFER_INDEX_BODY

3DSTATE_SO_BUFFER_INDEX_2

3DSTATE_SO_BUFFER_INDEX_2		
Source:	RenderCS	
Length Bias:	2	
This command is to program the SO buffer addresses and Attributes for Index 2.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 62h 3DSTATE_SO_BUFFER_INDEX_2 Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 6h Excludes DWord (0,1) Format: =n Total Length - 2
1..7	223:0	**SO Buffer Index State Body** Format: 3DSTATE_SO_BUFFER_INDEX_BODY

3DSTATE_SO_BUFFER_INDEX_3

3DSTATE_SO_BUFFER_INDEX_3

Source: RenderCS

Length Bias: 2

This command is to program the SO buffer addresses and Attributes for Index 3.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 63h 3DSTATE_SO_BUFFER_INDEX_3 Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 6h Excludes DWord (0,1) Format: =n Total Length - 2
1..7	223:0	**SO Buffer Index State Body** Format: 3DSTATE_SO_BUFFER_INDEX_BODY

3DSTATE_SO_BUFFER

3DSTATE_SO_BUFFER

Source: RenderCS

Length Bias: 2

Programming Notes

Foreach SO Buffer, the 3DSTATE_SO_BUFFER must only be sent once prior to each 3DPRIMITIVE command.

Workaround

Workaround : This command must be followed by a PIPE_CONTROL with CS Stall bit set.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 18h 3DSTATE_SO_BUFFER Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 6h Excludes DWord (0,1) Format: =n Total Length - 2
1	31	**SO Buffer Enable** Format: Enable If set, stream output to SO Buffer is enabled, , if 3DSTATE_STREAMOUT::SO Function ENABLE is also enabled..If clear, the SO Buffer is considered "not bound" and effectively treated as a zero-length buffer for the purposes of SO output and overflow detection. If an enabled stream's Stream to Buffer Selects includes this buffer it is by definition an overflow condition. That stream will cause no writes to occur, and only SO_PRIM_STORAGE_NEEDED[<stream>] will increment.

3DSTATE_SO_BUFFER

	30:29	**SO Buffer Index** Format: U2 Specifies which of the four SO Buffers is being defined.
	28:22	**SO Buffer Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for the SO buffer.
	21	**Stream Offset Write Enable** Format: Enable When set, this field allows the hardware to write SO_WRITE_OFFSET[Buffer#] as specified in the Stream Offset field. **Programming Notes** The field is operates irrespective of whether SO Buffer Enable is set or clear.
	20	**Stream Output Buffer Offset Address Enable** Format: Enable When set, this field allows the hardware to read/write the stream output buffer offset as specified in the "Stream Output Buffer Offset Address" field. **Programming Notes** The field is operates irrespective of whether SO Buffer Enable is set or clear.
	19:0	**Reserved** Access: RO Format: MBZ
2..3	63:2	**Surface Base Address** Format: VIRTUAL_ADDR[63:2] This field specifies the starting address of the buffer in Graphics Memory.
	1:0	**Reserved** Access: RO Format: MBZ
4	31:30	**Reserved** Access: RO Format: MBZ
	29:0	**Surface Size** Format: U30-1 This field specifies the size of buffer in number DWords minus 1 of the buffer in Graphics Memory.

<table>
<tr><th colspan="3">3DSTATE_SO_BUFFER</th></tr>
<tr><td>5..6</td><td>63:2</td><td>Stream Output Buffer Offset Address
<table><tr><td>Format:</td><td>VIRTUAL_ADDR[63:2]</td></tr></table>This field specifies the starting address of the buffer in Graphics Memory where the Stream Output Buffer Offset is stored when all the data has been written. It is also used to fetch the stream Output buffer Offset when needed.</td></tr>
<tr><td></td><td>1:0</td><td>Reserved
<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>7</td><td>31:0</td><td>Stream Offset
This field specifies the Offset in stream output buffer to start at, or whether to append to the end of an existing buffer. The Offset must be DWORD aligned. If Stream Offset is equal to 0xFFFFFFFF then load the value at the Stream Output Buffer Offset address into SO_WRITE_OFFSET[Buffer#]. Otherwise, SO_WRITE_OFFSET[Buffer#] = Stream Offset.</td></tr>
</table>

3DSTATE_SO_DECL_LIST

3DSTATE_SO_DECL_LIST

Source: RenderCS

Length Bias: 2

Workaround
Workaround : This command must be followed by a PIPE_CONTROL with CS Stall bit set.,

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 17h 3DSTATE_SO_DECL_LIST Format: OpCode
	15:9	**Reserved** Access: RO Format: MBZ
	8:0	**DWord Length** Format: =n Value | Name | Description [1,257] | Excludes DWORD (0,1) 0-128 Entries | Value = 2 * (# of SO_DECL quads) + 1
1	31:16	**Reserved** Access: RO Format: MBZ
	15:12	**Stream to Buffer Selects [3]** Format: U4 Identifies to which SO Buffers stream 3 outputs. See Stream To Buffer Selects [0] field description. Value | Name 1xxxb | SO Buffer 3 x1xxb | SO Buffer 2

	xx1xb	SO Buffer 1
	xxx1b	SO Buffer 0

11:8 Stream to Buffer Selects [2]

Format:	U4

Identifies to which SO Buffers stream 2 outputs. See Stream To Buffer Selects [0] field description.

Value	Name
1xxxb	SO Buffer 3
x1xxb	SO Buffer 2
xx1xb	SO Buffer 1
xxx1b	SO Buffer 0

7:4 Stream to Buffer Selects [1]

Format:	U4

Identifies to which SO Buffers stream 1 outputs. See Stream To Buffer Selects [0] field description.

Value	Name
1xxxb	SO Buffer 3
x1xxb	SO Buffer 2
xx1xb	SO Buffer 1
xxx1b	SO Buffer 0

3:0 Stream to Buffer Selects [0]

Format:	U4

Identifies to which SO Buffers stream 0 outputs (irrespective of whether those buffers are enabled via 3DSTATE_STREAMOUT). Software is required to scan the SO_DECL list in order to provide this summary information.Note: For "inactive" streams, software must program this field to all zero (no buffers written to) and the corresponding Num Entries field to zero (no valid SO_DECLs).

Value	Name
1xxxb	SO Buffer 3
x1xxb	SO Buffer 2
xx1xb	SO Buffer 1
xxx1b	SO Buffer 0

2 31:24 Num Entries [3]

Format:	U8

Specifies the number of valid SO_DECL entries for Stream 3. (See notes in Num Entries [0] field description).

Value	Name
[0,128]	entries

3DSTATE_SO_DECL_LIST

	23:16	**Num Entries [2]** Format: U8 Specifies the number of valid SO_DECL entries for Stream 2. (See notes in Num Entries [0] field description). **Value** / **Name** [0,128] entries
	15:8	**Num Entries [1]** Format: U8 Specifies the number of valid SO_DECL entries for Stream 1. (See notes in Num Entries [0] field description). **Value** / **Name** [0,128] entries
	7:0	**Num Entries [0]** Format: U8 Specifies the number of valid SO_DECL entries for Stream 0.Note that the SO_DECLs are programmed in groups of four (one SO_DECL for each of the four streams). Therefore the number of 2-DWord groups of SO_DECLs supplied in this command is derived from the stream(s) with the most valid SO_DECLs. The NumEntries value specific to each stream will indicate how many SO_DECLS are valid for that particular stream. Any trailing invalid SO_DECLs supplied for streams with fewer valid SO_DECLs will be ignored. It is legal to specify Num Entries = 0 for all four streams simultaneously. In this case there will be no SO_DECLs included in the command (only DW 0-2). Note that all Stream to Buffer Selects bits must be zero in this case (as no streams produce output). **Value** / **Name** [0,128] entries
3..n	63:0	**Entry** Format: SO_DECL_ENTRY

3DSTATE_STENCIL_BUFFER

3DSTATE_STENCIL_BUFFER

Source: RenderCS

Length Bias: 2

The stencil buffer surface state is delivered as a pipelined state packet. However, the state change pipelining isn't completely transparent (see restriction below).

WM HW will internally manage the draining pipe and flushing of the caches when this command is issued. The PIPE_CONTROL restrictions are removed.

Programming Notes

If the Stencil surface is not present, SW must set the Surface Type fieldto SURFTYPE_NULL.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 6h 3DSTATE_STENCIL_BUFFER Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 6h Excludes Dword (0,1) Format: =n Excludes DWord(0,1)
1..7	223:0	**Stencil Buffer State Body** Format: 3DSTATE_STENCIL_BUFFER_BODY

3DSTATE_STREAMOUT

3DSTATE_STREAMOUT		
Source: RenderCS Length Bias: 2		
This command contains pipelined state required by the SOL unit.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 1Eh 3DSTATE_STREAMOUT Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 3h Excludes DWord (0,1) Format: =n
1..4	127:0	**Streamout State Body** Format: 3DSTATE_STREAMOUT_BODY

3DSTATE_SUBSLICE_HASH_TABLE

3DSTATE_SUBSLICE_HASH_TABLE

Source: RenderCS

Length Bias: 2

Programmable DualSubSlice hashing control and tables. First DW indicates the mode how the last 4DW are configured and selects the programmable dualsubslice hashing mode for each slice.

Programming Notes

All slice references are physical slice. Instruction must be programmed based on which slices and subslices are enabled.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: Opcode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: Opcode
	26:24	**3D Command Opcode** Default Value: 1h 3DSTATE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 1Fh 3DSTATE_SUBSLICE_HASH_TABLE Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Format: =n Value: Ch — Name: Excludes DWord (0,1) **[Default]**
1	31:30	**TableMode** Format: U2 Value: 0h — Name: Single table **[Default]** — Description: DW2-5 is Table[0] - a single 2-way 128 entry [Y][X] table. Value: 1h — Name: Dual tables — Description: DW2-3 is 'Table[0]' and is 2-way 64 entry [Y][X] table. DW4-5 is 'Table[1]' and is 2-way 64 entry [Y][X] table.

3DSTATE_SUBSLICE_HASH_TABLE

	29:16	**Reserved** Access: RO Format: MBZ
	15:0	**SliceHashCtrl** PerSlice[7:0]SliceHashControl
2..5	127:0	**64 Entry 2-way Tables** Exists If: Instruction[3DSTATE_SUBSLICE_HASH_TABLE][TableMode]==Dual Format: U1[8][8][2] **Description** 2-way pixel hashing tables. Tables are 64-entries:8X,8Y in [Y][X] format. Each entry is a single bit that indicates which sub-slice hardware block the indicated xy pixel block is mapped. pixelhash_id maps to color-pipe. A value of 0 indicates the larger color-pipe, or first enabled color-pipe if both enabled color-pipes are balanced
	127:0	**128 Entry 2-way Table** Exists If: Instruction[3DSTATE_SUBSLICE_HASH_TABLE][TableMode]==Single Format: U1[8][16] 2-way pixel hashing table. Table is 128-entries:16X,8Y in [Y][X] format. Each entry is a single bit that indicates which sub-slice hardware block the indicated xy pixel block is mapped.
6..13	255:0	**64 Entry 3-way Tables** Exists If: Instruction[3DSTATE_SUBSLICE_HASH_TABLE][TableMode]==Dual Format: U2[8][8][2] **Description** 3-way or 4-way pixel hashing tables. Tables are 64-entries:8X,8Y in [Y][X] format. Each entry is two bits that indicates which sub-slice hardware block the indicated xy pixel block is mapped. pixelhash_id maps to color-pipe. A value of 0 indicates the largest color-pipe, or first enabled color-pipe if all enabled color-pipes are balanced. A value of 2 indicates the smallest color-pipe, or last enabled color-pipe if all enabled color-pipes are balanced.
	255:0	**128 Entry 3-way Table** Exists If: Instruction[3DSTATE_SUBSLICE_HASH_TABLE][TableMode]==Single Format: U2[16][8] **Description** 3-way or 4-way pixel hashing table. Table is 128-entries:16X,8Y in [Y][X] format. Each entry is two bits that indicates which sub-slice hardware block the indicated xy pixel block is mapped. pixelhash_id maps to color-pipe. A value of 0 indicates the largest color-pipe, or first enabled color-pipe if all enabled color-pipes are balanced. A value of 2 indicates the smallest color-pipe, or last enabled color-pipe if all enabled color-pipes are balanced.

3DSTATE_TE

3DSTATE_TE

Source: RenderCS

Length Bias: 2

the state used by TE is defined with this inline state packet.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 1Ch 3DSTATE_TE Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Format: =n Value: 3h — Name: Excludes DWord (0,1) **[Default]**
1..4	127:0	**TE State Body** Format: 3DSTATE_TE_BODY

3DSTATE_URB_ALLOC_DS

3DSTATE_URB_ALLOC_DS

Source: RenderCS

Length Bias: 2

When executed from the RCS command stream, this command provides the state variables associated with the URB region used by the DS pipeline stage.

Separate state variables are provided to define the URB region for Slice0 as well as additional URB space for additional slices (if any are enabled). Hardware will use those values to automatically compute the URB allocation within the total URB space based on the number of enabled slices. Software shall ensure that the values programmed do not exceed the URB capacity of a slice. Refer to the L3 allocation and programming guide for valid URB configurations.

Programming Notes

SW shall ensure that the ordering of URB allocations is consistent between the Slice0 and SliceN state settings across all FF stage URB allocations.

SW shall issue allocation state for all FF stages, i.e., 3DSTATE_URB_ALLOC_VS, 3DSTATE_URB_ALLOC_HS, 3DSTATE_URB_ALLOC_DS, and 3DSTATE_URB_ALLOC_GS commands.

If Domain Shader Thread Dispatch is Enabled then the minimum number of handles that must be allocated is 50 URB entries.

SW shall ensure that the URB region specified by this command does not overlap with the push constant allocation in the URB, as defined by the 3DSTATE_PUSH_CONSTANT_ALLOC_VS, 3DSTATE_PUSH_CONSTANT_ALLOC_DS, 3DSTATE_PUSH_CONSTANT_ALLOC_HS, and 3DSTATE_PUSH_CONSTANT_ALLOC_GS commands.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 5Ah 3DSTATE_URB_ALLOC_DS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ

3DSTATE_URB_ALLOC_DS		
	7:0	**DWord Length** Default Value: 1h DWORD_COUNT_n Format: =n
1..2	63:0	**URB Alloc DS State Body** Format: 3DSTATE_URB_ALLOC_DS_BODY

3DSTATE_URB_ALLOC_GS

3DSTATE_URB_ALLOC_GS

Source: RenderCS

Length Bias: 2

When executed from the RCS command stream, this command provides the state variables associated with the URB region used by the GS pipeline stage.

Separate state variables are provided to define the URB region for Slice0 as well as additional URB space for additional slices (if any are enabled). Software shall ensure that the values programmed do not exceed the URB capacity of a slice. Refer to the L3 allocation and programming guide for valid URB configurations.

Programming Notes

SW shall ensure that the ordering of URB allocations is consistent between the Slice0 and SliceN state settings across all FF stage URB allocations.

SW shall issue allocation state for all FF stages, i.e., 3DSTATE_URB_ALLOC_VS, 3DSTATE_URB_ALLOC_HS, 3DSTATE_URB_ALLOC_DS, and 3DSTATE_URB_ALLOC_GS commands.

SW shall ensure that the URB region specified by this command does not overlap with the push constant allocation in the URB, as defined by the 3DSTATE_PUSH_CONSTANT_ALLOC_VS, 3DSTATE_PUSH_CONSTANT_ALLOC_DS, 3DSTATE_PUSH_CONSTANT_ALLOC_HS, and 3DSTATE_PUSH_CONSTANT_ALLOC_GS commands.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 5Bh 3DSTATE_URB_ALLOC_GS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 1h DWORD_COUNT_n Format: =n

3DSTATE_URB_ALLOC_GS			
1..2	63:0	**URB Alloc GS State Body**	
		Format:	3DSTATE_URB_ALLOC_GS_BODY

3DSTATE_URB_ALLOC_HS

3DSTATE_URB_ALLOC_HS

Source: RenderCS

Length Bias: 2

When executed from the RCS command stream, this command provides the state variables associated with the URB region used by the HS pipeline stage.

Separate state variables are provided to define the URB region for Slice0 as well as additional URB space for additional slices (if any are enabled). Hardware will use those values to automatically compute the URB allocation within the total URB space based on the number of enabled slices. Software shall ensure that the values programmed do not exceed the URB capacity of a slice. Refer to the L3 allocation and programming guide for valid URB configurations.

Programming Notes

SW shall ensure that the ordering of URB allocations is consistent between the Slice0 and SliceN state settings across all FF stage URB allocations.

SW shall issue allocation state for all FF stages, i.e., 3DSTATE_URB_ALLOC_VS, 3DSTATE_URB_ALLOC_HS, 3DSTATE_URB_ALLOC_DS, and 3DSTATE_URB_ALLOC_GS commands.

SW shall ensure that the URB region specified by this command does not overlap with the push constant allocation in the URB, as defined by the 3DSTATE_PUSH_CONSTANT_ALLOC_VS, 3DSTATE_PUSH_CONSTANT_ALLOC_DS, 3DSTATE_PUSH_CONSTANT_ALLOC_HS, and 3DSTATE_PUSH_CONSTANT_ALLOC_GS commands.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 59h 3DSTATE_URB_ALLOC_HS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 1h DWORD_COUNT_n

3DSTATE_URB_ALLOC_HS			
		Format:	=n
1..2	63:0	**URB Alloc HS State Body**	
		Format:	3DSTATE_URB_ALLOC_HS_BODY

3DSTATE_URB_ALLOC_VS

3DSTATE_URB_ALLOC_VS

Source: RenderCS, PositionCS

Length Bias: 2

When executed from the RCS command stream, this command provides the state variables associated with the URB region used by the VF and VS pipeline stages.

When executed from the POCS command stream, this command provides the state variables associated with the URB region used by the VFR and VSR pipeline stages. The POCS command stream will only execute 3DSTATE_URB_ALLOC_VS command with respect to URB programming for the POCS 3D pipeline. 3DSTATE_URB_ALLOC_HS, 3DSTATE_URB_ALLOC_DS and 3DSTATE_URB_ALLOC_GS commands are ignored by the POCS command stream.

Separate state variables are provided to define the URB region for Slice0 as well as additional URB space for additional slices (if any are enabled). Hardware will use those values to automatically compute the URB allocation within the total URB space based on the number of enabled slices. Software shall ensure that the values programmed do not exceed the URB capacity of a slice. Refer to the L3 allocation and programming guide for valid URB configurations.

Programming Notes

SW shall ensure that the ordering of URB allocations is consistent between the Slice0 and SliceN state settings across all FF stage URB allocations.

The VSR URB region shall never overlap any other URB region. As POCS and RCS command streams are not implicitly synchronized, if POCS is used SW shall reserve a region of the URB for use by VSR. Both pipelines must be flushed and synchronized before expanding the VSR URB region such that the new VSR URB region overlaps URB space previously used by the render pipeline, or the URB space to be used by the render pipeline overlaps the previous VSR URB region.

When specifying URB allocation state for the RCS 3D pipe, SW shall issue allocation state for all FF stages, i.e., 3DSTATE_URB_ALLOC_VS, 3DSTATE_URB_ALLOC_HS, 3DSTATE_URB_ALLOC_DS, and 3DSTATE_URB_ALLOC_GS commands.

SW shall ensure that the URB region specified by this command does not overlap with the push constant allocation in the URB, as defined by the 3DSTATE_PUSH_CONSTANT_ALLOC_VS, 3DSTATE_PUSH_CONSTANT_ALLOC_DS, 3DSTATE_PUSH_CONSTANT_ALLOC_HS, and 3DSTATE_PUSH_CONSTANT_ALLOC_GS commands.

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	**Command SubType**	
		Default Value:	3h GFXPIPE_3D
		Format:	OpCode
	26:24	**3D Command Opcode**	
		Default Value:	0h 3DSTATE_PIPELINED

3DSTATE_URB_ALLOC_VS		
		Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 58h 3DSTATE_URB_ALLOC_VS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 1h DWORD_COUNT_n Format: =n
1..2	63:0	**URB Alloc VS State Body** Format: 3DSTATE_URB_ALLOC_VS_BODY

3DSTATE_URB_DS

3DSTATE_URB_DS

Source: RenderCS

Length Bias: 2

This command may not overlap with the push constants in the URB defined by the 3DSTATE_PUSH_CONSTANT_ALLOC_VS, 3DSTATE_PUSH_CONSTANT_ALLOC_DS, 3DSTATE_PUSH_CONSTANT_ALLOC_HS, and 3DSTATE_PUSH_CONSTANT_ALLOC_GS commands.

The URB Starting Address and Number of URB Entries fields shall be programmed as if there is only one slice enabled. When more than one slice is enabled, hardware will (a) recompute the actual URB Starting Address based on the number of enabled slices and (b) multiply the Number of URB Entries by the number of enabled slices. Software shall ensure that the values programmed do not exceed the URB capacity of a single slice. Refer to the L3 allocation and programming guide for valid URB configurations.

Programming Notes

When programming DS URB state for the RCS 3D pipe, 3DSTATE_URB_VS, 3DSTATE_URB_HS, and 3DSTATE_URB_GS must also be programmed in order for the programming of this state to be valid.

Please see **3DSTATE_URB_ALLOC_DS** for any new programming notes related to URB programming.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 32h 3DSTATE_URB_DS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**URB DS State Body** Format: **3DSTATE_URB_DS_BODY**

3DSTATE_URB_GS

3DSTATE_URB_GS

Source: RenderCS

Length Bias: 2

This command may not overlap with the push constants in the URB defined by the 3DSTATE_PUSH_CONSTANT_ALLOC_VS, 3DSTATE_PUSH_CONSTANT_ALLOC_DS, 3DSTATE_PUSH_CONSTANT_ALLOC_HS, and 3DSTATE_PUSH_CONSTANT_ALLOC_GS commands.

The URB Starting Address and Number of URB Entries fields shall be programmed as if there is only one slice enabled. When more than one slice is enabled, hardware will (a) recompute the actual URB Starting Address based on the number of enabled slices and (b) multiply the Number of URB Entries by the number of enabled slices. Software shall ensure that the values programmed do not exceed the URB capacity of a single slice. Refer to the L3 allocation and programming guide for valid URB configurations

Programming Notes

When programming GS URB state for the RCS 3D pipe, 3DSTATE_URB_VS, 3DSTATE_URB_HS, and 3DSTATE_URB_DS must also be programmed in order for the programming of this state to be valid.

Please see **3DSTATE_URB_ALLOC_GS** for any new programming notes related to URB programming.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 33h 3DSTATE_URB_GS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**URB GS State Body** Format: **3DSTATE_URB_GS_BODY**

3DSTATE_URB_HS

3DSTATE_URB_HS

Source: RenderCS

Length Bias: 2

This command may not overlap with the push constants in the URB defined by the 3DSTATE_PUSH_CONSTANT_ALLOC_VS, 3DSTATE_PUSH_CONSTANT_ALLOC_DS, 3DSTATE_PUSH_CONSTANT_ALLOC_HS, and 3DSTATE_PUSH_CONSTANT_ALLOC_GS commands.

The URB Starting Address and Number of URB Entries fields shall be programmed as if there is only one slice enabled. When more than one slice is enabled, hardware will (a) recompute the actual URB Starting Address based on the number of enabled slices and (b) multiply the Number of URB Entries by the number of enabled slices. Software shall ensure that the values programmed do not exceed the URB capacity of a single slice. Refer to the L3 allocation and programming guide for valid URB configurations

Programming Notes

When programming HS URB state for the RCS 3D pipe, 3DSTATE_URB_VS, 3DSTATE_URB_DS, and 3DSTATE_URB_GS must also be programmed in order for the programming of this state to be valid.

Please see **3DSTATE_URB_ALLOC_HS** for any new programming notes related to URB programming.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 31h 3DSTATE_URB_HS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**URB HS State Body** Format: **3DSTATE_URB_HS_BODY**

3DSTATE_URB_VS

3DSTATE_URB_VS

Source: RenderCS, PositionCS

Length Bias: 2

Description
VS URB Entry Allocation Size equal to 4(5 512-bit URB rows) may cause performance to decrease due to banking in the URB. Element sizes of 16 to 20 should be programmed with six 512-bit URB rows.
This command may not overlap with the push constants in the URB defined by the 3DSTATE_PUSH_CONSTANT_ALLOC_VS, 3DSTATE_PUSH_CONSTANT_ALLOC_DS, 3DSTATE_PUSH_CONSTANT_ALLOC_HS, and 3DSTATE_PUSH_CONSTANT_ALLOC_GS commands.
The offset and size should be programmed as if there is only one slice enabled. Hardware will grow the size based on the slice configuration. Software shall ensure that the values programmed do not exceed the URB capacity of one slice. Refer to the L3 allocation and programming guide for valid URB configurations.
When executed from the POCS command stream, state for the VSR stage is set. The VSR URB region shall never overlap any other URB region. As POCS and RCS command streams are not implicitly synchronized, if POCS is used SW shall reserve a region of the URB for use by VSR. Both pipelines must be flushed and synchronized before expanding the VSR URB region such that the new VSR URB region overlaps URB space previously used by the render pipeline, or the URB space to be used by the render pipeline overlaps the previous VSR URB region. The POCS command stream will only execute 3DSTATE_URB_VS command with respect to URB programming for the POCS 3D pipeline. 3DSTATE_URB_HS, 3DSTATE_URB_DS and 3DSTATE_URB_GS commands are ignored by the POCS command stream.

Programming Notes
When programming VS URB state for the RCS 3D pipe, 3DSTATE_URB_HS, 3DSTATE_URB_DS, and 3DSTATE_URB_GS must also be programmed in order for the programming of this state to be valid.
Please see **3DSTATE_URB_ALLOC_VS** for any new programming notes related to URB programming.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 30h 3DSTATE_URB_VS Format: OpCode

<table>
<tr><th colspan="3">3DSTATE_URB_VS</th></tr>
<tr><td rowspan="2"></td><td>15:8</td><td>Reserved<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>7:0</td><td>DWord Length<table><tr><td>Default Value:</td><td>0h DWORD_COUNT_n</td></tr><tr><td>Format:</td><td>=n</td></tr></table></td></tr>
<tr><td>1</td><td>31:0</td><td>URB VS State Body<table><tr><td>Format:</td><td>3DSTATE_URB_VS_BODY</td></tr></table></td></tr>
</table>

3DSTATE_VERTEX_BUFFERS

3DSTATE_VERTEX_BUFFERS

Source: RenderCS

Length Bias: 2

This command is used to specify VB state used by the VF function.

Can specify from 1 to 33 VBs.

The VertexBufferID field within a VERTEX_BUFFER_STATE structure indicates the specific VB. If a VB definition is not included in this command, its associated state is left unchanged and is available for use if previously defined.

Programming Notes

It is possible to have individual vertex elements sourced completely from generated ID values and therefore not require any vertex buffer accesses for that vertex element. In this case, VF function will simply ignore the VB state associated with that vertex element. If all enabled vertex elements have this characteristic, no VBs are required to process 3DPRIMITIVE commands. For example, this might arise when the user wants to perform all data lookups in the first shader, so only generated index values need to be passed down to it. In this extreme case, SW would not need to program any VB state, and therefore not need to issue any 3DSTATE_VERTEX_BUFFERS commands.

For any 3DSTATE_VERTEX_BUFFERS command, at least one VERTEX_BUFFER_STATE structure must be included.

VERTEX_BUFFER_STATE structures are 4 DWords for both VERTEXDATA buffers and INSTANCEDATA buffers.

Inclusion of partial VERTEX_BUFFER_STATE structures is UNDEFINED.

The order in which VBs are defined within this command can be arbitrary, though a vertex buffer must be defined only once in any given command (otherwise operation is UNDEFINED).

When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 03h GFXPIPE Format: Opcode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: Opcode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: Opcode
	23:16	**3D Command Sub Opcode** Default Value: 08h 3DSTATE_VERTEX_BUFFERS Format: Opcode

3DSTATE_VERTEX_BUFFERS

	15	**Reserved** Access: RO Format: MBZ
	14:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Format: =n n = 4b-1 (where b = # of buffer states included) Value: 3 — Name: DWORD_COUNT_n **[Default]** Value: [3,131] — Name: 1-33 Buffers
1..n	127:0	**Vertex Buffer State** Format: VERTEX_BUFFER_STATE

3DSTATE_VERTEX_ELEMENTS

3DSTATE_VERTEX_ELEMENTS

Source: RenderCS

Length Bias: 2

This is a variable-length command used to specify the active vertex elements. Each VERTEX_ELEMENT_STATE structure contains a Valid bit which determines which elements are used. Any elements not programmed by this command are disabled.

Up to 34 elements.

Programming Notes

- At least one VERTEX_ELEMENT_STATE structure must be included.
- Inclusion of partial VERTEX_ELEMENT_STATE structures is UNDEFINED.
- SW must ensure that at least one vertex element is defined prior to issuing a 3DPRIMTIVE command, or operation is UNDEFINED.
- There are no 'holes' allowed in the destination vertex: NOSTORE components must be overwritten by subsequent components unless they are the trailing DWords of the vertex. Software must explicitly chose some value (probably 0) to be written into DWords that would otherwise be 'holes'.
- Within a VERTEX_ELEMENT_STATE structure, if a Component Control field is set to something other than VFCOMP_STORE_SRC, no higher-numbered Component Control fields may be set to VFCOMP_STORE_SRC. In other words, only trailing components can be set to something other than VFCOMP_STORE_SRC.
- See additional restrictions listed in the command fields and VERTEX_ELEMENT_STATE description.
- Element[0] must be valid.
- All elements must be valid from Element[0] to the last valid element. (E.g.. if Element[2] is valid then Element[1] and Element[0] must also be valid).
- The pitch between elements packed in the URB will always be 128 bits.
- When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 03h GFXPIPE Format: Opcode
	28:27	**Command SubType** Default Value: 3h 3D Format: Opcode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: Opcode
	23:16	**3D Command Sub Opcode** Default Value: 09h 3DSTATE_VERTEX_ELEMENTS

3DSTATE_VERTEX_ELEMENTS

		Format: Opcode
	15	**Reserved** Access: RO Format: MBZ
	14:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Format: =n Vertex Element Count = (DWord Count + 1) / 2 **Value** / **Name** / **Description** 1 / DWORD_COUNT_n **[Default]** / excludes DWords 0,1 [1,67] / Range / 1-34 Elements
1..n	63:0	**Element** Format: VERTEX_ELEMENT_STATE

3DSTATE_VF_COMPONENT_PACKING

3DSTATE_VF_COMPONENT_PACKING

Source: RenderCS

Length Bias: 2

This command is used to specify, separately for Vertex Elements [0-31], which post-conversion, 32-bit components are "enabled" to be stored in the URB , and which are "disabled" (not stored). 128 per-component enable bits are provided. Disabling all four components for a given Vertex Element will result in no data stored for that element. Note that any insertion of SGVs (3DSTATE_VF_SGVS) is performed before the packing operation. The **Component Packing Enable** bit (3DSTATE_VF) controls the overall packing process. If that bit is set, the packing process is enabled and the bit mask provided in this command is used to control which components are stored. If that bit is clear, the packing process is disabled - all four components of "valid" Vertex Elements will be stored.

Programming Notes

Programming Restrictions:

- The Vertex Elements referenced in this command correspond to the first 32 VERTEX_ELEMENT structures passed in 3DSTATE_VERTEX_ELEMENTS. A Vertex Element must be marked as "Valid" via 3DSTATE_VERTEX_ELEMENTS or be an SGV or an element between the last valid element and the last SGV in order for the corresponding Component Enable bits of this command to be utilized.
- No enable bits are provided for Vertex Elements [32-33], and therefore no packing is performed on these elements (if Valid, all 4 components are stored).
- If a Vertex Element has Edge Flag Enable set no packing is performed for that element and the corresponding packing state is ignored.
- Component packing is probably only useful for SIMD8 VS thread execution.

At least one component of one "valid" Vertex Element must be enabled.

When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.

Software shall enable all components (XYZW) for any and all VERTEX_ELEMENTs associated with a 256-bit SURFACE_FORMAT. It is INVALID to disable any components in these cases.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode

3DSTATE_VF_COMPONENT_PACKING

	23:16	**3D Command Sub Opcode** Default Value: 55h 3DSTATE_VF_COMPONENT_PACKING Format: OpCode
	15	**Reserved** Access: RO Format: MBZ
	14:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 3h Excludes DWord (0,1) Format: =n
1..4	127:0	**VF Component Packing State Body** Format: 3DSTATE_VF_COMPONENT_PACKING_BODY

3DSTATE_VF_INSTANCING

3DSTATE_VF_INSTANCING

Source: RenderCS

Length Bias: 2

This command is used to control the "instancing" state associated with a specific vertex element.

Programming Notes

When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 49h 3DSTATE_VF_INSTANCING Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Format: =n Value / Name 1h: Excludes DWord (0,1) **[Default]** 43h: Context Restore
1..2	63:0	**VF Instancing State Body** Format: 3DSTATE_VF_INSTANCING_BODY

3DSTATE_VF

3DSTATE_VF	
Source:	RenderCS
Length Bias:	2

This command is used to set various state variables in the VF stage.

The use of the component packing mask is specified via 3DSTATE_VF_COMPONENT_PACKING

Programming Notes

When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 0Ch 3DSTATE_VF Format: OpCode
	15	**Reserved** Access: RO Format: MBZ
	14	**Force Sequential Access Enable** Format: Enable If ENABLED, the VERTEXDATA buffers are accessed sequentially, regardless of the value of 3DPRIMITIVE::VertexAccessType. The VertexID will still be equal to the index obtained from the Index Buffer if 3DPRIMITIVE::VertexAccessType is RANDOM. The3DSTATE_VF_TOPOLOGY::Primitive Topology Type must be set to patchlist 1. If DISABLED, the VERTEXDATA buffers are accessed according to the value of 3DPRIMITIVE::VertexAccessType. **Programming Notes** Software shall not enable both Force Sequential Access Enable AND either of the Sequential/Indexed Draw Cut Index Enable bits. I.e., Force Sequential Access and Cut Index processing are mutually exclusive functions.

	13	**InstanceID Offset Enable** Format: Enable If ENABLED, the InstanceID value optionally inserted into the vertex data, and used as an index for vertex element addressing when Instance Stride Enable is ENABLED, is offset by StartInstanceLocation. If DISABLED, InstanceID is not offset by StartInstanceLocation and instead is always 0-based.
	12	**Reserved** Access: RO Format: MBZ
	11	**VertexID Offset Enable** Format: Enable If ENABLED, the VertexID value optionally inserted into the vertex data is offset by StartVertexLocation (SEQUENTIAL draws) or BaseVertexLocation (RANDOM draws).If DISABLED, VertexID is not offset by these values and instead is always 0-based
	10	**Sequential Draw Cut Index Enable** Format: Enable If ENABLED, vertex indices in SEQUENTIAL 3DPRIMITIVE commands are compared to the Cut Index (specified below). When the vertex index matches the Cut Index any previous topology is terminated. If DISABLED, vertex indices are not compared to the Cut Index. This field can only be enabled for certain primitive topology types. Refer to the table later in this section for details. **Programming Notes** Software shall not enable both Force Sequential Access Enable AND either of the Sequential/Indexed Draw Cut Index Enable bits. I.e., Force Sequential Access and Cut Index processing are mutually exclusive functions.
	9	**Component Packing Enable** Format: Enable If ENABLED, vertex element component packing (as specified by 3DSTATE_VF_COMPONENT_PACKING) is performed before vertices are written into the URB. If DISABLED, no component packing is performed - all components of valid vertex elements will be stored in the URB.
	8	**Indexed Draw Cut Index Enable** Format: Enable If ENABLED, vertex indices in RANDOM 3DPRIMITIVE commands are compared to the Cut Index (specified below). When the vertex index matches the Cut Index any previous topology is terminated. If DISABLED, vertex indices are not compared to the Cut Index and are used strictly as indices into vertex buffers. This field can only be enabled for certain primitive topology types. Refer to the table later in this section for details. **Programming Notes** Software shall not enable both Force Sequential Access Enable AND either of the Sequential/Indexed Draw Cut Index Enable bits. I.e., Force Sequential Access and Cut Index processing are mutually exclusive functions.

3DSTATE_VF			
	7:0	**DWord Length**	
		Default Value:	0h Excludes DWord (0,1)
		Format:	=n
1	31:0	**VF State Body**	
		Format:	3DSTATE_VF_BODY

3DSTATE_VF_SGVS_2

3DSTATE_VF_SGVS_2

Source: RenderCS

Length Bias: 2

This command is used to control the insertion of the Extended Parameter (XP0-2) System-Generated Values (SGVs) into an input Vertex URB Entry (VUE) (available as input to a VS thread). The insertions are individually controlled. The insertion locations are specified as 128-bit element locations (starting at the beginning of the VUE) and 32-bit component within those specified elements. The SGV values can be inserted either (a) within a valid vertex element (in which case the value overwrites the value specified via 3DSTATE_VERTEX_ELEMENTS) or (b) beyond the last valid vertex element written to the URB. This permits some orthogonality between the programming of vertex elements (which typically is known at draw time) and programming of SGV insertion (which is associated with the shader). There are some restrictions however (see below). If an SGV is inserted beyond the last valid vertex element, zeroes are first inserted in the VUE after the last valid vertex element up to and including the vertex element receiving an SGV. If both of the SGVs are enabled for insertion, the zeroes will extend to the last vertex element receiving and SGV. Then the SGV(s) are inserted.
The sources for these SGV values are derived from 3DPRIMITIVE command parameters. Controls in the 3DPRIMITIVE command determine whether (a) the parameters are directly defined via in-line command DWords, (b) the parameters are indirectly specified by command stream registers, or (c) the parameters are not included in the 3DPRIMITIVE command and therefore default to 0. Refer to the 3DPRIMITIVE command description for details on these different cases.
The states included in this command are used to (a) enable/disable specific XP* SGV insertions and (b) for XP0 and XP1, specify which 3DPRIMITIVE parameters are used to source the inserted SGV value.

The insertion of SGV values occurs before any component packing (3DSTATE_VF_COMPONENT_PACKING). Therefore, the Element Offsets and Component Numbers specified in this command refer to the pre-packed data, following 3DSTATE_VERTEX_ELEMENT processing.

Programming Notes

Programming Restrictions:

- It is INVALID to specify that more than one SGV is to be stored in the same element/component location within the VUE.
- The states programmed by this command overwrite the state programmed by any previous commands. I.e., a specific SGV (if enabled) can only be inserted in one component of a vertex.
- It is INVALID to insert an SGV value past the end of the VUE entry (as determined by VS URB Entry Allocation Size) or past the 33rd vertex element. Therefore, the programming of VS URB Entry Allocation Size needs to comprehend any SGV insertion requirements.
- It is INVALID to use this command to overwrite any portion of a 64-bit vertex element component.

When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE

3DSTATE_VF_SGVS_2

		Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 56h 3DSTATE_VF_SGVS_2 Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 1h Excludes DWord (0,1) Format: =n
1..2	63:0	**VF SGVS 2 State Body** Format: 3DSTATE_VF_SGVS_2_BODY

3DSTATE_VF_SGVS

3DSTATE_VF_SGVS

Source: RenderCS

Length Bias: 2

This command is used to control the insertion of the VertexID and InstanceID System-Generated Values (SGVs) into an input Vertex URB Entry (VUE) (available as input to a VS thread). VertexID and InstanceID insertion can be individually controlled. The insertion locations are specified as 128-bit element locations (starting at the beginning of the VUE) and the 32-bit component within those specified elements. The SGV values can be inserted either (a) within a valid vertex element (in which case the value overwrites the value specified via 3DSTATE_VERTEX_ELEMENTS) or (b) beyond the last valid vertex element written to the URB. This permits some orthogonality between the programming of vertex elements (which typically is known at draw time) and programming of SGV insertion (which is associated with the shader). There are some restrictions however (see below). If an SGV is inserted beyond the last valid vertex element, zeroes are first inserted in the VUE after the last valid vertex element up to and including the vertex element receiving an SGV. If both of the SGVs are enabled for insertion, the zeroes will extend to the last (largest index) vertex element receiving an SGV. Then the SGV(s) are inserted.

The insertion of SGV values occurs before any component packing (3DSTATE_VF_COMPONENT_PACKING). Therefore, the Element Offsets and Component Numbers specified in this command refer to the pre-packed data, following 3DSTATE_VERTEX_ELEMENT processing.

Programming Notes

Programming Restrictions:

- It is INVALID to store both the VertexID and InstanceID in the same element/component location within the VUE.
- The states programmed by this command overwrite the state programmed by any previous commands. I.e., VertexID and InstanceID (if enabled) can only be inserted in one component of a vertex.
- It is INVALID to insert an SGV value past the end of the VUE entry (as determined by VS URB Entry Allocation Size) or past the 33rd vertex element. Therefore, the programming of VS URB Entry Allocation Size needs to comprehend any SGV insertion requirements.
- It is INVALID to use this command to overwrite any portion of a 64-bit vertex element component.
- It is INVALID to use this command to overwrite a EdgeFlag vertex element component or any vertex element beyond it.

When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D

3DSTATE_VF_SGVS

		Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 4Ah 3DSTATE_VF_SGVS Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n
1	31:0	**VF SGVS State Body** Format: 3DSTATE_VF_SGVS_BODY

3DSTATE_VF_STATISTICS

3DSTATE_VF_STATISTICS

Source: RenderCS

Length Bias: 1

The VF stage tracks two pipeline statistics, the number of vertices fetched and the number of objects generated. VF will increment the appropriate counter for each when statistics gathering is enabled by issuing the 3DSTATE_VF_STATISTICS command with the [Statistics Enable] bit set.

Programming Notes

When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: Opcode
	28:27	**Command SubType** Default Value: 1h GFXPIPE_SINGLE_DW Format: Opcode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: Opcode GFXPIPE[28:27 = 1h, 26:24 = 0h, 23:16 = 0Bh] (Pipelined, Single DWord)
	23:16	**3D Command Sub Opcode** Default Value: 0Bh 3DSTATE_VF_STATISTICS Format: Opcode GFXPIPE[28:27 = 1h, 26:24 = 0h, 23:16 = 0Bh] (Pipelined, Single DWord)
	15:1	**Reserved** Access: RO Format: MBZ
	0	**Statistics Enable** Format: Enable If ENABLED, VF will increment the pipeline statistics counters IA_VERTICES_COUNT and IA_PRIMITIVES_COUNT for each vertex fetched and each object output, respectively, for 3DPRIMITIVE commands issued subsequently.If DISABLED, these counters will not be incremented for subsequent 3DPRIMITIVE commands. **Programming Notes** When a 3DPRIMITIVE command with POSH Enable set is executed from the RCS command stream, VF statistics gathering is inhibited for that command.

3DSTATE_VF_TOPOLOGY

3DSTATE_VF_TOPOLOGY

Source: RenderCS

Length Bias: 2

This command specifies the VF stage's Topology state which can be used to override the Primitive Topology Type in subsequent 3DPRIMITIVE commands.

Programming Notes

When executed in the POCS command stream, this command programs state for the VFR stage of the POCS pipeline, and only the following topologies are valid: 3DPRIM_TRILIST, 3DPRIM_TRISTRIP, 3DPRIM_TRISTRIP_REV.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 4Bh 3DSTATE_VF_TOPOLOGY Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n
1	31:0	**VF Topology State Body** Format: 3DSTATE_VF_TOPOLOGY_BODY

3DSTATE_VIEWPORT_STATE_POINTERS_CC

3DSTATE_VIEWPORT_STATE_POINTERS_CC

Source: RenderCS

Length Bias: 2

The 3DSTATE_VIEWPORT_STATE_POINTERS_CC command is used to define the location of fixed functions' viewport state table.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 23h 3DSTATE_VIEWPORT_STATE_POINTERS_CC Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**Viewport State Pointers CC State Body** Format: 3DSTATE_VIEWPORT_STATE_POINTERS_CC_BODY

3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP

3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP

Source: RenderCS

Length Bias: 2

The 3DSTATE_VIEWPORT_STATE_POINTERS_CLIP command is used to define the location of fixed functions' viewport state table.

Restriction
When executed in POCS command stream, this programs viewport state of the POCS pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 21h 3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:0	**Viewport State Pointers SF Clip State Body** Format: 3DSTATE_VIEWPORT_STATE_POINTERS_SF_CLIP_BODY

3DSTATE_VS

3DSTATE_VS

Source: RenderCS, PositionCS

Length Bias: 2

This command specifies most of the state used by the Vertex Shader (VS) stage.

Programming Notes

When executed in the POCS command stream, this command programs state for the VSR stage of the POCS pipeline. When executed in the RCS command stream, this command programs state for the VS stage of the RCS pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 10h 3DSTATE_VS Format: OpCode
	15	**Reserved** Access: RO Format: MBZ
	14:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 7h Excludes DWord (0,1) Format: =n
1..8	255:0	**VS State Body** Format: 3DSTATE_VS_BODY

3DSTATE_WM_CHROMAKEY

3DSTATE_WM_CHROMAKEY		
Source:	RenderCS	
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 4Ch 3DSTATE_WM__CHROMAKEY Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**Dword Length** Default Value: 0h Excludes Dword (0,1) Format: =n Total Length - 2
1	31:0	**WM Chromakey State Body** Format: 3DSTATE_WM_CHROMAKEY_BODY

3DSTATE_WM_DEPTH_STENCIL

3DSTATE_WM_DEPTH_STENCIL		
Source:	RenderCS	
Length Bias:	2	
This command replaces the indirect state DEPTH_STENCIL_STATE with an inline state command.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 4Eh 3DSTATE_WM_DEPTH_STENCIL Format: OpCode
	15:13	**Reserved** Access: RO Format: MBZ
	12	**Depth State Modify Disable** Format: Disable When this bit is set, the following fields will be ignored: • Depth Test Function • Depth Test Enable • Depth Buffer Write Enable
	11	**Stencil State Modify Disable** Format: Disable When this bit is set, the following fields will be ignored: • Stencil Fail Op • Stencil Pass Depth Fail Op • Stencil Pass Depth Pass Op • Backface Stencil Test Function • Backface Stencil Fail Op • Backface Stencil Pass Depth Fail Op

3DSTATE_WM_DEPTH_STENCIL

		• Backface Stencil Pass Depth Pass Op • Stencil Test Function • Double Sided Stencil Enable • Stencil Test Enable • Stencil Buffer Write Enable
	10	**Stencil Write Mask Modify Disable** Format: Disable When this bit is set, the following fields will be ignored: • Stencil Write Mask • Backface Stencil Write Mask
	9	**Stencil Test Mask Modify Disable** Format: Disable When this bit is set, the following fields will be ignored: • Stencil Test Mask • Backface Stencil Test Mask
	8	**Stencil Reference Value Modify Disable** Format: Disable When this bit is set, the following fields will be ignored: • Stencil Reference Value • Backface Stencil Reference Value
	7:0	**Dword Length** Default Value: 02h Excludes Dword (0,1) Format: =n Total Length - 2
1..3	95:0	**WM Depth Stencil State Body** Format: 3DSTATE_WM_DEPTH_STENCIL_BODY

3DSTATE_WM_HZ_OP

3DSTATE_WM_HZ_OP

Source: RenderCS

Length Bias: 2

This command provides for clearing Z and/or stencil or resolving either HZ buffer or Z buffer.

Programming Notes

As this command generates an implicit rectangle, SW must make sure any MMIO register writes following WM_HZ_OP must be preceded by **PIPE_CONTROL** with **Command Streamer Stall Enable** bit set.

3DSTATE_DRAWING_RECTANGLE must be programmed such that it does not clip the HZ_OP command's rectangle. See programming notes for X/Y Min and X/Y Max below.
3DSTATE_DRAWING_RECTANGLE command must come before 3DSTATE_WM_HZ_OP in the command buffer
Caution: There is a difference in how X/Y coordinates are interpreted by 3DSTATE_DRAWING_RECTANGLE vs. 3DSTATE_WM_HZ_OP.
HZ_OP rectangle parameters are exclusive on max side, for example to have 8x4 rectangle we would program X Min = 0, Y Min = 0, X Max = 8, Y Max = 4.
Draw Rectangle parameters are inclusive on max side, meaning, for 8x4 rectangle the values would be X Min = 0, Y Min = 0, X Max = 7, Y Max = 3.

3DSTATE_MULTISAMPLE packet must be used prior to this packet to change the Number of Multisamples. This packet must not be used to change Number of Multisamples in a rendering sequence.
3DSTATE_RASTER if used must be programmed prior to using this packet.

SW should flush the Depth Cache AFTER the following WM_HZ_OP commands
1. Depth Clear
2. HIZ Resolve

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 52h 3DSTATE_WM_HZ_OP Format: OpCode
	15:9	**Reserved** Access: RO Format: MBZ

3DSTATE_WM_HZ_OP

	8	**Reserved** Access: RO Format: MBZ
	7:0	**Dword Length** Format: =n Total Length - 2 Value: 03h — Name: Excludes Dword (0,1) **[Default]**
1..4	127:0	**WM HZ OP State Body** Format: 3DSTATE_WM_HZ_OP_BODY

3DSTATE_WM

3DSTATE_WM		
Source:	RenderCS	
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 0h 3DSTATE_PIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 14h 3DSTATE_WM Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:0	**WM State Body** Format: 3DSTATE_WM_BODY

A64 Byte Scattered Read MSD

MSD1R_A64_BS - A64 Byte Scattered Read MSD		
Source:	EuSubFunctionDataPort1	
Length Bias:	1	
DWord	**Bit**	**Description**
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 10h Format: Opcode A64 Scattered Read message
	13	**Invalidate After Read** Format: MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	11:10	**Data Elements** Format: MDC_A64_DS Specifies the number of data elements to be read or written
	9:8	**A64 Scattered Message Subtype** Default Value: 0h Format: Opcode Byte Read/Write subtype
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Byte Scattered Write MSD

MSD1W_A64_BS - A64 Byte Scattered Write MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 1Ah Format: Opcode A64 Scattered Write message
	13	**Reserved** Access: RO Format: MBZ
	12	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	11:10	**Data Elements** Format: MDC_A64_DS Specifies the number of data elements to be read or written
	9:8	**A64 Scattered Message Subtype**

MSD1W_A64_BS - A64 Byte Scattered Write MSD

		Default Value: 0h Format: Opcode Byte Read/Write subtype
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Dword Scattered Read MSD

MSD1R_A64_DWS - A64 Dword Scattered Read MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 10h Format: Opcode A64 Scattered Read message
	13	**Invalidate After Read** Format: MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	11:10	**Data Elements** Format: MDC_A64_DS Specifies the number of data elements to be read or written

MSD1R_A64_DWS - A64 Dword Scattered Read MSD

	9:8	**A64 Scattered Message Subtype** Default Value: 1h Format: Opcode Dword Read/Write subtype
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Dword Scattered Write MSD

MSD1W_A64_DWS - A64 Dword Scattered Write MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 1Ah Format: Opcode A64 Scattered Write message
	13	**Reserved** Access: RO Format: MBZ
	12	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	11:10	**Data Elements** Format: MDC_A64_DS Specifies the number of data elements to be read or written

MSD1W_A64_DWS - A64 Dword Scattered Write MSD

	9:8	**A64 Scattered Message Subtype** Default Value: 1h Format: Opcode Dword Read/Write subtype
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Dword Untyped Atomic Float with Return Data Operation MSD

MSD1R_A64_DWAF - A64 Dword Untyped Atomic Float with Return Data Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 1Dh Format: Opcode A64 Untyped Atomic Float Operation message
	13	**Return Data Control** Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2S Only SIMD8 operations are supported.
	11	**Data Width** Default Value: 0h

MSD1R_A64_DWAF - A64 Dword Untyped Atomic Float with Return Data Operation MSD

		Format: Opcode Operations are on 32-bit floats.
	10:8	**Atomic Float Operation** Format: MDC_FOP Specifies the atomic float operation to be performed.
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Dword Untyped Atomic Float Write Only Operation MSD

MSD1W_A64_DWAF - A64 Dword Untyped Atomic Float Write Only Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 1Dh Format: Opcode A64 Untyped Atomic Float Operation message
	13	**Return Data Control** Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2S Description: Only SIMD8 operations are supported.

MSD1W_A64_DWAF - A64 Dword Untyped Atomic Float Write Only Operation MSD

	11	**Data Width** Default Value: 0h Format: Opcode Operations are on 32-bit floats.
	10:8	**Atomic Float Operation Type** Format: MDC_FOP Specifies the atomic float operation to be performed.
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Dword Untyped Atomic Integer with Return Data Operation MSD

MSD1R_A64_DWAI - A64 Dword Untyped Atomic Integer with Return Data Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF The message forbids a header
	18:14	**Message Type** Default Value: 12h Format: Opcode A64 Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Format: Opcode Operations are on 32-bit integers

MSD1R_A64_DWAI - A64 Dword Untyped Atomic Integer with Return Data Operation MSD

	11:8	**Atomic Integer Operation** Format: **MDC_AOP** Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

A64 Dword Untyped Atomic Integer Write Only Operation MSD

MSD1W_A64_DWAI - A64 Dword Untyped Atomic Integer Write Only Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF The message forbids a header
	18:14	**Message Type** Default Value: 12h Format: Opcode A64 Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Format: Opcode Operations are on 32-bit integers

MSD1W_A64_DWAI - A64 Dword Untyped Atomic Integer Write Only Operation MSD

	11:8	**Atomic Integer Operation** Format: **MDC_AOP** Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

A64 Hword Block Read MSD

MSD1R_A64_HWB - A64 Hword Block Read MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 14h Format: Opcode A64 Oword Block Read message
	13	**Invalidate After Read** Format: MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12:11	**A64 Block Message Subtype** Default Value: 3h Format: Opcode Hword Block Read/Write subtype
	10:8	**Data Elements** Format: MDC_A64_DB_HW Specifies the number of contiguous Hwords to be read or written

MSD1R_A64_HWB - A64 Hword Block Read MSD		
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Hword Block Write MSD

MSD1W_A64_HWB - A64 Hword Block Write MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 15h Format: Opcode A64 Hword Block Write message
	13	**Reserved** Access: RO Format: MBZ
	12:11	**A64 Block Message Subtype** Default Value: 3h Format: Opcode Hword Block Read/Write subtype
	10:8	**Data Elements** Format: MDC_A64_DB_HW Specifies the number of contiguous Hwords to be read or written

MSD1W_A64_HWB - A64 Hword Block Write MSD		
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

A64 Oword Aligned Block Read MSD

MSD1R_A64_OWAB - A64 Oword Aligned Block Read MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 14h Format: Opcode A64 Oword Block Read message
	13	**Reserved** Access: RO Format: MBZ
	12:11	**A64 Block Message Subtype** Default Value: 1h Format: Opcode Oword Aligned Block Read subtype
	10:8	**Data Elements** Format: MDC_A64_DB_OW Specifies the number of contiguous Owords to be read

MSD1R_A64_OWAB - A64 Oword Aligned Block Read MSD		
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Oword Aligned Block Write MSD

MSD1W_A64_OWAB - A64 Oword Aligned Block Write MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 15h Format: Opcode A64 Oword Block Write message
	13	**Reserved** Access: RO Format: MBZ
	12:11	**A64 Block Message Subtype** Default Value: 1h Format: Opcode Oword Aligned Block Write subtype
	10:8	**Data Elements** Format: MDC_A64_DB_OW Specifies the number of contiguous Owords to be written

MSD1W_A64_OWAB - A64 Oword Aligned Block Write MSD		
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Oword Block Read MSD

MSD1R_A64_OWB - A64 Oword Block Read MSD		
Source:	EuSubFunctionDataPort1	
Length Bias:	1	
DWord	**Bit**	**Description**
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 14h Format: Opcode A64 Oword Block Read message
	13	**Invalidate After Read** Format: MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12:11	**A64 Block Message Subtype** Default Value: 0h Format: Opcode Oword Block Read/Write subtype
	10:8	**Data Elements** Format: MDC_A64_DB_OW Specifies the number of contiguous Owords to be read or written

MSD1R_A64_OWB - A64 Oword Block Read MSD		
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Oword Block Write MSD

MSD1W_A64_OWB - A64 Oword Block Write MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 15h Format: Opcode A64 Oword Block Write message
	13	**Reserved** Access: RO Format: MBZ
	12:11	**A64 Block Message Subtype** Default Value: 0h Format: Opcode Oword Block Read/Write subtype
	10:8	**Data Elements** Format: MDC_A64_DB_OW Specifies the number of contiguous Owords to be read or written

MSD1W_A64_OWB - A64 Oword Block Write MSD		
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Qword Scattered Read MSD

MSD1R_A64_QWS - A64 Qword Scattered Read MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 10h Format: Opcode A64 Scattered Read message
	13	**Invalidate After Read** Format: MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	11:10	**Data Elements** Format: MDC_A64_DS Specifies the number of data elements to be read or written

MSD1R_A64_QWS - A64 Qword Scattered Read MSD

	9:8	**A64 Scattered Message Subtype** Default Value: 2h Format: Opcode Qword Read/Write subtype
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Qword Scattered Write MSD

MSD1W_A64_QWS - A64 Qword Scattered Write MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 1Ah Format: Opcode A64 Scattered Write message
	13	**Reserved** Access: RO Format: MBZ
	12	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	11:10	**Data Elements** Format: MDC_A64_DS Specifies the number of data elements to be read or written

MSD1W_A64_QWS - A64 Qword Scattered Write MSD		
	9:8	**A64 Scattered Message Subtype** Default Value: 2h Format: Opcode Qword Read/Write subtype
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Qword Untyped Atomic Integer with Return Data Operation MSD

MSD1R_A64_QWAI - A64 Qword Untyped Atomic Integer with Return Data Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF The message forbids a header
	18:14	**Message Type** Default Value: 12h Format: Opcode A64 Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**Data Width** Default Value: 1h Format: Opcode Operations are on 64-bit integers

MSD1R_A64_QWAI - A64 Qword Untyped Atomic Integer with Return Data Operation MSD

	11:8	**Atomic Integer Operation** Format: **MDC_AOP** Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

A64 Qword Untyped Atomic Integer Write Only Operation MSD

MSD1W_A64_QWAI - A64 Qword Untyped Atomic Integer Write Only Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF The message forbids a header
	18:14	**Message Type** Default Value: 12h Format: Opcode A64 Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Data Width** Default Value: 1h Format: Opcode Operations are on 64-bit integers

MSD1W_A64_QWAI - A64 Qword Untyped Atomic Integer Write Only Operation MSD

	11:8	**Atomic Integer Operation** Format: **MDC_AOP** Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

A64 Untyped Surface Read MSD

MSD1R_A64_US - A64 Untyped Surface Read MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 11h Format: Opcode A64 Untyped Surface Read message
	13:12	**SIMD Mode** Format: MDC_SM3 Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Channel Mask** Format: MDC_CMASK Specifies which RGBA channels are included in the message payload.
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Untyped Surface Write MSD

MSD1W_A64_US - A64 Untyped Surface Write MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 19h Format: Opcode A64 Untyped Surface Write message
	13:12	**SIMD Mode** Format: MDC_SM3 Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Channel Mask** Format: MDC_UW_CMASK Specifies which RGBA channels are included in the message payload.
	7:0	**Binding Table Index** Format: MDC_STATELESS Specifies the message is stateless

A64 Word Untyped Atomic Float with Return Data Operation MSD

MSD1R_A64_WAF - A64 Word Untyped Atomic Float with Return Data Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 1Eh Format: Opcode A64 Untyped Atomic Half Float Operation message
	13	**Return Data Control** Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**Reserved** Access: RO Format: MBZ
	11	**Data Width** Default Value: 0h Format: Opcode

MSD1R_A64_WAF - A64 Word Untyped Atomic Float with Return Data Operation MSD

		Operations are on 32-bit floats.
	10:8	**Atomic Float Operation** Format: **MDC_FOP** Specifies the atomic float operation to be performed.
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

A64 Word Untyped Atomic Float Write Only Operation MSD

MSD1W_A64_WAF - A64 Word Untyped Atomic Float Write Only Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header
	18:14	**Message Type** Default Value: 1Eh Format: Opcode A64 Untyped Atomic Half Float Operation message
	13	**Return Data Control** Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Reserved** Access: RO Format: MBZ
	11	**Data Width** Default Value: 0h Format: Opcode

MSD1W_A64_WAF - A64 Word Untyped Atomic Float Write Only Operation MSD

		Operations are on 32-bit floats.
	10:8	**Atomic Float Operation Type** Format: **MDC_FOP** Specifies the atomic float operation to be performed.
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

A64 Word Untyped Atomic Integer with Return Data Operation MSD

MSD1R_A64_WAI - A64 Word Untyped Atomic Integer with Return Data Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF The message forbids a header
	18:14	**Message Type** Default Value: 13h Format: Opcode A64 Untyped Atomic Half Integer Operation message
	13	**Return Data Control** Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Format: Opcode Operations are on 16-bit integers

MSD1R_A64_WAI - A64 Word Untyped Atomic Integer with Return Data Operation MSD

	11:8	**Atomic Integer Operation** Format: **MDC_AOP** Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

A64 Word Untyped Atomic Integer Write Only Operation MSD

MSD1W_A64_WAI - A64 Word Untyped Atomic Integer Write Only Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF The message forbids a header
	18:14	**Message Type** Default Value: 13h Format: Opcode A64 Untyped Atomic Half Integer Operation message
	13	**Return Data Control** Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Data Width** Default Value: 0h Format: Opcode Operations are on 16-bit integers

MSD1W_A64_WAI - A64 Word Untyped Atomic Integer Write Only Operation MSD

	11:8	**Atomic Integer Operation** Format: **MDC_AOP** Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: **MDC_STATELESS** Specifies the message is stateless

Addition

add - Addition

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	true
Source Modifier:	true

The add instruction performs component-wise addition of src0 and src1 and stores the results in dst. Addition of two floating-point numbers follows rules in add (IEEE mode) or add (ALT mode).

Format:

```
[(pred)] add[.cmod] (exec_size) dst src0 src1
```

Programming Notes

Use a source modifier with add to implement subtraction.

Syntax

[(pred)] add[.cmod] (exec_size) reg reg reg
[(pred)] add[.cmod] (exec_size) reg reg imm32

Pseudocode

```
Evaluate(WrEn);

 for ( n = 0; n < exec_size; n++ ) {
   if ( WrEn.chan[n] ) {
     dst.chan[n] = src0.chan[n] + src1.chan[n];
   }
 }
```

Src Types	Dst Types
*B,*W,*D	*B,*W,*D
*B,*W,*D	F
F	F
HF	HF
*B,*W,*D	HF

DWord	Bit	Description
0..3	127:126	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	127:96	**Src1.ImmValue[31:0]** Exists If: ([Src1.IsImm]==true)

add - Addition

	125:122	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	121:120	**Src1.Mod** Exists If: ([Src1.IsImm]==false) Format: **SrcMod**
	119:116	**Src1.VertStride** Exists If: ([Src1.IsImm]==false) Format: **VertStride**
	115:113	**Src1.Width** Exists If: ([Src1.IsImm]==false) Format: **Width**
	112	**Src1.AddrMode** Exists If: ([Src1.IsImm]==false) Format: **AddrMode**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect) Format: **IndirectOperand**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct) Format: **DirectOperand**
	97:96	**Src1.HorzStride** Exists If: ([Src1.IsImm]==false) Format: **HorzStride**
	95:92	**CondCtrl** Format: **FlagModifier**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==true) Format: **ImmDataType**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==false) Format: **RegDataType**
	87:84	**Src0.VertStride** Format: **VertStride**
	83:81	**Src0.Width** Format: **Width**

add - Addition

	80	**Src0.AddrMode** Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
	45:44	**Src0.Mod** Format: **SrcMod**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: **RegDataType**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: **ImmDataType**

add - Addition

	39:36	**Dst.DataType** Format: **RegDataType**
	35	**Dst.AddrMode** Format: **AddrMode**
	34	**Saturate** Format: **Saturate**
	33	**AccWrCtrl** Format: **AccWrCtrl**
	32	**AtomicCtrl** Format: **AtomicCtrl**
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". **Value** / **Name** / **Description** 0 / Normal **[Default]** / Normal. Per channel write enable used for final write enable generation. 1 / NoMask / NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	30	**Reserved**
	29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. **Value** / **Name** / **Description** 0 / NoCompaction **[Default]** / No compaction. 128-bit native instruction supporting all instruction options. 1 / Compacted / Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
	28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields **Value** / **Name** / **Description**

		0 | Positive **[Default]** | Positive polarity of predication. Use the predication mask produced by PredCtrl. 1 | Negative | Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
	27:24	**PredCtrl** Format: **PredCtrl** This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.
	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Addition with Carry

addc - Addition with Carry

Source:	Eulsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	false
Source Modifier:	false

The addc instruction performs component-wise addition of src0 and src1 and stores the results in dst; it also stores the carry into acc. If the operation produces a carry out, 0x00000001 is stored in acc, else 0x00000000 is stored in acc.

Format:

```
[(pred)] addc[.cmod] (exec_size) dst src0 src1
```

Restriction

AccWrEn is required.

The accumulator is an implicit destination and thus cannot be an explicit destination operand.

Syntax

```
[(pred)] addc[.cmod] (exec_size) reg reg reg
 [(pred)] addc[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         dst.chan[n] = src0.chan[n] + src1.chan[n];
         acc.chan[n] = carry(src0.chan[n] + src1.chan[n]);
     }
 }
```

Src Types	Dst Types
UD	UD

DWord	Bit	Description
0..3	127:126	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	127:96	**Src1.ImmValue[31:0]** Exists If: ([Src1.IsImm]==true)
	125:122	**Reserved**

addc - Addition with Carry

	Exists If:	([Src1.IsImm]==false)
	Format:	MBZ
121:120	**Src1.Mod**	
	Exists If:	([Src1.IsImm]==false)
	Format:	**SrcMod**
119:116	**Src1.VertStride**	
	Exists If:	([Src1.IsImm]==false)
	Format:	**VertStride**
115:113	**Src1.Width**	
	Exists If:	([Src1.IsImm]==false)
	Format:	**Width**
112	**Src1.AddrMode**	
	Exists If:	([Src1.IsImm]==false)
	Format:	**AddrMode**
111:98	**Src1.Operand**	
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)
	Format:	**IndirectOperand**
111:98	**Src1.Operand**	
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)
	Format:	**DirectOperand**
97:96	**Src1.HorzStride**	
	Exists If:	([Src1.IsImm]==false)
	Format:	**HorzStride**
95:92	**CondCtrl**	
	Format:	**FlagModifier**
91:88	**Src1.DataType**	
	Exists If:	([Src1.IsImm]==true)
	Format:	**ImmDataType**
91:88	**Src1.DataType**	
	Exists If:	([Src1.IsImm]==false)
	Format:	**RegDataType**
87:84	**Src0.VertStride**	
	Format:	**VertStride**
83:81	**Src0.Width**	
	Format:	**Width**
80	**Src0.AddrMode**	

addc - Addition with Carry

	Format:	**AddrMode**
79:66	**Src0.Operand**	
	Exists If:	([Src0.AddrMode]==Direct)
	Format:	**DirectOperand**
79:66	**Src0.Operand**	
	Exists If:	([Src0.AddrMode]==Indirect)
	Format:	**IndirectOperand**
65:64	**Src0.HorzStride**	
	Format:	**HorzStride**
63:50	**Dst.Operand**	
	Exists If:	([Dst.AddrMode]==Direct)
	Format:	**DirectOperand**
63:50	**Dst.Operand**	
	Exists If:	([Dst.AddrMode]==Indirect)
	Format:	**IndirectOperand**
49:48	**Dst.HorzStride**	
	Format:	**HorzStride**
47	**Src1.IsImm**	
	This field indicate that Source 1 operand is carrying an immediate value.	
	Value	**Name**
	0	false **[Default]**
	1	true
46	**Src0.IsImm**	
	This field indicate that Source 0 operand is carrying an immediate value.	
	Value	**Name**
	0	false **[Default]**
	1	true
45:44	**Src0.Mod**	
	Format:	**SrcMod**
43:40	**Src0.DataType**	
	Exists If:	([Src0.IsImm]==false)
	Format:	**RegDataType**
43:40	**Src0.DataType**	
	Exists If:	([Src0.IsImm]==true)
	Format:	**ImmDataType**
39:36	**Dst.DataType**	

		Format: RegDataType
	35	**Dst.AddrMode** Format: AddrMode
	34	**Saturate** Format: Saturate
	33	**AccWrCtrl** Format: AccWrCtrl
	32	**AtomicCtrl** Format: AtomicCtrl
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

	30	**Reserved**
	29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

	28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.

		1 | Negative | Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
	27:24	**PredCtrl** Format: **PredCtrl** This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.
	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Arithmetic Shift Right

asr - Arithmetic Shift Right

Source:	Eulsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	true
Source Modifier:	true

Perform component-wise arithmetic right shift of the bits in src0 by the shift count indicated in src1, storing the results in dst. If src0 has a signed type, insert copies of src0's sign bit in the number of MSBs indicated by the shift count. Otherwise insert 0 bits. When src0 is accumulator and/or source modifier is used with src0 the sign bit is inserted in MSBs which come from the additional precision. **Note**: For Word and DWord operands, the accumulators have 33 bits.
In QWord mode, the shift count is taken from the low six bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 63. Otherwise the shift count is taken from the low five bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 31. The operation uses QWord mode if src0 or dst has the Q or UQ type but not if src1 is the only operand with the Q or UQ type. For positive values, this operation is src0 / 2shiftCount and for negative values, this operation is src0 / 2shiftCount - 1.

Format:

```
[(pred)] asr[.cmod] (exec_size) dst src0 src1
```

Programming Notes

If src0 is -1, the result is -1 regardless of the shift count.

For unsigned src0 types, asr and shr produce the same result.

Syntax

```
[(pred)] asr[.cmod] (exec_size) reg reg reg
 [(pred)] asr[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.channel[n] ) {

         shiftCnt = src0 or dst has Q or UQ type ? src1.chan[n] & 0x3F : src1.chan[n] &
0x1F
         if (src0.chan[n] >= 0) {
             dst.chan[n] = src0.chan[n] » shiftCnt;
         } else {
             int maskLSB = pow(2, shiftCnt) - 1;
             if ( maskLSB & src0.chan[n] == 0 ) {
                 dst.chan[n] = sign(src0.chan[n]) * ((abs)src0.chan[n] » shiftCnt);
             } else {
                 dst.chan[n] = sign(src0.chan[n]) * ((abs)src0.chan[n] » shiftCnt) - 1;
             }
         }
```

asr - Arithmetic Shift Right

```
        }
    }
```

Src Types	Dst Types
*B,*W,*D	*B,*W,*D

DWord	Bit	Description
0..3	127:126	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	127:96	**Src1.ImmValue[31:0]** Exists If: ([Src1.IsImm]==true)
	125:122	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	121:120	**Src1.Mod** Exists If: ([Src1.IsImm]==false) Format: **SrcMod**
	119:116	**Src1.VertStride** Exists If: ([Src1.IsImm]==false) Format: **VertStride**
	115:113	**Src1.Width** Exists If: ([Src1.IsImm]==false) Format: **Width**
	112	**Src1.AddrMode** Exists If: ([Src1.IsImm]==false) Format: **AddrMode**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect) Format: **IndirectOperand**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct) Format: **DirectOperand**
	97:96	**Src1.HorzStride** Exists If: ([Src1.IsImm]==false) Format: **HorzStride**
	95:92	**CondCtrl** Format: **FlagModifier**

asr - Arithmetic Shift Right

	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==true) Format: **ImmDataType**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==false) Format: **RegDataType**
	87:84	**Src0.VertStride** Format: **VertStride**
	83:81	**Src0.Width** Format: **Width**
	80	**Src0.AddrMode** Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value — Name

	0	false **[Default]**
	1	true

45:44 Src0.Mod

Format:	**SrcMod**

43:40 Src0.DataType

Exists If:	([Src0.IsImm]==false)
Format:	**RegDataType**

43:40 Src0.DataType

Exists If:	([Src0.IsImm]==true)
Format:	**ImmDataType**

39:36 Dst.DataType

Format:	**RegDataType**

35 Dst.AddrMode

Format:	**AddrMode**

34 Saturate

Format:	**Saturate**

33 AccWrCtrl

Format:	**AccWrCtrl**

32 AtomicCtrl

Format:	**AtomicCtrl**

31 MaskCtrl

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 Reserved

29 CmptCtrl

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

		<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>NoCompaction **[Default]**</td><td>No compaction. 128-bit native instruction supporting all instruction options.</td></tr><tr><td>1</td><td>Compacted</td><td>Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.</td></tr></table>
	28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields <table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>Positive **[Default]**</td><td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td></tr><tr><td>1</td><td>Negative</td><td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td></tr></table>
	27:24	**PredCtrl** <table><tr><td>Format:</td><td>**PredCtrl**</td></tr></table> This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.
	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** <table><tr><td>Format:</td><td>**ChanOff**</td></tr></table> This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** <table><tr><td>Format:</td><td>**ExecSize**</td></tr></table> This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** <table><tr><td>Format:</td><td>**Header**</td></tr></table>

Average

avg - Average

Source:	Eulsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	true
Source Modifier:	true

The avg instruction performs component-wise integer average of src0 and src1 and stores the results in dst. An integer average uses integer upward rounding. It is equivalent to increment one to the addition of src0 and src1 and then apply an arithmetic right shift to this intermediate value.

Format:

```
      The avg instruction performs component-wise integer average of src0 and src1 and
stores the results in dst. An integer average uses integer upward rounding. It is
equivalent to increment one to the addition of src0 and src1 and then apply an arithmetic
right shift to this intermediate value.
```

Syntax

```
[(pred)] avg[.cmod] (exec_size) reg reg reg
 [(pred)] avg[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         dst.chan[n] = (src0.chan[n] + src1.chan[n] + 1) » 1;// Use arithmetic shift
right.
     }
 }
```

Src Types	Dst Types
*B,*W,*D	*B,*W,*D

DWord	Bit	Description
0..3	127:126	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	127:96	**Src1.ImmValue[31:0]** Exists If: ([Src1.IsImm]==true)
	125:122	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	121:120	**Src1.Mod**

avg - Average

	Exists If:	([Src1.IsImm]==false)
	Format:	**SrcMod**
119:116	**Src1.VertStride**	
	Exists If:	([Src1.IsImm]==false)
	Format:	**VertStride**
115:113	**Src1.Width**	
	Exists If:	([Src1.IsImm]==false)
	Format:	**Width**
112	**Src1.AddrMode**	
	Exists If:	([Src1.IsImm]==false)
	Format:	**AddrMode**
111:98	**Src1.Operand**	
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)
	Format:	**IndirectOperand**
111:98	**Src1.Operand**	
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)
	Format:	**DirectOperand**
97:96	**Src1.HorzStride**	
	Exists If:	([Src1.IsImm]==false)
	Format:	**HorzStride**
95:92	**CondCtrl**	
	Format:	**FlagModifier**
91:88	**Src1.DataType**	
	Exists If:	([Src1.IsImm]==true)
	Format:	**ImmDataType**
91:88	**Src1.DataType**	
	Exists If:	([Src1.IsImm]==false)
	Format:	**RegDataType**
87:84	**Src0.VertStride**	
	Format:	**VertStride**
83:81	**Src0.Width**	
	Format:	**Width**
80	**Src0.AddrMode**	
	Format:	**AddrMode**
79:66	**Src0.Operand**	
	Exists If:	([Src0.AddrMode]==Direct)

	Format:	**DirectOperand**
79:66	**Src0.Operand**	
	Exists If:	([Src0.AddrMode]==Indirect)
	Format:	**IndirectOperand**
65:64	**Src0.HorzStride**	
	Format:	**HorzStride**
63:50	**Dst.Operand**	
	Exists If:	([Dst.AddrMode]==Direct)
	Format:	**DirectOperand**
63:50	**Dst.Operand**	
	Exists If:	([Dst.AddrMode]==Indirect)
	Format:	**IndirectOperand**
49:48	**Dst.HorzStride**	
	Format:	**HorzStride**
47	**Src1.IsImm**	
	This field indicate that Source 1 operand is carrying an immediate value.	
	Value	**Name**
	0	false **[Default]**
	1	true
46	**Src0.IsImm**	
	This field indicate that Source 0 operand is carrying an immediate value.	
	Value	**Name**
	0	false **[Default]**
	1	true
45:44	**Src0.Mod**	
	Format:	**SrcMod**
43:40	**Src0.DataType**	
	Exists If:	([Src0.IsImm]==false)
	Format:	**RegDataType**
43:40	**Src0.DataType**	
	Exists If:	([Src0.IsImm]==true)
	Format:	**ImmDataType**
39:36	**Dst.DataType**	
	Format:	**RegDataType**
35	**Dst.AddrMode**	
	Format:	**AddrMode**

34 **Saturate**

Format:	Saturate

33 **AccWrCtrl**

Format:	AccWrCtrl

32 **AtomicCtrl**

Format:	AtomicCtrl

31 **MaskCtrl**

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 **Reserved**

29 **CmptCtrl**

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

avg - Average

	27:24	**PredCtrl** Format: **PredCtrl** This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.
	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

AVP_BSD_OBJECT

AVP_BSD_OBJECT

Source: VideoCS

Length Bias: 2

The AVP Pipeline is selected with the **Media Instruction Opcode "8h"** for all AVP Commands. Each AVP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

The AVP_BSD_OBJECT command sends to HW a tile at a time from an AV1 bitstream, starting with the first coded byte of the tile, not including the prefixed tile byte size. The bit stream of a tile, tile group, and of a frame may end with trailing bits and extra padding zero bytes. The prefixed tile byte size includes all the trailing bits and padding zero bytes at the end of a tile.
Each tile's coded/compressed bitstream is started and ended at a byte boundary.
HW is not required to parse the trailing bits and padding zero bytes. HW can stop processing right after it has completed the decoding of the last block in the tile. Potentially, error checking can be implemented to detect the trailing bits and padding zeros, but is not implemented in this generation of AVP Pipeline.
here can be multiple tiles in an AV1 frame and thus this command can be issued multiple times per frame. A coded frame minimum has at least 1 tile definition, i.e., a tile can cover the entire frame, unless the frame size exceeds the max allowed tile size limits in pixels, then the frame must contain more than 1 tile. There is no compressed header in AV1, hence AVP_BSD_OBJECT command is only used to process the bitstream of each individual tile of a frame.
The AVP_BSD_OBJECT command must be the last command issued in the sequence of batch commands before the AVP Pipeline starts decoding. Prior to issuing this command, it is assumed that all configuration parameters needed by the AVP Pipeline have been loaded in a specific order, including workload configuration registers and configuration tables. When this command is issued, the AVP Pipeline is waiting for bitstream data to be presented to its bitstream input shift register.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 3h Codec/Engine Name Format: OpCode Codec/Engine Name = AV1 = 3h
	22:16	**Media Instruction Command** Default Value: 20h AVP_BSD_OBJECT_STATE Format: OpCode
	15:12	**Reserved** Access: RO

AVP_BSD_OBJECT

		Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). **Value** / **Name** 1h
1	31:0	**Tile Indirect BSD Data Length** Format: U32 It specifies the compressed bitstream byte size of a tile. Each tile is started and ended at a byte boundary. It has the same value as the prefixed tile byte size read from the bitstream at the beginning of a tile. The Data Length does not include this prefixed tile byte size, but does include any zero padding bytes at the end of a tile. Error Checking : 1) only when tile bitstream has run out when the last block of the current tile has not been decoded. Set the error bit in MMIO, and perform error concealment to fill in the missing decoded block(s). 2) there is no checking when there are bytes remaining after the last block of the current tile has been decoded. Note : HW AVP decoding pipeline is not required to parse the trailing bits and padding zeros at the end of a tile, tile group, and of a frame.

Value	Name	Description
[1,4294967295]	Tile_Bitstream_Data_Length	It has a valid range of 1 to 4294967295 (2^32-1) bytes. Zero byte length is not allowed. 2^31 should be sufficient to handle a 16Kx16K,12-bit,4:4:4 frame size with a compression ratio of 1. Note : different bitstream LEVELs are having different requirement on the minimum compression ratio.

2	31:0	**Tile Indirect Data Start Address** Format: U32 Specifies the byte-aligned graphics memory starting address of a tile in the bitstream of a frame relative to the **BSD Indirect Object Base Address**. Each tile's coded bitstream is started and ended at a byte boundary. A frame with multiple tiles is coded as one bitstream. Indirect Data Start Address for each tile is equivalent to an address offseted from the starting address of the frame bitstream.

AVP_IND_OBJ_BASE_ADDR_STATE

AVP_IND_OBJ_BASE_ADDR_STATE

Source: VideoCS

Length Bias: 2

The AVP Pipeline is selected with the **Media Instruction Opcode "8h"** for all AVP Commands. Each AVP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

The AVP_IND_OBJ_BASE_ADDR_STATE command is used to define the indirect object base address of the AV1 compressed bitstream in graphics memory. This is a frame level command issued in both encoding and decoding processes.
Although a frame is coded as separate tiles, all tiles' compressed bitstream are still required to line up sequentially as one AV1 bitstream. Hence, there is only one Indirect Object Base Address for the entire AV1 coded frame. If the frame contains more than 1 tiles, the BSD Object Command will be issued multiple times, once for each tile and with its own tile bitstream starting memory address.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 3h Codec/Engine Name Format: OpCode Codec/Engine Name = AVP = 3h
	22:16	**Media Instruction Command** Default Value: 3h AVP_IND_OBJ_BASE_ADDR_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 04h; Name:
1..2	63:0	**AVP Indirect Bitstream Object Base Address** Format: SplitBaseAddress4KByteAligned

AVP_IND_OBJ_BASE_ADDR_STATE		
		Description Decoder: Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the AVP_BSD_OBJECT command for reading each tile's compressed bitstream in a frame.
3	31:0	**AVP Indirect Bitstream Object Memory Address Attributes** Format: **MemoryAddressAttributes**
4..5	63:0	**Reserved** Access: RO Format: MBZ

AVP_INLOOP_FILTER_STATE

AVP_INLOOP_FILTER_STATE

Source: VideoCS

Length Bias: 2

The AVP_INLOOP_FILTER_STATE command provides all the frame level syntax elements and derived parameters that are needed for the processing of all the post in-loop filters present in the AV1 codec, except the Luma and Chroma x0_qn which are tile based derived parameters. This includes the Deblocker, the CDEF (Constrained Directional Enhancement Filter), the HSRF (Horizontal-only Super-Resolution Filter), and the LRF (Loop Restoration Filter). These syntax elements can be changed in the bitstream from frame to frame.

All Post In-Loop Filters are inherently frame-based filtering, but when implemented in a HW pipeline, the filtering process is performed in tile based and in a block by block fashion. In the addition to these frame and tile level states, there are additional syntax elements and derived parameters that are generated at SuperBlock level, and are not described here.

Each of these 4 Post In-Loop Filters can be controlled independently and each can be enabled or disabled independently. Except the HSRF, all the other 3 filters have separate controls for each color plane as well. To disable a Post In-Loop Filter, its control parameter(s) are set to 0 - the default state.

This command should be issued once per tile, even if no post in-loop filter is enabled for decoding the current frame. When in frame lossless mode or when IntraBC is enabled, all the Post In-Loop Filters are disabled for all color planes, this command will provide the default values for all parameters. All syntax elements are then assumed a value of 0, except otherwise specified in each field of this State Command.

When it is in monochrome video, no filter parameter for the two chroma planes is present in the bitstream.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: Opcode
	28:27	**Pipeline Type** Default Value: 2h Format: Opcode
	26:23	**Media Instruction Opcode** Default Value: 3h Codec/Engine Name Format: OpCode Codec/Engine Name = AV1 = 3h
	22:16	**Media Instruction Command** Default Value: 33h AVP_INLOOP_FILTER_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ

AVP_INLOOP_FILTER_STATE

	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: Dh; Name:
1	31	**Deblocker Filter Delta LF Present Flag** Format: U1 set to 1, specifies that additional loop filter delta values are present at the superblock level in the bitstream. set to 0, specifies that no additional loop filter delta values are present at the superblock level in the bitstream. It is the frame level syntax element delta_if_present flag. It is present in the bitstream, only if delta_q_present is set to 1. If delta_q_present is set to 0, delta_lf_present flag is not present, and is defaulted to 0.
	30	**Deblocker Filter Delta LF Multi Flag** Format: U1 set to 1, specifies that at the Superblock level, separate loop filter deltas (mulitple deltas)are sent for 1) horizontal Luma Y edges, 2) vertical Luma edges, 3) both horizontal and vertical Chroma U edges, and 4) for both horizontal and vertical Chroma V edges. set to 0, specifies that at the Superblock level, the same loop filter delta is used for all edges of all color planes (Y, U and V). It is the frame level syntax delta_lf_multi. It is present in the bitstream, when delta_lf_present flag is set to 1. If delta_lf_present flag is set to 0,delta_lf_multi flag is not present, and is defaulted to 0.
	29:28	**Deblocker Delta LF Resolution** Format: U2 It specifies the number of left shift which should be applied to decoded loop filter delta values. It is in the range of [0..3]. (no shft, and therefore no scaling). It is theframe level syntax element delta_lf_res. It is present in the bitstream, when delta_lf_present flag is set to 1. If delta_lf_present flag is set to 0,delta_lf_resis not present, and is defaulted to 0. Note : it is used to derive one part of the final filter levels: lvl +=read_delta_lflevel() * (1 « delta_lf_res). Note :But in the reference C model, the same name is used for the derived parameter: delta_lf_res = 1 « (delta_lf_res), which can take a 4-bit of value[1, 2, 4 or 8] instead.
	27	**Deblocker Filter Mode Ref Delta Enable Flag** Format: U1

		set to 1, means that the filter levels depends on the mode and reference frame used to predict a block. set to 0, means that the filter level does not depend on the mode and reference frame. Default is 0. It is the frame level syntax elementloop_filter_delta_enabled, or named asmode_ref_delta_enabled.
	26:24	**Deblocker Filter Sharpness Level** Format: U3 It specifies the sharpness level of the deblocker. It is used to compute the deblocker filter limits (lim and mblim) for each of the 64 possible values of a filter level.The deblocker filter levels and the deblock filter sharpness together determine when a block edge is filtered, and by how much the filtering can change the sample values. It is the frame level syntax elementloop_filter_sharpness, or named as sharpness_level. It is in the range of [0..7]. Default is 0.
	23:18	**Chroma V Deblocker Filter Level** Format: U6 It specifies the deblocker filter strength for both the horizontal (using vertical filters) and the vertical (using horizontal filters) edges of the Chroma Vplane. It is the frame level syntax element loop_filter_level[3], or named as filter_level_v.It is present, [only if filter_level[0] and filter_level[1] are not both set to 0 AND only if the current frame is not a monochrome]. If not present, it is default to 0. It is in the range of [0..63]. Default is 0.
	17:12	**Chroma U Deblocker Filter Level** Format: U6 It specifies the deblocker filter strength for both the horizontal (using vertical filters) and the vertical (using horizontal filters) edges of the Chroma U plane. It is the frame level syntax element loop_filter_level[2], or named as filter_level_u. It is present, [only if filter_level[0] and filter_level[1] are not both set to 0 AND only if the current frame is not a monochrome].If not present, it is default to 0. It is in the range of [0..63]. Default is 0.
	11:6	**Luma Y Deblocker Filter Level Horizontal** Format: U6 It specifies the deblocker filter strength for the horizontal edges (using vertical filters) of the Luma plane. It is the frame syntax element loop_filter_level[1]. It is in the range of [0..63]. Default is 0. Settingloop_filter_level[0] =loop_filter_level[1] = 0, disables the deblocker filter.

AVP_INLOOP_FILTER_STATE

	5:0	**Luma Y Deblocker Filter Level Vertical** Format: U6 It specifies the deblocker filter strength for the vertical edges (using horizontal filters) of the Luma plane. It is the frame syntax element loop_filter_level[0]. It is in the range of [0..63]. Default is 0. Settingloop_filter_level[0] =loop_filter_level[1] = 0, disables the deblocker filter.
2	31	**Reserved** Access: RO Format: MBZ
	30:24	**Deblocker Filter Ref Deltas[3]** Format: S6 It is in 7-bits 2's complement within the range of [-64to +63]. Initialize to 0 bysetup_past_independence(). If this syntax element is not present in the bitstream (i.e. update_ref_delta=0), Deblocker Filter Ref Deltas[Ref=0 to 7] maintains its value from previous frame. Refer to the full description in Deblocker Filter Ref Deltas[0].
	23	**Reserved** Access: RO Format: MBZ
	22:16	**Deblocker Filter Ref Deltas[2]** Format: S6 It is in 7-bits 2's complement within the range of [-64to +63]. Initialize to 0 bysetup_past_independence(). If this syntax element is not present in the bitstream (i.e. update_ref_delta=0), Deblocker Filter Ref Deltas[Ref=0 to 7] maintains its value from previous frame. Refer to the full description in Deblocker Filter Ref Deltas[0].
	15	**Reserved** Access: RO Format: MBZ
	14:8	**Deblocker Filter Ref Deltas[1]** Format: S6 It is in 7-bits 2's complement within the range of [-64to +63]. Initialize to 0 bysetup_past_independence(). If this syntax element is not present in the bitstream (i.e. update_ref_delta=0), Deblocker Filter Ref Deltas[Ref=0 to 7] maintains its value from previous frame. Refer to the full description in Deblocker Filter Ref Deltas[0].

AVP_INLOOP_FILTER_STATE

	7	**Reserved** Access: RO Format: MBZ
	6:0	**Deblocker Filter Ref Deltas[0]** Format: S6 It specifiesthe adjustmentneeded for the deblockerfilter level based on which one of the 8 possible reference framesis in used - Deblocker Filter Ref Deltas[Ref=0 to 7]. Ref=0 to 7 is defined as follows: [0] = INTRA_FRAME [1] = LAST_FRAME [2] = LAST2_FRAME [3] = LAST3_FRAME [4] = GOLDEN_FRAME [5] = BWDREF_FRAME [6] = ALTREF2_FRAME [7] = ALTREF_FRAME Deblocker Filter Ref Deltas[] is used to pre-computethe final deblocker filter level for all blocks within a segment, lvl[seg_id][horz_or_vert][ref_frame[0]][block_coding_mode]. The pre-compute can take place either at the frame header (if delta_lf_present_flag ==0), or at the block level(if delta_lf_present_flag ==0); but the equations and clamping used are different in these 2 cases. It is the frame level syntax elementloop_filter_ref_deltas[Ref=0 to 7], or named asref_deltas[Ref=0 to 7]. It is in 7-bits 2's complement within the range of [-64to +63]. Initialize to 1bysetup_past_independence(). If this syntax element is not present in the bitstream (i.e. update_ref_delta=0), Deblocker Filter Ref Deltas[Ref=0 to 7] maintains its value from previous frame. Note : The 8 elements of the Deblock Filter Ref Deltas[Ref=0 to 7] are initialized differently inside thesetup_past_independence(). Deblock Filter Ref Deltas[INTRA_FRAME] is set equal to 1. Deblock Filter Ref Deltas[LAST_FRAME] is set equal to 0. Deblock Filter Ref Deltas[LAST2_FRAME] is set equal to 0. Deblock Filter Ref Deltas[LAST3_FRAME] is set equal to 0. Deblock Filter Ref Deltas[BWDREF_FRAME] is set equal to 0. Deblock Filter Ref Deltas[GOLDEN_FRAME] is set equal to -1. Deblock Filter Ref Deltas[ALTREF_FRAME] is set equal to -1. Deblock Filter Ref Deltas[ALTREF2_FRAME] is set equal to -1.
3	31	**Reserved** Access: RO Format: MBZ
	30:24	**Deblocker Filter Ref Deltas[7]** Format: S6 It is in 7-bits 2's complement within the range of [-64to +63]. Initialize to -1bysetup_past_independence().

		If this syntax element is not present in the bitstream (i.e. update_ref_delta=0), Deblocker Filter Ref Deltas[Ref=0 to 7] maintains its value from previous frame. Refer to the full description in Deblocker Filter Ref Deltas[0].
	23	**Reserved** Access: RO Format: MBZ
	22:16	**Deblocker Filter Ref Deltas[6]** Format: S6 It is in 7-bits 2's complement within the range of [-64to +63]. Initialize to -1bysetup_past_independence(). If this syntax element is not present in the bitstream (i.e. update_ref_delta=0), Deblocker Filter Ref Deltas[Ref=0 to 7] maintains its value from previous frame. Refer to the full description in Deblocker Filter Ref Deltas[0].
	15	**Reserved** Access: RO Format: MBZ
	14:8	**Deblocker Filter Ref Deltas[5]** Format: S6 It is in 7-bits 2's complement within the range of [-64to +63]. Initialize to -1bysetup_past_independence(). If this syntax element is not present in the bitstream (i.e. update_ref_delta=0), Deblocker Filter Ref Deltas[Ref=0 to 7] maintains its value from previous frame. Refer to the full description in Deblocker Filter Ref Deltas[0].
	7	**Reserved** Access: RO Format: MBZ
	6:0	**Deblocker Filter Ref Deltas[4]** Format: S6 It is in 7-bits 2's complement within the range of [-64to +63]. Initialize to 0 bysetup_past_independence(). If this syntax element is not present in the bitstream (i.e. update_ref_delta=0), Deblocker Filter Ref Deltas[Ref=0 to 7] maintains its value from previous frame. Refer to the full description in Deblocker Filter Ref Deltas[0].
4	31:15	**Reserved** Access: RO Format: MBZ

AVP_INLOOP_FILTER_STATE

	14:8	**Deblocker Filter Mode Deltas[1]** Format: S6 It is in 7-bits 2's complement within the range of [-64to +63]. Initialize to 0 bysetup_past_independence(). If this syntax element is not present in the bitstream (i.e. update_mode_delta=0), Deblocker Filter ModeDeltas[BPM=0 to 1] maintains its value from previous frame. Refer to the full description in Deblocker Filter Mode Deltas[0].
	7	**Reserved** Access: RO Format: MBZ
	6:0	**Deblocker Filter Mode Deltas[0]** Format: S6 It specifiesthe adjustmentneeded for the deblockerfilter level based on which block prediction mode is in used - Deblocker Filter Mode Deltas[BPMode=0 to 1]. BPM=0 to 1is defined by mode_lf_lut[Block Prediction Mode] ; Block Prediction Mode can be one of the 25 possible block prediction modes being defined in AV1. BPM=0, for all 13 intra block prediction modes, for the GLOBALMV inter block prediction mode, and for the GLOBAL_GLOBALMV inter compound block prediction mode. BPM =1, for all the other 10 inter block prediction modes (3 inter block prediction modes and 7 inter compound block prediction modes) Deblocker Filter Mode Deltas[] is used to pre-computethe final deblocker filter level for all blocks within a segment, lvl[seg_id][horz_or_vert][ref_frame[0]][mode_lf_lut[block_prediction_mode]]. The pre-compute can take place either at the frame header (if delta_lf_present_flag ==0), or at the block level(if delta_lf_present_flag ==0); but the equations and clamping used are different in these 2 cases. It is the frame level syntax elementloop_filter_mode_deltas[BPM=0 to 1], or named as mode_deltas[BPM=0 to 1]. It is in 7-bits 2's complement within the range of [-64to +63]. Initialize to 0 bysetup_past_independence(). If this syntax element is not present in the bitstream (i.e. update_mode_delta=0), Deblocker Filter Mode Deltas[BPM=0 to 1] maintains its value from previous frame.
5	31:30	**CDEF Filter Damping Factor Minus3** Format: U2 It specifies the amount of damping in the deringing filter. It is theframe level syntax element, cdef_damping_minus_3. It is also named as cdef_damping (=cdef_damping_minus3+3). It takes on value in the range of [0 to 3]. Default is 0.
	29:28	**CDEF Bits** Format: U2 It is used for 2 purposes:

		1) [1 « cdef_bits] specifies the number of frame level CDEF Y and UV strengths to be read from the bitstream in the uncompressed header. 2) cdef_bits specifies the number of bits in the bitstream, at the block level, that needto be read for the CDEF strength to be applied. It is the frame level syntax element, cdef_bits. It is in the range of [0..3]. Default is 0. Note: to disable CDEF Filtering process, set cdef_bits to 0, and set cdef_y_strengths[0]=cdef_uv_strengths[0]=0.
	27:24	**Reserved** Access: RO Format: MBZ
	23:18	**CDEF Y Strength[3]** Format: U6 Refer to the full description in CDEF Y Strength[0].
	17:12	**CDEF Y Strength[2]** Format: U6 Refer to the full description in CDEF Y Strength[0].
	11:6	**CDEF Y Strength[1]** Format: U6 Refer to the full description in CDEF Y Strength[0].
	5:0	**CDEF Y Strength[0]** Format: U6 It specifies one of the 8 possible filter strengths for the CDEF Luma Y Filter. It is the frame level syntax element cdef_y_strength[i=0 to 7], or also named simply as cdef_strengths[i=0 to 7]. The number of active filter strengths in used for the current frame is equal to [1 « cdef_bits], which can only be [1, 2, 4 or 8]. Each strength is in the range of [0 to 63]. For allfilter strengths that are not present in the bitstream, they aredefaulted to 0. Note: to disable CDEF Filtering process, set cdef_bits to 0, and set cdef_y_strengths[0]=cdef_uv_strengths[0]=0.
6	31:24	**Reserved** Access: RO Format: MBZ
	23:18	**CDEF Y Strength[7]** Format: U6 Refer to the full description in CDEF Y Strength[0].

AVP_INLOOP_FILTER_STATE

	17:12	**CDEF Y Strength[6]** Format: U6 Refer to the full description in CDEF Y Strength[0].
	11:6	**CDEF Y Strength[5]** Format: U6 Refer to the full description in CDEF Y Strength[0].
	5:0	**CDEF Y Strength[4]** Format: U6 Refer to the full description in CDEF Y Strength[0].
7	31:24	**Reserved** Access: RO Format: MBZ
	23:18	**CDEF UV Strength[3]** Format: U6 Refer to the full description in CDEF UVStrength[0].
	17:12	**CDEF UV Strength[2]** Format: U6 Refer to the full description in CDEF UVStrength[0].
	11:6	**CDEF UV Strength[1]** Format: U6 Refer to the full description in CDEF UVStrength[0].
	5:0	**CDEF UV Strength[0]** Format: U6 It specifies one of the 8 possible filter strengths for the CDEF Chroma U/VFilter. Chroma U and V share the same strength specification. It is the frame level syntax element cdef_uv_strength[i=0 to 7], or named as cdef_uv_strengths[i=0 to 7]. The number of active filter strengths in used for the current frame is equal to [1 « cdef_bits], which can only be [1, 2, 4 or 8]. For monochrome video, there is no Chroma UV strength in the bitstream, either. Each strength is inthe range of [0to 63]. The 7th and 8th-bit are ignored. For allfilter strengths that are not present in the bitstream, they aredefaulted to 0. Note: to disable CDEF Filtering process, set cdef_bits to 0, and set cdef_y_strengths[0]=cdef_uv_strengths[0]=0.

AVP_INLOOP_FILTER_STATE

8	31:24	**Reserved** Access: RO Format: MBZ
	23:18	**CDEF UV Strength[7]** Format: U6 Refer to the full description in CDEF UVStrength[0].
	17:12	**CDEF UV Strength[6]** Format: U6 Refer to the full description in CDEF UVStrength[0].
	11:6	**CDEF UV Strength[5]** Format: U6 Refer to the full description in CDEF UVStrength[0].
	5:0	**CDEF UV Strength[4]** Format: U6 Refer to the full description in CDEF UVStrength[0].
9	31:21	**Reserved** Access: RO Format: MBZ
	20:16	**Super-Res Denom** Format: U5 It specifies an integer denominator of a fixed point fractional number (scaling factor) which is used to scale down the current frame width (in pixels) for the subsequent block level decoding process of its bitstream. Super-resolution is only applied to the horizontal direction, so this scaling factor is applied to the frame width only. The numerator of this down-scaling factor is always set to 8. The reduced (scaled down) frame width in pixels = (upscaled frame width in pixels * 8 + (super-res denom»1)) / super-res denom. This divsion is done by Driver. It is derived from the 3-bits frame level syntax element, coded_denom. Super-res denom = coded_denom + 9, which is always > 8 (i.e. downscaling factor is always < 1.0). That is, the super-resolution down-scaling factor can go from [8/9 to 8/16], and the corresponding up-scaling factor can go from [9/8 to 16/8]. If super-resolution is not enabled, super-res denom is not present in the bitstream,its valueis defaulted to 8 (i.e no scaling). Hence, Super-Rest Denom is in the range of [8.. 16]. Value 0 to 7 are not allowed. Different frames in a video sequence can have different super-res denom. Adjacent frames can all have different super-res denoms. Super-res denom is used in multiple functions of the Super-resolution and the Loop Restoration processes, but no fixed-point division is needed.

		Note : if not monochrome video, the same scaling factor and super-resolution process are applied to the two chroma planes as well; otherwise, only the Luma plane is being processed. Following restrictions apply in Encoder Mode if Wiener Filter enabled: In Single pipe: Only valid values are 10, 12, 14 and 16 In Multi pipe: Only valid value is 16 otherwise all values are valid..
	15:0	**Super-Res Upscaled Frame Width Minus1** Format: U16 It specifies the super-resolution upscaled frame width in pixels, without padding on the right and bottom of the frame. Since, super-resolution is only applied to the horizontal frame width, there is no scaling to the vertical frame height. The input and output of the Loop Restoration Filter and all reference frames in the Display Buffer (DPB) are in Super-Res Upscaled Frame Width. When super-resolution is enabled, the bitstream is coded with the reduced frame width. Hence, the current frame width programmed in the AVP_PIC_STATE is the derived reduced frame width without padding on the right and bottom border of the frame. The scaled downframe width (AVP_PIC_STATE frame width minus1 + 1)= (upscaled frame width * 8 + (super-res denom»1)) / super-res denom. Upscaled frame width is derived as follows: A) In KEY FRAME or INTRA-ONLY NON-KEY FRAME: 1) when frame_size_override_flag is set to 1, it is the frame level syntax element frame_width_minus1. 2) when frame_size_override_flag is set to 1, it is the sequence level syntax element max_frame_width_minus1. B) In INTER FRAME: 1) if the current frame size can be inferred from one of the 7 possible reference frames, it is the derived frame width minus1. Otherwise, 2) when frame_size_override_flag is set to 1, it is the frame level syntax element frame_width_minus1. 3) when frame_size_override_flag is set to 1, it is the sequence level syntax element max_frame_width_minus1. Max frame size is 64K, but intel only support up to 16K. Hence, bit 14 and 15 are not used. Default is 7, for minimum frame width is 8 pixels. Note : if upscaling factor > 1.0, AVP_PIC_STATE Frame Width must < 16K. Note : both the upscaled frame width and the downscaled frame width are provided to the AVP HW pipeline, so that HW does not need to perform the division in the scaling process.
10	31:11	**Reserved** Access: RO Format: MBZ
	10	**Use Same Loop Restoration Unit Size for Chromas UV Flag** Format: U1 It specifies how the Chroma UV Loop Restoration Unit size should be derived from that of the

		Luma. Chroma U and Chroma V are having the same LRU Size. Set to 1, if Chroma LRU size is the same as that of the Luma. Set to 0, if Chroma LRU size is subsampled that of the Luma in 4:2:0 by half in each dimension. Default is 0. It is the frame level syntax element, lr_uv_shift, which is present only if current video is a 4:2:0 and not both Chroma plane having filter type set to RESTORE_NONE. For 4:2:2 and 4:4:4, LRU size for Chromas UV is always the same as that of Luma. For monochrome, this field is ignored. Intel Encoder PAK only support LRU size = SuperBlock size = 64x64 pixels, this field must also be set to 0.
	9:8	**Loop Restoration Unit Size for Luma Y** Format: U2 It specifies the Loop Restoration Unit size in pixels for Luma Y. Loop Restoration Unit is a squaregrid inthe current frame for each color plane. Each Loop Restoration Unit of a color plane has its own set ofLoop Restoration Filter parameters. It is derived from at most 2 1-bit syntax element (lr_unit_shift) in the frame header which are present only if not all 3 color planes having filter type set to RESTORE_NONE. If all 3 color planes having filter type set to RESTORE_NONE, then Loop Restoration Unit Size for Luma plane is defaulted to 256x256. It is also named as LoopRestorationSize[i=0], or restoration_unit_size for Luma plane. But intel has mapped them to the following code: 0, for 0 size (Default, when LR is not enabled) 1, for 64x64 pixels LRU nominal size. 2, for 128x128 pixels LRU nominal size. 3, for 256x256 pixels LRU nominal size. Intel Encoder PAK only supportLRU size = SuperBlock size = 64x64 pixels. Note : LRU Luma Size cannot < Superblock Size. If Superblock Size is 128x128, LRU Luma Size cannot be 64x64.
	7:6	**Reserved** Access: RO Format: MBZ
	5:4	**Frame Loop Restoration Filter Type for Chroma V** Format: U2 It specifies the Frame Level Loop Restoration Filter Type for the Chroma Vplane. It is derived from 2 bits syntax element (lr_type) in the frame header, if not in a monochrome video. It is also named as framerestorationtype[i=2], orframe_restoration_type for Chroma V plane. It is in the range of [0..3]. For a monochrome video, it is defaulted to 0. 0, for RESTORE_NONE. 1, for RESTORE_WIENER. 2, for RESTORE_SGRPROJ (Dual Self-Guided Projection Filter).

		3, for RESTORE_SWITCHABLE (LRU level choice of NONE, WIENER or SGRPROJ). Each color plane can have its ownFrame Level Loop Restoration Filter Type specification. Note : to disable Loop Restoration Filtering in the current frame, the Frame Restoration Filter Type for all color planes must be set to RESTORE_NONE. Note : that the syntax element lr_type uses a different enum order for the four LR filter types (0 for RESTORE_NONE, 1 for SWITCHABLE, 2 for WIENER, and 3 for SGRPROJ). In Encoder Mode, Loop Restoration Filter is not supported, this field can only be set to RESTORE_NONE.
	3:2	**Frame Loop Restoration Filter Type for Chroma U** Format: U2 It specifies the Frame Level Loop Restoration Filter Type for the Chroma U plane. It is derived from 2 bits syntax element (lr_type) in the frame header, if not in a monochrome video. It is also named as framerestorationtype[i=1], orframe_restoration_type for Chroma U plane. It is in the range of [0..3]. For a monochrome video, it is defaulted to 0. 0, for RESTORE_NONE. 1, for RESTORE_WIENER. 2, for RESTORE_SGRPROJ (Dual Self-Guided Projection Filter). 3, for RESTORE_SWITCHABLE (LRU level choice of NONE, WIENER or SGRPROJ). Each color plane can have its ownFrame Level Loop Restoration Filter Type specification. Note : to disable Loop Restoration Filtering in the current frame, the Frame Restoration Filter Type for all color planes must be set to RESTORE_NONE. Note : that the syntax element lr_type uses a different enum order for the four LR filter types (0 for RESTORE_NONE, 1 for SWITCHABLE, 2 for WIENER, and 3 for SGRPROJ). In Encoder Mode, Loop Restoration Filter is not supported, this field can only be set to RESTORE_NONE.
	1:0	**Frame Loop Restoration Filter Type for Luma Y** Format: U2 It specifies the Frame Level Loop Restoration Filter Type for the Luma plane. It is derived from 2 bits syntax element (lr_type) in the frame header. It is also named as framerestorationtype[i=0], or frame_restoration_type for Luma plane. It is in the range of [0..3]. Default is 0. 0, for RESTORE_NONE. 1, for RESTORE_WIENER. 2, for RESTORE_SGRPROJ (Dual Self-Guided Projection Filter). 3, for RESTORE_SWITCHABLE (LRU level choice of NONE, WIENER or SGRPROJ). Each color plane can have its ownFrame Level Loop Restoration Filter Type specification. Note : to disable Loop Restoration Filtering in the current frame, the Frame Restoration Filter Type for all color planes must be set to RESTORE_NONE. Note : that the syntax element lr_type uses a different enum order for the four LR filter types (0 for RESTORE_NONE, 1 for SWITCHABLE, 2 for WIENER, and 3 for SGRPROJ). In Encoder Mode, Loop Restoration Filter is not supported, this field can only be set to

AVP_INLOOP_FILTER_STATE

		RESTORE_NONE.
11	31:16	**Reserved (for higher precision of x_step_qn)** Format: MBZ
	15:0	**Luma Plane x_step_qn** Format: U16 Derived parameter, specifies the increment of the super-res luma upscaling step in pixel position for the current frame.
12	31:0	**Luma Plane x0_qn** Format: S31 Derived parameters, specifies the starting offset (in 2's complement signed integer) of the super-res upscaling process for each tile in the original frame for Luma.
13	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Chroma Plane x_step_qn** Format: U16 Derived parameter, specifies the increment of the super-res chroma upscaling step in pixel position for the current frame.
14	31:0	**Chroma Plane x0_qn** Format: S31 Derived parameters, specifies the starting offset (in 2's complement signed integer) of the super-res upscaling process for each tile in the original frame for chroma.

AVP_INTER_PRED_STATE

AVP_INTER_PRED_STATE

Source: VideoCS

Length Bias: 2

The AVP Pipeline is selected with the **Media Instruction Opcode "8h"** for all AVP Commands. Each AVP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.
AVP supports a 8-reference frames display buffer. But at any given frame being decoded, only up to 7reference frames out of the 8 can be active. There are also further constraints on which of these 7frames can be used for forward and backward reference in the compound mode.
To simplify the decoder command sequence programming, this command is issued once for each inter-coded tile as well as for each intra-coded tile (such as in a KEY_FRAME, a DELAY_KEY_FRAME, an INTRA_ONLY_FRAME).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 3h Codec/Engine Name Format: OpCode Codec/Engine Name = AVP = 3h
	22:16	**Media Instruction Command** Default Value: 12h AVP_INTER_PRED_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: Dh — Name:
1	31:24	**Saved Order Hints for All References[0][3]** Format: U8
	23:16	**Saved Order Hints for All References[0][2]** Format: U8

AVP_INTER_PRED_STATE

	15:8	**Saved Order Hints for All References[0][1]** Format: U8
	7:0	**Saved Order Hints for All References[0][0]** Format: U8 This array specifies the saved order hints for all references for the purpose of computing the Motion Field Projections. Array indices are defined as [a reference frame index][the 7 reference indices of the reference frame]. Hence, the array indices' range are [0..6] [0..6]. Index value = 0, is set for LAST frame. Index value = 6 is set for ALTREF frame. Since a current frame can have up to 7 references, when each reference was previously decoded as a current frame, it could also have up to 7 references (references of a reference frame), hence totally max. there can be 7x7=49 references in the past. With each reference has its own order hint (previously read from the bitstream), a total of max. 49 order hints need to be saved and programmed here. Order Hint is acting like a form of time step, and can take on a value of [0..255] (8-bits unsigned). All order hints are syntax elements previously read from the frame headers in the bitstream (not a derived value). Note : for tiles of a KEY_FRAME, saved order hints should be all set to a value of 0.
2	31:24	**Active Reference Bitmask for Motion Field Projection** Format: U8 This field specifies which ones of the 7 references are involved in the Motion Field Projection. bit 0 corresponding to the LAST reference frame for the decoding current frame. bit 6corresponding to the ALTREF reference frame for decoding the current frame. bit 7 is reserved and must set to 0. This is an intel derived parameters. A reference has its corresponding bit in the bitmaskset to 1 if 1) motion_field_projection() is called for that reference from av1_setip_motion_field(); AND 2) motion_field_projection() for that reference does not return 0, meaning the following conditions have to be satisfied for that reference: a) Reference is available (i.e. frame buffer index has a valid value) b) Reference is not intra-only (i.e. neither a KEY FRAME nor an INTRA-ONLY FRAME) c) Reference is not scaled. Otherwise set to 0.
	23:16	**Saved Order Hints for All References[0][6]** Format: U8
	15:8	**Saved Order Hints for All References[0][5]** Format: U8
	7:0	**Saved Order Hints for All References[0][4]** Format: U8
3	31:24	**Saved Order Hints for All References[1][3]** Format: U8

AVP_INTER_PRED_STATE

	23:16	**Saved Order Hints for All References[1][2]** Format: U8
	15:8	**Saved Order Hints for All References[1][1]** Format: U8
	7:0	**Saved Order Hints for All References[1][0]** Format: U8 This array specifies the saved order hints for all references for the purpose of computing the Motion Field Projections. Array indices are defined as [a reference frame index][the 7 reference indices of the reference frame]. Hence, the array indices' range are [0..6] [0..6]. Index value = 0, is set for LAST frame. Index value = 6 is set for ALTREF frame. Since a current frame can have up to 7 references, when each reference was previously decoded as a current frame, it could also have up to 7 references (references of a reference frame), hence totally max. there can be 7x7=49 references in the past. With each reference has its own order hint (previously read from the bitstream), a total of max. 49 order hints need to be saved and programmed here. Order Hint is acting like a form of time step, and can take on a value of [0..255] (8-bits unsigned). All order hints are syntax elements previously read from the frame headers in the bitstream (not a derived value). Note : for tiles of a KEY_FRAME, saved order hints should be all set to a value of 0.
4	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**Saved Order Hints for All References[1][6]** Format: U8
	15:8	**Saved Order Hints for All References[1][5]** Format: U8
	7:0	**Saved Order Hints for All References[1][4]** Format: U8
5	31:24	**Saved Order Hints for All References[2][3]** Format: U8
	23:16	**Saved Order Hints for All References[2][2]** Format: U8
	15:8	**Saved Order Hints for All References[2][1]** Format: U8
	7:0	**Saved Order Hints for All References[2][0]** Format: U8 This array specifies the saved order hints for all references for the purpose of computing the Motion Field Projections.

AVP_INTER_PRED_STATE

		Array indices are defined as [a reference frame index][the 7 reference indices of the reference frame]. Hence, the array indices' range are [0..6] [0..6]. Index value = 0, is set for LAST frame. Index value = 6 is set for ALTREF frame. Since a current frame can have up to 7 references, when each reference was previously decoded as a current frame, it could also have up to 7 references (references of a reference frame), hence totally max. there can be 7x7=49 references in the past. With each reference has its own order hint (previously read from the bitstream), a total of max. 49 order hints need to be saved and programmed here. Order Hint is acting like a form of time step, and can take on a value of [0..255] (8-bits unsigned). All order hints are syntax elements previously read from the frame headers in the bitstream (not a derived value). Note : for tiles of a KEY_FRAME, saved order hints should be all set to a value of 0.
6	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**Saved Order Hints for All References[2][6]** Format: U8
	15:8	**Saved Order Hints for All References[2][5]** Format: U8
	7:0	**Saved Order Hints for All References[2][4]** Format: U8
7	31:24	**Saved Order Hints for All References[3][3]** Format: U8
	23:16	**Saved Order Hints for All References[3][2]** Format: U8
	15:8	**Saved Order Hints for All References[3][1]** Format: U8
	7:0	**Saved Order Hints for All References[3][0]** Format: U8 This array specifies the saved order hints for all references for the purpose of computing the Motion Field Projections. Array indices are defined as [a reference frame index][the 7 reference indices of the reference frame]. Hence, the array indices' range are [0..6] [0..6]. Index value = 0, is set for LAST frame. Index value = 6 is set for ALTREF frame. Since a current frame can have up to 7 references, when each reference was previously decoded as a current frame, it could also have up to 7 references (references of a reference frame), hence totally max. there can be 7x7=49 references in the past. With each reference has its own order hint (previously read from the bitstream), a total of max. 49 order hints need to be saved and programmed here. Order Hint is acting like a form of time step, and can take on a value of [0..255] (8-bits unsigned).

AVP_INTER_PRED_STATE

		All order hints are syntax elements previously read from the frame headers in the bitstream (not a derived value). Note : for tiles of a KEY_FRAME, saved order hints should be all set to a value of 0.
8	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**Saved Order Hints for All References[3][6]** Format: U8
	15:8	**Saved Order Hints for All References[3][5]** Format: U8
	7:0	**Saved Order Hints for All References[3][4]** Format: U8
9	31:24	**Saved Order Hints for All References[4][3]** Format: U8
	23:16	**Saved Order Hints for All References[4][2]** Format: U8
	15:8	**Saved Order Hints for All References[4][1]** Format: U8
	7:0	**Saved Order Hints for All References[4][0]** Format: U8 This array specifies the saved order hints for all references for the purpose of computing the Motion Field Projections. Array indices are defined as [a reference frame index][the 7 reference indices of the reference frame]. Hence, the array indices' range are [0..6] [0..6]. Index value = 0, is set for LAST frame. Index value = 6 is set for ALTREF frame. Since a current frame can have up to 7 references, when each reference was previously decoded as a current frame, it could also have up to 7 references (references of a reference frame), hence totally max. there can be 7x7=49 references in the past. With each reference has its own order hint (previously read from the bitstream), a total of max. 49 order hints need to be saved and programmed here. Order Hint is acting like a form of time step, and can take on a value of [0..255] (5-bits unsigned). All order hints are syntax elements previously read from the frame headers in the bitstream (not a derived value). Note : for tiles of a KEY_FRAME, saved order hints should be all set to a value of 0.
10	31:24	**Reserved** Access: RO Format: MBZ

AVP_INTER_PRED_STATE

	23:16	**Saved Order Hints for All References[4][6]** Format: U8
	15:8	**Saved Order Hints for All References[4][5]** Format: U8
	7:0	**Saved Order Hints for All References[4][4]** Format: U8
11	31:24	**Saved Order Hints for All References[5][3]** Format: U8
	23:16	**Saved Order Hints for All References[5][2]** Format: U8
	15:8	**Saved Order Hints for All References[5][1]** Format: U8
	7:0	**Saved Order Hints for All References[5][0]** Format: U8 This array specifies the saved order hints for all references for the purpose of computing the Motion Field Projections. Array indices are defined as [a reference frame index][the 7 reference indices of the reference frame]. Hence, the array indices' range are [0..6] [0..6]. Index value = 0, is set for LAST frame. Index value = 6 is set for ALTREF frame. Since a current frame can have up to 7 references, when each reference was previously decoded as a current frame, it could also have up to 7 references (references of a reference frame), hence totally max. there can be 7x7=49 references in the past. With each reference has its own order hint (previously read from the bitstream), a total of max. 49 order hints need to be saved and programmed here. Order Hint is acting like a form of time step, and can take on a value of [0..255] (8-bits unsigned). All order hints are syntax elements previously read from the frame headers in the bitstream (not a derived value). Note : for tiles of a KEY_FRAME, saved order hints should be all set to a value of 0.
12	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**Saved Order Hints for All References[5][6]** Format: U8
	15:8	**Saved Order Hints for All References[5][5]** Format: U8
	7:0	**Saved Order Hints for All References[5][4]** Format: U8

AVP_INTER_PRED_STATE

13	31:24	**Saved Order Hints for All References[6][3]** Format: U8
	23:16	**Saved Order Hints for All References[6][2]** Format: U8
	15:8	**Saved Order Hints for All References[6][1]** Format: U8
	7:0	**Saved Order Hints for All References[6][0]** Format: U8 This array specifies the saved order hints for all references for the purpose of computing the Motion Field Projections. Array indices are defined as [a reference frame index][the 7 reference indices of the reference frame]. Hence, the array indices' range are [0..6] [0..6]. Index value = 0, is set for LAST frame. Index value = 6 is set for ALTREF frame. Since a current frame can have up to 7 references, when each reference was previously decoded as a current frame, it could also have up to 7 references (references of a reference frame), hence totally max. there can be 7x7=49 references in the past. With each reference has its own order hint (previously read from the bitstream), a total of max. 49 order hints need to be saved and programmed here. Order Hint is acting like a form of time step, and can take on a value of [0..255] (8-bits unsigned). All order hints are syntax elements previously read from the frame headers in the bitstream (not a derived value). Note : for tiles of a KEY_FRAME, saved order hints should be all set to a value of 0.
14	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**Saved Order Hints for All References[6][6]** Format: U8
	15:8	**Saved Order Hints for All References[6][5]** Format: U8
	7:0	**Saved Order Hints for All References[6][4]** Format: U8

AVP_PIC_STATE

AVP_PIC_STATE

Source: VideoCS

Length Bias: 2

All AVP_PIC_STATE should stay the same for the whole frame even if AVP_PIC_STATE is re-programmed for every tiles.
The bitfields of AVP_PIC_STATE are defined either from
1) syntax elements of the uncompressed sequence header (received from sequence_header_obu) and of the uncompressed frame header (received from frame_header_obu),
2) or, parameters derived from 1).
Note : Bitstreams may contain several copies of the frame header (there can only be one frame_header_obu, but multiple redundant_frame_header_obu)interspersed with tile_group_obu to allow for greater error resilience. However, the copies must contain identical contents to the original frame_header_obu.
Note : there should be only one sequence_header_obu per video sequence.
Note : AVP pipeline is invoked to decode a frame from the bitstream, only if that frame has show_existing_frame flag (syntax element in the frame header) set to 0. For the case thatshow_existing_frame flag is set to 1, application and driver process the frame instead, no block level decoding is needed.
Note : Unlike VP9, AV1 does not have a compressed header. All the syntax elements defined in the AV1 sequence and frame level headers are not arithmetic coded, hence application and driver can directly read them off from the bitstream.
Note : the values of the sequence header/level syntax elements and their derived parameters are to last throughout all frames in the video sequence, until the next Sequence Header OBU is received that may change them. But some sequence header/level syntax elements or their derived parameters may further qualified by frame header/level syntax elements and their derived parameters, then these type of syntax elements and their derived parameters can be changed frame to frame.
Note : the values of the frame header/level syntax elements and their derived parameters can be changed from frame to frame.
Note : there are some syntax elements and their derived parameters can be changed only at KEY FRAME. Hence, the values of these type of syntax elements and their derived parameters can last for the entire GOP, i.e. until the next KEY FRAME that may change them.
Note : there is no separate profile for Still Picture. Still Picture is coded and decoded as a KEY FRAME, with all coding tools supported (tiling, all post in-loop filters, film grain injection, monochrome, intraBC, palette prediction mode, etc.). There is no restriction in coding Still Picture as a KEY FRAME.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 3h Codec/Engine Name Format: OpCode

AVP_PIC_STATE

		Codec/Engine Name = AVP = 3h
	22:16	**Media Instruction Command** Default Value: 30h AVP_PIC_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value / Name / Programming Notes: 31h – Decoder DW Length – Only Up to DW12 should be programmed for decoder 48h – Encoder DW Length – All DWs should be programmed for encoder
1	31:16	**Frame Height In Pixel Minus 1** Format: U16 Specifies the height of the frame to be decoded in unit of pixel. It is the same as the frame height in luma samples. AV1 supports up to 64Kx64K frame size. Intel supports up to 16Kx16K frame size. Valid range [15, 16383]. Min frame height is 16pixels. Hence, Luma is 16and Chroma is 8(in 4:2:0). It is a frame-level derived parameters, after taking into account of the syntax elementmax_frame_height in the sequence header and frame_size_override_flag in the frame header. For SWITCH Frame,frame_size_override_flag is always set to 1; for all other frame types (KEY Frame, INTRA-ONLY Frame and INTER Frame),frame_size_override_flag is read from the bitstream (and can be 0 or 1).For inter-frame, it also takes into account of the possibility of using frame size inferred from a reference frame. Note : this field is not affected by Horizontal Super-Resolution coding. Note : this frame height may be odd or even number, and may not be divisible by 4, 8 , superblock size, tile size, or LRU size. Note : if frame height is an odd number, the corresponding height of the Chroma planes in 4:2:0 is rounded up. For example, if Luma height is 13 pixels, Chroma height is 7 pixels. Note : internally the frame height is rounded up to be divisible by 8 in both the HW encoder and decoder. The padding in the encoder side is not normative, but will be coded into the bitstream. When decoder reconstructs the decoded frame from the bitstream, the padding is being reconstructed as well. The padding at the right and bottom borders of the decoded frame can be cropped away before display. But they are not cropped away when the decoded frame is added into the DPB as a reference frame. Note : this is NOT the Render Frame Height, which is used to crop the decoded frame size for display.

AVP_PIC_STATE

	15:0	**Frame Width In Pixel Minus 1** Format: U16 Specifies the width of the decoded frame in unit of pixel. It is the same as the frame width in uma samples. AV1 supports up to 64Kx64K frame size. Intel supports up to 16Kx16K frame size. Valid range [15, 16383]. Min frame width is 16pixels. Hence, Luma is 16and Chroma is 8(in 4:2:0). It is a frame-level derived parameters, after taking into account of the syntax elementmax_frame_width in the sequence header and frame_size_override_flag in the frame header. For inter-frame and s-frame, it also takes into account of the possibility of using frame size inferred from a reference frame. When the Horizontal Super-Resolution is active, this frame width is the downscaled frame width. Note : if Horizontal-Only Super-Resolution is ON, this field specifies the reduced width of the downscaled frame size. The number of Superblocks to be processed per row in this case is derived from this reduced frame width. The tile configuration is also specified in this reduced frame width. But after deblocker and CDEFfiltering (if either or both are active), this reduced frame width is scaled back up to the original frame width for the subsequent Loop Restoration filtering (if active), which then is outputted for display or become a reference frame. Hence, all reference frames in the DPB are always stored in full frame resolution, and dynamic on-the-fly frame resizing is always invoked during Motion Comp at block level, which is coded with the reduced frame width. Note that the upscaled frame width cannot exist 16K range. upscaling frame width = reduced frame width * upscaling factor. The upscaled frame width is provided in the AVP_INLOOP_FILTER_STATE Command. Note that this frame width may be odd or even number, and may not be divisible by 4, 8 , superblock size, tile size, or LRU size. Note : if frame width is an odd number, the corresponding width of the Chroma planes in 4:2:0 is rounded up. For example, if Luma width is 13 pixels, Chroma width is 7 pixels. Note : internally the frame width is rounded up to be divisible by 8 in both the HW encoder and decoder. The padding in the encoder side is not normative, but will be coded into the bitstream. When decoder reconstructs the decoded frame from the bitstream, the padding is being reconstructed as well. The padding at the right and bottom borders of the decoded frame can be cropped away before display. But they are not cropped away when the decoded frame is added into the DPB as a reference frame. Note : this is NOT the Render Frame Width, which is used to crop the decoded frame size for display.
2	31:25	**Reserved** Access: RO Format: MBZ
	24:22	**Reserved** Access: RO Format: MBZ

AVP_PIC_STATE

21 **Sequence Enable Joint Compound Flag**

Format:	U1

It specifies whether the Joint Compound coding tool is enabled for the video sequence, or not.
It is the sequence level syntax element, enable_jnt_comp, which is only present in the bitstream when the sequence level syntax element enable_order_hint is set to 1. It is defaulted to0, whenenable_order_hint is set to 0.

Value	Name	Description
0h	**[Default]**	Indicates that the distance weights process is NOT used for inter prediction.
1h		Indicate that the distance weights process may be used for inter prediction.

20 **Sequence Enable Masked Compound Flag**

Format:	U1

It specifies whether the Masked Compound coding tool is enabled for the video sequence, or not.
It is the sequence level syntax element, enable_masked_compound.
Valid only in Decoder Mode

Value	Name	Description
0h	**[Default]**	Indicates that the mode info for inter blocks do NOT contain the block level syntax element compound_type and others.
1h		Indicates the block level syntax elements: compound_type and others, are present in the bitstream.

19 **Sequence Enable Inter-Intra Compound Flag**

Format:	U1

It specifies whether the Inter-Intra Compound coding tool is enabled for the video sequence, or not.
It is the sequence level syntax element, enable_interintra_compound.
Valid in Decoder Mode only

Value	Name	Description
0h	**[Default]**	Indicates that the mode info for inter blocks do NOT contain the block level syntax element interintra and others.
1h		Indicates that the block level syntax elements: interintra and others, are present in the bitstream.

18 **Sequence Enable Dual_Filter Flag**

Format:	U1

It specifies whether the Motion Comp horizontal and vertical interpolation filters are specified independently in the bitstream at the block level (inter-coded block)or not, for the video sequence.
Set to 0 ; indicate only ONE MC filter type is specified in the bitstream, which is then used in both directions . (Default)
Set to 1 ; indicate the MC filter type may be specified independently in the horizontal and vertical directions.
It is the sequence level syntax element, enable_dual_filter.
Note : MC filter type can be [EIGHTTAP, EIGHTTAP_SMOOTH, EIGHTTAP_SHARP, BILINEAR, and

		SWITCHABLE]. Must be set to 0 in encoder mode
	17	**Sequence Enable Intra Edge Filter Flag** Format: U1 It specifies whether the Intra Edge Filtering process is enabled for the video sequence or not. Set to 0 ; indicate that the Intra Edge Filter tool is DISabled. (Default) Set to 1 ; indicate that the Intra Edge Filter tool is ENabled. It is the sequence level syntax element, enable_intra_edge_filter.
	16	**Sequence Enable Filter_Intra Flag** Format: U1 It specifies if the Filter_Intra coding tool is enabled for the video sequence. Set to 0 ; indicate the block level syntax element: use_filter_intra, is NOT present in the bitstream. (Default) Set to 1 ; indicate the block level syntax element: use_filter_intra, is present in the bitstream. It is the sequence level syntax element, enable_filter_intra. Note : if both enable_filter_intra and use_filter_intra are set to 1, the corresponding block is coded with filter_intra Must be set to 0 in encoder mode
	15:13	**Reserved (for the expansion of Sequence Order Hint Bits Minus1)** Format: MBZ
	12:10	**Sequence Order Hint Bits Minus1** Format: U3 It specifies the number of bits to be read from the bitstream for the frame header syntax elements: order_hint and ref_order_hint[i=0 to 6]. It is in the range of [0..7] . It is the sequence level syntax element, order_hint_bits_minus1, which is only present in the bitstream if the sequence level syntax element enable_order_hint_flag is set to 1. It is defaulted to 0, if not present. It can be used to generate the bitmask in computing the relative distance between two order hints. Note: it is also used for other purposes, as detail in the bitfield Sequence Enable Order Hint Flag.
	9	**Sequence Enable Order Hint Flag** Format: U1 Set to 1 ;enables the use of Order Hint in the decoding process of all INTER frames in the video sequence. Set to 0 ; disables the use of Order Hint. (Default) It is the sequence level syntax element, enable_order_hint. It controls 1) the bitstream reading of the syntax elements: enable_jnt_comp, enable_ref_frame_mvs, and

		order_hint_bits_minus1 in the sequence header. 2) the bitstream reading of the syntax elements: ref_order_hint[i=0 to 6] and frame_refs_short_signaling in the frame header. 3) the setting of reference frames in the frame header for the decoding of the current frame. Note : these 2 sequence level syntax elements: enable_order_hint flag and the 3-bits order_hint_bits_minus1 are used in : 1) skip_mode derivation, 2) compound prediction temporal weighing factor derivation, 3) motion field estimation process and MV projection, 4) motion_field MV storage setting, 5) reference frame bias derivation, 6) forward, second_forward, and backward reference frame selection, 7) and forward, second_forward and backward reference frame order hint setting.
	8:7	**Sequence Superblock Size Used** Format: U2 It specifies one of the two possible Superblock sizes that is used to code the video sequence. Set to 0, if the SuperBlock size is 64x64 pixels. (Default) Set to 1, if the SuperBlock size is 128x128 pixels. Value 2-3 are reserved. It is the sequence level syntax element, use_128x128_superblock; and is also named as sb_size. Supports SB size 64x64 only in encoder mode
	6	**Reserved** Access: RO Format: MBZ
	5	**Reserved (for expansion of Sequence Pixel Bit-Depth Idc)** Format: MBZ
	4:3	**Sequence Pixel Bit-Depth Idc** Format: U2 It specifies the pixel bit depth for all frames in the video sequence being decoded. [4:3] = 00 ; specifies 8-bit per pixel (bpp). (Default) [4:3] = 01 ; specifies 10-bit per pixel. [4:3] = 10 ; specifies 12-bit per pixel It is a sequence-level parameter derived from the sequence header syntax elements:seq_profile,high_bitdepth and twelve_bit. Note : Only Bit-Depth = 8 and 10 are allowed for this generation of AVP. Note : refer to the description under Sequence Chroma Sampling Format for detail on allowed bit depth for each profile.
	2	**Reserved (for expansion of Chroma SubSampling Format)** Format: MBZ

AVP_PIC_STATE

	1:0	**Sequence Chroma SubSampling Format** Format: U2 It specifies the chroma subsampling format for all frames in the video sequence being decoded. [1:0] = 00 ; stands for Monochrome 4:0:0, no Chroma planes at all, but [subsampling_x and subsampling_y] is defaulted to [1, 1], as only Profile 0 can support monochrome video coding. [1:0] = 01 ; stands for 4:2:0, with[subsampling_x and subsampling_y] defining as[1, 1]. It is supported in all profiles (seq_profile=0, 1, 2 - syntax element in the sequence header) [1:0] = 10 ; stands for 4:2:2, with[subsampling_x and subsampling_y] defining as[1, 0]. It is supported only in seq_profile=2. [1:0] = 11 ; stands for 4:4:4 with[subsampling_x and subsampling_y] defining as[0, 0]. It is supported in both seq_profile=1 and 2. It is a sequence-level parameter derived from the sequence header syntax elements: seq_profile, subsampling_x, subsampling_y, monochome, high_bitdepth and twelve_bit. Default is 1, i.e. 4:2:0.. Note : AV1 supports 3 profiles: seq_profile Bit_depth Chroma Subsampling 0 (Main Profile) 8 / 10 YUV 4:2:0 and 4:0:0 1 (High Profile) 8 / 10 YUV 4:4:4 (4:0:0 is not allowed) 2 (Pro Profile) 8 / 10 /12 YUV 4:2:2 AND 12 YUV 4:2:0/4:4:4/4:0:0 Note : for AV1 decoder: • A profile 0 compliant decoder must be able to decode all bitstreams labeled profile 0 • A profile 1 compliant decoder must be able to decode all bitstreams labeled profile 0 or 1 • A profile 2 compliant decoder must be able to decode all bitstreams labeled profile 0, 1, or 2 "01" -- Chroma Sampling 4:2:0 is supported. **Value** / **Name** / **Description** 1h / 4:2:0 / Chroma Sampling 4:2:0
3	31	**Reserved (for future expansion of Primary Reference Frame Idx)** Format: MBZ
	30:28	**Primary Reference Frame Idx** Format: U3 It specifies which one of the 7 possible reference frames [reference frame ID0 to 6]contains the frame context(CDF table set for all syntax elements) and other state that should be loaded at the start of the frame. The normal range of values for Primary Reference Frame is [0 to 6]. The value of 7 (defined as PRIMARY_REF_NONE) is used to signal when there is no primary reference frame. It is the frame level syntax element, primary_ref_frame, which is present in the bitstream when (! FrameIsIntra && ! error_resilient_mode). If it is not present in the bitstream, it is defaulted to PRIMARY_REF_NONE (=7).

		There is no primary reference frame: 1) if (FrameIsIntra || error_resilient_mode). Hence only INTER Frame and SWITCH Frame can have primary reference frame ID specified in the bitstream when not in error resilient mode. 2) Or when it receives a value of 7from the bitstream. Note : load_cdfs(ref_frame_idx[primary_ref_frame])is a function call that indicates that the frame context (CDF table set)is loaded from the frame context attached to one of the 8 possible reference frames in the DPB. Note : load_previous( ) is a function call that indicates that information from a previous frame may be loaded for use in decoding the current frame. Note : if (primary_ref_frame == PRIMARY_REF_NONE), it is required to set segmentation_update_map = 1 , segmentation_temporal_update = 0 , and segmentation_update_data = 1. Note : If primary_ref_frame is set to PRIMARY_REF_NONE, it is a requirement of bitstream conformance that loop_filter_delta_update is equal to 1.
	27:24	**Reserved** Access: RO Format: MBZ
	23	**Allow IntraBC Flag** Format: U1 It specifies whether intra block copy prediction mode is allowed to be used in the current frame or not. Set to 0 to disallow intra block copy prediction mode. (Default) Set to 1 to allow intra block copy prediction mode. It is the frame level syntax element, allow_intrabc. It is present in the bitstream, only if the sequence syntax element allow_screen_content_tools is set to 1 AND there is no super-resolution frame width scaling (but super-resolution can still be enabled). When it is not present in the bitstream, it is defaulted to 0. Note : intra block copy is only allowed in KEY Frame and INTRA-ONLY Non-Key Frame. For all other frame types (INTER Frame and SWITCH Frame), this flag is set to 0. Note : when this flag is set to 1, all post in loop filters (deblocker, CDEF, Super-Resolution and Loop Restoration) are all disabled. Note : when this flag is set to 1, force_integer_mv is set to 1 in KEY Frame and INTRA-ONLY Non-Key Frame. Valid only in Decoder Mode
	22	**Error Resilient Mode Flag** Format: U1 It specifies whether all syntax decoding of the current frame is independent of the previous frames, or not. Set to 0 to disable error resilient mode Set to 1 to enable error resilient mode (for independent syntax decoding) It is the frame-level syntax element, error_resilient_mode. Default is 0. It is read from the bitstream for all frame types (KEY Frame, INTRA-ONLY Frame and INTER

Frame), except when frame_type is set to SWITCH_FRAME, in which it is forced to 1 instead of reading from the bitstream.
When error resilient mode is set to 1 (active), Refresh Frame Context is set to0. When error resilient is set to 0, Refresh Frame Context is read from the bit stream.
Valid only in Decoder Mode

Value	Name
0	Disable
1	Enable

21:20 **Reserved**

Access:	RO
Format:	MBZ

19 **IntraOnly Flag**

Format:	U1

It specifies if the current frame being decoded is a KEY Frame or a INTRA-ONLY Non-Key Frame. It is a derived parameter from the frame level syntax element, frame_type, and is also named as FrameIsIntra.
IntraOnly Flag = (frame_type == INTRA_ONLY_FRAME) || (frame_type == KEY_FRAME).

18 **Reserved (for the expansion of Frame Type)**

Format:	MBZ

17:16 **Frame Type**

Format:	U2

Specifies one of the four possible AV1 frame types.
[17:16] = [00] ; specifies KEY Frame. (Default)
[17:16] = [01] ; specifies INTER Non-Key Frame.
[17:16] = [10] ; specifies INTRA-ONLY Non-Key Frame.
[17:16] = [11] ; specifies SWITCH Non-Key Frame (or called S-Frame, is a different type of INTER Frame).
It is the frame level syntax element, frame_type.
Note: Encoder does not support INTRA-ONLYand S-Frame

15:14 **Reserved**

Access:	RO
Format:	MBZ

13:9 **Reserved**

Access:	RO
Format:	MBZ

8:7 **Reserved**

Access:	RO
Format:	MBZ

	6	**Frame Level Loop Restoration Filter Enable Flag** Format: U1 It specifies whether the Post In-Loop Loop Restoration Filter tool is enabled for the current frame, or not. Set to 0 ; indicate that the Loop Restoration Filter is DISabled for the current frame.(Default) Set to 1 ; indicate that the Loop Restoration Filter is ENabled for the current frame. Frame Level Loop Restoration Filter Enable Flag is an intel derived parameter, and is set to 1 whenLuma frame_restoration_type != RESTORE_NONE || Chroma Cb frame_restoration_type != RESTORE_NONE || Chroma Cr frame_restoration_type != RESTORE_NONE; It is derived from 1) the sequence level syntax element, enable_restoration, 2) the frame level derived coding parameters : frame level all_lossless and allow_intrabc, 3) and also the frame level syntax for loop restoration filter type of each color component. Although Use Loop Restoration Filter Flag = ! (AllLossless || allow_intrabc || !enable_restoration) can be used to signal thedisabling ofthe loop restoration filter for the current frame, it is still possible at the frame level to read in the loop restoration filter type syntax elements that indicate the loop restoration filter is not enabled. To simplify hardware, this field is used to give HW the derived final decision if the loop restoration filter is enabled or not. When Large Scale Tile Enable flag is set to 1, this Loop Restoration Filterflag must set to 0, to disable the loop restoration filtering completely. Note : When individual frames in a video sequence are coded in Alllossless or with intraBC enabled, all post in-loop filtering processes (deblocker, CDEF, horizontal super-res, and loop restoration) should be disabled. Note : When Loop Restoration filter is disabled, all its filter types should be programmed to RESTORE_NONE.
	5	**Use Super-Res Flag** Format: U1 It specifies whether the Post In-Loop Horizontal-Only Super-Resolution tool is enabled for the current frame being decoded, or not. Set to 0 ; indicate that the Super-Res Frame Upscaling process is NOT performed. This sequence level setting cannot be overridden. (Default) Set to 1 ; indicate that the Super-Res Frame Upscaling process is to be performed. But this sequence level setting can be overridden when in frame coded lossless or when intraBC is enabled. It is derived from the sequence level syntax element, enable_superres, from the frame level syntax element of the same name: use_superres, and from the frame level derived coding parameters : frame coded_lossless and allow_intrabc. If the frame level syntax element of the same name, use_superres is not present in the bitstream, it is defaulted to 0. When Large Scale Tile Enable flag is set to 1, this Use Super-Res flag must set to 0, to disable super-resolution coding method. Note : it is legal to set the sequence level syntax element: enable_superresequal to 1, even when use_superres is set to 0 (i.e. no superres is performed)on all frames in the coded video sequence. Note : When individual frames in a video sequence are coded in coded_lossless or with intraBC enabled, all post in-loop filtering processes (deblocker, CDEF, horizontal super-res, and loop restoration) should be disabled.

		Note : When Super-res is disabled, the Denom of the scaling factor should be set to 8 (to have the same value as the default NUMERATOR).
	4	**Use CDEF Filter Flag** Format: U1 It specifies whether the Post In-Loop CDEF Filter tool is enabled for the current frame being decoded, or not. Set to 0 ; indicate that the CDEF Filter is DISabled. This sequence level setting cannot be overriden. (Default) Set to 1 ; indicate that the CDEF Filter is ENabled. But this sequence level setting can be overridden when in frame-level coded lossless or when intraBC is enabled. It is derived from the sequence level syntax element of the same name, enable_cdef and the frame level derived coding parameters : coded_lossless and allow_intrabc. Use CDEFFilter Flag = ! (Codedlossless|| allow_intrabc || !enable_cdef). When Large Scale Tile Enable flag is set to 1, this Use CDEF Filter flag must set to 0, to disable CDEF Filtering operation completely. Note : it is legal to set enable_cdef equal to 1 even when cdef filtering is not used on any frame in the coded video sequence. Note : When individualframes in a video sequence are coded in lossless or with intraBC enabled, all post in-loop filtering processes (deblocker, CDEF, horizontal super-res, and loop restoration) should bedisabled. Note : When CDEF filter is disabled, all its filter parameters should be reset to 0.
	3	**Reserved** Access: RO Format: MBZ
	2	**Allow Warped Motion Flag** Format: U1 Set to 0 ; indicates that the block level syntax element: motion_mode is NOT present in the bitstream. Hence, the block cannot signal for LOCALWARP coding. Set to 1 ; indicates that the block level syntax element: motion_mode is present in the bitstream. It is the frame level syntax element, allow_warp_motion, which is present in the bitstream only if thiscondition is met: (! FrameIsIntra && ! error_resilient_mode && enable_warped_motion). If it is not present in the bitstream, it is defaulted to 0. Note : motion_mode can take on a value of [SIMPLE. OBMC, or LOCALWARP]. LOCALWARP not supported in encoder mode
	1	**Force Integer MV Flag** Format: U1 Set to 0 ; indicates that motion vectors used in the Motion Comp can contain fractional bits (sub-pel precision). (Default) Set to 1 ; indicates that motion vectors used in the Motion Comp is always integers (NO sub-pel precision for Luma only). It is derived from thesequence level syntax elements: seq_choose_screen_content_tools

		andseq_force_screen_content_tools, and from the frame level syntax element : allow_screen_content_tools.
		if (allow_screen_content_tools = =1) AND (seq_force_integer_mv = = 2), read the value of force_integer_mv flag from the bitstream (in the frame header).
		If(allow_screen_content_tools = =1) AND (seq_force_integer_mv != 2) , force_integer_mv flag is set to the value of seq_force_integer_mv (which is read from the bitstream in the sequence header).
		if(allow_screen_content_tools = =0) , force_integer_mv is set to0.
		Note : for 4:2:0 and 4:2:2, the chroma subsampling factors (subsampling_x and subsampling_y) are also applied to the Luma MVs. Hence, integer Luma MVs will still give half-pel precision Chroma MVs (i.e. with fractional bit).
		Note : if intraBC is enabled, force_integer_mv is always set to 1 in the intra frames (KEY Frame or Intra-Only Non-Key Frame).
		Note : when force_integer_mv is set to 1, some fractional bits are still read for the translation components. However, these fractional bits will be discarded during the Setup Zero MV process.
		Valid only in decoder Mode

	0	**Allow Screen Content Tools Flag** Format: U1 Set to 0 ; indicates that the two block level screen content coding tools (palette prediction coding and intraBC coding). (Default) Set to 1 ; indicates that the block level syntax element: motion_mode is present in the bitstream. It is derived from thesequence level syntax elements: seq_choose_screen_content_tools andseq_force_screen_content_tools, and from the frame level syntax element of the same name: allow_screen_content_tools. ifseq_choose_screen_content_tools = = 1, read the value of allow_screen_ccontent_tools from the bitstream (in the frame header). Ifseq_choose_screen_content_tools = = 0, allow_screen_content_tools is set to the value of seq_force_screen_content_tools (which is read from the bitstream in the sequence header). Note : at the frame header, allow_screen_content_tools flag controls the setting of intraBC and integer_mv coding tools, but at the block level, it controls the use of the palette prediction coding mode. Valid in Decoder Mode only
4	31	**Reserved** Access: RO Format: MBZ
	30:24	**Y_dc_delta_q** Format: S6 **Programming Notes** 2's complement sign number of range -63 to +63.

AVP_PIC_STATE

	23:16	**Base Qindex** Format: U8 It can take on value range from 0 to 255. This is the same as the Y_ac_q, becasue y_ac_delta_q is always set to 0.
	15:14	**Reserved** Access: RO Format: MBZ

	13	**Segment ID Buffer Stream-Out Enable Flag**
		Format: U1
		It indicates if the processing of the current frame requires to write out the segment IDs at the block level to a segment ID Buffer.
		It is set to 1, if stream-outis going to happen.
		It is set to 0, if stream-outis not going to happen at all.
		It is an intel derived frame level parameters based on only frame level syntax and other derived parameters. Its meaning is defined as follows:
		if(SegmentIdStreamOutFlag)
		{
		HW needs to write out segment ID to Segment ID Write Buffer;
		}
		else
		{
		HW does not need to write out segment ID;
		}
		Driver is responsible to set this Segment ID Buffer Stream-Out Enable Flag based on the following conditions (using Driver variable names) :
		Frame Level derived parameter : (seg->enable = = 1) &&
		Frame Level SE : seg->update_map = = 1
		Valid in Decoder only Mode

	12	**Segment ID Buffer Stream-In Enable Flag**
		Format: U1
		It indicates if the processing of the current frame requires to read in the segment IDs at the block level from a segment ID Buffer.
		It is set to 1, if stream-in is going to happen.
		It is set to 0, if stream-in is not going to happen at all.
		It is an intel derived frame level parameters based on only frame level syntax and other derived parameters. Its meaning is defined as follows:
		Valid in Decoder only Mode
		if (SegmentMapIsZeroFlag)
		{
		Segment ID is all 0s.
		Driver wlill programSegmentIdStreamInFlagto 0. (HW checks on the SegmentMapIsZeroFlag=1, and set all temporal segment ID to 0)

}
else if (SegmentIdStreamInFlag)
{
Segment ID is streamed in from Segment ID read buffer;
}
else
{
No Segment ID stream-in
}
Driver is responsible to set this Segment ID Buffer Stream-InEnable Flag based on the following conditions (using Driver variable names) :
Frame Level derived parameter : (seg->enable = = 1) &&
Frame Level SE : (primary_ref_frame != PRIMARY_REF_NONE) &&
Frame Level derived parameters :
((seg->update_map = = seg->temporal_update = = 1) ||
(seg->update_map = = seg->temporal_update = = 0)) &&
((DPB[current_frame].resolution = = DPB[primary_ref_frame].resolution) &&
(DPB[primary_ref_frame].seg_enable) &&
(DPB[primary_ref_frame].segment_ID_buffer != NULL))

11 **Segment Map Is Zero Flag**

Format:	U1

In a video sequence or within a GOP,some frames can have segmentation disabled. Their corresponding segment map is defaulted to have its content set to all 0. If later, this all zero segment map will be referenced by HW for temporal segment map prediction, it will be a waste of bandwidth to read in zeros.
Segment Map Is Zero flag is an intel added parameter at frame level.
If it is set to 1, ittells HW that the segment map is containing all zero. AVP HW will check this bit together with if segment map streamin is enabled, then HW will not actually read from the segment map buffer, but will internally generate the read of 0 instead, to save bandwidth.
Driver needs to keep track of which reference frame(s) (max 7) have segmentation disabled for decoding the current frame,and set this flag accordingly.
Driver is responsible to set this Segment Map Is Zero Flag based on the following conditions (using Driver variable names) :
Frame Level derived parameter : (seg->enable = = 1) &&
Frame Level SE : (primary_ref_frame != PRIMARY_REF_NONE) &&
Frame Level derived parameters :
((seg->update_map = = seg->temporal_update = = 1) ||
(seg->update_map = = seg->temporal_update = = 0)) &&
! [((DPB[current_frame].resolution = = DPB[primary_ref_frame].resolution) &&
(DPB[primary_ref_frame].seg_enable) &&
(DPB[primary_ref_frame].segment_ID_buffer != NULL))]

10 **Frame Coded Lossless Mode**

Format:	U1

This bitSet to indicate lossless coding mode at frame level.
Frame Coded Lossless Mode is set to 1, if all active segment's segment lossless flag are set to 1.
The equation for deriving coded lossless mode is presented in the AVP_SEGMENT_STATE Command.
AllLossless = CodedLossless && (FrameWidth == UpscaledWidth). The second condition in this equation is equivalent tohaving Super-res NOT enabled.
Only CodedLossless flag is sent to HW. AllLossless flag is not.
CodedLossless directly control the enabling/disabling of deblocker, CDEF in-loop filters.
But AllLossless is used to control the enabling/disabling of Loop Restoration filter. Hence, when super-res is ON, Loop Restoration filter can still be ON/OFF, regardless of CodedLossless.

Value	Name
0	Normal Mode
1	Coded Lossless Mode

9:8 **Delta Q RES**

Format:	U2

if delta_q_present_flag = 1, 2 bits from bitstream are read.
Delta_q_res = 0, 1, 2 and 3.
and the mulitple factor= 1 « (delta_q_res) = [1, 2, 4 or 8].
Here, only the 2 bits are programmed to the HW.

7 **Delta Q Present Flag**

Format:	Enable

if delta_q_present_flag = 1, delta_q_res is present in the bitstream - to specify SB level delta q.
In encoder mode, this flag must be set to 1 for BRC purpose so the QP can be changed at SB level.
However, it can be set to 0 for CQP across frame

6:4 **Last Active Segment ID**

Format:	U3

It specifies the highest numbered segment id that has some enabled feature. This is used when decoding the segment id to only decode choices corresponding to used segments.

3 **Pre-Skip Segment ID Flag**

Format:	U1

Set to 1 indicates that the segment id will be read before the skip syntax element. SegIdPreSkip equal to 0 indicates that the skip syntax element will be read first.

2 **Segmentation Temporal Update Flag**

Format:	U1

Indicates whether segID is decoding from bitstream or predicted from previous frame.
In encoder Mode it should use either from previous frame or streamIn

Value	Name
0h	Decode segID from bitstream
1h	Get segID either from bitstream or from previous frame

Programming Notes
Decoder Only: For KEY_FRAME or INTRA_ONLY frame [OR ERROR_RESILIENCE], this bit should be set to "0". [Temporary add in the condition ERROR_RESILIENCE, to be removed later] Note: Driver should override this flag to "0" in KEY_FRAME or INTRA_ONLY frame even if this bit decoded from bitstream is different. This is for hardware optimization. This override does not affect bitstream decoding other than uncompressed header.

1 **Segmentation Update Map Flag**

Format:	U1

Set to 0 means that the segmentation map from the previous frame is used, and the current frame decode is not changing the segmentation map. (Default)
Set to 1 means that the segmentation map is updated during the decoding of the current frame. Hence SegmentIDs of current frame arestreamout to a write only surface, which will be used as the segmentation map for a future frame(s).
It is theframe level syntax element, segmentation_update_map. It is present in the bitstream only if, the frame level syntax elements: segmentation is enabled AND primary_ref_frame is NOT PRIMARY_REF_NONE. Ifprimary_ref_frame is set to PRIMARY_REF_NONE, segmentation_update_map is always set to 1.
Note : the segmentation map that has streamout (surface buffer) is attached to the current frame, after it has become a reference frame in the Display Buffer (DPB).
Note : HW needs to detect one of these 3 conditions (the current frame is in error resilient mode OR the current frame type is KEY Frame or INTRA-ONLY Frame), and if TRUE, then HW will not read the segment map for decoding the current frame, all segment ID is considered as 0. But during the decoding of the current frame, the segment map can still be modified is segmentation map update flag is ON.

0 **Segmentation Enable Flag**

Format:	U1

Indicate if segmentation is enabled or not

Value	Name
0h	All blocks are implied to belong to segment 0
1h	SegID determination depends on segmentation_update_map setting

5 31 **Reserved**

Access:	RO
Format:	MBZ

30:24 **V_ac_delta_q**

Format:	S6

AVP_PIC_STATE

		Programming Notes 2's complement sign number of range -63 to +63.
	23	**Reserved** Access: RO Format: MBZ
	22:16	**V_dc_delta_q** Format: S6 **Programming Notes** 2's complement sign number of range -63 to +63.
	15	**Reserved** Access: RO Format: MBZ
	14:8	**U_ac_delta_q** Format: S6 **Programming Notes** 2's complement sign number of range -63 to +63.
	7	**Reserved** Access: RO Format: MBZ
	6:0	**U_dc_delta_q** Format: S6 **Programming Notes** 2's complement sign number of range -63 to +63.
6	31:28	**Reserved** Access: RO Format: MBZ
	27:24	**Reserved (for future expansion of Frame Order Hint)** Format: MBZ
	23:16	**Current Frame Order Hint** Format: U8 It specifies "OrderHintBits" least significant bits of the expected output order for the current frame being decoded. It is the frame level syntax element, order_hint. Default value is 0. Order_hint is a variable bit length syntax element; its bit length is in the range of

		OrderHintBits=[1..8]. Hence, maximum order_hint can have a value of 255. Note: There is no requirement that OrderHint should reflect the true output order.
	15:8	**Reference Frame Sign Bias [i=0 to 7]** Format: U8 This is a bit array that specifies the Reference Frame Sign Bias for Reference 0 to 7 (only 1 to 7 are valid, OR 0 to 6 ???) Reference Picture 0 is INTRA frame and there is no sign bias for it. Bit 9 : Last Frame (Reference Picture 1) Bit 10: Last2 Frame (Reference Picture 2) Bit 11: Last3 Frame (Reference Picture 3) Bit 12: GoldenFrame (Reference Picture 4) Bit 13: Bwdref Frame (Reference Picture 5) Bit 14: Altref2 Frame (Reference Picture 6) Bit 15: Altref Frame (Reference Picture 7) It is a frame-level derived parameter for each reference frame, RefFrameSignBias[i=0 to 7]. 1-bit per reference frame. It is derived from the frame level parameters order hint and ref order hints. If the sequence syntax element enable_order_hint is set to 0, RefFrameSignBias[i=0 to 7] are all set to 0.
	7	**Use Reference Frame MV Set Flag** Format: U1 Set to 0, specifies that motion vector information from a previous frame (stored inside the temporal MV buffer) CANNOT be used when decoding the current frame. (Default) Set to 1, specifies that motion vector information from a previous frame can be usedwhen decoding the current frame. It is the frame level syntax element, use_ref_frame_mvs, which is present in the bitstream only if the sequence level syntax elements: enable_order_hint and enable_ref_frame_mvs are both set to 1 and the frame level syntax element: error_resilient_mode is set to 0. If it is not present in the bitstream, it is defaulted to 0.
	6	**Motion Mode Switchable Flag** Format: U1 Set to 0, specifies only SIMPLE motion mode can be used. (Default) Set to 1, specifies the motion mode being used is determined at the block level. It is the frame level syntax element, is_motion_mode_switchable. It is present only in INTER Frame or SWITCH Frame. For all other frame types (KEY Frame and INTRA-ONLY Frame), it is defaulted to 0. Note :Motion Mode for motion comp can be SIMPLE, OBMC or LOCALWARP.
	5	**Reserved (for future expansion of Mcomp Filter Type)** Format: MBZ

AVP_PIC_STATE

	4:2	**Mcomp Filter Type** Format: U3 It specifies which Motion Compensation Filter type is to be used for the current frame. It is a frame-level derived parameters. It is derived from the frame level syntax elements (is_filter_switchable flag and the 2-bits interpolation_filter). Default is 0 (i.e. use the eight-tap basic filter). **Value** / **Name** 0h – Eight-tap 1h – Eight-tap-Smooth 2h – Eight-tap-Sharp 3h – Bilinear 4h – Switchable **Programming Notes** In VDENC mode, setting Switchable results in Eight-tap filter across the whole frame.
	1	**Frame Level Reference Mode Select** Format: U1 Set to 0 specifies that all inter blocks in the current frame will use single prediction (SINGLE_REFERENCE). Default is 0, use SINGLE_REFERENCE. Set to 1 specifies that the frame level prediction mode for inter blocks is REFERENCE_MODE_SELECT, which will cause reading the syntax element comp_mode at PartU level to specify whether to use single or compound reference prediction for that partU. It is the frame level syntax element, reference_select. If it is NOT present in the bitstream, it is defaulted to 0.
	0	**Allow High Precision MV** Format: U1 It specifies whether the high precision MV mode is used for the Luma Motion Vector prediction or not. Set to 0, specifies that motion vectors are in quarter-pel precision. (Default, the normal mode) Set to 1, specifies that motion vectors are in eighth-pel precision. It is the frame-level syntax element, allow_high_precision_mv. It is present in the bitstream, only if the frame level parameter force_integer_mv is set to 0. If it is not present in the bitstream, it is default to 0.
7	31:24	**Reference Frame Side [i=0 to 7]** Format: U8 This is a bit array that specifies the Reference Frame Sidefor Reference 0 to 7. Bit 24: Intra Frame (Reference Picture 0) Bit 25: Last Frame (Reference Picture 1) Bit 26: Last2 Frame (Reference Picture 2) Bit 27: Last3 Frame (Reference Picture 3)

		Bit 28: GoldenFrame (Reference Picture 4) Bit 29: Bwdref Frame (Reference Picture 5) Bit 30: Altref2 Frame (Reference Picture 6) Bit 31: Altref Frame (Reference Picture 7) It is a intel frame-level derived parameter for each reference frame, For each reference frame the corresponding bit is set to 0 when the corresponding bit in ref_frame_side is 0, else the bit will be set to 1. Each individual bit (i=0 to 7)of the original reference frame side parameter can take on a value of -1, 0, or 1. But intel version of the same parameter can only take on a value of 0 or 1. Both the original value of -1 and1 is mapped to 1 here, and the original value of 0 remains mapped to 0. Default all 8-bits are set to 0.
	23:13	**Reserved** Access: RO Format: MBZ
	12	**Reserved (for future expansion of Skip Mode Frame[1])** Format: MBZ
	11:9	**Skip Mode Frame [1]** Format: U3 It specifies the reference frames to use for compound prediction when skip_mode is set to 1. It is the frame level derived parameter SkipModeFrame[1]. Skip mode tries to use the closest forward (past) and backward (future) references (as measured by values in the RefOrderHint array). If no backward reference is found, then the second closest forward reference is used. If no forward reference is found, then skip mode is disabled.
	8	**Reserved (for future expansion of Skip Mode Frame[0])** Format: MBZ
	7:5	**Skip Mode Frame [0]** Format: U3 It specifies the reference frames to use for compound prediction when skip_mode is set to 1. It is the frame level derived parameter SkipModeFrame[0]. Skip mode tries to use the closest forward (past) and backward (future) references (as measured by values in the RefOrderHint array). If no backward reference is found, then the second closest forward reference is used. If no forward reference is found, then skip mode is disabled.
	4	**Skip Mode Present Flag** Format: U1 Set to 0 specifies that skip_mode will not be used for this frame. Default is 0, no PartU level skip flag. Set to 1 specifies that the syntax element skip_mode will be coded in the bitstream at the PartU level. It is the frame-level syntax element skip_mode_present. It is present in the bitstream based on an algorithm using RefOrderHint. If it is not present in the bitstream, it is defaulted to 0.

AVP_PIC_STATE

	3	**Reserved** Access: RO Format: MBZ
	2:1	**Frame Transform Mode** Format: U2 It specifies the Luma-only transform size to be used for the entire current frame to be decoded. 1) tx_mode=0 for ONLY_4X4, if the current frameis coded with frame coded lossless and only Hadamard 4x4 transform is being used for the whole frame. This is also applied to Chroma planes as well. 2) tx_mode=1 for TX_MODE_LARGEST, the inverse transform will use the largest transform size that fits inside the block. 3) tx_mode=2 for TX_MODE_SELECT, the choice of transform size is specified explicitly for each block. It is the frame-level derived parameters, TxMode. It is derived from the frame-level syntax element, tx_mode_select and the frame level derived parameter Frame Coded Lossless. Default is TxMode=1. Chroma Tx Mode is no longer derived from that of Luma Tx size. Chroma Tx Mode has no explicit setting in the bitstream,andChroma Tx Mode can be viewed as always equivalent to TX_MODE_LARGEST for the two Chroma planes.
	0	**Reduced Tx Set Used** Format: U1 **Programming Notes** set to 1 to allow the use of a reduced tx set. set to 0 to allow the full tx set to be used for each tx type. It is the frame level syntax element, reduced_tx_set.
8	31:24	**Frame Level Global Motion Invalid Flags** Format: U8 It indicates to the AV1 HW pipeline (at block level processing) that the result of parsing the frame level global motion parameters of each reference frame is invalid or not. Each bitrepresents the invalid flag of a reference frame : it is set to 1, if the global motion parameters for the corresponding reference frame is invalid. Otherwise it is set to 0 for valid (default). Frame Level Global Motion Invalid Flags[31] indicates the validity of reference frame ALTREF. ... Frame Level Global Motion Invalid Flags[25] indicates the validity of reference frame LAST. Frame Level Global Motion Invalid Flags[24] indicates the validity of reference frame INTRA (this bit is reserved and not being used by HW)... This is an intel defined parameter to give HW a hint of frame header parsing result that cansimplify the HW design. It takes on the same value as of theglobal_motion[ALTRF...LAST].invalid flag, which is set insideread_global_motion() function of the reference C model.

	23	**Reserved (for future expansion of Global Motion Type[7]** Format: MBZ
	22:21	**Global Motion Type[7]** Format: U2 It specifies the global motion type associated with each reference [LAST_FRAME ... ALTREF_FRAME] that can be used to decode the current frame. Set to 0, specifies IDENTITY(requires no additional warp projection parameter) Set to 1, specifies TRANSLATION (requires 2 additional warp projection parameters) Set to 2, specifies ROTZOOM (requires 4 additional warp projection parameters) Set to 3, specifies AFFINE (requires 6additional warp projection parameters) It is the frame level derived parameter gmtype[LAST_FRAME ... ALTREF_FRAME = 1..7], gmtype[0] is not being used.
	20	**Reserved (for future expansion of Global Motion Type[6]** Format: MBZ
	19:18	**Global Motion Type[6]** Format: U2 It specifies the global motion type associated with each reference [LAST_FRAME ... ALTREF_FRAME] that can be used to decode the current frame. Set to 0, specifies IDENTITY(requires no additionalprojection parameter) Set to 1, specifies TRANSLATION (requires 2 additionalprojection parameters) Set to 2, specifies ROTZOOM (requires 4 additionalprojection parameters) Set to 3, specifies AFFINE (requires 6additionalprojection parameters) It is the frame level derived parameter gmtype[LAST_FRAME ... ALTREF_FRAME = 1..7], gmtype[0] is not being used.
	17	**Reserved (for future expansion of Global Motion Type[5]** Format: MBZ
	16:15	**Global Motion Type[5]** Format: U2 It specifies the global motion type associated with each reference [LAST_FRAME ... ALTREF_FRAME] that can be used to decode the current frame. Set to 0, specifies IDENTITY(requires no additionalprojection parameter) Set to 1, specifies TRANSLATION (requires 2 additionalprojection parameters) Set to 2, specifies ROTZOOM (requires 4 additionalprojection parameters) Set to 3, specifies AFFINE (requires 6additionalprojection parameters) It is the frame level derived parameter gmtype[LAST_FRAME ... ALTREF_FRAME = 1..7], gmtype[0] is not being used.
	14	**Reserved (for future expansion of Global Motion Type[4]** Format: MBZ

AVP_PIC_STATE

	13:12	**Global Motion Type[4]** Format: U2 It specifies the global motion type associated with each reference [LAST_FRAME ... ALTREF_FRAME] that can be used to decode the current frame. Set to 0, specifies IDENTITY(requires no additional projection parameter) Set to 1, specifies TRANSLATION (requires 2 additional projection parameters) Set to 2, specifies ROTZOOM (requires 4 additional projection parameters) Set to 3, specifies AFFINE (requires 6additionalprojection parameters) It is the frame level derived parameter gmtype[LAST_FRAME ... ALTREF_FRAME = 1..7], gmtype[0] is not being used.
	11	**Reserved (for future expansion of Global Motion Type[3]** Format: MBZ
	10:9	**Global Motion Type[3]** Format: U2 It specifies the global motion type associated with each reference [LAST_FRAME ... ALTREF_FRAME] that can be used to decode the current frame. Set to 0, specifies IDENTITY(requires no additional projection parameter) Set to 1, specifies TRANSLATION (requires 2 additional projection parameters) Set to 2, specifies ROTZOOM (requires 4 additional projection parameters) Set to 3, specifies AFFINE (requires 6additionalprojection parameters) It is the frame level derived parameter gmtype[LAST_FRAME ... ALTREF_FRAME = 1..7], gmtype[0] is not being used.
	8	**Reserved (for future expansion of Global Motion Type[2]** Format: MBZ
	7:6	**Global Motion Type[2]** Format: U2 It specifies the global motion type associated with each reference [LAST_FRAME ... ALTREF_FRAME] that can be used to decode the current frame. Set to 0, specifies IDENTITY(requires no additional projection parameter) Set to 1, specifies TRANSLATION (requires 2 additional projection parameters) Set to 2, specifies ROTZOOM (requires 4 additional projection parameters) Set to 3, specifies AFFINE (requires 6additionalprojection parameters) It is the frame level derived parameter gmtype[LAST_FRAME ... ALTREF_FRAME = 1..7], gmtype[0] is not being used.
	5	**Reserved (for future expansion of Global Motion Type[1]** Format: MBZ
	4:3	**Global Motion Type[1]** Format: U2 It specifies the global motion type associated with each reference [LAST_FRAME ... ALTREF_FRAME] that can be used to decode the current frame.

		Set to 0, specifies IDENTITY(requires no additional projection parameter) Set to 1, specifies TRANSLATION (requires 2 additional projection parameters) Set to 2, specifies ROTZOOM (requires 4 additional projection parameters) Set to 3, specifies AFFINE (requires 6additionalprojection parameters) It is the frame level derived parameter gmtype[LAST_FRAME ... ALTREF_FRAME = 1..7], gmtype[0] is not being used.
	2:0	**Reserved (for future expansion of Global Motion Type[0])** Format: MBZ
9..29	671:0	**Warp Parameters Array [Reference=1 to 7][ProjectionCoeff=0 to 5]** It specifies the Warp Parameter set for each of the 7 reference frames [LAST_FRAME .. ALTREF_FRAME] Each Warp Parameter set is contains of 6 warp projection coefficient [projection_coeff = 0 to 5] Each projection coefficient is a 16-bit signed integer (2's component). Total 7 references * 6 proj coeff / 2 = 21 Dwords. Different Global Motion Type is reading different number of projection coefficients from the bitstream. All projection coefficients are coded with signed_subexp_with_ref(). The number of bits read for each projection coefficients from the bitstream, depends on the Global Motion Type too. After decoded the signed_subexp_with_ref(), the projection coefficents are further upshifted and round to the final precision. Allowed range for each coeff is [-4096, 4096] for Warp Motion. For translation-only motion type, the max range for each coeff is [-512 to +512]. All the upper bits are assumed to have sign extended. For IDENTITY motion type, all the coefficients should set to 0. Dword 9 15:0 - Warp Parameters Array[1][0] Dword 9 31:16 -Warp Parameters Array[1][1] Dword 10 15:0 -Warp Parameters Array[1][2] Dword 10 31:16-Warp Parameters Array[1][3] ... ... Dword 29 15:0 -Warp Parameters Array[7][4] Dword 29 31:16-Warp Parameters Array[7][5]
30	31:0	**Reserved** Format: MBZ
31	31:16	**Intra Frame Height In Pixel Minus 1** Format: U16 Specifies the height of the INTRA picture(Reference Picture0) in pixels. The INTRA picture height in units of luma samples equals (INTRA_FRAME_HeightInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383].
	15:0	**Intra Frame Width In Pixel Minus 1** Format: U16

AVP_PIC_STATE

		Specifies the width of the INTRA picture(Reference Picture0) in pixels. The INTRA picture widthin units of luma samples equals (INTRA_FRAME_WidthInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383].
32	31:16	**Last Frame Height In Pixel Minus 1** Format: U16 Specifies the height of the LAST picture(Reference Picture1) in pixels. The LAST picture height in units of luma samples equals (LASTFRAME_HeightInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383].
	15:0	**Last Frame Width In Pixel Minus 1** Format: U16 Specifies the width of the LAST picture(Reference Picture1) in pixels. The LAST picture widthin units of luma samples equals (LASTFRAME_WidthInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383].
33	31:16	**Last2 Frame Height In Pixel Minus 1** Format: U16 Specifies the height of the LAST2(Reference Picture 2) picture in pixel. The LAST2 picture height in units of luma samples equals (LAST2FRAME_HeightInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383]. For detail description on frame size and programming notes, refer to the bitfield Frame Height In Pixel Minus 1 and Frame Width In Pixel Minus 1.
	15:0	**Last2 Frame Width In Pixel Minus 1** Format: U16 Specifies the width of the LAST2 picture(Reference Picture 2) in pixels. The LAST2 picture widthin units of luma samples equals (LAST2FRAME_WidthInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383]. For detail description on frame size and programming notes, refer to the bitfield Frame Height In Pixel Minus 1 and Frame Width In Pixel Minus 1.
34	31:16	**Last3 Frame Height In Pixel Minus 1**

		Format: U16 Specifies the height of the LAST3 picture(Reference Picture 3) in pixels. The LAST3 picture height in units of luma samples equals (LAST3FRAME_HeightInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383]. For detail description on frame size and programming notes, refer to the bitfield Frame Height In Pixel Minus 1 and Frame Width In Pixel Minus 1.
	15:0	**Last3 Frame Width In Pixel Minus 1** Format: U16 Specifies the width of the LAST3 picture(Reference Picture 3) in pixels. The LAST3 picture widthin units of luma samples equals (LAST3FRAME_WidthInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383]. For detail description on frame size and programming notes, refer to the bitfieldFrame Height In Pixel Minus 1 and Frame Width In Pixel Minus 1.
35	31:16	**Golden Frame Height In Pixel Minus 1** Format: U16 Specifies the height of the GOLDEN picture(Reference Picture 4) in pixels. The GOLDEN picture height in units of luma samples equals (GOLDENFRAME_HeightInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383].
	15:0	**Golden Frame Width In Pixel Minus 1** Format: U16 Specifies the width of the GOLDEN picture(Reference Picture 4) in pixels. The GOLDEN picture widthin units of luma samples equals (GOLDENFRAME_WidthInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383].
36	31:16	**BWDREF Frame Height In Pixel Minus 1** Format: U16 Specifies the height of the BWDREF picture(Reference Picture 5) in pixels. The BWDREF picture height in units of luma samples equals (BWDREFFRAME_HeightInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383].

AVP_PIC_STATE

	15:0	**BWDREF Frame Width In Pixel Minus 1** Format: U16 Specifies the width of the BWDREF picture(Reference Picture 5) in pixels. The BWDREF picture widthin units of luma samples equals (BWDREFFRAME_WidthInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383].
37	31:16	**ALTREF2 Frame Height In Pixel Minus 1** Format: U16 Specifies the height of the ALTREF2 picture(Reference Picture 6) in pixels. The ALTREF picture height in units of luma samples equals (ALTREFFRAME_HeightInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383]. For detail description on frame size and programming notes, refer to the bitfieldFrame Height In Pixel Minus 1 and Frame Width In Pixel Minus 1.
	15:0	**ALTREF2 Frame Width In Pixel Minus 1** Format: U16 Specifies the width of the ALTREF2 picture(Reference Picture 6) in pixels. The ALTREF picture widthin units of luma samples equals (ALTREFFRAME_WidthInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383]. For detail description on frame size and programming notes, refer to the bitfieldFrame Height In Pixel Minus 1 and Frame Width In Pixel Minus 1.
38	31:16	**ALTREF Frame Height In Pixel Minus 1** Format: U16 Specifies the height of the ALTREF picture(Reference Picture 7) in pixels.T he ALTREF2 picture height in units of luma samples equals (ALTREF2FRAME_HeightInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size. Valid Value = [15, 16383]. For detail description on frame size and programming notes, refer to the bitfield Frame Height In Pixel Minus 1 and Frame Width In Pixel Minus 1.
	15:0	**ALTREF Frame Width In Pixel Minus 1** Format: U16 Specifies the width of the ALTREF picture(Reference Picture 7) in pixels. The ALTREF2 picture widthin units of luma samples equals (ALTREF2FRAME_WidthInPixelMinus1+ 1) AV1 supports up to 64K frame size. Intel supports up to 16K frame size.

		Valid Value = [15, 16383]. For detail description on frame size and programming notes, refer to the bitfield Frame Height In Pixel Minus 1 and Frame Width In Pixel Minus 1.
39	31:16	**Horizontal Scale Factor for INTRA** Format: U2.14 This indicates the scaling factor between current frame and the INTRA reference frame (Reference Picture0). Set to (INTRA_Width * 2^14 + (CurrentWidth / 2)) / CurrentWidth Scaling Factor can be [1/16 to 2].
	15:0	**Vertical Scale Factor for INTRA** Format: U2.14 This indicates the scaling factor between current frame and the INTRA reference frame (Reference Picture 0). Set to (INTRA_Height * 2^14 + (CurrentHeight / 2)) / CurrentHeight Scaling Factor can be [1/16 to 2].
40	31:16	**Horizontal Scale Factor for LAST** Format: U2.14 This indicates the scaling factor between current frame and the LAST reference frame (Reference Picture1). Set to (LAST_Width * 2^14 + (CurrentWidth / 2)) / CurrentWidth Scaling Factor can be [1/16 to 2].
	15:0	**Vertical Scale Factor for LAST** Format: U2.14 This indicates the scaling factor between current frame and the LAST reference frame (Reference Picture 1). Set to (LAST_Height * 2^14 + (CurrentHeight / 2)) / CurrentHeight Scaling Factor can be [1/16 to 2].
41	31:16	**Horizontal Scale Factor for LAST2** Format: U2.14 This indicates the scaling factor between current frame and the LAST2 reference frame (Reference Picture 2). Set to (LAST2_Width * 2^14 + (CurrentWidth / 2)) / CurrentWidth Scaling Factor can be [1/16 to 2].
	15:0	**Vertical Scale Factor for LAST2** Format: U2.14 This indicates the scaling factor between current frame and the LAST2 reference frame (Reference Picture 2). Set to (LAST2_Height * 2^14 + (CurrentHeight / 2)) / CurrentHeight Scaling Factor can be [1/16 to 2].

AVP_PIC_STATE		
42	31:16	**Horizontal Scale Factor for LAST3** Format: U2.14 This indicates the scaling factor between current frame and the LAST3 reference frame (Reference Picture 3). Set to (LAST3_Width * 2^14 + (CurrentWidth / 2)) / CurrentWidth Scaling Factor can be [1/16 to 2].
	15:0	**Vertical Scale Factor for LAST3** Format: U2.14 This indicates the scaling factor between current frame and the LAST3 reference frame (Reference Picture 3). Set to (LAST3_Height *2^14 + (CurrentHeight / 2)) / CurrentHeight Scaling Factor can be [1/16 to 2].
43	31:16	**Horizontal Scale Factor for GOLDEN** Format: U2.14 This indicates the scaling factor between current frame and the GOLDEN reference frame (Reference Picture 4). Set to (GOLDEN_Width * 2^14 + (CurrentWidth / 2)) / CurrentWidth Scaling Factor can be [1/16 to 2].
	15:0	**Vertical Scale Factor for GOLDEN** Format: U2.14 This indicates the scaling factor between current frame and the GOLDEN reference frame (Reference Picture 4). Set to (GOLDEN_Height * 2^14 + (CurrentHeight / 2)) / CurrentHeight Scaling Factor can be [1/16 to 2].
44	31:16	**Horizontal Scale Factor for BWDREF_FRAME** Format: U2.14 This indicates the scaling factor between current frame and the BWDREF_FRAME reference frame (Reference Picture 5). Set to (BWDREF_FRAME_Width * 2^14 + (CurrentWidth / 2)) / CurrentWidth Scaling Factor can be [1/16 to 2].
	15:0	**Vertical Scale Factor for BWDREF_FRAME** Format: U2.14 This indicates the scaling factor between current frame and the BWDREF_FRAME reference frame (Reference Picture 5). Set to (BWDREF_FRAME_Height * 2^14 + (CurrentHeight / 2)) / CurrentHeight Scaling Factor can be [1/16 to 2].
45	31:16	**Horizontal Scale Factor for ALTREF2** Format: U2.14 This indicates the scaling factor between current frame and the ALTREF2 reference frame (Reference Picture 6). Set to (ALTREF_FRAME_Width * 2^14 + (CurrentWidth / 2)) / CurrentWidth Scaling Factor can be [1/16 to 2].

AVP_PIC_STATE		
	15:0	**Vertical Scale Factor for ALTREF2** Format: U2.14 This indicates the scaling factor between current frame and the ALTREF2 reference frame(Reference Picture 6). Set to (ALTREF_FRAME_Height * 2^14 + (CurrentHeight / 2)) / CurrentHeight Scaling Factor can be [1/16 to 2].
46	31:16	**Horizontal Scale Factor for ALTREF** Format: U2.14 This indicates the scaling factor between current frame and the ALTREFreference frame (Reference Picture 7). Set to (ALTREF2_FRAME_Width * 2^14 + (CurrentWidth / 2)) / CurrentWidth Scaling Factor can be [1/16 to 2].
	15:0	**Vertical Scale Factor for ALTREF** Format: U2.14 This indicates the scaling factor between current frame and the ALTREFreference frame(Reference Picture 7). Set to (ALTREF2_FRAME_Height * 2^14 + (CurrentHeight / 2)) / CurrentHeight Scaling Factor can be [1/16 to 2].
47	31:24	**Reference Frame Order Hint [3] for Last3 Frame** Format: U8 It specifies the expected output order hint for each reference buffer. This is the LAST3 Reference Frame Order Hint (ReferenceFrame 3). Note : The values in the ref_order_hint array are provided to allow implementations to gracefully handle cases when some frames have been lost.
	23:16	**Reference Frame Order Hint [2] for Last2 Frame** Format: U8 It specifies the expected output order hint for each reference buffer. This is the LAST2 Reference Frame Order Hint (ReferenceFrame 2). Note : The values in the ref_order_hint array are provided to allow implementations to gracefully handle cases when some frames have been lost.
	15:8	**Reference Frame Order Hint [1] for Last Frame** Format: U8 It specifies the expected output order hint for each reference buffer. This is the LAST Reference Frame Order Hint (ReferenceFrame 1). Note : The values in the ref_order_hint array are provided to allow implementations to gracefully handle cases when some frames have been lost.

	7:0	**Reference Frame Order Hint [0] for Intra Frame** Format: U8 It specifies the expected output order hint for each reference buffer. This is the INTRA Reference Frame Order Hint (ReferenceFrame 0). It is a derived frame-level parameter, which can be equal to 1) the frame-level syntax element, ref_order_hint[i=0 to 7], which is only present in the bitstream if the current frame type (frame-level syntax)is NOT a KEY Frame, AND frame level syntax element: error-resilient mode is set to 1 AND sequence level syntax element: enable_order_hint is set to 1. 2) OR, the saved order hint, when the reference frame was the current frame being decoded. Note : The values in the ref_order_hint array can be usedto implementto gracefully handle cases when some frames have been lost. It is done at the SW level, not inside AVP HW pipeline.
48	31:24	**Reference Frame Order Hint [7] for ALTREF Frame** Format: U8 It specifies the expected output order hint for each reference buffer. This is the ALTREF Reference Frame Order Hint (ReferenceFrame 7). Note : The values in the ref_order_hint array are provided to allow implementations to gracefully handle cases when some frames have been lost.
	23:16	**Reference Frame Order Hint [6] for ALTREF2 Frame** Format: U8 It specifies the expected output order hint for each reference buffer. This is the ALTREF2 Reference Frame Order Hint (ReferenceFrame 6). Note : The values in the ref_order_hint array are provided to allow implementations to gracefully handle cases when some frames have been lost.
	15:8	**Reference Frame Order Hint [5] for BWDREF Frame** Format: U8 It specifies the expected output order hint for each reference buffer. This is the BWDREF Reference Frame Order Hint (ReferenceFrame 5). Note : The values in the ref_order_hint array are provided to allow implementations to gracefully handle cases when some frames have been lost.
	7:0	**Reference Frame Order Hint [4] for Golden Frame** Format: U8 It specifies the expected output order hint for each reference buffer. This is the GOLDEN Reference Frame Order Hint (ReferenceFrame 4). Note : The values in the ref_order_hint array are provided to allow implementations to gracefully handle cases when some frames have been lost.
49	31:0	**Reserved** Access: RO Format: MBZ

AVP_PIC_STATE		
50	31:0	**Reserved** Access: RO Format: MBZ

AVP_PIPE_BUF_ADDR_STATE

AVP_PIPE_BUF_ADDR_STATE

Source: VideoCS

Length Bias: 2

The AVP Pipeline is selected with the **Media Instruction Opcode "8h"** for all AVP Commands. Each AVP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

This state command provides the physical memory base addresses for all row store buffers, column store buffers, reconstructed output and reference frame buffers, and auxiliary data buffers (MV, segment map, etc.) that are required by the AV1 decoding and encoding process.
This is a frame level state command and is shared by both encoding and decoding processes.
AVP is a tile based pipeline and is a stateless pipeline, hence all sequence level, frame level, and segment level state commands must be resent to process each tile.
Memory compression may be applicable to some of these buffers for BW saving.
Note : there is no buffer to store the 16 QM table sets, they are implemented directly inside the HW pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 3h Codec/Engine Name Format: OpCode Codec/Engine Name = AVP = 3h
	22:16	**Media Instruction Command** Default Value: 2h AVP_PIPE_BUF_ADDR_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: C0h Name:

AVP_PIPE_BUF_ADDR_STATE

1..16	511:0	**Reference Frame Buffer Base Address (RefAddr[0-7])** Format: **SplitBaseAddress64ByteAligned[8]** This array specifies the physical memory base addresses of the 8 visible reference frame buffers using for inter-prediction. They all contain previously reconstructed/decoded frames. However, this Reference Frame Buffer array is a remapped version (see below the mapping equations)of the DPB buffer array specified in the AV1 Spec and Reference C Model. Intel implementation has converted the 2 level of reference frame indexing (using RefFrame[0..6] and ref_frame_idx[0..6]) into a single level indexing (using the enumINTRA_FRAME, LAST_FRAME, ..., ALTREF_FRAME, in thisfixed order). In the process, the ref_frame_idx[0..6] array is eliminated by intel. Application and Driver will continue to receive or parse the bitstream header based on the2-level indexing AV1 DPB buffer array definition, which can containmore than 8 frame buffers, but only at most 8 of them are visible to the AVP HW pipeline and is in the formed of single level indexing intel-remapped-DPB array. Typically, these reference frame buffers are read-only, for the purpose of Motion Comp. Memory compression is applied in accessing these buffers. At most only 7 out of the 8 visible reference frames can be used for Motion Comp.in decoding inter-coded blocks of the current frame. The subset of reference frames being used in decoding the current frame is setup by the Application and Driver. But it is recommended to set all Reference Frame Buffer Base Address to valid andknown addresses for error handling. AV1 Reference Frames are defined in the following order: DW 0-1 RefAddr[0] - INTRA Frame (Reference Frame 0) DW 2-3 RefAddr[1] - LAST Frame (Reference Frame 1) In AV1 Spec, it is mapped intoDPB [ref_frame_idx[LAST_FRAME-LAST_FRAME]] DW 4-5 RefAddr[2] - LAST2 Frame (Reference Frame 2) In AV1 Spec, it is mapped intoDPB [ref_frame_idx[LAST2_FRAME-LAST_FRAME]] DW 6-7 RefAddr[3] - LAST3 Frame (Reference Frame 3) In AV1 Spec, it is mapped intoDPB [ref_frame_idx[LAST3_FRAME-LAST_FRAME]] DW 8-9 RefAddr[4] - GOLDEN Frame (Reference Frame 4) In AV1 Spec, it is mapped intoDPB [ref_frame_idx[GOLDEN_FRAME-LAST_FRAME]] DW 10-11RefAddr[5] - BWDREF Frame (Reference Frame 5)In AV1 Spec, it is mapped intoDPB [ref_frame_idx[BWDREF_FRAME-LAST_FRAME]] DW 12-13RefAddr[6] - ALTREF2 Frame (Reference Frame 6)In AV1 Spec, it is mapped intoDPB [ref_frame_idx[ALTREF2_FRAME-LAST_FRAME]] DW 14-15RefAddr[7] - ALTREF Frame (Reference Frame 7) In AV1 Spec, it is mapped intoDPB [ref_frame_idx[ALTREF_FRAME-LAST_FRAME]] Note : for 48 bit address, a pair of Dwords (i.e. 2) are needed to store each base address. Note : This reference frame naming convention is a legacy specification in the reference C model. Although the very first reference frame is labeled as INTRA, it is not reserved only for a previously decoded KEY frame or a decoded INTRA-ONLY NON-KEY frame. Any newly decoded frame can be stored in any one of these 8 reference frame buffers for future reference - it is up to the Application and Driver that handle the DPB management according to the bitstream and normative rules specified in AV1. Note : When IntraBC coding mode is used in the KEY Frame or in the INTRA-ONLY NON-KEY Frame, the intraBC coded block is decoded as a regular inter-coded block, and one of these 8 reference frame buffers will be directly used in a READ/WRITE fashion. ??? Note : the format of the pixels (Y, U and V components)stored inside these reference frame buffers is defined in the AV1_Surface_State Command. For monochrome video, all reference frame buffers can only have Luma Y component. Note :All reference frame buffers' surface addresses must be 4K byte aligned. There is a max. of 8 Reference Picture Buffer Addresses, and all share the same third address DW in specifying 48-

AVP_PIPE_BUF_ADDR_STATE

17	31:0	**Reference Frame Buffer Base Address Attributes** Format: MemoryAddressAttributes All reference frame buffers' surface addresses must be 4K byte aligned. There is a max. of 8 Reference Picture Buffer Addresses, and all share this same third address DW in specifying 48-bit address.
18..19	63:0	**Decoded Output Frame Buffer Address** Format: SplitBaseAddress4KByteAligned It specifies the physical memory base addresses of the frame buffer that stores the final pixel output of the AVP pipeline, just before the Film Grain unit(could be sent for display). Typically, it is a WRITE-only surface with pixel format specified in the AV1_SURFACE_STATE. It can be 4:0:0, 4:2:0, 4:2:2 or 4:4:4. Memory compression is applied to this surface. This buffer holds the result of reconstructing the current frame, including all the Post In-Loop Filtering Processes (Deblocker, CDEF, Super-Resolution, and/or Loop Restoration) that are enabled. There can be a few actions to follow: 1) Application and Driver will then place this newly decoded frame into the Display Buffer (DPB), as one of the 8 reference frames, only if it will be used for the decoding of a later frame(s) (??? a bit in the frame header is set to indicate this ???). 2) At the same time, if this newly decoded frame is to be displayed immediately, it will also be copied and sent to other system for further processing or directly to the Display Controller. 3) If Film Grain Injection is enabled in the sequence header, this newly decoded frame is further processed by the Out of Loop Film Grain Synthesis unit in a block-based pipeline fashion (continued from the block-based pixel reconstruction pipeline of the AV1 decoder). Note: when intraBC is active, it can also be a READ surface for the Motion Comp operation. ??? Note : there is only one write location along the in-loop decoding pipeline, and it is at the output of the Loop Restoration Filter. If any or all of the Post In-Loop Filters (Deblocker, CDEF, Super-Resolution, Loop Restoration)are disabled, the reconstructed pixels still need to pipethrough those disabled filters without change (i.e. simply bypass), until they reach the output of the Loop Restoration Filter. Note : if it is decided to implement some of the Post In-Loop Filters in software (as a separate pass - not being implemented yet), this buffer will hold the final pixel output frame of the HW pipeline. Note : this decoded output buffer if going to be used as a reference frame, then it is added into the Display Buffer (DPB) at the location specified by the frame level syntax element refresh_frame_flags. If the current frame is a KEY Frame or a SWITCH Frame, refresh_frame_flags is set to all 1's, so the entire DPB is initialized (KEY Frame)/re-initialized (SWITCH Frame) to the current frame.
20	31:0	**Decoded Output Frame Buffer Address Attributes** Format: MemoryAddressAttributes
21..23	95:0	**Reserved** Access: RO Format: MBZ

AVP_PIPE_BUF_ADDR_STATE		
24..25	63:0	**IntraBC Decoded Output Frame Buffer Address** Format: **SplitBaseAddress4KByteAligned** When IntraBC (frame level) is ON, the current decoded and reconstructed partial pixel frame is needed for motion compensation. The normal Decoder Output Frame Buffer is typically written out with memory compression ON, as such it is not suitable to be read back for performing motion compensation within the decoding of the same frame. A separate decoded output frame buffer is needed, whose surface buffer does not have memory compression turned ON. That is, when IntraBC is ON (frame level), AVP will write out two decoded output frame with identical content.
26	31:0	**IntraBC Decoded Output Frame Buffer Address Attributes** Format: **MemoryAddressAttributes**
27..28	63:0	**CDF Tables Initialization Buffer Address** Format: **SplitBaseAddress64ByteAligned** Base address for the CDF Tables Initialization Buffer. This Buffer is read-only. It is programmable with the initial CDF table set for the entire frame and for all its tiles at the very beginning before any decoding process. The content of this buffer must be the same for all tiles for the frame
29	31:0	**CDF Tables Initialization Buffer Address Attributes** Format: **MemoryAddressAttributes**
30..31	63:0	**CDF Tables Backward Adaptation Buffer Address** Format: **SplitBaseAddress64ByteAligned** Baseaddress of the CDF Tables Backward Adaptation Buffer. This Buffer stores the updated frame context of the largest tile in the current decoded frame. This is a WRITE-Only buffer.
32	31:0	**CDF Tables Backward Adaptation Buffer Address Attributes** Format: **MemoryAddressAttributes**
33..34	63:0	**AV1 Segment ID Read Buffer Address** Format: **SplitBaseAddress64ByteAligned** Specifies the 64 byte aligned buffer address for AV1 SegmentID buffer. This should contain the writeout SegmentID from previous frame and will be used to predict SegmentID for the current frame. Hardware will write out SegmentID of the current frame in the same address for the next frame. It is used for temporal prediction of segment ID in the current frame. The segment map has a granularity of 4x4 blocks.
35	31:0	**AV1 Segment ID Read Buffer Address Attributes** Format: **MemoryAddressAttributes**

AVP_PIPE_BUF_ADDR_STATE

36..37	63:0	**AV1 Segment ID Write Buffer Address** Format: **SplitBaseAddress64ByteAligned** Specifies the 64 byte aligned buffer address for AV1 SegmentID buffer. This should contain the writeout SegmentID of the current frame and will be used to predict SegmentID for later frame. This segment map buffer is attached to the current decoded frame as its auxiliary data, and are both stored together in the Display Buffer (DPB), if the current frame is to be used as a reference frame to decode later frame(s). The segment map has a granularity of 4x4 blocks.
38	31:0	**AV1 Segment ID Write Buffer Address Attributes** Format: **MemoryAddressAttributes**
39..54	511:0	**Collocated Motion Vector Temporal Buffer Base Address (TmvAddr[0-7])** Format: **SplitBaseAddress64ByteAligned[8]** Base address for the Collocated Motion Vector Temporal buffer. The 8 Temporal Buffersare defined in the following order: DW 38-39 TmvAddr[0] - INTRA Frame (Reference Frame 0) DW 40..41 TmvAddr1] - LAST Frame (Reference Frame 1) DW 42..43 TmvAddr[2] - LAST2 Frame (Reference Frame 2) DW 44..45 TmvAddr[3] - LAST3 Frame (Reference Frame 3) DW 46..47 TmvAddr4] - GOLDEN Frame (Reference Frame 4) DW 48..49 TmvAddr[5] - BWDREF Frame (Reference Frame 5) DW 50..51 TmvAddr[6] - ALTREF2 Frame (Reference Frame 6) DW 52..53 TmvAddr[7] - ALTREF Frame (Reference Frame 7) Note : There is a max. of 8 Collocated MV TemporalBuffer Addresses, and all share the same third address DW in specifying 48-bit address
55	31:0	**Collocated Motion Vector Temporal Buffer Base Address Attributes** Format: **MemoryAddressAttributes** Note : There is a max. of 8 Collocated MV TemporalBuffer Addresses, and all share this same third address DW in specifying 48-bit address.
56..57	63:0	**Current Frame Motion Vector Write Buffer Address** Format: **SplitBaseAddress64ByteAligned** Base address for the Current Motion Vector Temporal buffer.
58	31:0	**Current Frame Motion Vector Write Buffer Address Attributes** Format: **MemoryAddressAttributes**
59..61	95:0	**Reserved** Access: RO Format: MBZ

AVP_PIPE_BUF_ADDR_STATE		
62..63	63:0	**Bitstream Decoder/Encoder Line Rowstore Read/Write Buffer Address** Format: SplitBaseAddress64ByteAligned Specifies the 64 byte aligned buffer address for Bitstream Decode Line Rowstore
64	31:0	**Bitstream Decoder/Encoder Line Rowstore Read/Write Buffer Address Attributes** Format: MemoryAddressAttributes
65..66	63:0	**Bitstream Decoder/Encoder Tile Line Rowstore Read/Write Buffer Address** Format: SplitBaseAddress64ByteAligned Specifies the 64 byte aligned buffer address for Bitstream Decode Tile Line Buffer
67	31:0	**Bitstream Decoder/Encoder Tile Line Rowstore Read/Write Buffer Address Attributes** Format: MemoryAddressAttributes
68..69	63:0	**Intra Prediction Line Rowstore Read/Write Buffer Address** Format: SplitBaseAddress64ByteAligned Specifies the 64 byte aligned buffer address forIntra Prediction Line Rowstore
70	31:0	**Intra Prediction Line Rowstore Read/Write Buffer Address Attributes** Format: MemoryAddressAttributes
71..72	63:0	**Intra Prediction Tile Line Rowstore Read/Write Buffer Address** Format: SplitBaseAddress64ByteAligned Specifies the 64 byte aligned buffer address for Intra Prediction Tile Line Rowstore
73	31:0	**Intra Prediction Tile Line Rowstore Read/Write Buffer Address Attributes** Format: MemoryAddressAttributes
74..75	63:0	**Spatial Motion Vector Line Read/Write Buffer Address** Format: SplitBaseAddress64ByteAligned Base address for the Spatial Motion Vector Line buffer.
76	31:0	**Spatial Motion Vector Line Read/Write Buffer Address Attributes** Format: MemoryAddressAttributes
77..78	63:0	**Spatial Motion Vector Coding Tile Line Read/Write Buffer Address** Format: SplitBaseAddress64ByteAligned Base address for the Spatial Motion Vector Tile Line buffer.
79	31:0	**Spatial Motion Vector Tile Line Read/Write Buffer Address Attributes** Format: MemoryAddressAttributes
80..81	63:0	**Loop Restoration Meta Tile Column Read/Write Buffer Address** Format: SplitBaseAddress64ByteAligned Base Address forLoop Restoration Meta Tile Column Read/Write Buffer

AVP_PIPE_BUF_ADDR_STATE		
82	31:0	**Loop Restoration Meta Tile Column Read/Write Buffer Address Attributes** Format: **MemoryAddressAttributes**
83..84	63:0	**Loop Restoration Filter Tile Read/Write Line Y Buffer Address** Format: **SplitBaseAddress64ByteAligned** Base address for the Loop Restoration Filter Tile Line Y Buffer
85	31:0	**Loop Restoration Filter Tile Read/Write Line Y Buffer Address Attributes** Format: **MemoryAddressAttributes**
86..87	63:0	**Loop Restoration Filter Tile Read/Write Line U Buffer Address** Format: **SplitBaseAddress64ByteAligned** Base address for the Loop Restoration Filter Tile Line U Buffer
88	31:0	**Loop Restoration Filter Tile Read/Write Line U Buffer Address Attributes** Format: **MemoryAddressAttributes**
89..90	63:0	**Loop Restoration Filter Tile Read/Write Line V Buffer Address** Format: **SplitBaseAddress64ByteAligned** Base address for the Loop Restoration Filter Tile Line V Buffer
91	31:0	**BitField: Loop Restoration Filter Tile Read/Write Line V Buffer Address Attributes** Format: **MemoryAddressAttributes**
92..93	63:0	**Deblocker Filter Line Read/Write Y Buffer Address** Format: **SplitBaseAddress64ByteAligned** Base address of the filter line buffer (read/write) used by the Deblocking Filter.
94	31:0	**Deblocker Filter Line Read/Write Y Buffer Address Attributes** Format: **MemoryAddressAttributes**
95..96	63:0	**Deblocker Filter Line Read/Write U Buffer Address** Format: **SplitBaseAddress64ByteAligned** Base address of the filter line buffer (read/write) used by the Deblocking Filter.
97	31:0	**Deblocker Filter Line Read/Write U Buffer Address Attributes** Format: **MemoryAddressAttributes**
98..99	63:0	**Deblocker Filter Line Read/Write V Buffer Address** Format: **SplitBaseAddress64ByteAligned** Base address of the filter line buffer (read/write) used by the Deblocking Filter.
100	31:0	**Deblocker Filter Line Read/Write V Buffer Address Attributes** Format: **MemoryAddressAttributes**

AVP_PIPE_BUF_ADDR_STATE		
101..102	63:0	**Deblocker Filter Tile Line Read/Write Y Buffer Address** Format: **SplitBaseAddress64ByteAligned** Base address of the tile line buffer (read/write) used by the Deblocking Filter.
103	31:0	**Deblocker Filter Tile Line Read/Write Y Buffer Address Attributes** Format: **MemoryAddressAttributes**
104..105	63:0	**Deblocker Filter Tile Line Read/Write V Buffer Address** Format: **SplitBaseAddress64ByteAligned** Base address of the tile line buffer (read/write) used by the Deblocking Filter.
106	31:0	**Deblocker Filter Tile Line Read/Write V Buffer Address Attributes** Format: **MemoryAddressAttributes**
107..108	63:0	**Deblocker Filter Tile Line Read/Write U Buffer Address** Format: **SplitBaseAddress64ByteAligned** Base address of the tile line buffer (read/write) used by the Deblocking Filter.
109	31:0	**Deblocker Filter Tile Line Read/Write U Buffer Address Attributes** Format: **MemoryAddressAttributes**
110..111	63:0	**Deblocker Filter Tile Column Read/Write Y Buffer Address** Format: **SplitBaseAddress64ByteAligned** Base address of the tile column buffer (read/write) used by the Deblocking Filter.
112	31:0	**Deblocker Filter Tile Column Read/Write Y Buffer Address Attributes** Format: **MemoryAddressAttributes**
113..114	63:0	**Deblocker Filter Tile Column Read/Write U Buffer Address** Format: **SplitBaseAddress64ByteAligned** Base address of the tile column buffer (read/write) used by the Deblocking Filter.
115	31:0	**Deblocker Filter Tile Column Read/Write U Buffer Address Attributes** Format: **MemoryAddressAttributes**
116..117	63:0	**Deblocker Filter Tile Column Read/Write V Buffer Address** Format: **SplitBaseAddress64ByteAligned** Base address of the tile column buffer (read/write) used by the Deblocking Filter.
118	31:0	**Deblocker Filter Tile Column Read/Write V Buffer Address Attributes** Format: **MemoryAddressAttributes**
119..120	63:0	**CDEF Filter Line Read/Write Buffer Address** Format: **SplitBaseAddress64ByteAligned** Base address for the CDEF Filter Line buffer. It include YUV data

AVP_PIPE_BUF_ADDR_STATE

121	31:0	**CDEF Filter Line Read/Write Buffer Address Attributes** Format: **MemoryAddressAttributes**
122..127	191:0	**Reserved** Access: RO Format: MBZ
128..129	63:0	**CDEF Filter Tile Line Read/Write Buffer Address** Format: **SplitBaseAddress64ByteAligned** Base address for the CDEF Filter Tile Line buffer. It includes YUV data
130	31:0	**CDEF Filter Tile Line Read/Write Buffer Address Attributes** Format: **MemoryAddressAttributes**
131..136	191:0	**Reserved** Access: RO Format: MBZ
137..138	63:0	**CDEF Filter Tile Column Read/Write Buffer Address** Format: **SplitBaseAddress64ByteAligned** Base address for the CDEF Filter Tile Column buffer. It includes YUV data
139	31:0	**CDEF Filter Tile Column Read/Write Buffer Address Attributes** Format: **MemoryAddressAttributes**
140..141	63:0	**CDEF Filter Meta Tile Line Read/Write Buffer Address** Format: **SplitBaseAddress64ByteAligned** Base address for the CDEF Filter Meta Tile Line buffer.
142	31:0	**CDEF Filter Meta Tile Line Read/Write Buffer Address Attributes** Format: **MemoryAddressAttributes**
143..144	63:0	**CDEF Filter Meta Tile Column Read/Write Buffer Address** Format: **SplitBaseAddress64ByteAligned** Base address for the CDEF Filter Meta Tile Column buffer.
145	31:0	**CDEF Filter Meta Tile Column Read/Write Buffer Address Attributes** Format: **MemoryAddressAttributes**
146..147	63:0	**CDEF Filter Top-Left Corner Read/Write Buffer Address** Format: **SplitBaseAddress64ByteAligned** Base address for the CDEF Filter Tile Column buffer.
148	31:0	**CDEF Filter Top-Left Corner Read/Write Buffer Address Attributes** Format: **MemoryAddressAttributes**

AVP_PIPE_BUF_ADDR_STATE		
149..150	63:0	**Super-Res Tile Column Read/Write Y Buffer Address** Format: SplitBaseAddress64ByteAligned Base address for the Super-ResolutionTile Column buffer.
151	31:0	**Super-Res Tile Column Read/Write Y Buffer Address Attributes** Format: MemoryAddressAttributes
152..153	63:0	**Super-Res Tile Column Read/Write U Buffer Address** Format: SplitBaseAddress64ByteAligned Base address for the Super-Resolution Tile Column buffer.
154	31:0	**Super-Res Tile Column Read/Write U Buffer Address Attributes** Format: MemoryAddressAttributes
155..156	63:0	**Super-Res Tile Column Read/Write V Buffer Address** Format: SplitBaseAddress64ByteAligned Base address for the Super-Resolution Tile Column buffer.
157	31:0	**Super-Res Tile Column Read/Write V Buffer Address Attributes** Format: MemoryAddressAttributes
158..159	63:0	**Loop Restoration Filter Tile Column Read/Write Y Buffer Address** Format: SplitBaseAddress64ByteAligned Base address for the Loop Restoration Filter Tile Column buffer.
160	31:0	**Loop Restoration Filter Tile Column Read/Write Y Buffer Address Attributes** Format: MemoryAddressAttributes
161..162	63:0	**Loop Restoration Filter Tile Column Read/Write U Buffer Address** Format: SplitBaseAddress64ByteAligned Base address for the Loop Restoration Filter Tile Column buffer.
163	31:0	**Loop Restoration Filter Tile Column Read/Write U Buffer Address Attributes** Format: MemoryAddressAttributes
164..165	63:0	**Loop Restoration Filter Tile Column Read/Write V Buffer Address** Format: SplitBaseAddress64ByteAligned Base address for the Loop Restoration Filter Tile Column buffer.
166	31:0	**Loop Restoration Filter Tile Column Read/Write V Buffer Address Attributes** Format: MemoryAddressAttributes
167..169	95:0	**Reserved** Access: RO

AVP_PIPE_BUF_ADDR_STATE

		Format: MBZ
170..175	191:0	**Reserved** Access: RO Format: MBZ
176..177	63:0	**Decoded Frame Status/Error Buffer Base Address** Format: **SplitBaseAddress64ByteAligned** **Decoder Mode :** Specifies the 64 byte aligned buffer address for writing a single status/error cache-line sized record into memory when the Pic Status/Error Report Enable is set in the AVP_PIPE_MODE_SELECT command. The pic status/error record is written by hardware after the picture is decoded. The content of this memory location can later be read by the driver only.
178	31:0	**Decoded Frame Status/Error Buffer Base Address Attributes** Format: **MemoryAddressAttributes**
179..180	63:0	**Decoded Block Data Streamout Buffer Address** Exists If: //Decoder Only Format: **SplitBaseAddress64ByteAligned** Buffer address for outputting the per-block indirect data to memory when **StreamOutEnable** is set in the AVP_PIPE_MODE_SELECT command. For Decoder: this field is used for transcoding purpose.
181	31:0	**Decoded Block Data Streamout Buffer Address Attributes** Exists If: //Decoder Only Format: **MemoryAddressAttributes**
182..184	95:0	**Reserved**
185..187	95:0	**Reserved**

AVP_PIPE_MODE_SELECT

AVP_PIPE_MODE_SELECT

Source: VideoCS

Length Bias: 2

The AVP Pipeline is selected with the **Media Instruction Opcode "8h"** for all AVP Commands. Each AVP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

The workload for the AVP pipeline is tile based. Once the bit stream DMA is configured with the AVP_BSD_OBJECT command for a tile in a frame, and the tile's bitstream is presented to the AVP, the tile decoding will begin.
AVP pipeline is stateless, i.e. there is no states saved between the decoding of each tile. Hence all sequence, frame and segment state commands have to be resent before the tile coding command and the BSD object command.
The AVP_PIPE_MODE_SELECT command is responsible for general pipeline level configuration that would normally be set once for a single stream encode or decode and would not be modified on a frame workload basis.
This is a frame level state command and is shared by both encoding and decoding processes.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 3h Codec/Engine Name Format: OpCode Codec/Engine Name = AVP = 3h
	22:16	**Media Instruction Command** Default Value: 0h AVP_PIPE_MODE_SELECT Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Format: =n (Excludes Dwords 0, 1). Value: 4h Name:

AVP_PIPE_MODE_SELECT

1	31:24	**Reserved**
	23	**Reserved**
	22:17	**Reserved** Access: RO Format: MBZ
	16:15	**Pipe working Mode** This programs the working mode for AVP pipe. **Value** / **Name** / **Description** 00b / Legacy decoder/encoder mode (Single pipe) / This is for single pipe mode standalone mode. It is used by both decoder and encoder. 01b / Reserved / 11b / Decoder Scalable mode with MSAC in real tiles (Scalable Multi-pipe) / This is for multiple-pipe scalable mode decoder mode in real tiles. MSAC and reconstruction will run together. Each pipes will run in real tiles vertically.
	14:13	**Multi-Engine Mode** This indicates the current pipe is in single pipe mode or if in scalable mode is in left/right/middle pipe in multi-engine mode. **Value** / **Name** / **Description** 00b / Single Engine Mode or MSAC FE only decode mode / This is for single engine mode (legacy) OR MSAC FE only decode mode During AV1Decoder Scalability Real Tile Mode, for the last phase, it is possible to have single tile column left. In this case, it should be programmed with pipe as a single engine mode (using this value). For example, for 9 tile column running on 4 pipes. The first two phases will use all 4 pipes and finish 8 tile column. The remaining one column will be processed as last third phase as single tile column. 01b / Pipe is the left engine in a Multi-engine mode / Current pipe is the most left engine while running in scalable multi-engine mode 10b / Pipe is the right engine in a Multi-engine mode / Current pipe is the most right engine while running in scalable multi-engine mode 11b / Pipe is one of the middle engine in a Multi-engine mode / Current pipe is in one of the middle engine while running in scalable multi-engine mode
	12	**Reserved** Access: RO Format: MBZ

AVP_PIPE_MODE_SELECT

11 — Reserved

Access:	RO
Format:	MBZ

10 — Reserved

Access:	RO
Format:	MBZ

9:8 — Reserved

Access:	RO
Format:	MBZ

7:5 — Codec Standard Select

Value	Name
2	AV1

4 — Reserved

Access:	RO
Format:	MBZ

3 — Pic Status/Error Report Enable

Format:	Enable

Value	Name	Description
0	Disable	Disable status/error reporting
1	Enable	Status/Error reporting is written out once per picture. The Pic Status/Error Report ID in DWord3 along with the status/error status bits are packed into one cache line and written to the Status/Error Buffer address in the AVP_PIPE_BUF_ADDR_STATE command. Must be zero for encoder mode.

2:1 — Reserved

Format:	MBZ

0 — Codec Select

Format:	U1

Value	Name
0	Decode

2 — 31:0 — Reserved

Access:	RO
Format:	MBZ

3 — 31:0 — Pic Status/Error Report ID

Format:	U32

The Pic Status/Error Report ID is a unique 32-bit unsigned integer assigned to each picturestatus/error output. Must be zero for encoder mode.

AVP_PIPE_MODE_SELECT		
		Programming Notes Software must program different Status/Error Buffer addresses between pictures; otherwise thehardware might overwrite previously written data.
4	31:0	**Reserved** Access: RO Format: MBZ
5	31:0	**Reserved** Access: RO Format: MBZ

AVP_SEGMENT_STATE

AVP_SEGMENT_STATE

Source: VideoCS

Length Bias: 2

When segmentation is enabled, this Segment State command is issued once per segment. There can be maximum 8 segments specified to decode a given frame, so this Segment State Command can be issued maximum 8 times. It is assumed that there is no gap in segment IDs. So, when the AVP PIC States specified that the number of active

When segmentation is disabled, driver still sends out this command once for segment id = 0. HW needs to check the segmentation enable flag from AVP_PIC_STATE Command as well to distinguish from the case when segmentation is enabled for segment id = 0.

Each segment can have its own specification of enabling any one of the 8 features defined in AV1 and their corresponding feature data. When a feature is not enabled, its feature data is defaulted to 0. When segmentation is not enabled, all the features are disabled and their corresponding feature data are set to 0.

Segment State Command also provides other segment related parameters.

It is assumed that HW is keeping a copy of the complete AV1 QM Matrix Table for all color components inside its internal memory, and Driver only needs to send the qm_level as index into this Table.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 3h Codec/Engine Name Format: OpCode Codec/Engine Name = AVP = 3h
	22:16	**Media Instruction Command** Default Value: 32h AVP_SEGMENT_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 2h Name:

AVP_SEGMENT_STATE

1	31:3	**Reserved** Access: RO Format: MBZ
	2:0	**Segment ID** Format: U3 The Segment ID specifies which one of the 8 possible segments that the current Segment State Command is associated with. Segment ID is in the range of [0 ... 7]. Maximum, there can be 8 segments specified for decoding a given frame. Segment ID=0, even when segmentation is disabled.
2	31:28	**Segment Chroma V QM Level** Format: U4 It specifies one of the 16 QM matrices to be used for the Chroma V component of the current segment. It is in the range of [0..15]. QM Level = 15, is a bypass, with no scaling. Default is 15. If the current segment is coded as lossless or if qm matrix is not enabled (frame level SE : using_qmatrix = = 0), then Segment QM Level is set to 15. Chroma V qmlevel = (segment lossless flag || using_qmatrix == 0) ? 15: qm_v (frame level SE). If separate_uv_delta_q (sequence level SE) is set to 0, Segment Chroma V QM Level is set to the same value as of Segment Chroma UQM Level. If separate_uv_delta_q (sequence level SE) is set to 1, Segment Chroma V QM Level can be set independently to a different value as of Segment Chroma U QM Level. For a given frame, all lossy segments should have the same value for Vqm_level.
	27:24	**Segment Chroma U QM Level** Format: U4 It specifies one of the 16 QMmatrices to be used for the Chroma U component of the current segment. It is in the range of [0..15]. QM Level = 15, is a bypass, with no scaling. Default is 15. If the current segment is coded as lossless or if qm matrix is not enabled ((frame level SE : using_qmatrix = = 0), then Segment QM Level is set to 15. Chroma U qmlevel = (segment lossless flag || using_qmatrix == 0) ? 15: qm_u (frame level SE). For a given frame, all lossy segments should have the same value for Uqm_level.
	23:20	**Segment Luma Y QM Level** Format: U4 It specifies one of the 16 QMmatrices to be used for the Luma Y component of the current segment. It is in the range of [0..15]. QM Level = 15, is a bypass, with no scaling. Default is 15. If the current segment is coded as lossless or if qm matrix is not enabled (frame level SE : using_qmatrix = = 0), then Segment QM Level is set to 15. Luma qmlevel = (segment lossless flag || using_qmatrix == 0) ? 15: qm_y (frame level SE).

		For a given frame, all lossy segments should have the same value forY qm_level.
	19	**Segment Lossless Flag** Format: U1 0 : the current segment is not coded as lossless. Default is 0. 1 : the current segment is coded as lossless. Segment Lossless Flag = (clamp[(base_qindex + optional segment delta qindex), 0, 255] == 0) && (y_dc_delta_q == 0) && (u_dc_delta_q == 0) && (u_ac_delta_q == 0) && (v_dc_delta_q == 0) && (v_ac_delta_q == 0) It is computing using syntax elements all from the bitstream, read in the uncompressed header. In encoder mode, if one of the segments is lossless then all segments must be lossless in the frame.
	18	**Segment Block GlobalMV Flag** Format: U1 It specifies the segment feature data corresponding to the feature id = SEG_LVL_GLOBALMV = 7. Segment Block GlobalMV Flag = 0, specifies nothing. Segment Block GlobalMV Flag = 1, specifies the blocks of the current segment are all coded with Y inter prediction mode set to GLOBALMV. There is no feature data read from the bitstream for setting the Segment Block GlobalMV Flag. It iis derived from it corresponding segment feature mask. Segment Block GlobalMV Flag =seg_feature_mask & (1 « SEG_LVL_GLOBALMV) If Segment Block GlobalMV Flag = 1, The Y inter prediction mode is set to ZeroMV (for using zero MV), and RefFrame[0] (??? elaborate) is set to LAST_FRAME, and RefFrame[1] is set to NONE (-1).
	17	**Segment Block Skip Flag** Format: U1 It specifies the segment feature data corresponding to the feature id = SEG_LVL_SKIP = 6. Segment Block Skip Flag = 0, specifies there is no segment block skip. Segment Block Skip Flag = 1, specifies the blocks of the current segment are all skipped. These blocks are then having all coefficients and MV set to 0 (Y inter prediction mode is set to zeroMV mode), but these blocks can still have coding mode specified in the bitstream. There is no feature data read from the bitstream for setting the Segment Block Skip Flag. It iis derived from it corresponding segment feature mask. Segment Block Skip Flag =seg_feature_mask & (1 « SEG_LVL_SKIP) If Segment Block Skip Flag = 1, no coefficient is present in the bitstream, and are all set to 0. The Y inter prediction mode is set to ZeroMV (for using zero MV), and RefFrame[0] is set to LAST_FRAME , and RefFrame[1] is set to NONE (-1). It is used to derived the skip_mode flag and skip flag.
	16:8	**Segment Delta Qindex** Format: S8 It specifies the segment feature data corresponding to the feature id =SEG_LVL_ALT_Q = 0.

		It is a 9-bit signed 2's complement number, in the range of [-255, +255]. Default is 0. Value -256 is not allowed.
	7:0	**Segment Feature Mask** Format: U8 It is an 8-bit mask specifies which of the 8 segment features are active in the current segment specifications. It is also known as feature enable mask or feature active mask. This mask can indicate 0 or up to 8 features are enabled. Any feature can be active or inactive independent of the other features. seg_feature_mask & (1 « feature_id) = 0, the feature (identified by feature_id) is not enabled/active in the current segment. seg_feature_mask & (1 « feature_id) = 1, the feature (identified by feature_id) is enabled/active in the current segment. Feature_ID (enum)Feature Name 0 SEG_LVL_ALT_Q, // Use alternate Quantizer. 1 SEG_LVL_ALT_LF_Y_V, // Use alternate loop filter value on y plane vertical. 2 SEG_LVL_ALT_LF_Y_H, // Use alternate loop filter value on y plane horizontal. 3 SEG_LVL_ALT_LF_U, // Use alternate loop filter value on u plane. 4 SEG_LVL_ALT_LF_V, // Use alternate loop filter value on v plane. 5 SEG_LVL_REF_FRAME, // Optional Segment reference frame. 6 SEG_LVL_SKIP,// Optional Segment zeroMV (0,0) + skip mode. 7 SEG_LVL_GLOBALMV. // Optional Segment zeroMV (0,0). Each segment has its own seg_feature_mask. When a segment is not being used or when segmentation is disabled, the corresponding seg_feature_mask has all 8-bits set to 0. Default all 8-bits are 0.
3	31	**Reserved** Access: RO Format: MBZ
	30:28	**Segment Reference Frame** Format: U3 It specifies the segment feature data corresponding to the feature id =SEG_LVL_REF_FRAME = 5. It is an alternate specification of one of the 8 possible reference frames for a Motion Compensation. =0 for INTRA_FRAME =1 for LAST_FRAME =2 for LAST2_FRAME =3 for LAST3_FRAME =4 for GOLDEN_FRAME =5 for BWDREF_FRAME =6 for ALTREF2_FRAME =7 for ALTREF_FRAME It is in the range of [0..7]. Default is 0.

AVP_SEGMENT_STATE		
	27:21	**Segment Delta Loop Filter Level Chroma V** Format: S6 It specifies the segment feature data corresponding to the feature id =SEG_LVL_ALT_LF_V= 4. It is associated with filter_level_v read from the uncompressed header. It is a 7-bit signed 2's complement number, in the range of [-63, +63]. Default is 0.
	20:14	**Segment Delta Loop Filter Level Chroma U** Format: S6 It specifies the segment feature data corresponding to the feature id =SEG_LVL_ALT_LF_U= 3. It is associated with filter_level_u read from the uncompressed header. It is a 7-bit signed 2's complement number, in the range of [-63, +63]. Default is 0.
	13:7	**Segment Delta Loop Filter Level Luma Horizontal** Format: S6 It specifies the segment feature data corresponding to the feature id =SEG_LVL_ALT_LF_Y_H= 2. It is associated with filter_level[1] read from the uncompressed header. It is a 7-bit signed 2's complement number, in the range of [-63, +63]. Default is 0.
	6:0	**Segment Delta Loop Filter Level Luma Vertical** Format: S6 It specifies the segment feature data corresponding to the feature id =SEG_LVL_ALT_LF_Y_V = 1. It is associated with filter_level[0] read from the uncompressed header. It is a 7-bit signed 2's complement number, in the range of [-63, +63]. Default is 0.

AVP_SURFACE_STATE

AVP_SURFACE_STATE

Source: VideoCS

Length Bias: 2

The AVP Pipeline is selected with the **Media Instruction Opcode "8h"** for all AVP Commands. Each AVP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

The AVP_SURFACE_STATE command is responsible for defining the frame buffer pitch and the offset of the chroma component.
This is a picture level state command and is shared by both encoding and decoding processes.
For Decoder, this command is issued once per surface type. There is one reconstructed surface, 8 reference pictures surfaces and one optional IntraBC Decoded Surface (only if IBC is ON).
For Encoder, this command is issued once per surface type. There are4 surface types :source down scaled, source original, reference and reconstructed picture. All reference frames are defined with the same surface command.
Tile-Yf and Tile-Ys are not supported, but HW interface still need to keep these bits as reserved bits.
Note : When NV12 and Tile Y are being used, full pitch and interleaved UV is always in use. U and V Xoffset must be set to 0; U and V Yoffset must be 8-pixel aligned. For 10-bit pixel, P010 surface definition is being used.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 3h Codec/Engine Name Format: OpCode Codec/Engine Name = AVP = 3h
	22:16	**Media Instruction Command** Default Value: 1h SURFACE_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1).

AVP_SURFACE_STATE

		Value / **Name**: 1h
1	31:28	**Surface Id** Format: U4

Value	Name	Description	Programming Notes
0h	Reconstructed Picture	This is for the reconstructed picture surface state	
6h	AV1 INTRA FRAME	This is for AV1 Intra Frame (Reference Picture 0). Each AV1 Reference Pictures can have different size so a separate ID is needed.	
7h	AV1 Last Frame	This is for AV1Last Frame (Reference Picture 1). Each AV1 Reference Pictures can have different size so a separate ID is needed.	
8h	AV1 Last2 Frame	This is for AV1 Last2 Frame (Reference Picture 2). Each AV1 Reference Pictures can have different size so a separate ID is needed.	
9h	AV1 Last3 Frame	This is for AV1 Last3 Frame (Reference Picture 3). Each AV1 Reference Pictures can have different size so a separate ID is needed.	
Ah	AV1 Golden Frame	This is for AV1 Golden Frame (Reference Picture 4). Each AV1 Reference Pictures can have different size so a separate ID is needed.	
Bh	AV1 Bwdref Frame	This is for AV1 Bwdref Frame (Reference Picture 5). Each AV1 Reference Pictures can have different size so a separate ID is needed.	
Ch	AV1 Altref2 Frame	This is for AV1 Altref2 Frame (Reference Picture 6). Each AV1 Reference Pictures can have different size so a separate ID is needed.	
Dh	AV1 Altref Frame	This is for AV1 Altref Frame (Reference Picture 7). Each AV1 Reference Pictures can have different size so a separate ID is needed.	
Eh	IntraBC Decoded Frame	This is for AV1 IntraBC Decoded Frame. It will be used both Read/Write at the same time. This surface requires multiple of 8 pixels on both width and height.	This surface must be programmed as uncompressed

	27:17	**Reserved** Access: RO Format: MBZ
	16:0	**Surface Pitch Minus1** Format: U17-1

This field specifies the surface pitch in (#Bytes - 1).

Programming Notes
For TileYF and TileYS surfaces, the range is dependent on the Cu parameter (refer to Memory Data Formats section for the definition of the Cu parameter depending on the case). The range in bytes is [2^{Cu}-1, 131071] -> [(2^{Cu})B, 128KB] = [1 tile, 128KB/(2^{Cu}) tiles] The field specifies the surface pitch in (#Bytes - 1)
For tiled surfaces, the pitch must be a multiple of the tile width (i.e.128 bytes aligned). If Half Pitch for Chroma is set, this field must be a multiple of two tile widths for tiled surfaces, or a multiple of 2 bytes for linear surfaces. For Y-tiled surfaces: Range = [127, 131071] to [128B,128KB] = [1 tile, 1024 tiles]

2 — 31:27 — Surface Format

Format:	U5

Specifies the format of the surface.
P010V is only applied to the Surface Id=Fh (AV1 CDEF pixels streamout), encoder only. And also the 2 LSBs are removed, so this P010V surface is actually an 8-bit surface.

Value	Name
4h	PLANAR_420_8
Dh	P010

2 — 26 — Reserved

Access:	RO
Format:	MBZ

2 — 25 — Reserved

Access:	RO
Format:	MBZ

2 — 24:15 — Reserved

Access:	RO
Format:	MBZ

2 — 14:0 — Y Offset for U(Cb) in pixel

Format:	U15

This field specifies the vertical offset in rows from the **Surface Base Address** to the start(origin) of the U(Cb) plane or the interleaved UV plane if **Interleave Chroma** is enabled. This field is only used for PLANAR surface formats.

Programming Notes
• For PLANAR_420 surface formats, the alignment of this field follows the tile mode described in bits 14:13 of the **Memory Address Attributes** table. • TileY (legacy 4k) - 8 pixel aligned • TileYF (New 4k) - 64 pixel aligned • TileYS (64k) - 256 pixel aligned

3	31:16	**Y Offset for V(Cr)** Format: U16 Row Offset in Pixels This field specifies the vertical offset in rows from the Surface Base Address to the start (origin) of the V(Cr) plane. This field is only used for PLANAR surface formats with Interleave Chroma disabled. **Programming Notes** • TileY (legacy 4k) - 8 pixel aligned • TileYF (New 4k) - 64 pixel aligned • TileYS (64k) - 256 pixel aligned
	15:0	**Default Alpha Value**
4	31:21	**Reserved** Access: RO Format: MBZ
	20:16	**Reserved** Access: RO Format: MBZ
	15	**Compression Type for Altref Frame** This bit is for AV1 Altref Frame (Reference Picture 7). Valid only when Memory Compression for Altref Frame is enabled. Value: 0 — Name: Media compression Enabled **[Default]** Value: 1 — Name: Render compression Enabled
	14	**Compression Type for Altref2 Frame** This bit is for AV1 Altref2 Frame (Reference Picture 6). Valid only when Memory Compression for Altref2 Frame is enabled. Value: 0 — Name: Media compression Enabled **[Default]** Value: 1 — Name: Render Compression Enabled
	13	**Compression Type for Bwdref Frame** This bit is for AV1 Bwdref Frame (Reference Picture 5). Valid only when Memory Compression for Bwdref Frame is enabled. Value: 0 — Name: Media compression Enabled **[Default]** Value: 1 — Name: Render Compression Enabled
	12	**Compression Type for Golden Frame** This bit is for AV1 Golden Frame (Reference Picture 4). Valid only when Memory Compression for Golden Frame is enabled.

Value	Name
0	Media compression Enabled **[Default]**
1	Render Compression Enabled

11 **Compression Type for Last3 Frame**
This bit is for AV1 Last3 Frame (Reference Picture 3). Valid only when Memory Compression for Last3 Frameis enabled.

Value	Name
0	Media compression Enabled **[Default]**
1	Render Compression Enabled

10 **Compression Type for Last2 Frame**
This bit is for AV1 Last2 Frame (Reference Picture 2). Valid only when Memory Compression for Last2 Frame is enabled.

Value	Name
0	Media compression Enabled **[Default]**
1	Render Compression Enabled

9 **Compression Type for Last Frame**
This bit is for AV1 Last Frame (Reference Picture 1). Valid only when Memory Compression for Last Frameis enabled.

Value	Name
0	Media compression Enabled **[Default]**
1	Render Compression Enabled

8 **Compression Type for Intra Frame**
This bit is for Intra Frame (Reference Picture 0). Valid only when Memory Compression for Intra Frame is enabled.

Value	Name
0	Media compression Enabled **[Default]**
1	Render Compression Enabled

7 **Memory Compression Enable for AV1 Altref Frame**
This bit is for AV1Altref Frame (Reference Picture 7).

Value	Name
1	Memory Compression Enable
0	Memory Compression Disable

6 **Memory Compression Enable for AV1 Altref2 Frame**
This bit is for AV1Altref2 Frame (Reference Picture 6).

Value	Name
1	Memory Compression Enable
0	Memory Compression Disable

	5	**Memory Compression Enable for AV1 Bwdref Frame** This bit is for AV1 Bwdref Frame (Reference Picture 5). **Value** / **Name** 1 – Memory Compression Enable 0 – Memory Compression Disable
	4	**Memory Compression Enable for AV1 Golden Frame** This bit is for AV1 Golden Frame (Reference Picture 4). **Value** / **Name** 1 – Memory Compression Enable 0 – Memory Compression Disable
	3	**Memory Compression Enable for AV1 Last3 Frame** This bit is for AV1 Last3 Frame (Reference Picture 3). **Value** / **Name** 1 – Memory Compression Enable 0 – Memory Compression Disable
	2	**Memory Compression Enable for AV1 Last2 Frame** This bit is for AV1 Last2 Frame (Reference Picture 2). **Value** / **Name** 1 – Memory Compression Enable 0 – Memory Compression Disable
	1	**Memory Compression Enable for AV1 Last Frame** This bit is for AV1 Last Frame (Reference Picture 1). **Value** / **Name** 1 – Memory Compression Enable 0 – Memory Compression Disable
	0	**Memory Compression Enable for AV1 Intra Frame** This bit is for AV1 Intra Frame (Reference Picture 0). **Value** / **Name** 1 – Memory Compression Enable 0 – Memory Compression Disable

AVP_TILE_CODING

AVP_TILE_CODING

Source: BSpec

Length Bias: 1

Programming Notes

This command is used only for AV1codec. It is issued for every tile of a frame. If a frame is composed of only 1 tile, it is still being issued. Tiling and Tile Group organization in AV1 cannot be disabled, a frame minimum must have 1 tile. Currently, each batch buffer can contain only 1 tile to be processed, it cannot contain more than 1 tile or the entire tile group of tiles.

When the tile width exceeds 4096 pixels or the tile area exceeds 4096x2304 pixels, tiling must be performed and number of tiles in such frame must be >1. There is no mandatory tiling driven by tile height. The frame height in pixels will limit the allowed tile height in extreme situation. Hence, the AVP_TILE_CODING can be issued multiple times for decoding a frame.

Since AVP HW pipeline is stateless, all sequence, frame and segment level states (coding parameters in all Frame Level State Commands) must be resent before sending each TILE_CODING_STATE command.

Although tile size is specified in SuperBlock unit, the minimum tile size is actually set to be 8x8 pixels (which is the same as the minimum frame size in pixels). It can also happen to the rightmost tile column and bottommost tile row of a frame which is not divisible by the SuperBlock size - this leads to the presence of partial tile and partial SuperBlock handling.

AV1 supports both

1) a uniform-spacing tiling scheme (as in VP9, which is always in the form of 2^N x 2^M number of tiles, for the entire frame), and

2) a non-uniform-spacing tiling scheme. Bitstream syntax elements will specify the width and height of each tile size in the frame.

AVP HW pipeline is a tile-based codec engine, it does not need to distinguish between these two tiling schemes. Driver will take care of the difference and details of these tiling schemes. At the end, Driver will send out one tile at a time with all the related tile information to the HW through this TILE_CODING State Command.

In AV1, a frame is partitioned by tile row and tile column. That is, a tile boundary must go across the full frame width or the full frame height only. There is no tiling within a tile.

For AV1, the max number of tiles per frame is set to 256 in the LEVEL definition for regular video decoding. The ext-tile (Virtual Reality mode, currently not supported) has a different tiling configuration, constraints and definition.

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	Opcode
	28:27	**Pipeline Type**	
		Default Value:	2h
		Format:	Opcode
	26:23	**Media Instruction Opcode**	
		Default Value:	3h Codec/Engine Name
		Format:	Opcode

AVP_TILE_CODING

	22:16	**Media Instruction Command** Default Value: 15h AVP_TILE_CODING Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: U12 Excludes Dwords 0 & 1 Value: 4h — Name: **[Default]**
1	31:24	**Tile Group ID** Format: U8 It specifies the Tile Group the current tile belongs to. It is intel specific parameter, not present in the AV1 bitstream. Each tile is belonging to a Tile Group, and there can be multiple TGs in a frame. All TGs of a frame are lined up in raster order within the frame. Each TG is received from the bitstream as a separate TG_OBU. So, a frame can receive multiple TG_OBUs. Currently, all TG_OBUs of a frame must be received in the correct sequential order (not arbitrary order is defined yet). A TG can start at any tile position of a frame (e.g. at the beginning of a tile row, in the middle of a tile row, or at the end of a tile row). A TG does not break by the frame width, and can continue to the next row of tiles below. A TG can span multiple tile rows. A TG can end at any tile position of a frame(e.g. at the beginning of a tile row, in the middle of a tile row, or at the end of a tile row). A TG can be viewed as a linear chain of tiles (not necessary a 2-D rectangular/square region of a frame). A TG can maximum contain the whole frame of tiles, or minimum contain only a single tile of the frame. A frame can be entirely coded as a single tile in a single TG. Each TG is assigned an unique ID number, in raster increasing order. This numbering is intel specific, not defined in the spec. There is no jump or gap in TG ID value between two adjacent TG. The top-left most tile of a frame is always has TG ID = 0. The bottom-right most tile of a frame is always has the highest TG ID value. Tile Group ID is in the range [0..255]. The max value 255 is intel specific. Note : the max number of TG per frame is set equal to the max number of tiles allowed in a frame, since each tile can belong to a different TG.
	23:12	**TG Tile Num** Format: U12 Specify the Tile Number inside a Tile Group. This numbering is intel specific, not present in the AV1 bitstream. All tiles of a TG are numbered in sequential raster order, starting from 0. TG Tile Num is reset at each TG boundary. That is, the very first tile of a TG is always assigned a TG Tile Num of 0. The last tile of a TG has the largest Tile Num of that TG. So, the total number of tiles in a frame = (the largest Frame Tile ID + 1) = sum for all TG [largest

		TG Tile Num+ 1]. Max number of tiles in a TG is currently limited to 128 at the highest level 6.3 being defined for regular video. TG Tile Num is in the range [0..255], starting from 0. The upper 5bits are reserved for future expansion.
	11:0	**Frame Tile ID** Format: U12 All the tiles of a frame are labeled with a Frame Tile ID in a sequence raster order, starting from 0. The very first tile of a frame (top-left most corner) has Frame Tile ID = 0, and the last tile of a frame (bottom-right most corner)has the largest Frame Tile ID. So, the total number of tiles in a frame = (the largest Frame Tile ID + 1) = sum for all TG [largest Tile Num in a TG + 1]. Frame Tile ID is numbered from the 2 TG syntax elements TG_START and TG_END, or their default value when no present in the bitstream. Max number of tiles per frame is currently limited to 128 at the highest level 6.3 being defined for regular video. But for VR Large Scale Tile, max number of tiles per tile list is 512.Frame Tile ID is in the range [0..511], starting from 0.The upper 3bits are reserved for future expansion. Frame Tile ID numbering is consistent with the spec definition of tile ordering in a Tile Group and in a frame. Frame Tile ID does not get reset to 0 at Tile Group boundary (as the TG Tile Num does). Frame Tile ID numbering is not affected by dependency setting across tiles.
2	31:26	**Reserved** Access: RO Format: MBZ
	25:16	**Tile Row Position in SB Unit** Format: U10 Specify the row (y-) position of the current tile to be processed in a frame. The position is specified in SuperBlock unit. The SuperBlock unit is specified in the Sequence Header; it can be either 64x64 pixels or 128x128 pixels. For regular video, the decoded tile pixels are placed in the same tile location in the decoded output frame as the coded tile in the bitstream. But for VR Large Scale Tile, the decoded tile pixels can be placed in a different location predefined in the decoded pseudo output frame as the coded tile in the bitstream. This field is used for both regular video and VR Large Scale Tile. For VR Large Scale Tile, this field is programmed to be the coded tile position for MotionCompensation purpose, and the Output Decoded Tile Row Position field is then programmed to be the position in the decoded pseudo output frame for placing the decoded pixels of the tile, A Tile contains a 2D array of SB units. A Tile min. size is 1 SB unit. The Tile position is based on the top-left most corner SB unit of the Tile. The very first tile of a frame (top-left most corner) has the position [Tile Column Position in SB Unit , Tile Row Position in SB Unit] = [0, 0] The frame height is rounded up to an integer multiple of the Superblock unit. A frame height that is less than a SB unit (but must be >=8 pixels), is rounded up to 1 SB unit for

the purpose of defining a tile.
A frame height that is not divisible by tile height, the last tile row of the frame will have a smaller tile height, but still an integer multiple of SB unit. It does not affect the Tile Row Position in SB Unit of these partial Tiles.
Tile Row Position in SB unit is derived from the frame level syntax elements in tiling specifications (tile_info()).
Max Frame Height = 64K, hence the max number of SB unit= 1024, when the SB unit is 64x64 pixels, and the max Tile Row Position in SB unit can be 1023.
Intel supports Tile Row Position in SB unit only in the range of [0..255], for 16K Max Frame Height. starting from 0 at the top-left most corner of a frame.

Value	Name	Description
[0,255]	16K_Below	This allows support up to 16K picture

15:10 **Reserved**

Access:	RO
Format:	MBZ

9:0 **Tile Column Position in SB Unit**

Format:	U10

Specify the column (x-) position of the current tile to be processed in a frame. The position is specified in SuperBlock unit. The SuperBlock unit is specified in the Sequence Header; it can be either 64x64 pixels or 128x128 pixels.
For regular video, the decoded tile pixels are placed in the same tile location in the decoded output frame as the coded tile in the bitstream. But for VR Large Scale Tile, the decoded tile pixels can be placed in a different location predefined in the decoded pseudo output frame as the coded tile in the bitstream. This field is used for both regular video and VR Large Scale Tile.
For VR Large Scale Tile, this field is programmed to be the coded tile position for MotionCompensation purpose, and the Output Decoded Tile Column Position field is then programmed to be the position in the decoded pseudo output frame for placing the decoded pixels of the tile,
A Tile contains a 2D array of SB units. A Tile min. size is 1 SB unit. The Tile position is based on the top-left most corner SB unit of the Tile.
The very first tile of a frame (top-left most corner) has the position [Tile Column Position in SB Unit , Tile Row Position in SB Unit] = [0, 0]
The frame width is rounded up to an integer multiple of the Superblock unit.
A frame width that is less than a SB unit (but must be >= 8 pixels), is rounded up to 1 SB unit for the purpose of defining a tile.
A frame width that is not divisible by tile width, the last tile column of the frame will have a smaller tile width, but still an integer multiple of SB unit. It does not affect the Tile Column Position in SB Unit of these partial Tiles.
Tile Column Position in SB unit is derived from the frame level syntax elements in tiling specifications (tile_info()).
Max Frame Width= 64K, hence the max number of SB unit= 1024, when the SB unit is 64x64 pixels, and the max Column Row Position in SB unit can be 1023.
Intel supports Tile Column Position in SB unit only in the range of [0..255], for 16K Max Frame Width. starting from 0 at the top-left most corner of a frame.

AVP_TILE_CODING

		Value / **Name** / **Description**: [0,255] / 16K_Below / This allows support up to 16K picture.
3	31:26	**Reserved** Access: RO Format: MBZ
	25:16	**Tile Height in SuperBlock Unit Minus1** Format: U10 **Description**: Tile Size is measured in SuperBlock Unit. The SuperBlock unit is specified in the Sequence Header; it can be either 64x64 pixels or 128x128 pixels. When large-scale tile is ON, tile height must be 1 SB in size (i.e. can be 64x64 or 128x128, depending on the SB size flag). The minimum Tile Size is 1 SB unit x 1 SB unit The frame height is rounded up to an integer multiple of the Superblock unit. If a frame height is not divisible by the SuperBlock unit, the bottommost row of SBs of the frame is partial in size, but for the purpose of tile size definition, the partial SB is still counted as 1 unit. A frame height that is less than a SB unit, is rounded up to 1 SB unit for the purpose of defining a tile. A frame height that is not divisible by tile height, the last tile row of the frame will have a smaller tile height, but still an integer multiple of SB unit. Tile Height in SB unit is derived from the frame level syntax elements in tiling specifications (tile_info()). In AV1, there are two max tile size constraints : Max Tile Width <= 4096 pixels and Max Tile Area <= 4096 width x2304 height pixels. But there is no separate constraint for Max Tile Height. Intel set a limit for frame height to be 16K pixels. In AV1, the following restrictions apply: 1) Last SB (if partial in size)at frames bottommost edge must align to 8x8 block (partial SB) In AV1, the following additional restrictions apply: 1) In Scalability Mode, the minimum tile size is 2 width x1 height SBs. (Intel restriction) **Value** / **Name** / **Description**: [0,135] / 8704_and_Below / This supports upto 8704 (in pixels) for level 6.3.
	15:6	**Reserved** Access: RO Format: MBZ
	5:0	**Tile Width in SuperBlock Unit Minus1** Format: U6 **Description**: Tile Size is measured in SuperBlock Unit.The SuperBlock unit is specified in the Sequence

		Header; it can be either 64x64 pixels or 128x128 pixels. The minimum Tile Size is 1 SB unit x 1 SB unit The frame width is rounded up to an integer multiple of the Superblock unit. If a width height is not divisible by the SuperBlock unit, the rightmost column of SBs of the frame is partial in size, but for the purpose of tile size definition, the partial SB is still counted as 1 unit. A frame width that is less than a SB unit, is rounded up to 1 SB unit for the purpose of defining a tile. A frame width that is not divisible by tile width, the last tile column of the frame will have a smaller tile width, but still an integer multiple of SB unit. Tile Width in SB unit is derived from the frame level syntax elements in tiling specifications (tile_info()). In AV1, there are two max tile size constraints : Max Tile Width <= 4096 pixels and Max Tile Area <= 4096 width x2304 height pixels. But there is no separate constraint for Max Tile Height. When super-res is ON, these tile constraints are applied to the downscaled frame's tiles. So, Tile Width in SuperBlock Unit Minus1 is in the range of [0..63]. Tile Width = (Tile Width in SuperBlock Unit Minus1 + 1) SBs. In AV1, the following restrictions apply. 1) Last SB (if partial in size)at frames rightmost edge must align to 8x8 block (partial SB)
		In AV1, the following additional restrictions apply: 1) In Scalability Mode, the minimum tile size is 2 width x1 height SBs. (Intel restriction)
4	31	**Disable Frame Context Update Flag** Format: U1 **Description** Set to 0, the current tile being decoded is writing out its updated Frame Context to memory (surface buffer), at the end of its decoding. Set to 1, the current tile being decoded is not writing out its updated Frame Context to memory, for use in the next frame to be decoded. It is the frame level syntax element, disable_frame_end_update_cdf, or named as [! refresh_frame_context]. Default is 0. The arithmetic decoding of each tile of a frame is started with the same Frame Context (CDF Cumulative Probability Table Set for all AV1 Syntax Elements) provided by the Driver. This is from a read-only memory surface. After each arithmetic decoding of a Syntax Element from the bitstream,its corresponding CDF Table will be updated (if Disable CDF Update Flag = 0). As such, the frame context is changed after decoding each tile. When Backward Adaptation of the current frame context (CDF Cumulative Probability Table Set for all AV1 Syntax Elements) is enabled, only the largest tile in byte size (bitstream size) of the current frame being decoded will update the frame context for the decoding of the next frame. Driver will determine which tile in the current frame will need to write out its updated frame context to memory by setting Disable Frame Context Update Flag to 0. This is to a write-only memory surface. For all other tiles of the frame, Driver will set Disable Frame Context Update Flag to 1. If Driver knows, ahead of time, which tile has the largest bitstream size, then the frame context update can only be done once. If Driver does not know, ahead of time,which tile has the largest bitstream size, then the frame

		context update may be done more than once, and each time overwritten the previous one, until the largest tile has found. Note : Even when Error Resilient Mode is ON, this field can still be 0 or 1.
		When in Intel Scalability mode (multiple HW pipes), there are still be one read-only and one write-only frame context buffers allocated for decoding the current frame. And at anyone time, Driver will only enable one pipe to update/write out the frame context. Again it is possible that this update can be done more than once until the largest tile among all pipes has found.
	30	**Disable CDF Update Flag** Format: U1 Set to 1, the current tile being decoded is not updating the CDF table of each syntax element after parsing(multisymbol arithmetic decode) from the bitstream. Hence, the frame context is not being changed. And no need to write out the frame context at the end of decoding the frame. Set to 0, the current tile being decoded is updating the CDF table of each syntax element after parsing (multisymbol arithmetic decode) from the bitstream. Hence. the frame context is changed. If Disable Frame Context Update Flag = 0 for this tile, the new frame context is writing out to memory for subsequent frame decoding. It is the frame level syntax element, disable_cdf_update. Default is 0.
	29	**IsLastTileOfFrame Flag** Format: U1 Indicates if current tile being decoded is the last tile of a frame. Default is 0. The last tile is the bottom-right most corner region of a frame. This is intel specific frame level parameter.
	28	**IsEndTileOfTileGroup Flag** Format: U1 Indicates if current tile is the last tile of a Tile Group. Default is 0. 0 - is not the end tile of a tile group 1 - is the end tile of a tile group. It is derived as : isEndTileOfTileGroup = (Frame Tile ID = = TG_END syntax element) A Tile Group can end at any tile in a tile row (e.g. the first tile of a tile row, the last tile of a tile row, or a tile anywhere in the middle of a tile row). The End tile of a Tile Group may not be in the same tile row as the Start tile of the same Tile Group, but must come after in raster order. This is intel specific frame level parameter.
	27	**IsStartTileOfTileGroup Flag** Format: U1 Indicates if current tile is the first tile of a Tile Group. Default is 0. 0 - is not the start tile of a tile group 1 - is the start tile of a tile group. It is derived as : isStartTileOfTileGroup = (Frame Tile ID = = TG_START syntax element) A Tile Group can start at any tile in a tile row (e.g. the first tile of a tile row, the last tile of a tile

		row, or a tile anywhere in the middle of a tile row). The End tile of a Tile Group may not be in the same tile row as the Start tile of the same Tile Group. This is intel specific frame level parameter.
	26	**IsLastTileOfRow Flag** Format: U1 Indicates if the current tile is the last tile of the current tile row. Default is 0. 0 - is not the last tile of a tile row 1 - is the last tile of a tile row. It is derived from Frame Tile ID and Num of Tile Columns Minus1. It is the tile at the right frame boundary. The bottom-right most tile of a frame is having both IsLastTileOfRow and IsLastTileOfColumn set to 1. This is intel specific frame level parameter.
	25	**IsLastTileOfColumn Flag** Format: U1 Indicates if the current tile is the last tile of the current tile column. Default is 0. 0 - is not the last tile of a tile column 1 - is the last tile of a tile column. It is derived from Frame Tile ID, Num of Tile Columns Minus1and Num of Tile Rows Minus1. is the tile at the bottom frame boundary. The bottom-right most tile of a frame is having both IsLastTileOfRow and IsLastTileOfColumn set to 1. This is intel specific frame level parameter.
	24	**Reserved** Access: RO Format: MBZ
	23	**Reserved** Access: RO Format: MBZ
	22:4	**Reserved** Access: RO Format: MBZ
	3:0	**Reserved** Access: RO Format: MBZ
5	31:22	**Num of Tile Rows Minus1 in a Frame** Format: U10

Specify the total number of Tile Rows in a frame.
TileRows =Num of Tile Rows Minus1 in a Frame + 1.
Max Frame Height is 64K, the smallest tile size is 1SB unit, hence max number of tile rows in a frame can be 1024.
Valid Num of Tile Rows Minus1 in a Frame is in the range of [0..63] only. The max tile number in a frame is governed by the Level definition for a compliant bitstream.
For regular video, highest Level defined is 6.3 (max. 8K video), which specifies max total 128 tiles in a frame and max number of tile columns is 16. The corresponding tile configuration is flexible, it can be 16x8, 8x16, 16x4, 4x16, 8x8, etc. There is no constraints on max number of tile rows.
For VR large scale tile application, the tile configuration can be max total 64x64=4K tiles in a pseudo output frame and an anchor frame. But the tile list can max. contain 512 tiles at a time.
This is the same as the variable tile_rows (minus1)defined in the reference C model.
Num of Tile Rows in a frame is derived from the frame level syntax elements in tiling specifications (tile_info()).

21:12 **Num of Tile Columns Minus1 in a Frame**

Format:	U10

Description

Specify the total number of Tile Columns in a frame.
TileColumns=Num of Tile Columns Minus1 in a Frame + 1.
Max Frame Width is 64K, the smallest tile size is 1SB unit, hence max number of tile columns in a frame can be 1024.
Valid Num of Tile Columns Minus1 in a Frame is in the range of [0..63] only. The max tile number in a frame is governed by the Level definition for a compliant bitstream.
For regular video, highest Level defined is 6.3 (max. 8K video), which specifies max total 128 tiles in a frame and max number of tile columns is 16. The corresponding tile configuration is flexible, it can be 16x8, 8x16, 16x4, 4x16, 8x8, etc. There is no constraints on max number of tile rows.
For VR large scale tile application, the tile configuration can be max total 64x64=4K tiles in a pseudo output frame and an anchor frame. But the tile list can max. contain 512 tiles at a time.
This is the same as the variable tile_cols (minus1) defined in the reference C model.
Num of Tile Columns in a frame is derived from the frame level syntax elements in tiling specifications (tile_info()).

In Intel Scalability Mode (multiple HW pipes), execution across multiple pipes is done in tile column fashion. When the number of tile columns in a frame > the number of HW pipes, the tile columns are cycled through the pipes in multiple phases as described below:
Tile Col0/Row0 | Tile Col1/Row0 | Tile Col2/Row0
Tile Col0/Row1 | Tile Col1/Row1 | Tile Col2/Row1
Tile Col0/Row2 | Tile Col1/Row2 | Tile Col2/Row2
Assume there is only 2 HW pipes (Pipe0 and Pipe1)
Phase 0:
Pipe0 :Tile Col0/Row0 ; Pipe1 : Tile Col1/Row0
Phase 1:

		Pipe0 :Tile Col2/Row0 Phase 2: Pipe0 :Tile Col0/Row1 ; Pipe1 : Tile Col1/Row1 Phase 3: Pipe0 :Tile Col2/Row1 Phase 4: Pipe0 :Tile Col0/Row2 ; Pipe1 : Tile Col1/Row2 Phase 5: Pipe0 :Tile Col2/Row2.
	11:8	**Reserved MBZ** Format: MBZ
	7:0	**Number of Active BE Pipes** Indicates the number of active, consecutive positioned Scalable VDBOXs to be used for the current frame decoding or encoding. BE Pipe partitioning, SW must guarantee the minimum width is at least two full SBs for each tiles This field in general should be smaller or equal to Num of Tile columns in a Frame. This field is ignored by HW This field is not used by HW

Value	Comment
0	ignored
1	ignored
2	Supported by Encoder / Decoder.
3	Supported only by Decoder.
4	Supported only by Encoder

AVP_VD_CONTROL_STATE

AVP_VD_CONTROL_STATE

Source: VideoCS

Length Bias: 2

For AVP, it is selected with the **Media Instruction Opcode "3h".**

This command is used to modify the control of HCP pipe. It can be inserted anywhere within a frame. It can be inserted multiple times within a frame as well.

This command is also used for AVP pipe.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 3h Codec/Engine Name for AVP Format: OpCode Codec/EngineName = AVP = 3h
	22:16	**Media Instruction Command** Default Value: Ah VD_CONTROL_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 2h — Name:
1..2	63:0	**VD Control State Body** Format: VD_CONTROL_STATE_BODY

Barrier

MSD_BARRIER - Barrier

Source: EuSubFunctionGateway

Length Bias: 1

Record an additional thread reaching the barrier.

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of GRF registers sent as the message payload. Value 1 — Name: One **[Default]** — Description: See MDP_Barrier Barrier Data Payload definition.
	24:20	**Response Length** Format: U5 Specifies the number of GRF registers expected as the message response payload. Value 0 — Name: None **[Default]** — Description: Barrier completion notification is signaled with ARF N0.0. Value 1 — Name: One — Description: Ignore the data in the register. The update of the register is used as the barrier completion notification
	19:16	**Reserved** Access: RO Format: MBZ
	15:12	**Reserved** Access: RO Format: MBZ
	11:3	**Reserved** Access: RO Format: MBZ
	2:0	**Barrier Subfunction** Default Value: 0x4 Format: OpCode

Bit Field Extract

bfe - Bit Field Extract

Source:	Eulsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

Component-wise extract a bit field from src2 using the bit field width from src0 and the bit field offset from src1. Store the extracted bit field value in the low bits of dst and sign extend (if D type) or zero extend (if UD type). The width and offset values are from the low five bits of src0 and src1 respectively, or src0 & 0x1f and src1 & 0x1f. If width is zero, the result is zero. If offset + width > 32 then the extracted bit field is bits offset to 31 of src2, extracting only 32 - offset bits, less than width as the bit field cannot extend past the MSB of the source value. Otherwise extract width bits extending from bit positions offset to offset + width - 1.

Format:

```
[(pred)] bfe (exec_size) dst src0 src1 src2
```

Restriction

No accumulator access, implicit or explicit.

All three-source instructions have certain restrictions, described in Instruction Formats.

Syntax

```
[(pred)] bfe (exec_size) reg reg reg reg
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         UD width = src0.chan[n][4:0];
         UD offset = src1.chan[n][4:0];
         if ( width == 0 ) {
             dst.chan[n] = 0x00000000;
         } else if ( (width + offset) < 32 ) {
             dst.chan[n] = src2.chan[n] « (32 - width - offset);
             if (src2 is signed) {
                 dst.chan[n] = dst.chan[n] » (32 - width); // pad sign bit of dst.chan
             } else {
                 dst.chan[n] = dst.chan[n] » (32 - width); // pad 0
             }
         } else {
             if ( src2 is signed ) {
                 dst.chan[n] = src2.chan[n] » offset; // pad sign bit
             } else {
                 dst.chan[n] = src2.chan[n] » offset; // pad 0
             }
         }
     }
 }
```

bfe - Bit Field Extract

Src Types	Dst Types
UD	UD
D	D

DWord	Bit	Description
0..3	127:114	**Src2.Operand** Exists If: ([Src2.IsImm]==false) AND ([Header][Opcode]!=madm) Format: **DirectOperand**
	127:114	**Src2.Operand** Exists If: ([Src2.IsImm]==false) AND ([Header][Opcode]==madm) Format: **MacroOperand**
	127:112	**Src2.ImmValue[15:0]** Exists If: ([Src2.IsImm]==true)
	113:112	**Src2.HorzStride** Exists If: ([Src2.IsImm]==false) Format: **HorzStride**
	111:98	**Src1.Operand** Exists If: ([Header][Opcode]!=madm) Format: **DirectOperand**
	111:98	**Src1.Operand** Exists If: ([Header][Opcode]==madm) Format: **MacroOperand**
	97:96	**Src1.HorzStride** Format: **HorzStride**
	95:92	**CondCtrl** Format: **FlagModifier**
	91	**Src1.VertStride[1]** Format: **TernaryVertStride[1:1]**
	90:88	**Src1.DataType** Format: **TernaryDataType**
	87:86	**Src1.Mod** Format: **SrcMod**
	85:84	**Src2.Mod** Format: **SrcMod**

bfe - Bit Field Extract

	83	**Src1.VertStride[0]** Format: **TernaryVertStride[0:0]**
	82:80	**Src2.DataType** Format: **TernaryDataType**
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) AND ([Header][Opcode]!=madm) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) AND ([Header][Opcode]==madm) Format: **MacroOperand**
	79:64	**Src0.ImmValue[15:0]** Exists If: ([Src0.IsImm]==true)
	65:64	**Src0.HorzStride** Exists If: ([Src0.IsImm]==false) Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Header][Opcode]!=madm) Format: **DirectOperand** **Programming Notes** The Dst.Operand must be 64 bit aligned. i.e. Dst.Operand.SubRegNum[2:0] must be zero,
	63:50	**Dst.Operand** Exists If: ([Header][Opcode]==madm) Format: **MacroOperand**
	49	**Reserved** Format: MBZ
	48	**Dst.HorzStride** This field provides the distance in unit of data elements between two adjacent data elements within a row (horizontal) in the register region for the operand. **Value** / **Name** 0 / 1 element 1 / 2 element
	47	**Src2.IsImm** This field indicate that Source 2 operand is carrying an immediate value. **Value** / **Name** 0 / false 1 / true

bfe - Bit Field Extract

	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. **Value** / **Name** 0 / false 1 / true
	45:44	**Src0.Mod** Format: SrcMod
	43	**Src0.VertStride[1]** Format: TernaryVertStride[1:1]
	42:40	**Src0.DataType** Format: TernaryDataType
	39	**ExecDataType** This field indicate the datatype mode of ternary instruction. Integer or Float. **Value** / **Name** 0 / Integer 1 / Float
	38:36	**Dst.DataType** Format: TernaryDataType
	35	**Src0.VertStride[0]** Format: TernaryVertStride[0:0]
	34	**Saturate** Format: Saturate
	33	**AccWrCtrl** Format: AccWrCtrl
	32	**AtomicCtrl** Format: AtomicCtrl
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". **Value** / **Name** / **Description** 0 / Normal **[Default]** / Normal. Per channel write enable used for final write enable generation. 1 / NoMask / NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	30	**Reserved**
	29	**CmptCtrl** Format: MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. This can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

bfe - Bit Field Extract		
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Bit Field Insert 1

bfi1 - Bit Field Insert 1

Source:	Eulsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

The bfi1 instruction is the first instruction in a two-instruction macro for bfi (Bit Field Insert). The bfi1 instruction component-wise generates mask with control from src0 and src1 and stores the results in dst. The mask is used in the bfi2 instruction to generate the final result of bfi. Create a bit mask corresponding to the bit field width and offset in src0 and src1. Store the bit mask in dst. The mask has all bits in the bit field set to 1 and all other bits as 0. The width and offset values are from the low five bits of src0 and src1 respectively, or src0 & 0x1f and src1 & 0x1f. If width is zero, the result is zero. The bfi macro has four source operands: src0 - bit field width in low five bits, src1 - bit field offset/starting bit position in low five bits, src2 - bit field value to insert, using only the number of least significant bits given by width in src0, and src3 - overall value into which the bit field is inserted, providing all bits other than the inserted bits for the result value. bfi dst src0 src1 src2 src3 // Translates to these two instructions: bfi1 dst src0 src1 bfi2 dst dst src2 src3

Format:

```
[(pred)] bfi1 (exec_size) dst src0 src1
```

Programming Notes

No accumulator access, implicit or explicit.

Syntax

```
[(pred)] bfi1 (exec_size) reg reg reg
 [(pred)] bfi1 (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
     for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
          UD width = src0.chan[n][4:0];
          UD offset = src1.chan[n][4:0];
          dst = ((1 « width) - 1) « offset;
     }
 }
```

Src Types	Dst Types
UD	UD
D	D

DWord	Bit	Description
0..3	127:126	**Reserved**

bfi1 - Bit Field Insert 1

		Exists If:	([Src1.IsImm]==false)
		Format:	MBZ
	127:96	**Src1.ImmValue[31:0]**	
		Exists If:	([Src1.IsImm]==true)
	125:122	**Reserved**	
		Exists If:	([Src1.IsImm]==false)
		Format:	MBZ
	121:120	**Src1.Mod**	
		Exists If:	([Src1.IsImm]==false)
		Format:	**SrcMod**
	119:116	**Src1.VertStride**	
		Exists If:	([Src1.IsImm]==false)
		Format:	**VertStride**
	115:113	**Src1.Width**	
		Exists If:	([Src1.IsImm]==false)
		Format:	**Width**
	112	**Src1.AddrMode**	
		Exists If:	([Src1.IsImm]==false)
		Format:	**AddrMode**
	111:98	**Src1.Operand**	
		Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)
		Format:	**IndirectOperand**
	111:98	**Src1.Operand**	
		Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)
		Format:	**DirectOperand**
	97:96	**Src1.HorzStride**	
		Exists If:	([Src1.IsImm]==false)
		Format:	**HorzStride**
	95:92	**CondCtrl**	
		Format:	**FlagModifier**
	91:88	**Src1.DataType**	
		Exists If:	([Src1.IsImm]==true)
		Format:	**ImmDataType**
	91:88	**Src1.DataType**	
		Exists If:	([Src1.IsImm]==false)
		Format:	**RegDataType**

bfi1 - Bit Field Insert 1

	87:84	**Src0.VertStride** Format: **VertStride**
	83:81	**Src0.Width** Format: **Width**
	80	**Src0.AddrMode** Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value. **Value** / **Name** 0 – false **[Default]** 1 – true
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. **Value** / **Name** 0 – false **[Default]** 1 – true
	45:44	**Src0.Mod** Format: **SrcMod**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false)

bfi1 - Bit Field Insert 1

	Format:	**RegDataType**
43:40	**Src0.DataType**	
	Exists If:	([Src0.IsImm]==true)
	Format:	**ImmDataType**
39:36	**Dst.DataType**	
	Format:	**RegDataType**
35	**Dst.AddrMode**	
	Format:	**AddrMode**
34	**Saturate**	
	Format:	**Saturate**
33	**AccWrCtrl**	
	Format:	**AccWrCtrl**
32	**AtomicCtrl**	
	Format:	**AtomicCtrl**

31 MaskCtrl

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 Reserved

29 CmptCtrl

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 PredInv

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits

bfi1 - Bit Field Insert 1

generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16 **ExecSize**

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

15:0 **Header**

Format:	Header

Bit Field Insert 2

bfi2 - Bit Field Insert 2

Source:	Eulsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

The bfi2 instruction is the second instruction in a two-instruction macro for bfi (Bit Field Insert). The bfi2 instruction component-wise performs the bitfield insert operation on src1 and src2 based on the mask in src0. Use the mask in src0 to take a bit field value from the low bits of src1 and combine it with the value from src2 (so src2 provides all bits other than those masked out and replaced by the bit field value). Store the result in dst. The bfi macro has four source operands: src0 - bit field width in low five bits, src1 - bit field offset/starting bit position in low five bits, src2 - bit field value to insert, using only the number of least significant bits given by width in src0, and src3 - overall value into which the bit field is inserted, providing all bits other than the inserted bits for the result value. bfi dst src0 src1 src2 src3 // Translates to these two instructions: bfi1 dst src0 src1 bfi2 dst dst src2 src3

Format:

```
[(pred)] bfi2 (exec_size) dst src0 src1 src2
```

Restriction

No accumulator access, implicit or explicit.

All three-source instructions have certain restrictions, described in Instruction Formats.

Syntax

```
[(pred)] bfi2 (exec_size) reg reg reg reg
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         UD offset = LZD(reverse(src0.chan[n]))-1;
         // offset is the number of LSB zero bits below the bit mask which has all 1s.
         // width (implied by the logic) is the number of 1 bits in the mask value, which
should be all 1s.
         dst.chan[n] = ((src1.chan[n] « offset) & src0.chan[n]) | (src2.chan[n] & !
src0.chan[n]);
     }
 }
```

Src Types	Dst Types
UD	UD
D	D

DWord	Bit	Description
0..3	127:114	**Src2.Operand** Exists If: ([Src2.IsImm]==false) AND ([Header][Opcode]!=madm) Format: **DirectOperand**
	127:114	**Src2.Operand** Exists If: ([Src2.IsImm]==false) AND ([Header][Opcode]==madm) Format: **MacroOperand**
	127:112	**Src2.ImmValue[15:0]** Exists If: ([Src2.IsImm]==true)
	113:112	**Src2.HorzStride** Exists If: ([Src2.IsImm]==false) Format: **HorzStride**
	111:98	**Src1.Operand** Exists If: ([Header][Opcode]!=madm) Format: **DirectOperand**
	111:98	**Src1.Operand** Exists If: ([Header][Opcode]==madm) Format: **MacroOperand**
	97:96	**Src1.HorzStride** Format: **HorzStride**
	95:92	**CondCtrl** Format: **FlagModifier**
	91	**Src1.VertStride[1]** Format: **TernaryVertStride[1:1]**
	90:88	**Src1.DataType** Format: **TernaryDataType**
	87:86	**Src1.Mod** Format: **SrcMod**
	85:84	**Src2.Mod** Format: **SrcMod**
	83	**Src1.VertStride[0]** Format: **TernaryVertStride[0:0]**
	82:80	**Src2.DataType** Format: **TernaryDataType**
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) AND ([Header][Opcode]!=madm) Format: **DirectOperand**

bfi2 - Bit Field Insert 2

	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) AND ([Header][Opcode]==madm) Format: **MacroOperand**
	79:64	**Src0.ImmValue[15:0]** Exists If: ([Src0.IsImm]==true)
	65:64	**Src0.HorzStride** Exists If: ([Src0.IsImm]==false) Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Header][Opcode]!=madm) Format: **DirectOperand** **Programming Notes** The Dst.Operand must be 64 bit aligned. i.e. Dst.Operand.SubRegNum[2:0] must be zero,
	63:50	**Dst.Operand** Exists If: ([Header][Opcode]==madm) Format: **MacroOperand**
	49	**Reserved** Format: MBZ
	48	**Dst.HorzStride** This field provides the distance in unit of data elements between two adjacent data elements within a row (horizontal) in the register region for the operand. Value: 0 — Name: 1 element Value: 1 — Name: 2 element
	47	**Src2.IsImm** This field indicate that Source 2 operand is carrying an immediate value. Value: 0 — Name: false Value: 1 — Name: true
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — Name: false Value: 1 — Name: true
	45:44	**Src0.Mod** Format: **SrcMod**

bfi2 - Bit Field Insert 2

	43	**Src0.VertStride[1]** Format: **TernaryVertStride[1:1]**
	42:40	**Src0.DataType** Format: **TernaryDataType**
	39	**ExecDataType** This field indicate the datatype mode of ternary instruction. Integer or Float. **Value** / **Name** 0 / Integer 1 / Float
	38:36	**Dst.DataType** Format: **TernaryDataType**
	35	**Src0.VertStride[0]** Format: **TernaryVertStride[0:0]**
	34	**Saturate** Format: **Saturate**
	33	**AccWrCtrl** Format: **AccWrCtrl**
	32	**AtomicCtrl** Format: **AtomicCtrl**
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". **Value** / **Name** / **Description** 0 / Normal **[Default]** / Normal. Per channel write enable used for final write enable generation. 1 / NoMask / NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	30	**Reserved**
	29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. **Value** / **Name** / **Description** 0 / NoCompaction **[Default]** / No compaction. 128-bit native instruction supporting all instruction options.

1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24	**PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
21:19	**ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16	**ExecSize**

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

15:0	**Header**

Format:	Header

Bit Field Reverse

bfrev - Bit Field Reverse

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

The bfrev instruction component-wise reverses all the bits in src0 and stores the results in dst.

Format:

```
[(pred)] bfrev (exec_size) dst src0
```

Restriction

No accumulator access, implicit or explicit.

Syntax

```
[(pred)] bfrev (exec_size) reg reg
 [(pred)] bfrev (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         for ( idx = 0; idx < 32; idx++ ) {
             dst.chan[n][idx] = src0.chan[n][31-idx];
         }
     }
 }
```

Src Types	Dst Types
UD	UD

DWord	Bit	Description
0..3	127:96	**Src0.ImmValue[31:0]** Exists If: ([Src0.IsImm]==true)
	95:92	**CondCtrl** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **FlagModifier**
	95:64	**Src0.ImmValue[63:32]** Exists If: ([Src0.IsImm]==true) AND (([Src0.DataType]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df))

bfrev - Bit Field Reverse

	87:84	**Src0.VertStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **VertStride**
	83:81	**Src0.Width** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **Width**
	80	**Src0.AddrMode** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Reserved** Access: RO Format: MBZ
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value.

bfrev - Bit Field Reverse

	Value	Name
	0	false **[Default]**
	1	true

45:44 Src0.Mod

Format:	SrcMod

43:40 Src0.DataType

Exists If:	([Src0.IsImm]==false)
Format:	RegDataType

43:40 Src0.DataType

Exists If:	([Src0.IsImm]==true)
Format:	ImmDataType

39:36 Dst.DataType

Format:	RegDataType

35 Dst.AddrMode

Format:	AddrMode

34 Saturate

Format:	Saturate

33 AccWrCtrl

Format:	AccWrCtrl

32 AtomicCtrl

Format:	AtomicCtrl

31 MaskCtrl

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 Reserved

29 CmptCtrl

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16 **ExecSize**

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

bfrev - Bit Field Reverse		
	15:0	**Header** Format: **Header**

Branch Converging

brc - Branch Converging

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

The brc instruction redirects the execution forward or backward to the instruction pointed by (current IP + offset). The jump will occur if all channels are branched away. UIP should reference the instruction where all channels are expected to come together. JIP should reference the end of the innermost conditional block.

In instruction binary, JIP and UIP use locations src1 and src0 respectively when immediate and location src0 when reg64, where reg64 is accessed as paired DWord (regioning being <2;2,1>). dst must be IP. When the offsets are immediate, src0 regfile must be immediate.

Format:

```
[(pred)] brc (exec_size) JIP UIP
```

Restriction

A brc instruction cannot use the Switch instruction option.

Syntax

```
[(pred)] brc (exec_size) imm32 imm32
 [(pred)] brc (exec_size) reg64
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < 32; n++ ) {
    if ( WrEn[n] ) {
       PcIP[n] = IP + UIP;
    } else {
       PcIP[n] = IP + 1;
    }
 }
 if ( all PcIP != IP + 1 ) {  // for all channels
    Jump(IP + JIP);
 }
```

DWord	Bit	Description
0..3	127:96	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	127:96	**JIP**

brc - Branch Converging

		Exists If: ([Src0.IsImm]==true) Format: S31 The byte-aligned jump distance if a jump is taken for the channel.
	95:80	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	95:64	**Reserved** Exists If: ([Src0.IsImm]==true) AND ([Src1.IsImm]==false) Format: MBZ
	95:64	**UIP** Exists If: ([Src0.IsImm]==true) AND ([Src1.IsImm]==true) Format: S31 The byte aligned jump distance if a jump is taken for the instruction.
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) Format: **DirectOperand**
	65:64	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	63:50	**Dst.Operand** Format: **DirectOperand**
	49:48	**Reserved** Access: RO Format: MBZ
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value Value: 0 — false Value: 1 — true
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value Value: 0 — false Value: 1 — true
	45:34	**Reserved**

	Access:	RO
	Format:	MBZ

33 BranchCtrl

This field is used by *goto*, *if*, and *else* instructions to control branching. See the goto instruction description for more information about BranchCtrl.

32 AtomicCtrl

Format:	AtomicCtrl

31 MaskCtrl

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 Reserved

29 CmptCtrl

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 PredInv

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

brc - Branch Converging

	27:24	**PredCtrl** Format: **PredCtrl** This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.
	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Branch Diverging

brd - Branch Diverging

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

The brd instruction redirects the execution forward or backward to the instruction pointed by (current IP + offset). The jump will occur if any channels are branched away.

In instruction binary, JIP is at location src1 when immediate and at location src0 when reg32, where reg32 is accessed as a scalar DWord. The ip register must be used (for example, by the assembler) as dst.

Format:

```
[(pred)] brd (exec_size) JIP
```

Restriction

A brd instruction cannot use the Switch instruction option.

Syntax

```
[(pred)] brd (exec_size) imm32
 [(pred)] brd (exec_size) reg32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < 32; n++ ) {
     if ( WrEn[n] ) {
         PcIP[n] = IP + JIP;
     } else {
         PcIP[n] = IP + 1;
     }
 }
 if ( any PcIP == ExIP + JIP ) {  // any channel
     Jump(ExIP + JIP);
 }
```

DWord	Bit	Description
0..3	127:96	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	127:96	**JIP** Exists If: ([Src0.IsImm]==true) Format: S31

brd - Branch Diverging

		The byte-aligned jump distance if a jump is taken for the channel
	95:80	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	95:64	**Reserved** Exists If: ([Src0.IsImm]==true) Format: MBZ
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) Format: **DirectOperand**
	65:64	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	63:50	**Dst.Operand** Format: **DirectOperand**
	49:47	**Reserved** Access: RO Format: MBZ
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. **Value** / **Name** 0 – false 1 – true
	45:34	**Reserved** Access: RO Format: MBZ
	33	**BranchCtrl** This field is used by *goto*, *if*, and *else* instructions to control branching. See the **goto** instruction description for more information about BranchCtrl.
	32	**AtomicCtrl** Format: **AtomicCtrl**
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". **Value** / **Name** / **Description** 0 – Normal **[Default]** – Normal. Per channel write enable used for final write enable generation.

1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 **Reserved**

29 **CmptCtrl**

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if

brd - Branch Diverging

		both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Break

break - Break

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

The break instruction is used to early-out from the inner most loop, or early out from the inner most switch block. When used in a loop, upon execution, the break instruction terminates the loop for all execution channels enabled. If all the enabled channels hit the break instruction, jump to the instruction referenced by JIP. JIP should be the offset to the end of the inner most conditional or loop block, UIP should be the offset to the while instruction of the loop block. If SPF is ON, the UIP must be used to update IP; JIP is not used in this case

The following table describes the two 32-bit instruction pointer offsets. Both the JIP and UIP are signed 32-bit numbers, added to IP pre-increment. In instruction binary, JIP and UIP are at locations src0 and src1 and must be of type DW (signed DWord integer). When the offsets are immediate, src0 regfile must be immediate.

Format:

```
[(pred)] break (exec_size) JIP UIP
```

Syntax

```
[(pred)] break (exec_size) imm32 imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.channel[n] ) {
         PcIP[n] = IP + UIP;
     } else {
         PcIP[n] = IP + 1;
     }
 }
 if ( PcIP != (IP + 1) ) { // all channels
     Jump(IP + JIP);
 }
```

DWord	Bit	Description
0..3	127:96	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	127:96	**JIP** Exists If: ([Src0.IsImm]==true) Format: S31

break - Break

		The byte-aligned jump distance if a jump is taken for the channel.
	95:80	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	95:64	**Reserved** Exists If: ([Src0.IsImm]==true) AND ([Src1.IsImm]==false) Format: MBZ
	95:64	**UIP** Exists If: ([Src0.IsImm]==true) AND ([Src1.IsImm]==true) Format: S31 The byte aligned jump distance if a jump is taken for the instruction.
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) Format: **DirectOperand**
	65:64	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	63:50	**Dst.Operand** Format: **DirectOperand**
	49:48	**Reserved** Access: RO Format: MBZ
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value Value: 0 — false Value: 1 — true
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value Value: 0 — false Value: 1 — true
	45:34	**Reserved** Access: RO Format: MBZ

break - Break

33 BranchCtrl

This field is used by *goto*, *if*, and *else* instructions to control branching. See the goto instruction description for more information about BranchCtrl.

32 AtomicCtrl

Format:	AtomicCtrl

31 MaskCtrl

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 Reserved

29 CmptCtrl

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 PredInv

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 PredCtrl

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask

<table>
<tr><th colspan="3">break - Break</th></tr>
<tr><td></td><td></td><td>for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.</td></tr>
<tr><td></td><td>23</td><td>FlagRegNum[0]
This field specifies bit[0] of the register number for a flag register operand.</td></tr>
<tr><td></td><td>22</td><td>FlagSubRegNum
This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.</td></tr>
<tr><td></td><td>21:19</td><td>ChanOff
<table><tr><td>Format:</td><td>ChanOff</td></tr></table>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</td></tr>
<tr><td></td><td>18:16</td><td>ExecSize
<table><tr><td>Format:</td><td>ExecSize</td></tr></table>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</td></tr>
<tr><td></td><td>15:0</td><td>Header
<table><tr><td>Format:</td><td>Header</td></tr></table></td></tr>
</table>

Byte Scattered Read MSD

MSD0R_BS - Byte Scattered Read MSD

Source: EuSubFunctionDataPort0

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: Enable If set, indicates that the message includes the header.

MSD0R_BS - Byte Scattered Read MSD

	18	**Legacy Message** Default Value: 0h Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 04h Format: Opcode Byte Scattered Read message
	13	**Reserved** Access: RO Format: MBZ
	12	**Reserved** Access: RO Format: MBZ
	11:10	**Data Elements** Format: MDC_DS Specifies the number of Bytes to be read or written per Dword
	9	**Reserved** Access: RO Format: MBZ
	8	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Byte Scattered Write MSD

MSD0W_BS - Byte Scattered Write MSD

Source: EuSubFunctionDataPort0

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: Enable If set, indicates that the message includes the header.

MSD0W_BS - Byte Scattered Write MSD

	18	**Legacy Message** Default Value: 0h Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 0Ch Format: Opcode Byte Scattered Write message
	13:12	**Reserved** Access: RO Format: MBZ
	11:10	**Data Elements** Format: MDC_DS Specifies the number of Bytes to be read or written per Dword
	9	**Reserved** Access: RO Format: MBZ
	8	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Call

call - Call

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

The call instruction jumps to a subroutine. It can be predicated or non-predicated. If non-predicated, all enabled channels jump to the subroutine. If predicated, only the channels enabled by PMask jump to the subroutine; the rest of the channels move to the next instruction after the call instruction. If none of the channels jump into the subroutine, the call instruction is treated as a nop. In case of a jump, the call instruction stores the return IP onto the first DWord of the destination register and stores the CallMask in the second DWord of the destination register. When SPF is on, the predication control must be scalar.

The following section describes JIP, the jump offset.
JIP can be an immediate or register value. When a jump occurs, this value is added to IP pre-increment. In instruction binary, JIP is at location src1 and src0 must be null. The GRF register must be put (for example, by the assembler) at dst location.
Format: [(pred)] call (exec_size) dst JIP

Format:

```
[(pred)] call (exec_size) dst JIP
```

Restriction

The call instruction must have DWord source and destination type, and the destination must be QWord aligned.

A call instruction must target an instruction with Switch set; in addition, the instruction following the call must also have Switch set.

When EU Fusion is enabled JIP of both EU's must be same.

Syntax

```
[(pred)] call (exec_size) reg imm32 [(pred)] call (exec_size) reg reg32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if (WrEn.chan[n] ) {
         PcIP[n] = IP + JIP;
         CallMask[n] = 1;
     } else {
         PcIP[n] = IP + 1;
         CallMask[n] = 0;
     }
 }
```

call - Call

```
if ( PcIP[n] != (IP + 1) ) {  // any channel jumped
    dst.chan[0] = IP + 1;
    dst.chan[1] = CallMask;
    Jump(IP + JIP);
}
```

DWord	Bit	Description
0..3	127:96	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	127:96	**JIP** Exists If: ([Src0.IsImm]==true) Format: S31 The byte-aligned jump distance if a jump is taken for the channel
	95:80	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	95:64	**Reserved** Exists If: ([Src0.IsImm]==true) Format: MBZ
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) Format: **DirectOperand**
	65:64	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	63:50	**Dst.Operand** Format: **DirectOperand**
	49:47	**Reserved** Access: RO Format: MBZ
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. 0 – false 1 – true

45:34 **Reserved**

Access:	RO
Format:	MBZ

33 **BranchCtrl**

This field is used by *goto*, *if*, and *else* instructions to control branching. See the **goto** instruction description for more information about BranchCtrl.

32 **AtomicCtrl**

Format:	**AtomicCtrl**

31 **MaskCtrl**

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 **Reserved**

29 **CmptCtrl**

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

call - Call

	27:24	**PredCtrl** Format: **PredCtrl** This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.
	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Call Absolute

calla - Call Absolute

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

The calla instruction jumps to a subroutine. It can be predicated or non-predicated. If non-predicated, all enabled channels jump to the subroutine. If predicated, only the channels enabled by PMask jump to the subroutine; the rest of the channels move to the next instruction after the calla instruction. If none of the channels jump into the subroutine, the calla instruction is treated as a nop. In case of a jump, the call instruction stores the return IP onto the first DWord of the destination register and stores the CallMask in the second DWord of the destination register. If SPF is ON, none of the PcIP are updated. When SPF is on, the predication control must be scalar. The difference between calla and call is that calla uses JIP as the IP value rather than adding it to the IP value.

Format:

```
[(pred)] calla (exec_size) dst JIP
```

Restriction

The calla instruction must have DWord source and destination type, and the destination must be QWord-aligned.

When EU Fusion is enabled JIP of both EU's must be same.

Syntax

```
[(pred)] calla (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.channel[n] ) {
         PcIP[n] = JIP;
         CallMask[n] = 1;
     } else {
         PcIP[n] = IP + 1;
         CallMask[n] = 0;
     }
 }
 if ( PcIP[n] != (IP + 1) ) {  // any channel jumped
     dst.chan[0] = IP + 1;
     dst.chan[1] = CallMask;
     Jump(JIP);
 }
```

DWord	Bit	Description

calla - Call Absolute

0..3	127:96	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	127:96	**JIP** Exists If: ([Src0.IsImm]==true) Format: S31 The byte-aligned jump distance if a jump is taken for the channel
	95:80	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	95:64	**Reserved** Exists If: ([Src0.IsImm]==true) Format: MBZ
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) Format: **DirectOperand**
	65:64	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	63:50	**Dst.Operand** Format: **DirectOperand**
	49:47	**Reserved** Access: RO Format: MBZ
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. 0 – false 1 – true
	45:34	**Reserved** Access: RO Format: MBZ
	33	**BranchCtrl** This field is used by *goto*, *if*, and *else* instructions to control branching. See the **goto** instruction description for more information about BranchCtrl.

calla - Call Absolute

	32	**AtomicCtrl** Format: **AtomicCtrl**
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". **Value** / **Name** / **Description** 0 / Normal **[Default]** / Normal. Per channel write enable used for final write enable generation. 1 / NoMask / NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	30	**Reserved**
	29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. **Value** / **Name** / **Description** 0 / NoCompaction **[Default]** / No compaction. 128-bit native instruction supporting all instruction options. 1 / Compacted / Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
	28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields **Value** / **Name** / **Description** 0 / Positive **[Default]** / Positive polarity of predication. Use the predication mask produced by PredCtrl. 1 / Negative / Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
	27:24	**PredCtrl** Format: **PredCtrl** This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

calla - Call Absolute

	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Compare

cmp - Compare

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	false
Source Modifier:	true

The cmp instruction performs component-wise comparison of src0 and src1 and stores the results in the selected flag register and in dst. It takes component-wise subtraction of src0 and src1, evaluating the conditional code (excluding NS signal) based on the conditional modifier, and storing the conditional bits in bit-packed form in the destination flag register and all bits of dst channels. If the dst is not null, for the enabled channels, then all bits of the destination channel will contain the flag value for the channel. When the instruction operates on packed word format, one general register may store up to 16 such comparison results. In DWord format, one general register may store up to 8 results. A conditional modifier must be specified; the conditional modifier field cannot be 0000b. The comparison does not use the NS (NaN source) signals, as described in the Creating Conditional Flags section. Accordingly the conditional modifier should not be .u (unordered). For each enabled channel 0b or 1b is assigned to the appropriate flag bit and 0/all zeros or all ones (e.g, byte 0xFF, word 0xFFFF, DWord 0xFFFFFFFF) is assigned to dst. When any source type is floating-point, the cmp instruction obeys the rules described in the tables in the Floating Point Modes section of the Data Types chapter.

Format:

```
[(pred)] cmp[.cmod] (exec_size) dst src0 src1
```

Syntax

```
[(pred)] cmp[.cmod] (exec_size) reg reg reg
 [(pred)] cmp[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         results[n] = src0.chan[n] - src1.chan[n];
         bitMask[n] = Condition(results[n]); // details in Assigning Conditional Flags
dst.chan[n] = bitMask[n]; // All bits for dst channel
         flag#.bit[n]= bitMask[n];
     }
 }
```

Src Types	Dst Types
*B,*W,*D	*B,*W,*D
F	F
HF	HF

cmp - Compare

HF, F	HF, F

DWord	Bit	Description
0..3	127:126	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	127:96	**Src1.ImmValue[31:0]** Exists If: ([Src1.IsImm]==true)
	125:122	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	121:120	**Src1.Mod** Exists If: ([Src1.IsImm]==false) Format: **SrcMod**
	119:116	**Src1.VertStride** Exists If: ([Src1.IsImm]==false) Format: **VertStride**
	115:113	**Src1.Width** Exists If: ([Src1.IsImm]==false) Format: **Width**
	112	**Src1.AddrMode** Exists If: ([Src1.IsImm]==false) Format: **AddrMode**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect) Format: **IndirectOperand**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct) Format: **DirectOperand**
	97:96	**Src1.HorzStride** Exists If: ([Src1.IsImm]==false) Format: **HorzStride**
	95:92	**CondCtrl** Format: **FlagModifier**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==true) Format: **ImmDataType**

cmp - Compare

	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==false) Format: **RegDataType**
	87:84	**Src0.VertStride** Format: **VertStride**
	83:81	**Src0.Width** Format: **Width**
	80	**Src0.AddrMode** Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value. **Value** / **Name** 0 – false **[Default]** 1 – true
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. **Value** / **Name** 0 – false **[Default]** 1 – true
	45:44	**Src0.Mod**

	Format:	SrcMod
43:40	**Src0.DataType**	
	Exists If:	([Src0.IsImm]==false)
	Format:	RegDataType
43:40	**Src0.DataType**	
	Exists If:	([Src0.IsImm]==true)
	Format:	ImmDataType
39:36	**Dst.DataType**	
	Format:	RegDataType
35	**Dst.AddrMode**	
	Format:	AddrMode
34	**Saturate**	
	Format:	Saturate
33	**AccWrCtrl**	
	Format:	AccWrCtrl
32	**AtomicCtrl**	
	Format:	AtomicCtrl

31 **MaskCtrl**

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 **Reserved**

29 **CmptCtrl**

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.

1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16 **ExecSize**

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

15:0 **Header**

Format:	Header

Compare NaN

cmpn - Compare NaN

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	false
Source Modifier:	true

The cmpn instruction performs component-wise special-NaN comparison of src0 and src1 and stores the results in the selected flag register and in dst. It takes component-wise subtraction of src0 and src1, evaluating the conditional signals including NS based on the conditional modifier, and storing the conditional flag bits in bit-packed form in the destination flag register and all bits of dst channels. If the dst is not null, for the enabled channels, then all bits of the destination channel will contain the flag value for the channel. When the instruction operates on packed word format, one general register may store up to 16 such comparison results. In DWord format, one general register may store up to 8 results. A conditional modifier must be specified; the conditional modifier field cannot be 0000b. More information about the conditional signals used is in the Creating Conditional Flags section. For each enabled channel 0b or 1b is assigned to the appropriate flag bit and 0/all zeros or all ones (e.g, byte 0xFF, word 0xFFFF, DWord 0xFFFFFFFF) is assigned to dst. Min/Max instructions use cmpn to select the destination from the input sources (see the Min Max of Floating Point Numbers section for details).

Format:

```
[(pred)] cmpn[.cmod] (exec_size) dst src0 src1
```

Restriction

.l and .ge are the only two conditional modifiers are supported for this instruction.

Syntax

```
[(pred)] cmpn[.cmod] (exec_size) reg reg reg
 [(pred)] cmpn[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         results[n] = src0.chan[n] - src1.chan[n];
         bitMask[n] = ConditionNaN(results[n]); // details in Assigning Conditional Flags
dst.chan[n][0] = bitMask[n]; // All bits for dst channel
         flag#.bit[n] = bitMask[n];
     }
 }
```

Src Types	Dst Types
*B,*W,*D	*B,*W,*D

cmpn - Compare NaN

F	F
HF	HF

DWord	Bit	Description
0..3	127:126	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	127:96	**Src1.ImmValue[31:0]** Exists If: ([Src1.IsImm]==true)
	125:122	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	121:120	**Src1.Mod** Exists If: ([Src1.IsImm]==false) Format: **SrcMod**
	119:116	**Src1.VertStride** Exists If: ([Src1.IsImm]==false) Format: **VertStride**
	115:113	**Src1.Width** Exists If: ([Src1.IsImm]==false) Format: **Width**
	112	**Src1.AddrMode** Exists If: ([Src1.IsImm]==false) Format: **AddrMode**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect) Format: **IndirectOperand**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct) Format: **DirectOperand**
	97:96	**Src1.HorzStride** Exists If: ([Src1.IsImm]==false) Format: **HorzStride**
	95:92	**CondCtrl** Format: **FlagModifier**

cmpn - Compare NaN

	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==true) Format: **ImmDataType**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==false) Format: **RegDataType**
	87:84	**Src0.VertStride** Format: **VertStride**
	83:81	**Src0.Width** Format: **Width**
	80	**Src0.AddrMode** Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value.

		Value / **Name** 0 / false **[Default]** 1 / true
	45:44	**Src0.Mod** Format: **SrcMod**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: **RegDataType**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: **ImmDataType**
	39:36	**Dst.DataType** Format: **RegDataType**
	35	**Dst.AddrMode** Format: **AddrMode**
	34	**Saturate** Format: **Saturate**
	33	**AccWrCtrl** Format: **AccWrCtrl**
	32	**AtomicCtrl** Format: **AtomicCtrl**
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". **Value** / **Name** / **Description** 0 / Normal **[Default]** / Normal. Per channel write enable used for final write enable generation. 1 / NoMask / NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	30	**Reserved**
	29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

cmpn - Compare NaN

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28

PredInv

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24

PredCtrl

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23

FlagRegNum[0]

This field specifies bit[0] of the register number for a flag register operand.

22

FlagSubRegNum

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19

ChanOff

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16

ExecSize

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

cmpn - Compare NaN		
	15:0	**Header** Format: **Header**

Conditional Select

csel - Conditional Select

Source:	EuIsa
Length Bias:	4
Predication:	false
Conditional Modifier:	true
Saturation:	true
Source Modifier:	true

The csel instruction selectively moves components in src0 or src1 to the dst based on the result of the compare of src2 with zero. If the channel condition is true, data in src0 is moved into dst. Otherwise, data in src1 is moved into dst. The csel instruction provides the function of a cmp followed by sel. The instruction must not be used if cmpn is required. The instruction does not update the flag register.
The comparison follows the same rule as cmp instruction for that data type.

When Access Mode is Align1, accumulator may be used as source or destination.

Format:

```
csel (exec_size) dst src0 src1 src2
```

Syntax

```
csel[.cmod] (exec_size) reg reg reg reg
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     bitMask[n] = 0;
     if ( EMask.chan[n] ) {
         result[n] = src2.chan[n] - 0;
         bitMask[n] = Condition(result[n]);
         if (bitMask[n] = 1) {
             dst.chan[n] = src0.chan[n];
         } else {
             dst.chan[n] = src1.chan[n];
         }
     }
 }
```

Src Types	Dst Types
F	F
HF	HF
D	D
W	W

DWord	Bit	Description
0..3	127:114	**Src2.Operand** Exists If: ([Src2.IsImm]==false) AND ([Header][Opcode]!=madm)

csel - Conditional Select

	Format:	**DirectOperand**
127:114	**Src2.Operand**	
	Exists If:	([Src2.IsImm]==false) AND ([Header][Opcode]==madm)
	Format:	**MacroOperand**
127:112	**Src2.ImmValue[15:0]**	
	Exists If:	([Src2.IsImm]==true)
113:112	**Src2.HorzStride**	
	Exists If:	([Src2.IsImm]==false)
	Format:	**HorzStride**
111:98	**Src1.Operand**	
	Exists If:	([Header][Opcode]!=madm)
	Format:	**DirectOperand**
111:98	**Src1.Operand**	
	Exists If:	([Header][Opcode]==madm)
	Format:	**MacroOperand**
97:96	**Src1.HorzStride**	
	Format:	**HorzStride**
95:92	**CondCtrl**	
	Format:	**FlagModifier**
91	**Src1.VertStride[1]**	
	Format:	**TernaryVertStride[1:1]**
90:88	**Src1.DataType**	
	Format:	**TernaryDataType**
87:86	**Src1.Mod**	
	Format:	**SrcMod**
85:84	**Src2.Mod**	
	Format:	**SrcMod**
83	**Src1.VertStride[0]**	
	Format:	**TernaryVertStride[0:0]**
82:80	**Src2.DataType**	
	Format:	**TernaryDataType**
79:66	**Src0.Operand**	
	Exists If:	([Src0.IsImm]==false) AND ([Header][Opcode]!=madm)
	Format:	**DirectOperand**
79:66	**Src0.Operand**	
	Exists If:	([Src0.IsImm]==false) AND ([Header][Opcode]==madm)

csel - Conditional Select

	Format:	**MacroOperand**
79:64	**Src0.ImmValue[15:0]**	
	Exists If:	([Src0.IsImm]==true)
65:64	**Src0.HorzStride**	
	Exists If:	([Src0.IsImm]==false)
	Format:	**HorzStride**
63:50	**Dst.Operand**	
	Exists If:	([Header][Opcode]!=madm)
	Format:	**DirectOperand**

Programming Notes
The Dst.Operand must be 64 bit aligned. i.e. Dst.Operand.SubRegNum[2:0] must be zero,

63:50	**Dst.Operand**	
	Exists If:	([Header][Opcode]==madm)
	Format:	**MacroOperand**
49	**Reserved**	
	Format:	MBZ

48 Dst.HorzStride

This field provides the distance in unit of data elements between two adjacent data elements within a row (horizontal) in the register region for the operand.

Value	Name
0	1 element
1	2 element

47 Src2.IsImm

This field indicate that Source 2 operand is carrying an immediate value.

Value	Name
0	false
1	true

46 Src0.IsImm

This field indicate that Source 0 operand is carrying an immediate value.

Value	Name
0	false
1	true

45:44	**Src0.Mod**	
	Format:	**SrcMod**
43	**Src0.VertStride[1]**	
	Format:	**TernaryVertStride[1:1]**

csel - Conditional Select

	42:40	**Src0.DataType** Format: **TernaryDataType**
	39	**ExecDataType** This field indicate the datatype mode of ternary instruction. Integer or Float. Value: 0 — Integer; 1 — Float
	38:36	**Dst.DataType** Format: **TernaryDataType**
	35	**Src0.VertStride[0]** Format: **TernaryVertStride[0:0]**
	34	**Saturate** Format: **Saturate**
	33	**AccWrCtrl** Format: **AccWrCtrl**
	32	**AtomicCtrl** Format: **AtomicCtrl**
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". Value: 0 — Normal **[Default]** — Normal. Per channel write enable used for final write enable generation. Value: 1 — NoMask — NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	30	**Reserved**
	29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. Value: 0 — NoCompaction **[Default]** — No compaction. 128-bit native instruction supporting all instruction options. Value: 1 — Compacted — Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

	28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

	27:24	**PredCtrl** Format: **PredCtrl** This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.
	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Constant Cache Dword Scattered Read MSD

MSD_CC_DWS - Constant Cache Dword Scattered Read MSD

Source: EuSubFunctionReadOnlyDataPort

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: Enable If set, indicates that the message includes the header.

MSD_CC_DWS - Constant Cache Dword Scattered Read MSD

	18	**Legacy Message** Default Value: 0h Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 03h Format: Opcode Dword Scattered Read message
	13	**Invalidate After Read** Format: MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12:10	**Reserved** Access: RO Format: MBZ
	9	**Legacy SIMD Mode** Default Value: 1h Format: Opcode Must be set for compatibility.
	8	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Constant Cache Oword Aligned Block Read MSD

MSD_CC_OWAB - Constant Cache Oword Aligned Block Read MSD

Source: EuSubFunctionReadOnlyDataPort

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18	**Reserved** Access: RO Format: MBZ
	17:14	**Message Type** Default Value: 01h Format: Opcode Oword Aligned Block Read Constant Cache message
	13:11	**Reserved** Access: RO Format: MBZ
	10:8	**Data Elements** Format: MDC_DB_OW Specifies the number of contiguous Owords to be read

MSD_CC_OWAB - Constant Cache Oword Aligned Block Read MSD

	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Constant Cache Oword Block Read MSD

MSD_CC_OWB - Constant Cache Oword Block Read MSD

Source: EuSubFunctionReadOnlyDataPort

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18	**Reserved** Access: RO Format: MBZ
	17:14	**Message Type** Default Value: 00h Format: Opcode Oword Block Read Constant Cache message
	13	**Invalidate After Read** Format: MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12:11	**Reserved** Access: RO Format: MBZ

MSD_CC_OWB - Constant Cache Oword Block Read MSD		
	10:8	**Data Elements** Format: **MDC_DB_OW** Specifies the number of contiguous Owords to be read or written
	7:0	**Binding Table Index** Format: **MDC_BTS** Specifies the Binding Table Index for the message

Continue

cont - Continue

Source:	Eulsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

The cont instruction disables execution for the subset of channels for the remainder of the current loop iteration. Channels remain disabled until right before the while instuction or right before the condition check code block for the while instruction. If all enabled channels hit this instruction, jump to the instruction referenced by JIP where execution continues. UIP should always reference the loop's associated while instruction. JIP should point to the last instruction of the inner most conditional block if the cont instruction is inside a conditional block. In case of the break instruction directly under the loop, the JIP and the UIP are the same. If SPF is ON, the UIP must be used to update IP; JIP is not used in this case.

The following table describes the two 32-bit instruction pointer offsets. Both the JIP and UIP are signed 32-bit numbers, added to IP pre-increment. In instruction binary, JIP and UIP are at locations src0 and src1 and must be of type DW (signed DWord integer). When the offsets are immediate, src0 regfile must be immediate.

Format:

```
[(pred)] cont (exec_size) JIP UIP
```

Restriction

The execution size must be the same for the while, break, and cont instructions of the same code block.

Syntax

```
[(pred)] cont (exec_size) imm32 imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.channel[n] ) {
         if ( PMask[n] ) {  // PMask is for all channels enabled for the cont instruction.
             PcIP[n] = IP + UIP;
         } else {
             PcIP[n] = IP + 1;
         }
     }
 }
 for ( n = exec_size; n < 32; n++ ) {
     PcIP[n] = IP + 1;
 }
 if ( PcIP != (IP + 1) ) {  // all channels true
     Jump(IP + JIP);
 }
```

DWord	Bit	Description

cont - Continue

0..3	127:96	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	127:96	**JIP** Exists If: ([Src0.IsImm]==true) Format: S31 The byte-aligned jump distance if a jump is taken for the channel.
	95:80	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	95:64	**Reserved** Exists If: ([Src0.IsImm]==true) AND ([Src1.IsImm]==false) Format: MBZ
	95:64	**UIP** Exists If: ([Src0.IsImm]==true) AND ([Src1.IsImm]==true) Format: S31 The byte aligned jump distance if a jump is taken for the instruction.
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) Format: **DirectOperand**
	65:64	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	63:50	**Dst.Operand** Format: **DirectOperand**
	49:48	**Reserved** Access: RO Format: MBZ
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value Value: 0 — false Value: 1 — true
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value

cont - Continue

Value	Name
0	false
1	true

45:34 Reserved

Access:	RO
Format:	MBZ

33 BranchCtrl

This field is used by*goto*,*if*, and*else*instructions to control branching. See the goto instruction description for more information about BranchCtrl.

32 AtomicCtrl

Format:	AtomicCtrl

31 MaskCtrl

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 Reserved

29 CmptCtrl

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 PredInv

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

cont - Continue

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16 **ExecSize**

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

15:0 **Header**

Format:	Header

Count Bits Set

cbit - Count Bits Set

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

The cbit instruction counts component-wise the total bits set in src0 and stores the resulting counts in dst.

Format:

```
[(pred)] cbit (exec_size) dst src0
```

Restriction

No accumulator access, implicit or explicit.

Syntax

```
[(pred)] cbit (exec_size) reg reg
 [(pred)] cbit (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         UD cnt = 0;
         UD val = src0.chan[n];
         while ( val ) {
             if ( val & 1 ) {
                 cnt ++;
             }
             val = val » 1;
         }
         dst.chan[n] = cnt;
     }
 }
```

Src Types	Dst Types
UB, UW, UD	UD

DWord	Bit	Description
0..3	127:96	**Src0.ImmValue[31:0]** Exists If: ([Src0.IsImm]==true)
	95:92	**CondCtrl** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))

cbit - Count Bits Set

		Format: **FlagModifier**
	95:64	**Src0.ImmValue[63:32]** Exists If: ([Src0.IsImm]==true) AND (([Src0.DataType]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df))
	87:84	**Src0.VertStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **VertStride**
	83:81	**Src0.Width** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **Width**
	80	**Src0.AddrMode** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Reserved**

cbit - Count Bits Set

	Access:	RO
	Format:	MBZ

46 Src0.IsImm

This field indicate that Source 0 operand is carrying an immediate value.

Value	Name
0	false **[Default]**
1	true

45:44 Src0.Mod

Format:	**SrcMod**

43:40 Src0.DataType

Exists If:	([Src0.IsImm]==false)
Format:	**RegDataType**

43:40 Src0.DataType

Exists If:	([Src0.IsImm]==true)
Format:	**ImmDataType**

39:36 Dst.DataType

Format:	**RegDataType**

35 Dst.AddrMode

Format:	**AddrMode**

34 Saturate

Format:	**Saturate**

33 AccWrCtrl

Format:	**AccWrCtrl**

32 AtomicCtrl

Format:	**AtomicCtrl**

31 MaskCtrl

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 Reserved

29 CmptCtrl

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact

instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

cbit - Count Bits Set		
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Dot Product 4 Accumulate

dp4a - Dot Product 4 Accumulate

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	true
Source Modifier:	false

DP4A is a packed four-wide integer dot product and accumulate operation. Each source's 32-bit channel value is treated as four element vector of 8-bit integer values. The operation performs a 32-bit precision dot product of those four bytes and adds it with a 32-bit accumulator (typically a GRF, not necessarily an acc# reg).

Format:

```
[(pred)] dp4a (exec_size) dst src0 src1 src2
```

Programming Notes

```
EXAMPLE (SIMD1 for simplicity):
    mov  (1) r1.0:d  0x0102037F:d
    // (char4)(0x1,0x2,0x3,0x7F)
    mov  (1) r2.0:d  50:d
    dp4a (1) r3.0:d  r2:d r1:d r1:d
    // r3.0 = 50 + (0x1*0x1 + 0x2*0x2 + 0x3*0x3 + 0x7F*0x7F)
    //      = 50 + (1 + 4 + 9 + 16129)
    //      = 16193
```

Restriction

All three-source instructions have certain restrictions, described in Instruction Formats.

Only one of src0 or src1 operand may be an the accumulator register (acc#).

Syntax

```
[(pred)] dp4a (exec_size) reg reg   reg reg
 [(pred)] dp4a (exec_size) reg imm16 reg reg
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
    if ( WrEn.chan[n] ) {
        dst[n] = src0.chan[n]  +
            src1.chan[n][7:0]   * src2.chan[n][7:0] +
            src1.chan[n][15:8]  * src2.chan[n][15:8] +
            src1.chan[n][23:16] * src2.chan[n][23:16] +
            src1.chan[n][31:24] * src2.chan[n][31:24];
    }
 }
```

dp4a - Dot Product 4 Accumulate

Src Types	Dst Types
*D	*D

DWord	Bit	Description
0..3	127:114	**Src2.Operand** Exists If: ([Src2.IsImm]==false) AND ([Header][Opcode]!=madm) Format: **DirectOperand**
	127:114	**Src2.Operand** Exists If: ([Src2.IsImm]==false) AND ([Header][Opcode]==madm) Format: **MacroOperand**
	127:112	**Src2.ImmValue[15:0]** Exists If: ([Src2.IsImm]==true)
	113:112	**Src2.HorzStride** Exists If: ([Src2.IsImm]==false) Format: **HorzStride**
	111:98	**Src1.Operand** Exists If: ([Header][Opcode]!=madm) Format: **DirectOperand**
	111:98	**Src1.Operand** Exists If: ([Header][Opcode]==madm) Format: **MacroOperand**
	97:96	**Src1.HorzStride** Format: **HorzStride**
	95:92	**CondCtrl** Format: **FlagModifier**
	91	**Src1.VertStride[1]** Format: **TernaryVertStride[1:1]**
	90:88	**Src1.DataType** Format: **TernaryDataType**
	87:86	**Src1.Mod** Format: **SrcMod**
	85:84	**Src2.Mod** Format: **SrcMod**
	83	**Src1.VertStride[0]** Format: **TernaryVertStride[0:0]**
	82:80	**Src2.DataType** Format: **TernaryDataType**

dp4a - Dot Product 4 Accumulate

79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) AND ([Header][Opcode]!=madm) Format: **DirectOperand**
79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) AND ([Header][Opcode]==madm) Format: **MacroOperand**
79:64	**Src0.ImmValue[15:0]** Exists If: ([Src0.IsImm]==true)
65:64	**Src0.HorzStride** Exists If: ([Src0.IsImm]==false) Format: **HorzStride**
63:50	**Dst.Operand** Exists If: ([Header][Opcode]!=madm) Format: **DirectOperand** **Programming Notes** The Dst.Operand must be 64 bit aligned. i.e. Dst.Operand.SubRegNum[2:0] must be zero,
63:50	**Dst.Operand** Exists If: ([Header][Opcode]==madm) Format: **MacroOperand**
49	**Reserved** Format: MBZ
48	**Dst.HorzStride** This field provides the distance in unit of data elements between two adjacent data elements within a row (horizontal) in the register region for the operand. Value: 0 — Name: 1 element Value: 1 — Name: 2 element
47	**Src2.IsImm** This field indicate that Source 2 operand is carrying an immediate value. Value: 0 — Name: false Value: 1 — Name: true
46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — Name: false

	1	true

45:44 Src0.Mod

Format:	SrcMod

43 Src0.VertStride[1]

Format:	TernaryVertStride[1:1]

42:40 Src0.DataType

Format:	TernaryDataType

39 ExecDataType

This field indicate the datatype mode of ternary instruction. Integer or Float.

Value	Name
0	Integer
1	Float

38:36 Dst.DataType

Format:	TernaryDataType

35 Src0.VertStride[0]

Format:	TernaryVertStride[0:0]

34 Saturate

Format:	Saturate

33 AccWrCtrl

Format:	AccWrCtrl

32 AtomicCtrl

Format:	AtomicCtrl

31 MaskCtrl

Mask Control (formerly Write Enable Control). This field determines if the the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 Reserved

29 CmptCtrl

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction

Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28

PredInv

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24

PredCtrl

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23

FlagRegNum[0]

This field specifies bit[0] of the register number for a flag register operand.

22

FlagSubRegNum

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19

ChanOff

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16

ExecSize

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

dp4a - Dot Product 4 Accumulate		
	15:0	**Header** Format: **Header**

Dword Atomic Counter with Return Data Operation MSD

MSD1R_DWAC - Dword Atomic Counter with Return Data Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header

MSD1R_DWAC - Dword Atomic Counter with Return Data Operation MSD

	18:14	**Message Type** Default Value: 0Bh Format: Opcode Atomic Counter Operation message
	13	**Return Data Control** Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2RS Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Dword Atomic Counter Write Only Operation MSD

MSD1W_DWAC - Dword Atomic Counter Write Only Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header

MSD1W_DWAC - Dword Atomic Counter Write Only Operation MSD

	18:14	**Message Type** Default Value: 0Bh Format: Opcode Atomic Counter Operation message
	13	**Return Data Control** Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2RS Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Dword Scattered Read MSD

MSD0R_DWS - Dword Scattered Read MSD

Source: EuSubFunctionDataPort0

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: Enable If set, indicates that the message includes the header.

MSD0R_DWS - Dword Scattered Read MSD

	18	**Legacy Message** Default Value: 0h Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 03h Format: Opcode Dword Scattered Read message
	13	**Invalidate After Read** Format: MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs
	12:10	**Reserved** Access: RO Format: MBZ
	9	**Legacy SIMD Mode** Default Value: 1h Format: Opcode Must be set for compatibility.
	8	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	7:0	**Binding Table Index** Format: MDC_BTS_A32 Specifies the Binding Table Index for the message

Dword Scattered Write MSD

MSD0W_DWS - Dword Scattered Write MSD

Source: EuSubFunctionDataPort0

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: Enable If set, indicates that the message includes the header.

MSD0W_DWS - Dword Scattered Write MSD

	18	**Legacy Message** Default Value: 0h Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 0Bh Format: Opcode Dword Scattered Write message
	13:10	**Reserved** Access: RO Format: MBZ
	9	**Legacy SIMD Mode** Default Value: 1h Format: Opcode Must be set for compatibility.
	8	**SIMD Mode** Format: MDC_SM2 Specifies the SIMD mode of the message (number of slots processed)
	7:0	**Binding Table Index** Format: MDC_BTS_A32 Specifies the Binding Table Index for the message

Dword Typed Atomic Integer with Return Data Operation MSD

MSD1R_DWTAI - Dword Typed Atomic Integer with Return Data Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.

MSD1R_DWTAI - Dword Typed Atomic Integer with Return Data Operation MSD

	18:14	**Message Type** Default Value: 06h Format: Opcode Typed Atomic Integer Operation message
	13	**Return Data Control** Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**Slot Group** Format: MDC_SG2 Specifies the Slot Group mode of the message (which slots are processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Dword Typed Atomic Integer Write Only Operation MSD

MSD1W_DWTAI - Dword Typed Atomic Integer Write Only Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.

MSD1W_DWTAI - Dword Typed Atomic Integer Write Only Operation MSD

	18:14	**Message Type** Default Value: 06h Format: Opcode Typed Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Slot Group** Format: MDC_SG2 Specifies the Slot Group mode of the message (which slots are processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Dword Untyped Atomic Float with Return Data Operation MSD

MSD1R_DWAF - Dword Untyped Atomic Float with Return Data Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.

MSD1R_DWAF - Dword Untyped Atomic Float with Return Data Operation MSD

	18:14	**Message Type** Default Value: 1Bh Format: Opcode Untyped Atomic Float Operation message
	13	**Return Data Control** Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2R Specifies the SIMD mode of the message (number of slots processed)
	11	**Data Width** Default Value: 0h Format: Opcode Operations are on 32-bit floats.
	10:8	**Atomic Float Operation** Format: MDC_FOP Specifies the atomic float operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Dword Untyped Atomic Float Write Only Operation MSD

MSD1W_DWAF - Dword Untyped Atomic Float Write Only Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.

MSD1W_DWAF - Dword Untyped Atomic Float Write Only Operation MSD

	18:14	**Message Type** Default Value: 1Bh Format: Opcode Untyped Atomic Float Operation message
	13	**Return Data Control** Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2R Specifies the SIMD mode of the message (number of slots processed)
	11	**Data Width** Default Value: 0h Format: Opcode Operations are on 32-bit floats.
	10:8	**Atomic Float Operation** Format: MDC_FOP Specifies the atomic float operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Dword Untyped Atomic Integer with Return Data Operation MSD

MSD1R_DWAI - Dword Untyped Atomic Integer with Return Data Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.

MSD1R_DWAI - Dword Untyped Atomic Integer with Return Data Operation MSD

	18:14	**Message Type** Default Value: 02h Format: Opcode Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2R Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Dword Untyped Atomic Integer Write Only Operation MSD

MSD1W_DWAI - Dword Untyped Atomic Integer Write Only Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.

MSD1W_DWAI - Dword Untyped Atomic Integer Write Only Operation MSD

	18:14	**Message Type** Default Value: 02h Format: Opcode Untyped Atomic Integer Operation message
	13	**Return Data Control** Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2R Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Else

else - Else

Source:	EuIsa
Length Bias:	4
Predication:	false
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false
Syntax:	JUMP_BINARY_IMM_IMM

The else instruction is an optional statement within an if/else/endif block of code. It restricts execution within the else/endif portion to the opposite set of channels enabled under the if/else portion. Channels which were inactive prior to entering the if/endif block remain inactive throughout the entire block. All enabled channels upon arriving the else instruction will be redirected to the matching endif. If all channels are redirected (by else or before else), a relative jump is performed to the location specified by <JIP>. The jump target should be the matching endif instruction for that conditional block. The following table describes the 32-bit <JIP>. In instruction binary, <JIP> is at location <src1> and must be of type D (signed dword integer). <JIP> must be an immediate operand, it is a signed 32-bit number and is intended to be forward referencing. This value is added to IP pre-increment. If the <branch_ctrl> bit is set, then the <JIP> points to the first join instruction within the else block and <UIP> points to the endif instruction. If the <branch_ctrl> bit is not set, <JIP> and <UIP>, both point to endif.

Format:

```
else (exec_size) JIP  UIP  branch_ctrl
```

Restriction

Predication is not allowed.

The execution size must be the same for the if, else, and endif instructions of the same code block.

The branch_ctrl must be set equal to (JIP != UIP).

Syntax

```
else (exec_size) imm32 imm32 branch_ctrl
```

Pseudocode

```
Evaluate(WrEn);
 for (n = 0; n < 32; n++) {
     if (WrEn.channel[n] == 1 || branch_ctrl) {
         PcIP[n] = IP + JIP;
     } else {
         PcIP[n] = IP + UIP;
     }
 }
 if (PcIP != (IP+1)) { // for all channels
     Jump(IP + JIP);
 }
```

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0..3</td><td>127:96</td><td>Reserved<table><tr><td>Exists If:</td><td>([Src0.IsImm]==false)</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>127:96</td><td>JIP<table><tr><td>Exists If:</td><td>([Src0.IsImm]==true)</td></tr><tr><td>Format:</td><td>S31</td></tr></table>The byte-aligned jump distance if a jump is taken for the channel.</td></tr>
<tr><td></td><td>95:80</td><td>Reserved<table><tr><td>Exists If:</td><td>([Src0.IsImm]==false)</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>95:64</td><td>Reserved<table><tr><td>Exists If:</td><td>([Src0.IsImm]==true) AND ([Src1.IsImm]==false)</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>95:64</td><td>UIP<table><tr><td>Exists If:</td><td>([Src0.IsImm]==true) AND ([Src1.IsImm]==true)</td></tr><tr><td>Format:</td><td>S31</td></tr></table>The byte aligned jump distance if a jump is taken for the instruction.</td></tr>
<tr><td></td><td>79:66</td><td>Src0.Operand<table><tr><td>Exists If:</td><td>([Src0.IsImm]==false)</td></tr><tr><td>Format:</td><td>DirectOperand</td></tr></table></td></tr>
<tr><td></td><td>65:64</td><td>Reserved<table><tr><td>Exists If:</td><td>([Src0.IsImm]==false)</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>63:50</td><td>Dst.Operand<table><tr><td>Format:</td><td>DirectOperand</td></tr></table></td></tr>
<tr><td></td><td>49:48</td><td>Reserved<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>47</td><td>Src1.IsImm
This field indicate that Source 1 operand is carrying an immediate value<table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>false</td></tr><tr><td>1</td><td>true</td></tr></table></td></tr>
<tr><td></td><td>46</td><td>Src0.IsImm
This field indicate that Source 0 operand is carrying an immediate value</td></tr>
</table>

else - Else

	Value	Name
	0	false
	1	true

45:34 Reserved

Access:	RO
Format:	MBZ

33 BranchCtrl

This field is used by *goto*, *if*, and *else* instructions to control branching. See the goto instruction description for more information about BranchCtrl.

32 AtomicCtrl

Format:	AtomicCtrl

31 MaskCtrl

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 Reserved

29 CmptCtrl

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 PredInv

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

else - Else

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16 **ExecSize**

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

15:0 **Header**

Format:	Header

End If

endif - End If

Source:	EuIsa
Length Bias:	4
Predication:	false
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

The endif instruction terminates an if/else/endif block of code. It restores execution to the channels that were active prior to the if/else/endif block. The endif instruction is also used to hop out of nested conditionals by jumping to the end of the next outer conditional block when all channels are disabled.

The following table describes the 32-bit JIP. In instruction binary, JIP is at location src1 and must be of type D (signed DWord integer). JIP must be an immediate operand, it is a signed 32-bit number. This value is added to IP pre-increment.

Format:

```
endif JIP
```

Restriction

Predication is not allowed.

The execution size must be the same for the if, else, and endif instructions of the same code block.

Syntax

```
endif (exec_size) imm32
```

Pseudocode

```
Evaluate(WrEn);
 if ( WrEn == 0 ) {  // all channels false
     Jump(IP + JIP);
 }
```

DWord	Bit	Description
0..3	127:96	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	127:96	**JIP** Exists If: ([Src0.IsImm]==true) Format: S31 The byte-aligned jump distance if a jump is taken for the channel

endif - End If

95:80	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
95:64	**Reserved** Exists If: ([Src0.IsImm]==true) Format: MBZ
79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) Format: **DirectOperand**
65:64	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
63:50	**Dst.Operand** Format: **DirectOperand**
49:47	**Reserved** Access: RO Format: MBZ
46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value.

Value	Name
0	false
1	true

45:34	**Reserved** Access: RO Format: MBZ
33	**BranchCtrl** This field is used by *goto*, *if*, and *else* instructions to control branching. See the **goto** instruction description for more information about BranchCtrl.
32	**AtomicCtrl** Format: **AtomicCtrl**
31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

endif - End If

	30	**Reserved**
	29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. **Value** / **Name** / **Description** 0 / NoCompaction **[Default]** / No compaction. 128-bit native instruction supporting all instruction options. 1 / Compacted / Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
	28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields **Value** / **Name** / **Description** 0 / Positive **[Default]** / Positive polarity of predication. Use the predication mask produced by PredCtrl. 1 / Negative / Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
	27:24	**PredCtrl** Format: **PredCtrl** This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.
	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

<table>
<tr><th colspan="3">endif - End If</th></tr>
<tr><td></td><td>21:19</td><td>ChanOff<table><tr><td>Format:</td><td>ChanOff</td></tr></table>This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.</td></tr>
<tr><td></td><td>18:16</td><td>ExecSize<table><tr><td>Format:</td><td>ExecSize</td></tr></table>This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.</td></tr>
<tr><td></td><td>15:0</td><td>Header<table><tr><td>Format:</td><td>Header</td></tr></table></td></tr>
</table>

Extended Math Function

math - Extended Math Function

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	true
Source Modifier:	true
Syntax:	GROUP
Subfunctions:	MathFC[95:92]

The math instruction performs extended math function on the components in src0, or src0 and src1, and write the output to the channels of dst. The type of extended math function are based on the FC[3:0] encoding in the table below.

Format:

```
[(pred)] math.<FC> (exec_size) dst src0
 [(pred)] math.<FC> (exec_size) dst src0 src1
```

Restriction
Accumulator access is allowed only for IEEE macro functions (invm and rsqtm).
The math instruction does not support indirect addressing modes.
The only supported rounding mode for math instruction is Round to Nearest Even.
INT DIV function does not support SIMD16. INT DIV function does not support simultaneous write to two registers. INT DIV function does not support source modifiers.
The FDIV function is not supported in ALT_MODE.
The math instruction can use GRF or immediates as source. The source formats for immediates must be one of the source formats supported by math operation.
DepCtrl must not be used.
The math instruction must use GRF as destination.
The supported regioning mode for math instruction is align1 and align16. The following restrictions apply for align1 mode: Scalar source is supported. Source and destination horizontal stride must be the same. Regioning must ensure Src.Vstride = Src.Width * Src.Hstride . Source and destination offset must be the same, except the case of scalar source.
Half-float denorms are always retained.
Math Operation rules when float and half-floats are mixed between source or between source and destination operands. The half-float operand must be interleaved in the register foralign1 and the source and destination register offset must be be the same to DW granularity. For align16, the half-float operand is allowed to be packed.
The execution size must be no more than 8 when half-floats are used in source or destination operand.

math - Extended Math Function

The source operand must not span two registers if the destination operand spans just one register Example: Case (a) // Allowed. The source must be strided by 2. the offset is allowed to select between lower/upper 16b math.inv (8) r10:f r11.0<16;8,2>:hf math.inv (8) r10:f r11.1<16;8,2>:hf math.invm (8) r10:f r11.0<16;8,2>:hf r12.1<16;8,2>:hf Case (b) // Allowed. The destination must be strided by 2. The offset is allowed to selecte between lower/upper 16b math.inv (8) r10.0<2>:hf r11.0<8;8,1>:f math.inv (8) r10.1<2>:hf r11.0<8;8,1>:f math.invm (8) r10.0<2>:hf r11.0<16;8,2>:hf r12.0<16;8,2>:hf Case (c) // Allowed. Destination has stride of 2. The offset is allowed to select between uppoer/lower 16b math.invm (8) r10.0<2>:hf r11.0<8;8,1>:f r12.1<16;8,2>:hf math.invm (8) r10.1<2>:hf r11.1<16;8,2>:hf r12.0<8;8,1>:f Case (d) // Not Allowed. Destination is half-float but is not interleaved. math.inv (8) r10.0<1>:hf r11.0<8;8,1>:f Case (e) // Not Allowed. Source is half-float but not interleaved math.invm (8) r10.0<2>:hf r11.0<8;8,1>:f r12.0<8;8,1>:hf Case (f) // Not Allowed. Source operand spans 2 registers while destination spans one register. math.sin (8) r83.8<1>:hf r12.4<4;4,1>:f

Math Operation rules when half-floats are used on both source and destination operands. The execution size must be 8. The half-float source must be packed or interleaved. When interleaving, both source and destination must be interleaved. Example: Case (a) // Allowed. The source and destination are packed or interleaved math.inv (8) r10.0:hf r11.0<8;8,1>:hf math.inv (8) r10.0<2>:hf r11.0<16;8,2>:hf math.inv (8) r10.8:hf r11.0<8;8,1>:hf math.inv (8) r10.8<2>:hf r11.0<16;8,2>:hf

For one source math operations src1 must encode the null register.

Syntax

```
[(pred)] math.<FC> (exec_size) reg reg reg
```

Pseudocode

```
Evaluate(WrEn);
 for (n = 0; n < exec_size; n++) {
     if (WrEn.channel[n] == 1) {
         switch FC[3:0] {
             case 1h: // math.inv
         dst.channel[n] = rcp(src0.channel[n]);
             case 2h: // math.log
         dst.channel[n] = log(src0.channel[n]);
             case 3h: // math.exp
         dst.channel[n] = exp(src0.channel[n]);
             case 4h: // math.sqt
         dst.channel[n] = sqt(src0.channel[n]);
             case 5h: // math.rsqt
         dst.channel[n] = rsqt(src0.channel[n]);
             case 6h: // math.sin
         dst.channel[n] = sin(src0.channel[n]);
             case 7h: // math.cos
         dst.channel[n] = cos(src0.channel[n]);
             case 9h: // math.fdiv  (src0 / src1)
         dst.channel[n] = fdiv(src0.channel[n], src1.channel[n]);
             case Ah: // math.pow
         dst.channel[n] = pow(src0.channel[n], src1/channel[n]);
             case Bh: // math.idiv (src0 / src1)
         idiv(src0.channel[n], src1.channel[n]);
         dst.channel[n] = quotient;
         dst+1.channel[n] = remainder;
             case Ch: // math.iqot
         idiv(src0.channel[n], src1.channel[n]);
         dst.channel[n] = quotient;
```

math - Extended Math Function

```
            case Dh: // math.irem
        idiv(src0.channel[n], src1.channel[n]);
        dst.channel[n] = remainder;
            case Eh: // math.invm
        dst.channel[n] = invm(src0.channel[n], src1.channel[n]);
            case Fh: // math.rsqtm
        dst.channel[n] = rsqtm(src0.channel[n]);
            }
        }
    }
}
```

Src Types	Dst Types
D	D
UD	UD
F, HF	F, HF

DWord	Bit	Description
0..3	127:126	**Reserved** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==false) Format: MBZ
	127:96	**Src0.ImmValue[31:0]** Exists If: ([Src0.IsImm]==true) AND ([Src1.IsImm]==false)
	127:96	**Src1.ImmValue[31:0]** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==true)
	125:122	**Reserved** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==false) Format: MBZ
	121:120	**Src1.Mod** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==false) Format: **SrcMod**
	119:116	**Src1.VertStride** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==false) Format: **VertStride**
	115:113	**Src1.Width** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==false) Format: **Width**
	112	**Reserved** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==false) Format: MBZ

math - Extended Math Function

	111:98	**Src1.Operand** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==false) AND ([FuncCtrl]==INVM) Format: **MacroOperand**
	111:98	**Src1.Operand** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==false) AND ([FuncCtrl]!=INVM) Format: **DirectOperand**
	97:96	**Src1.HorzStride** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==false) Format: **HorzStride**
	95:92	**FuncCtrl** Format: **MathFC**
	91:88	**Src1.DataType** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==true) Format: **ImmDataType**
	91:88	**Src1.DataType** Exists If: ([Src0.IsImm]==false) AND ([Src1.IsImm]==false) Format: **RegDataType**
	91:64	**Reserved** Exists If: ([Src0.IsImm]==true) Format: MBZ
	87:84	**Src0.VertStride** Exists If: ([Src0.IsImm]==false) Format: **VertStride**
	83:81	**Src0.Width** Exists If: ([Src0.IsImm]==false) Format: **Width**
	80	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) AND (([FuncCtrl]==INVM) OR ([FuncCtrl]==RSQTM)) Format: **MacroOperand**
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) AND (([FuncCtrl]!=INVM) AND ([FuncCtrl]!=RSQTM)) Format: **DirectOperand**
	65:64	**Src0.HorzStride**

math - Extended Math Function

	Exists If:	([Src0.IsImm]==false)
	Format:	**HorzStride**

63:50 Dst.Operand

Exists If:	([FuncCtrl]!=INVM) AND ([FuncCtrl]!=RSQTM)
Format:	**DirectOperand**

63:50 Dst.Operand

Exists If:	([FuncCtrl]==INVM) OR ([FuncCtrl]==RSQTM)
Format:	**MacroOperand**

49:48 Dst.HorzStride

Format:	**HorzStride**

47 Src1.IsImm

This field indicate that Source 1 operand is carrying an immediate value

Value	Name
0	false
1	true

46 Src0.IsImm

This field indicate that Source 0 operand is carrying an immediate value.

Value	Name
0	false
1	true

45:44 Src0.Mod

Exists If:	([Src0.IsImm]==false)
Format:	**SrcMod**

45:44 Reserved

Exists If:	([Src0.IsImm]==true)
Format:	MBZ

43:40 Src0.DataType

Exists If:	([Src0.IsImm]==false)
Format:	**RegDataType**

43:40 Src0.DataType

Exists If:	([Src0.IsImm]==true)
Format:	**ImmDataType**

39:36 Dst.DataType

Format:	**RegDataType**

	35	**Reserved** Access: RO Format: MBZ
	34	**Saturate** Format: **Saturate**
	33	**AccWrCtrl** Format: **AccWrCtrl**
	32	**AtomicCtrl** Format: **AtomicCtrl**
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". **Value** / **Name** / **Description** 0 / Normal **[Default]** / Normal. Per channel write enable used for final write enable generation. 1 / NoMask / NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	30	**Reserved**
	29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. **Value** / **Name** / **Description** 0 / NoCompaction **[Default]** / No compaction. 128-bit native instruction supporting all instruction options. 1 / Compacted / Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
	28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields **Value** / **Name** / **Description** 0 / Positive **[Default]** / Positive polarity of predication. Use the predication mask produced by PredCtrl.

math - Extended Math Function

		1 | Negative | Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
	27:24	**PredCtrl** Format: | **PredCtrl** This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.
	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: | **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: | **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: | **Header**

Find First Bit from LSB Side

fbl - Find First Bit from LSB Side

Source:	Eulsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

The fbl instruction counts component-wise the number of LSB 0 bits before the first 1 bit in src0, storing that number in dst.

Format:

```
[(pred)] fbl (exec_size) dst src0
```

Programming Notes

If src0 contains no 1 bits, store 0xFFFFFFFF in dst.

Restriction

No accumulator access, implicit or explicit.

Syntax

```
[(pred)] fbl (exec_size) reg reg
 [(pred)] fbl (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         UD cnt = 0;
         UD udScalar = src0.chan[n];
         while ( (udScalar & 1) == 0 && cnt != 32 ) {
             cnt ++;
             udScalar = udScalar » 1;
         }
         if ( src0.chan[n] == 0x00000000 ) {
             dst.chan[n] = 0xFFFFFFFF;
         } else {
             dst.chan[n] = cnt;
         }
     }
 }
```

Src Types	Dst Types
UD	UD

DWord	Bit	Description

fbl - Find First Bit from LSB Side

0..3	127:96	**Src0.ImmValue[31:0]** Exists If: ([Src0.IsImm]==true)
	95:92	**CondCtrl** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **FlagModifier**
	95:64	**Src0.ImmValue[63:32]** Exists If: ([Src0.IsImm]==true) AND (([Src0.DataType]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df))
	87:84	**Src0.VertStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **VertStride**
	83:81	**Src0.Width** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **Width**
	80	**Src0.AddrMode** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct)

fbl - Find First Bit from LSB Side

	Format:	**DirectOperand**
49:48	**Dst.HorzStride**	
	Format:	**HorzStride**
47	**Reserved**	
	Access:	RO
	Format:	MBZ
46	**Src0.IsImm**	
	This field indicate that Source 0 operand is carrying an immediate value.	

Value	Name
0	false **[Default]**
1	true

45:44	**Src0.Mod**	
	Format:	**SrcMod**
43:40	**Src0.DataType**	
	Exists If:	([Src0.IsImm]==false)
	Format:	**RegDataType**
43:40	**Src0.DataType**	
	Exists If:	([Src0.IsImm]==true)
	Format:	**ImmDataType**
39:36	**Dst.DataType**	
	Format:	**RegDataType**
35	**Dst.AddrMode**	
	Format:	**AddrMode**
34	**Saturate**	
	Format:	**Saturate**
33	**AccWrCtrl**	
	Format:	**AccWrCtrl**
32	**AtomicCtrl**	
	Format:	**AtomicCtrl**
31	**MaskCtrl**	
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

fbl - Find First Bit from LSB Side

	30	**Reserved**
	29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. **Value** / **Name** / **Description** 0 / NoCompaction **[Default]** / No compaction. 128-bit native instruction supporting all instruction options. 1 / Compacted / Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
	28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields **Value** / **Name** / **Description** 0 / Positive **[Default]** / Positive polarity of predication. Use the predication mask produced by PredCtrl. 1 / Negative / Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
	27:24	**PredCtrl** Format: **PredCtrl** This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.
	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

fbl - Find First Bit from LSB Side		
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Find First Bit from MSB Side

fbh - Find First Bit from MSB Side

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

If src0 is unsigned, the fbh instruction counts component-wise the leading zeros from src0 and stores the resulting counts in dst. If src0 is signed and positive, the fbh instruction counts component-wise the leading zeros from src0 and stores the resulting counts in dst. If src0 is signed and negative, the fbh instruction counts component-wise the leading ones from src0 and stores the resulting counts in dst.

Format:

```
[(pred)] fbh (exec_size) dst src0
```

Programming Notes

If src0 is zero, store 0xFFFFFFFF in dst.

If src0 is signed and is -1 (0xFFFFFFFF), store 0xFFFFFFFF in dst.

Restriction

No accumulator access, implicit or explicit.

Syntax

```
[(pred)] fbh (exec_size) reg reg
 [(pred)] fbh (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         UD cnt = 0;
         if ( src0 is unsigned ) {
             UD udScalar = src0.chan[n];
             while ( (udScalar & (1 « 31)) == 0 && cnt != 32 ) {
                 cnt ++;
                 udScalar = udScalar « 1;
             }
             if ( src0.chan[n] == 0x00000000 ) {
                 dst.chan[n] = 0xFFFFFFFF;
             } else {
                 dst.chan[n] = cnt;
             }
         } else {  // src0 is signed.
             D dScalar = src0.chan[n];
             bit cval = dScalar[31];
```

fbh - Find First Bit from MSB Side

```
            while ((dScalar & (1 « 31)) == cval && cnt != 32 ) {
                cnt ++;
                dScalar = dScalar « 1;
            }
            if ( (src0.chan[n] == 0xFFFFFFFF) || (src0.chan[n] == 0x00000000) ) {
                dst.chan[n] = 0xFFFFFFFF;
            } else {
                dst.chan[n] = cnt;
            }
        }
    }
}
```

Src Types	Dst Types
*D	UD

DWord	Bit	Description
0..3	127:96	**Src0.ImmValue[31:0]** Exists If: ([Src0.IsImm]==true)
	95:92	**CondCtrl** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **FlagModifier**
	95:64	**Src0.ImmValue[63:32]** Exists If: ([Src0.IsImm]==true) AND (([Src0.DataType]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df))
	87:84	**Src0.VertStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **VertStride**
	83:81	**Src0.Width** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **Width**
	80	**Src0.AddrMode** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand**

fbh - Find First Bit from MSB Side

		Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Reserved** Access: RO Format: MBZ
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value / Name 0 / false **[Default]** 1 / true
	45:44	**Src0.Mod** Format: **SrcMod**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: **RegDataType**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: **ImmDataType**
	39:36	**Dst.DataType** Format: **RegDataType**
	35	**Dst.AddrMode** Format: **AddrMode**
	34	**Saturate**

	Format:	Saturate

33 AccWrCtrl

Format:	AccWrCtrl

32 AtomicCtrl

Format:	AtomicCtrl

31 MaskCtrl

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 Reserved

29 CmptCtrl

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 PredInv

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

fbh - Find First Bit from MSB Side

	27:24	**PredCtrl** Format: **PredCtrl** This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.
	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Fraction

frc - Fraction

Source: Eulsa
Length Bias: 4
Predication: true
Conditional Modifier: true
Saturation: false
Source Modifier: true

The frc instruction computes, component-wise, the truncate-to-minus-infinity fractional values of src0 and stores the results in dst. The results, in the range of [0.0, 1.0], are the fractional portion of the source data. The result is in the range [0.0, 1.0] irrespective of the rounding mode. Floating-point fraction computation follows the rules in the following tables, based on the current floating-point mode.

Format:

```
[(pred)] frc[.cmod] (exec_size) dst src0
```

Syntax

```
[(pred)] frc[.cmod] (exec_size) reg reg
 [(pred)] frc[.cmod] (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         dst.chan[n] = src0.chan[n] - floor(src0.chan[n]);
     }
 }
```

Src Types	Dst Types
F	F

DWord	Bit	Description
0..3	127:96	**Src0.ImmValue[31:0]** Exists If: ([Src0.IsImm]==true)
	95:92	**CondCtrl** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **FlagModifier**
	95:64	**Src0.ImmValue[63:32]** Exists If: ([Src0.IsImm]==true) AND (([Src0.DataType]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df))
	87:84	**Src0.VertStride** Exists: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND

frc - Fraction

		If: ([Src0.DataType]!=:df) Format: **VertStride**
	83:81	**Src0.Width** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **Width**
	80	**Src0.AddrMode** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Reserved** Access: RO Format: MBZ
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. **Value** / **Name**

	0	false **[Default]**
	1	true
45:44	**Src0.Mod**	
	Format:	**SrcMod**
43:40	**Src0.DataType**	
	Exists If:	([Src0.IsImm]==false)
	Format:	**RegDataType**
43:40	**Src0.DataType**	
	Exists If:	([Src0.IsImm]==true)
	Format:	**ImmDataType**
39:36	**Dst.DataType**	
	Format:	**RegDataType**
35	**Dst.AddrMode**	
	Format:	**AddrMode**
34	**Saturate**	
	Format:	**Saturate**
33	**AccWrCtrl**	
	Format:	**AccWrCtrl**
32	**AtomicCtrl**	
	Format:	**AtomicCtrl**
31	**MaskCtrl**	

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30	**Reserved**	
29	**CmptCtrl**	
	Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description

0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16 **ExecSize**

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

15:0 **Header**

Format:	Header

Goto

goto - Goto

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

The goto instruction directs the instruction pointer to the offset specified by the UIP offset or to the next IP based on the BranchCtrl bit in the instruction. When BranchCtrl is set the active channels that are predicated on this instruction will take IP + UIP path, the others will continue with IP + 1, the active channels that are not predicated on this instruction will be made inactive. Irrespective of BranchCtrl when there are no active channels the instruction pointer will move to IP + JIP.
The goto instruction is used in conjunction with a join instruction. A goto deactivates some channels that are reactivated at some program-specified join instruction. See the join instruction for the activation rules.
The goto and join instructions enable unstructured program control flow. These instructions must be used with additional care where dangling channels can result without proper compiler checks, meaning that it is expected that programs will navigate through these paths to reactivate the channels. Hardware does not provide native checks or reconvergence.
The following table describes the two 32-bit instruction pointer offsets. Both the JIP and UIP are signed 32-bit numbers, added to IP pre-increment. In instruction binary, JIP and UIP are at locations src0 and src1 and must be of type DW (signed DWord integer).
If SPF is ON, none of the PcIP are updated.

Format:

```
[(pred)] goto (exec_size) JIP UIP branch_ctrl
```

Restriction

Cannot have a goto in the body and the corresponding join in the function or the subroutine. Similarly the brd and brc.

Syntax

```
[(pred)] goto (exec_size) imm32 imm32 branch_ctrl
```

Pseudocode

```
Evaluate(WrEn); for ( n = 0; n < exec_size; n++ ) { if ( WrEn.chan[n] ) { // for
the predicated active channels PcIP[n] = IP + UIP; } else { // join IP, for the
active non predicated channels PcIP[n] = IP + 1; } } if ( BranchCtrl ) { if (PcIP
!= (IP + UIP) ) { // for all channels if (PcIP != (IP + 1) ) { // for all
channels Jump(IP + JIP); } else { Jump(IP + 1); } } else { Jump(IP + UIP); } }
else { if (PcIP != (IP + 1) ) { // for all channels Jump(IP + JIP); } else {
Jump(IP + 1); } }
```

DWord	Bit	Description

goto - Goto

0..3	127:96	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	127:96	**JIP** Exists If: ([Src0.IsImm]==true) Format: S31 The byte-aligned jump distance if a jump is taken for the channel.
	95:80	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	95:64	**Reserved** Exists If: ([Src0.IsImm]==true) AND ([Src1.IsImm]==false) Format: MBZ
	95:64	**UIP** Exists If: ([Src0.IsImm]==true) AND ([Src1.IsImm]==true) Format: S31 The byte aligned jump distance if a jump is taken for the instruction.
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) Format: **DirectOperand**
	65:64	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	63:50	**Dst.Operand** Format: **DirectOperand**
	49:48	**Reserved** Access: RO Format: MBZ
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value Value: 0 — false Value: 1 — true
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value

goto - Goto

Value	Name
0	false
1	true

45:34 Reserved

Access:	RO
Format:	MBZ

33 BranchCtrl

This field is used by*goto*,*if*, and*else*instructions to control branching. See the goto instruction description for more information about BranchCtrl.

32 AtomicCtrl

Format:	AtomicCtrl

31 MaskCtrl

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 Reserved

29 CmptCtrl

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 PredInv

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16 **ExecSize**

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

15:0 **Header**

Format:	Header

GPGPU_CSR_BASE_ADDRESS

GPGPU_CSR_BASE_ADDRESS

Source: BSpec

Length Bias: 2

The GPGPU_CSR_BASE_ADDRESS command sets the base pointers for EU and L3 to Context Save and Restore EU State and SLM for **GPGPU** mid-thread preemption.

Programming Notes

Execution of this command causes a full pipeline flush, thus its use should be minimized for higher performance. State and instruction caches are flushed on completion of the flush.

SW must always program PIPE_CONTROL with "CS Stall" and "Render Target Cache Flush Enable" set prior to programming GPGPU_CSR_BASE_ADDRESS command for GPGPU workloads i.e when pipeline select is GPGPU via PIPELINE_SELECT command. This is required to achieve better GPGPU preemption latencies for certain programming sequences. If programming PIPE_CONTROL has performance implications then preemption latencies can be trade off against performance by not implementing this programming note.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: Opcode
	28:27	**Command SubType** Default Value: 0h GFXPIPE_COMMON Format: Opcode
	26:24	**3D Command Opcode** Default Value: 1h GFXPIPE_NONPIPELINED Format: Opcode
	23:16	**3D Command Sub Opcode** Default Value: 04h GPGPU_CSR_BASE_ADDRESS Format: Opcode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Format: =n Value / Name / Description 1h / Length_1 **[Default]** / Excludes DWord(0,1)
1..2	63:12	**GPGPU CSR Base Address** Format: GraphicsAddress[63:12]

GPGPU_CSR_BASE_ADDRESS		
		Specifies the 256K-byte aligned base address for GPGPU context GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	11:0	**Reserved** Access: RO Format: MBZ

GPGPU_WALKER

GPGPU_WALKER

Source: RenderCS

Length Bias: 2

Programming Notes

If the threads spawned by this command are required to observe memory writes performed by threads spawned from a previous command, software must precede this command with a command that performs a memory flush (e.g., MI_FLUSH).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h GPGPU_WALKER Format: OpCode
	23:16	**SubOpcode** Default Value: 05h GPGPU_WALKER SubOp Format: OpCode
	15:11	**Reserved** Access: RO Format: MBZ
	10	**Indirect Parameter Enable** Format: Enable If set, the values in DW 7, 10, 12are ignored and replaced by the current values of the corresponding GPGPU_xxx MMIO registers: • GPGPU_DISPATCHDIMX (instead of DW7) • GPGPU_DISPATCHDIMY (instead of DW10) • GPGPU_DISPATCHDIMZ (instead of DW12)
	9	**Reserved** Access: RO Format: MBZ
	8	**Predicate Enable** Format: Enable

		If set, this command is executed (or not) depending on the current value of the MI Predicate internal state bit. This command is ignored only if **PredicateEnable** is set and the Predicate state bit is 0. o
	7:0	**DWord Length** Format: =n **Value** / **Name** / **Description** 0Dh / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)
1	31:8	**Reserved**
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**Interface Descriptor Offset** Format: U6 This field specifies the offset from the interface descriptor base pointer to the interface descriptor which will be applied to this object. It is specified in units of interface descriptors.
2	31:17	**Reserved** Access: RO Format: MBZ
	16:0	**Indirect Data Length** Format: U17 This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. If Indirect Data is enabled, it is assumed that CURBE is not being used. This field must have the same alignment as the Indirect Object Data Start Address. It must be DQWord (32-byte) aligned.
3	31:6	**Indirect Data Start Address** Format: IndirectObjectOffset[31:6] This field specifies the Graphics Memory starting address of the data to be loaded into the kernel for processing. This pointer is relative to the **Indirect Object Base Address.** Hardware ignores this field if indirect data is not present. Alignment of this address depends on the mode of operation. It is the 64-byte aligned address of the indirect data. **Value** / **Name** / **Description** [0x0,0x3FFFFF] / / Range 0 .. 512 MB. (Bits 31:29 MBZ)
	5:0	**Reserved** Access: RO Format: MBZ

GPGPU_WALKER

<table>
<tr><td>4</td><td>31:30</td><td>SIMD Size
This field determines the size of the payload and the number of bits of the execution mask that are expected. The kernel pointed to by the interface descriptor should match the SIMD declared here.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>SIMD8</td><td>8 LSBs of the execution mask are used</td></tr>
<tr><td>1</td><td>SIMD16</td><td>16 LSBs used in execution mask</td></tr>
<tr><td>2</td><td>SIMD32</td><td>32 bits of execution mask used</td></tr>
</table></td></tr>
<tr><td></td><td>29:22</td><td>Reserved
<table>
<tr><td>Access:</td><td>RO</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>21:16</td><td>Thread Depth Counter Maximum
The maximum value of the thread depth counter. Since the counter starts at 0, the depth is this value + 1.
(Thread_Depth_Max+1)*(Thread_Height_Max+1)*(Thread_Width_Max+1) must equalNumber of Threads in GPGPU Thread Group in the Interface Descriptor.</td></tr>
<tr><td></td><td>15:14</td><td>Reserved
<table>
<tr><td>Access:</td><td>RO</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>13:8</td><td>Thread Height Counter Maximum
<table>
<tr><td>Format:</td><td>U6-1</td></tr>
</table>The maximum value of the thread height counter. The height is this value + 1.</td></tr>
<tr><td></td><td>7:6</td><td>Reserved
<table>
<tr><td>Access:</td><td>RO</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>5:0</td><td>Thread Width Counter Maximum
<table>
<tr><td>Format:</td><td>U6-1</td></tr>
</table>The maximum value of the thread width counter. The width is this value + 1.</td></tr>
<tr><td>5</td><td>31:0</td><td>Thread Group ID Starting X
This is the initial value of the X component of the thread group. When X reaches the maximum value it rolls around to this value.</td></tr>
<tr><td>6</td><td>31:0</td><td>Reserved
<table>
<tr><td>Access:</td><td>RO</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td>7</td><td>31:0</td><td>Thread Group ID X Dimension
The X dimension of the thread group (maximum X is dimension -1)</td></tr>
</table>

GPGPU_WALKER		
8	31:0	**Thread Group ID Starting Y** This is the initial value of the Y component of the thread group. When Y reaches the maximum value it rolls around to this value.
9	31:0	**Reserved** Access: RO Format: MBZ
10	31:0	**Thread Group ID Y Dimension** The Y dimension of the thread group (maximum Y is dimension -1)
11	31:0	**Thread Group ID Starting/Resume Z** This is the initial value of the Z component of the thread group.
12	31:0	**Thread Group ID Z Dimension** The Z dimension of the thread group (maximum Z is dimension -1)
13	31:0	**Right Execution Mask** The execution mask used for threads on the right (maximum X) of the thread group.
14	31:0	**Bottom Execution Mask** The execution mask used for threads on the bottom (maximum Y) of the thread group.

Half Precision HI8DS Render Target Write MSD

MSD_RTWH_HI8DS - Half Precision HI8DS Render Target Write MSD

Source: EuSubFunctionRenderDataPort

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Message Precision Subtype** Default Value: 1h Format: Opcode Half precision data message
	29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	**Per-Coarse Pixel PS outputs enable** Format: Enable This bit indicates the render target write is a coarse pixel write. **Programming Notes** This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.

MSD_RTWH_HI8DS - Half Precision HI8DS Render Target Write MSD

	17:14	**Message Type** Default Value: 0Ch Format: Opcode Render Target Write message
	13	**Per-Sample PS Enable** Format: Enable If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot. **Programming Notes** This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.
	12	**Last Render Target Select** Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.
	11	**Slot Group Select** Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 3h Format: Opcode SIMD8 dual source message. Use slots [15:8] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:24] are referenced instead of [15:8].

MSD_RTWH_HI8DS - Half Precision HI8DS Render Target Write MSD

	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Half Precision LO8DS Render Target Write MSD

MSD_RTWH_LO8DS - Half Precision LO8DS Render Target Write MSD

Source: EuSubFunctionRenderDataPort

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Message Precision Subtype** Default Value: 1h Format: Opcode Half precision data message
	29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	**Per-Coarse Pixel PS outputs enable** Format: Enable This bit indicates the render target write is a coarse pixel write. **Programming Notes** This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.

MSD_RTWH_LO8DS - Half Precision LO8DS Render Target Write MSD

	17:14	**Message Type** Default Value: 0Ch Format: Opcode Render Target Write message
	13	**Per-Sample PS Enable** Format: Enable If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot. **Programming Notes** This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.
	12	**Last Render Target Select** Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.
	11	**Slot Group Select** Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 2h Format: Opcode SIMD8 dual source message. Use slots [7:0] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [23:16] are referenced instead of [7:0].

MSD_RTWH_LO8DS - Half Precision LO8DS Render Target Write MSD

	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Half Precision REP16 Render Target Write MSD

MSD_RTWH_REP16 - Half Precision REP16 Render Target Write MSD

Source: EuSubFunctionRenderDataPort

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Message Precision Subtype** Default Value: 1h Format: Opcode Half precision data message
	29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	**Per-Coarse Pixel PS outputs enable** Format: Enable This bit indicates the render target write is a coarse pixel write. **Programming Notes** This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determied by Per-Sample PS outputs enable)

MSD_RTWH_REP16 - Half Precision REP16 Render Target Write MSD

		from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.
	17:14	**Message Type** Default Value: 0Ch Format: Opcode Render Target Write message
	13	**Per-Sample PS Enable** Format: Enable If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot. **Programming Notes** This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.
	12	**Last Render Target Select** Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.
	11	**Slot Group Select** Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 1h Format: Opcode SIMD16 Single source message with replicated data. Use slots [15:0] for pixel enables, X/Y

		addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:16] are referenced instead of [15:0].
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Half Precision SIMD8 Render Target Write MSD

MSD_RTWH_SIMD8 - Half Precision SIMD8 Render Target Write MSD

Source: EuSubFunctionRenderDataPort

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Message Precision Subtype** Default Value: 1h Format: Opcode Half precision data message
	29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	**Per-Coarse Pixel PS outputs enable** Format: Enable This bit indicates the render target write is a coarse pixel write. **Programming Notes** This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determied by Per-Sample PS outputs enable)

	from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.
17:14	**Message Type** Default Value: 0Ch Format: Opcode Render Target Write message
13	**Per-Sample PS Enable** Format: Enable If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot. **Programming Notes** This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.
12	**Last Render Target Select** Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.
11	**Slot Group Select** Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
10:8	**Render Target Message Subtype** Default Value: 4h Format: Opcode SIMD8 single source message. Use slots [7:0] for pixel enables, X/Y addresses, and oMask.

MSD_RTWH_SIMD8 - Half Precision SIMD8 Render Target Write MSD

		Programming Notes The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [23:16] are referenced instead of [7:0].
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Half Precision SIMD16 Render Target Write MSD

MSD_RTWH_SIMD16 - Half Precision SIMD16 Render Target Write MSD

Source: EuSubFunctionRenderDataPort

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Message Precision Subtype** Default Value: 1h Format: Opcode Half precision data message
	29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	**Per-Coarse Pixel PS outputs enable** Format: Enable This bit indicates the render target write is a coarse pixel write. **Programming Notes** This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determied by Per-Sample PS outputs enable)

	from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.
17:14	**Message Type** Default Value: 0Ch Format: Opcode Render Target Write message
13	**Per-Sample PS Enable** Format: Enable If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot. **Programming Notes** This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.
12	**Last Render Target Select** Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.
11	**Slot Group Select** Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
10:8	**Render Target Message Subtype** Default Value: 0h Format: Opcode SIMD16 Single source message. Use slots [15:0] for pixel enables, X/Y addresses, and oMask.

MSD_RTWH_SIMD16 - Half Precision SIMD16 Render Target Write MSD

		Programming Notes The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:16] are referenced instead of [15:0].
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Halt

halt - Halt

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

The halt instruction temporarily suspends execution for all enabled compute channels. Upon execution, the enabled channels are sent to the instruction at (IP + UIP), if all channels are enabled at HALT, jump to the instruction at (IP + JIP). If the halt instruction is not inside any conditional code block, the values of JIP and UIP should be the same. If the halt instruction is inside a conditional code block, the UIP should be the end of the program and the JIP should be the end of the inner most conditional code block. The UIP must point to a HALT Instruction. If SPF is ON, the UIP must be used to update IP; JIP is not used in this case.

The following table describes the two 32-bit instruction pointer offsets. Both the JIP and UIP are signed 32-bit numbers, added to IP pre-increment. In instruction binary, JIP and UIP are at locations src0 and src1 and must be of type DW (signed DWord integer). When the offsets are immediate, src0 regfile must be immediate and dst must be null.

Format:

```
[(pred)] halt (exec_size) JIP UIP
```

Syntax

```
[(pred)] halt (exec_size) imm32 imm32
```

pre>

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < 32; n++ ) {
     if ( WrEn.channel[n] ) {
         PcIP[n] = IP + UIP;
     } else {
         PcIP[n] = IP + 1;
     }
 }
 if ( PcIP != (IP + 1) ) {  // for all channels
     Jump(IP + JIP);
 }
```

pre>

DWord	Bit	Description
0..3	127:96	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	127:96	**JIP**

		Exists If: ([Src0.IsImm]==true) Format: S31 The byte-aligned jump distance if a jump is taken for the channel.
	95:80	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	95:64	**Reserved** Exists If: ([Src0.IsImm]==true) AND ([Src1.IsImm]==false) Format: MBZ
	95:64	**UIP** Exists If: ([Src0.IsImm]==true) AND ([Src1.IsImm]==true) Format: S31 The byte aligned jump distance if a jump is taken for the instruction.
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) Format: **DirectOperand**
	65:64	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	63:50	**Dst.Operand** Format: **DirectOperand**
	49:48	**Reserved** Access: RO Format: MBZ
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value **Value** / **Name** 0 – false 1 – true
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value **Value** / **Name** 0 – false 1 – true
	45:34	**Reserved**

halt - Halt

	Access:	RO
	Format:	MBZ

33 BranchCtrl

This field is used by *goto*, *if*, and *else* instructions to control branching. See the goto instruction description for more information about BranchCtrl.

32 AtomicCtrl

Format:	AtomicCtrl

31 MaskCtrl

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 Reserved

29 CmptCtrl

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 PredInv

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 PredCtrl

		Format: **PredCtrl** This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.
	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

HCP_BSD_OBJECT

HCP_BSD_OBJECT

Source: VideoCS

Length Bias: 2

The HCP is selected with the **Media Instruction Opcode "7h"** for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

The HCP_BSD_OBJECT command fetches the HEVC bit stream for a slice starting with the first byte in the slice. The bit stream ends with the last non-zero bit of the frame and does not include any zero-padding at the end of the bit stream. There can be multiple slices in a HEVC frame and thus this command can be issued multiple times per frame.

The HCP_BSD_OBJECT command must be the last command issued in the sequence of batch commands before the HCP starts decoding. Prior to issuing this command, it is assumed that all configuration parameters in the HCP have been loaded including workload configuration registers and configuration tables. When this command is issued, the HCP is waiting for bit stream data to be presented to the shift register.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 20h HCP_BSD_OBJECT_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 1h Name:

HCP_BSD_OBJECT		
1	31:0	**Indirect BSD Data Length** Format: U32 Specifies the length in bytes of the bitstream data for the current slice. It includes the first byte of the slice and the last non-zero byte of the in the slice. Specifically, the zero-padding bytes(if present) and the next start-code are excluded.
2	31:29	**Reserved** Access: RO Format: MBZ
	28:0	**Indirect Data Start Address** Format: U29 Specifies the byte-aligned graphics memory starting address of the slice bit stream relative to the **BSD Indirect Object Base Address**.

HCP_FQM_STATE

HCP_FQM_STATE

Source: VideoCS

Length Bias: 2

The HCP_FQM_STATE command loads the custom HEVC quantization tables into local RAM and may be issued up to8 times: 4 scaling list per intra and inter.

Driver is responsible for performing the Scaling List division. So, save the division HW cost in HW. The 1/x value is provided in 16-bit fixed-point precision as ((1«17)/QM +1) » 1. .

Note: FQM is computed as (2^16)/QM. If QM=1, FQM=all 1's.

To simplify the design, only a limited number of scaling lists are provided at the PAK interface: default two SizeID0 and two SizeID123 (one set for inter and the other set for intra), and the encoder only allows custom entries for these four matrices. The DC value of SizeID2 and SizeID3 will be provided.

When the scaling_list_enable_flag is set to disable, the scaling matrix is still sent to the PAK, and with all entries programmed to the same value of 16.

This is a picture level state command and is issued in encoding processes only.

Dwords 2-33 form a table for the DCT coefficients, 2 16-bit coefficients/DWord.

- Size 4x4 for SizeID0, DWords 2-9.
- Size 8x8 for SizeID1/2/3, DWords 2-33.

SizeID 0 (Table 4-13)

4x4	[31:16]	[15:0]
DWord 2	AC(0,1)	DC
DWord 3	AC(0,3)	AC(0,2)
DWord 4	AC(1,1)	AC(1,0)
DWord 5	AC(1,3)	AC(1,2)
DWord 6	AC(2,1)	AC(2,0)
DWord 7	AC(2,3)	AC(2,2)
DWord 8	AC(3,1)	AC(3,0)
DWord 9	AC(3,3)	AC(3,2)

SizeID 1, 2, 3 (Table 4-14)

8x8	[31:16]	[15:0]
DWord 2	AC(0,1)	DC
DWord 3	AC(0,3)	AC(0,2)
DWord 4	AC(0,5)	AC(0,4)
DWord 5	AC(0,7)	AC(0,6)
DWord 6	AC(1,1)	AC(1,0)
DWord 7	AC(1,3)	AC(1,2)

HCP_FQM_STATE

DWord 8	AC(1,5)	AC(1,4)
DWord 9	AC(1,7)	AC(1,6)
...		
DWord 30	AC(7,1)	AC(7,0)
DWord 31	AC(7,3)	AC(7,2)
DWord 32	AC(7,5)	AC(7,4)
DWord 33	AC(7,7)	AC(7,6)

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 5h HCP_FQM_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 20h Name:
1	31:16	**FQM DC Value: (1/DC):** Format: U16 Specifies DC value of the scaling list for 16x16 (SizeID=2) or 32x32 (SizeID=3). DC Value = scaling_list_dc_coef_minus8 + 8. Driver will do the division.

HCP_FQM_STATE

	15:5	**Reserved** Access: RO Format: MBZ
	4:3	**Color Component** Format: U2 Luma and Chroma's share the same scaling list and DC value for the same SizeID. 0 – Luma 1 – Chroma Cb 2 – Chroma Cr 3 – Reserved
	2:1	**SizeID** Format: U2 0 – SizeID 0 4x4 1 – SizeID 1, 2, 3 (8x8, 16x16, 32x32) 2 – SizeID 2 (for DC value in 16x16) 3 – SizeID 3 (for DC value in 32x32)
	0	**Intra/Inter** Format: U1 This field specifies the quant matrix intra or inter type. 0 – Intra 1 – Inter
2..33	1023:0	**QuantizerMatrix**

HCP_IND_OBJ_BASE_ADDR_STATE

HCP_IND_OBJ_BASE_ADDR_STATE

Source: VideoCS

Length Bias: 2

The HCP is selected with the **Media Instruction Opcode "7h"** for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

The HCP_IND_OBJ_BASE_ADDR_STATE command is used to define the indirect object base address of the stream in graphics memory. This is a frame level command. (Is it frame or picture level?)
This is a picture level state command and is issued in both encoding and decoding processes.

Compressed Header Format

Fields	Bits	
Bin	0	Kernel Binarized Syntax
Probability select	1	0 ->indicates probability 128 1 ->indicates probability 256
	Repeat to pack a Cacheline	

Partition1 and TileSize record

Fields	Bits	
Tile Size	31:0	Partition1 Size is 16-bit value, Tile Size is 32-bit value
AddressOffset	63:32	Cacheline Address Offset to be Modified
Offset	69:64	Byte offset to be Modified
16-bit vs 32-bit update	70	0: Update 16-bit; 1: Update 32-bit
Reserved	511:71	

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 3h HCP_IND_OBJ_BASE_ADDR_STATE

HCP_IND_OBJ_BASE_ADDR_STATE

		Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1).
1..2	63:0	**HCP Indirect Bitstream Object Base Address** Format: **SplitBaseAddress4KByteAligned** Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the HCP_BSD_OBJECT command for fetching (reading) the compressed Slice Data.
3	31:0	**HCP Indirect Bitstream Object Memory Address Attributes** Format: **MemoryAddressAttributes**
4..5	63:0	**HCP Indirect Bitstream Object Access Upper Bound** Format: **SplitBaseAddress4KByteAligned** Decoder only. This field specifies the 4K-byte aligned maximum memory address access by the indirect data object in the HCP_BSD_OBJECT command for the slice bit stream. Indirect data accessed at this address or greater will cause the HCP to stop issuing requests to the GAC and the BSP VLD will then only receive zeros until a slice done is received. Setting this field to 0 will cause this range to be ignored by the HCP.
6..7	63:0	**HCP Indirect CU Object Base Address** Format: **BaseAddress4KByteAligned** Encoder only. Specifies the 4K-byte aligned data buffer base address for the read-only indirect data object for fetching (reading) per CU data during the encoding process.
8	31:0	**HCP Indirect CU Object Object Memory Address Attributes** Format: **MemoryAddressAttributes**
9..10	63:0	**HCP PAK-BSE Object Base Address** Format: **BaseAddress4KByteAligned** Encoder only. Specifies the 4K-byte aligned memory base address for the write-only data pointed by the PAK engine for writing out the compressed bitstream.

HCP_IND_OBJ_BASE_ADDR_STATE

11	31:0	**HCP PAK-BSE Object Address Memory Address Attributes** Format: **MemoryAddressAttributes** Encoder only.
12..13	63:0	**HCP PAK-BSE Object Access Upper Bound** Format: **SplitBaseAddress4KByteAligned** Encoder only. This field specifies the 4K-byte aligned maximum memory address access by the HCP_PAK_OBJECT command for writing out the slice bit stream. Indirect data accessed at this address and beyond will be blocked by the hardware and ignored. This address must be greater than the HCP PAK-BSE Object Base Address state.
14..15	63:0	**HCP VP9 PAK Compressed Header Syntax Streamin- Base Address** Exists If: //Encoder Only Format: **BaseAddress4KByteAligned** Specifies the 4K-byte aligned data buffer base address for the read-only Probability counters fetching during the encoding process..
16	31:0	**HCP VP9 PAK Compressed Header Syntax StreamIn Memory Address Attributes** Format: **MemoryAddressAttributes**
17..18	63:0	**HCP VP9 PAK Probability Counter StreamOut- Base Address** Exists If: //Encoder Only Format: **BaseAddress4KByteAligned** Specifies the 4K-byte aligned data buffer base address for the write-only Probability counters fetching during the encoding process..
19	31:0	**HCP VP9 PAK Probability Counter StreamOut Memory Address Attributes** Exists If: //Encoder Only Format: **MemoryAddressAttributes**
20..21	63:0	**HCP VP9 PAK Probability Deltas StreamIn- Base Address** Exists If: //Encoder Only Format: **BaseAddress4KByteAligned** Specifies the 4K-byte aligned data buffer base address for the read-only Probability differences during the encoding process.
22	31:0	**HCP VP9 PAK Probability Deltas StreamIn Memory Address Attributes** Exists If: //Encoder Only Format: **MemoryAddressAttributes**

HCP_IND_OBJ_BASE_ADDR_STATE		
23..24	63:0	**HCP VP9 PAK Tile Record StreamOut- Base Address** Format: **BaseAddress4KByteAligned** Specifies the 4K-byte aligned memory base address for the write-only data pointed by the PAK engine for writing out the Tile Record.
25	31:0	**HCP VP9 PAK Tile Record StreamOut Memory Address Attributes** Exists If: //Encoder Only Format: **MemoryAddressAttributes**
26..27	63:0	**HCP VP9 PAK CU Level Statistic StreamOut- Base Address** Exists If: //Encoder Only Format: **BaseAddress4KByteAligned** Specifies the 4K-byte aligned memory base address for the write-only data pointed by the PAK engine for writing out the CU Record.
28	31:0	**HCP VP9 PAK CU Level Statistic StreamOut Memory Address Attributes** Exists If: //Encoder Only Format: **MemoryAddressAttributes**

HCP_PAK_INSERT_OBJECT

HCP_PAK_INSERT_OBJECT	
Source:	VideoCS
Length Bias:	2

It is an encoder only command, operating at bitstream level, before and after SliceData compressed bitstream. It is setup by the header and tail present flags in the Slice State command. If these flags are set and no subsequent PAK_INSERT_OBJECT commands are issued, the pipeline will hang.
The HCP_ PAK_ INSERT _OBJECT command supports both inline and indirect data payload, but only one can be active at any time. It is issued to insert a chunk of bits (payload) into the current compressed bitstream output buffer (specified in the HCP_PAK-BSE Object Base Address field of the HCP_IND_OBJ_BASE_ADDR_STATE command)starting at its current write pointer bit position. Hardware will keep track of this write pointer's byte position and the associated next bit insertion position index.
It is a variable length command when the payload (data to be inserted) is presented as inline data within the command itself. The inline payload is a multiple of 32-bit (1 DW), as the data bus to the compressed bitstream output buffer is 32-bit wide.
The payload data is required to be byte aligned on the left (first transmitted bit order) and may or may not be byte aligned on the right (last transmitted bits). The command will specify the bit offset of the last valid DW. Note that : Stitch Command is used if the beginning position of data is in bit position. When PAK Insert Command is used the beginning position must be in byte position.
Multiple insertion commands can be issued back to back in a series. It is host software's responsibility to make sure their corresponding data will properly stitch together to form a valid bitstream.
Internally, HCP hardware will keep track of the very last two bytes' (the very last byte can be a partial byte)values of the previous insertion. It is required that the next Insertion Object Command or the next PAK Object Command to perform the start code emulation sequence check and prevention 0x03 byte insertion with this end condition of the previous insertion.
The payload data may have already been processed for start code emulation byte insertion, except the possibility of the last 2 bytes plus the very last partial byte (if any). Hence, when hardware performing the concatenation of multiple consecutive insertion commands, or concatenation of an insertion command and a PAK object command, it must check and perform the necessary start code emulation byte insert at the junction.
Data to be inserted can be a valid NAL units or a partial NAL unit. It can be any encoded syntax elements bit data before the encoded Slice Data (PAK Object Command) of the current Slice - SPS NAL, PPS NAL, SEI NAL and Other Non-Slice NAL, Leading_Zero_8_bits (as many bytes as there is), Start Code , Slice Header. Any encoded syntax elements bit data after the encoded Slice Data (PAK Object Command) of the current Slice and prior to the next encoded Slice Data of the next Slice or prior to the end of the bitstream, whichever comes firstCabac_Zero_Word or Trailing_Zero_8bits (as many bytes as there is).
Certain NAL unit has a minimum byte size requirement. As such the hardware will optionally (enabled by SLICE STATE Command) determines the number of CABAC_ZERO_WORD to be inserted to the end of the current NAL, based on the minimum byte size of a NAL and the actual bin count of the encoded Slice. Since prior to the CABAC_ZERO_WORD insertion, the RBSP or EBSP is already byte-aligned, so each CABAC_ZERO_WORD insertion is actually a 3-byte sequence 0x00 00 03.
Context switch interrupt is not supported by this command.

DWord	Bit	Description

HCP_PAK_INSERT_OBJECT

0 — 31:29 — Command Type

Default Value:	3h PARALLEL_VIDEO_PIPE
Format:	OpCode

28:27 — Pipeline Type

Default Value:	2h
Format:	OpCode

26:23 — Media Instruction Opcode

Default Value:	7h Codec/Engine Name
Format:	OpCode

Codec/Engine Name = HCP = 7h

22:16 — Media Instruction Command

Default Value:	22h HCP_PAK_INSERT_OBJECT
Format:	OpCode

15:12 — Reserved

Access:	RO
Format:	MBZ

11:0 — Dword Length

Default Value:	[1h, FFFh] DWORD_COUNT_n
Format:	=n

(Excludes Dwords 0, 1) =Total Length - 2, soDWord Length = X, where X is in the size of the payload in DWs 2..n which has the range of [1,4095]

1 — 31 — Indirect Payload Enable

Format:	Enable

Only one of these two payload modes can be active at any time.
When Slice Size Conformance is enable the Payload(header) must be inline only so this bit set to MBZ.

Value	Name	Description
0	inline payload is used	
1	indirect payload is used	

Programming Notes
In VP9 scalability mode, tail should be inserted in inline mode only so this bit Must Be zero.

30:18 — Reserved

Access:	RO
Format:	MBZ

HCP_PAK_INSERT_OBJECT

17:16 **DataByteOffset - SrcDataStartingByteOffset[1:0]**

Format:	U2

Source Data Starting Byte Position within the very first inline DW.

15 **HeaderLengthExcludeFrmSize**

Format:	U1

In case this flag is on, bits are NOT accumulated during current access unit coding neither for Cabac Zero Word insertion bits counting or for output in MMIO register HCP_BITSTREAM_BYTECOUNT_FRAME_NO_HEADER.
When using HeaderLenghtExcludeFrmSize for header insertion, the software needs to make sure that data comes already with inserted start code emulation bytes. SW shouldn't set EmulationFlag bit (Bit 3 of DWORD1 of HCP_PAK_INSERT_OBJECT).

Value	Description
0	All bits accumulated
1	Bits during current call are not accumulated

Value	Name	Description
0	All bits accumulated	
1	Bits during current call are not accumulated	

Programming Notes
Must be set to 0 for JPEG encoder.

14 **Slice Header Indicator**

Format:	U1

This bit indicates if the insert object is a slice header. In the VDEnc mode, PAK only gets this command at the beginning of the frame for slice position X=0, Y=0. It internally generates the header that needs to be inserted per slice. For VDEnc mode, this bit should always be set.

Value	Name	Description
1	SLICE_HEADER	Insertion Object is a Slice Header. The command is stored internally by HW and is used for inserting slice headers.
0	LEGACY	Legacy Insertion Object command. The PAK Insertion Object command is not stored in HW.

Programming Notes
In VDENC mode, we support only Slice layer without partitioning RBSP syntax. The payload for PAK_INS_OBJ should contain only start code for Slice header followed by NAL_type and slice header (slice_header() in HEVC spec). The payload for PAK_INS_OBJ shouldn't contain CABAC Byte alignment bits. HW adds these alignment bits which are part of slice_data. Example PAK_INS_OBJ payload : 00 00 01 <NAL_type> <slice_header_Byte0> ..<slice_header_Byte LAST>

		Any zero_bytes that are added before slice header can be inserted by any preceding general PAK_INS_OBJ.
	13:8	**DataBitsInLastDW - SrCDataEndingBitInclusion[5:0]** Format: U6 Source Data to be included in the very last inline DW. Follows the MSBit is the upper bit of each byte within the DW. The lower byte is actually processed first.For example, SrCDataEndingBitInclusion = 9, bit 7:0 and bit 15 are included as valid header data. For VP9: The Driver has to give byte aligned Data for the last inline DW. (the driver pads Zeros to next byte boundary to the original header if it was not byte aligned on the last inline DW). Value: [1,32] — Name:
	7:4	**SkipEmulByteCnt - Skip Emulation Byte Count** Format: U4 Skip emulation check for number of starting bytes It can be programmed from 0 to 15 bytes. For example, to skip the start code that has already prefixed in the bitstream. Only valid for HEVC and reserved for VP9.
	3	**EmulationFlag - EmulationByteBitsInsertEnable** Format: U1 Only valid for HEVC and reserved for VP9.

Value	Name	Description
0		When this bit is set to 0, HW will disable the insertion of emulation bytes into the bitstream.
1		When this bit is set to 1, HW will enable the insertion of emulation bytes into the bitstream.

	2	**LastHeaderFlag - LastSrcHeaderDataInsertCommandFlag** Format: U1 To process a series of consecutive insertion commands, this flag (=1) indicates the current command isthe last 'header' insertion in the series. In CABAC, hardware must perform the "1" insert for byte alignfor Slice Header before Slice Data comes in in the next PAK-OBJECT command.
	1	**EndOfSliceFlag - LastDstDataInsertCommandFlag** Format: U1 No more insertion command and no more PAK-OBJECT command follows. Flush data out to memory.
	0	**Reserved** Access: RO Format: MBZ

<table>
<tr><th colspan="3">HCP_PAK_INSERT_OBJECT</th></tr>
<tr><td rowspan="2">2..n</td><td>127:0</td><td>Indirect Payload<table><tr><td>Exists If:</td><td>([Indirect Payload Enable]==1)</td></tr><tr><td>Format:</td><td>HCP_PAK_INSERT_OBJECT_INDIRECT_PAYLOAD</td></tr></table></td></tr>
<tr><td>31:0</td><td>Inline PayLoad<table><tr><td>Exists If:</td><td>([Indirect Payload Enable]==0)</td></tr><tr><td>Format:</td><td>U32</td></tr></table>Actual Data (inline) to be inserted to the output bitstream buffer.</td></tr>
</table>

HCP_PAK_OBJECT

HCP_PAK_OBJECT

Source: VideoCS

Length Bias: 2

This is a per-LCU command for encoder only.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 21h HCP_PAK_OBJECT Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 1h — Name: Value: 3h — Name:
1	31	**LastCtbOfSlice Flag** Format: Boolean Value: 0 — Name: False Value: 1 — Name: True

HCP_PAK_OBJECT

	30	**LastCtbOfTile Flag** Format: U1 Indicates last LCU or SB of a Tile
	29:24	**CU_count_minus1** Format: U6 Number of CUs in the current LCU = CU_count_minus1 + 1. Minimum, there must be 1 CU in a LCU.
	23:21	**Reserved** Access: RO Format: MBZ
	20:0	**split_coding_unit_flag[x0][y0]** Format: Split_coding_unit_flags If CU size=64x64 and DQP!=0, split should happen at least once
2	31:16	**Current LCU Y Addr** Format: U16
	15:0	**Current LCU X Addr** Format: U16
3	31:0	**Estimated LCU Size in Bytes** Format: U32 This parameter indicates the estimated LCU size in bytes. This is generated in the RD Estimator by adding the Coeff bytes of the LCU to the CU costs (pre-lambda input values).
4	31:18	**Reserved** Access: RO Format: MBZ
	17	**Reserved**
	16	**LcuForceZeroCoeff (Time Budget Overflow Occurred)** Format: Enable The bitstream packer pipeline will force all the coefficients to zero. Also, VDenc indicates PAK that time budget overflow has occurred.
	15:12	**SSE ClassID 32x32_3** Format: U4 This parameter indicates the SSE classID for the 32x32 block3 of the current LCU. Valid values: 0-8

HCP_PAK_OBJECT		
	11:8	**SSE ClassID 32x32_2** Format: U4 This parameter indicates the SSE classID for the 32x32 block2 of the current LCU. Valid values: 0-8
	7:4	**SSE ClassID 32x32_1** Format: U4 This parameter indicates the SSE classID for the 32x32 block1 of the current LCU. Valid values: 0-8
	3:0	**SSE ClassID 32x32_0** Format: U4 This parameter indicates the SSE classID for the 32x32 block1 of the current LCU. Valid values: 0-8

HCP_PALETTE_INITIALIZER_STATE

HCP_PALETTE_INITIALIZER_STATE

Source: VideoCS

Length Bias: 2

The HCP is selected with the**Media Instruction Opcode "7h"**for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

The HCP_PALETTE_INITIALIZER_STATE command loads in the SCC Palette Initializer Table to the HW.
Decoder only command.

Dword#2 - 193form a fixed size table for the Palette Initializer Table.
Max Palette Initializer Table is 128entries. Each entry has 3 components (Y, Cb and Cr) for a color.
Each component is 16-bits, even though currently only support up to 10-bit SCC extension. The upper (higher bits) 6 bits are set to zero - that is Least Significant Bit alignment.
Each entry of the Palette Initializer Table will consume 1.5 Dwords. Every two entries will consume 2 Dwords. Hence, total requires 96 Dwords.
Dword#2 Bit 31 Cb#0 15:0 Luma#0 15:0 Bit 0
Dword#3 Bit 31 Luma#115:0 Cr#015:0 Bit 0
Dword#4 Bit 31 Cr#115:0 Cb#115:0 Bit 0
Dword#2 corresponds to the entry# 0 of the Palette Initializer Table.
Dword#193correspondsto the entry# 127of the Palette Initializer Table.

Programming Notes

Palette Initialization needs to happen at the beginning of each frame/tiles or start of each independent slice. Palette initialization is not needed at the start of dependent slices (except the start of a new tiles since each tile needs to re-initialize the palette list) and the palette list is inherited from previous slice.
The following is the programming restriction:
(1) Palette Initialization command must be programmed in palette mode at the beginning of each frame and tiles (regardless if the slice is independent/dependent)and also the start of each independent slices.
(2) Palette Initialization command must not be programmed for dependent slices except the dependent slices are start of tiles (first slice in frame must be independent slice).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: Opcode
	28:27	**Pipeline Type** Default Value: 2h Format: Opcode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h

HCP_PALETTE_INITIALIZER_STATE

	22:16	**Media Instruction Command** Default Value: 9h HCP_PALETTE_INITIALIZER_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1) Value: C0h Name:
1	31:8	**Reserved** Access: RO Format: MBZ
	7:0	**Active Palette Initializer Table Entries** The number of entries in the Palette Initializer Table that is valid. The Palette Initializer Table always filled with only valid entries starting from entry 0 onwards, packed and no jump. Max allowed 128entries. This field is set to 0 if there is no active color entry.
2..97	3071:0	**First 64 Color Entries** This contains first 64 color entries (0 to 63), each with 16-bit Y, U, V entries. DW2 = [31:0]; DW3 = [63:32], etc for (i from 0 to 63) Palette Initializer Luma Value i = [(3 *i* 16 + 15) : (3 * i *16)] Palette Initializer Cb Value i = [((3 * i + 1) *16 + 15) : ((3 * i + 1) *16)] Palette Initializer CrValue i = [((3 * i + 2) *16 + 15) : ((3 * i + 2) *16)]
98..193	3071:0	**Second 64 Color Entries** This contains second64 color entries (64to 127), each with 16-bit Y, U, V entries. DW98= [31:0]; DW99= [63:32], etc for (i from 64to 127) Palette Initializer Luma Value i = [(3 * (i-64) * 16 + 15) : (3 * (i-64)*16)] Palette Initializer Cb Value i = [((3 * (i-64) + 1) *16 + 15) : ((3 * (i-64)+ 1) *16)] Palette Initializer CrValue i = [((3 * (i-64) + 2) *16 + 15) : ((3 * (i-64) + 2) *16)]

HCP_PIC_STATE

HCP_PIC_STATE

Source: VideoCS

Length Bias: 2

The HCP is selected with the **Media Instruction Opcode "7h"** for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

This is a picture level command and is issued only once per workload for both encoding and decoding processes.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 10h HCP_PIC_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 22h Name:
1	31:27	**Reserved** Access: RO Format: MBZ
	26:16	**FrameHeightInMinCbMinus1** Format: U11

Specifies the height of each decoded picture in units of minimum coding block size.

Programming Notes
The decoded picture height in units of luma samples equals • (FrameHeightInMinCbMinus1 + 1) * • (1 « (log2_min_coding_block_size_minus3 + 3)) • The maximum support picture size is 16384 pixels for both encoder and decoder.
The decoded picture height in units of luma samples equals (FrameHeightInMinCbMinus1 + 1) * (1 < (log2_min_coding_block_size_minus3 + 3)) The maximum support picture size is 16384 pixels.
In HEVC Encoder mode, the following restrictions apply. Note : for a frame width that is not an integer multiple of LCU : Kernel : on last LCU at frames right edge, it must align to CU boundary. This applies to all size of LCU: 16x16, 32x32, 64x64. Kernel does not count/include CUs outside of the picture in PakObj/CU_record respectively. Driver : sets up a LCU aligned (both in X/Y direction) surface.

15 **PAK Transform Skip Enable**

If global bit, **transform_skip_enabled_flag** in image state dw4.22 set to 1 and this bit set to,

1 -> HW will perform transform skip and over write the ENC decision in CU record

0-> HW will not perform transform skip

if **transform_skip_enabled_flag=0**, no transform skip performed in HW or ENC

Programming Notes
Encoder Only

14:11 **Reserved**

Access:	RO
Format:	MBZ

10:0 **FrameWidthInMinCbMinus1**

Format:	U11

Specifies the width of each decoded picture in units of minimum coding block size.

Programming Notes
The decoded picture width in units of luma samples equals (FrameWidthInMinCbMinus1 + 1) * (1 < (log2_min_coding_block_size_minus3 + 3)) The maximum support picture size is 16384pixels.
The picture width in units of luma samples equals • (FrameWidthInMinCbMinus1 + 1) * • (1 « (log2_min_coding_block_size_minus3 + 3)) • The maximum support picture size is 16384 pixels for both encoder and decoder.

HCP_PIC_STATE

2	31:29	**Chroma Subsampling** Specify the chroma subsampling of the current bitstream to be decoded or encoded. **Value** / **Name** / **Description** 0h / Reserved / This value is reserved for Monochrome Format which is not supported currently 1h / 4:2:0 / 4:2:0 Format 2h / 4:2:2 / 4:2:2 Format 3h / 4:4:4 / 4:4:4 Format
	28	**chroma_qp_offset_list_enabled_flag** **0** cu_chroma_qp_offset_flag is not present (default)**1** cu_chroma_qp_offset_flag is present. **Programming Notes** Only for 4:4:4 can it be program to non-zero. Decoder Only
	27:24	**diff_cu_chroma_qp_offset_depth** Difference between the luma coding tree block size and the minimum luma coding block size of coding units that convey cu_chroma_qp_offset_flag. Log2MinCuChromaQpOffsetSize = CtbLog2SizeY diff_cu_chroma_qp_offset_depth 0 to log2_diff_max_min_luma_coding_block_size default = 0. Decoder only feature
	23	**Reserved** Access: RO Format: MBZ
	22:20	**chroma_qp_offset_list_len_minus1** It is the index value i of the cb_qp_offset_list[i] and cr_qp_offset_list[i] syntax elements that are present in the PPS i = 0 to 5 6 to 7are invalid valid. default = 0 Decoder only feature
	19	**Reserved** Access: RO Format: MBZ
	18:16	**log2_sao_offset_scale_chroma** To scale SAO offset values for chroma samples. 0 to Max(0, BitDepth$_C$10) default = 0 Decoder Only

Value	Name
0	0 **[Default]**
1	1
2	2

15 **Reserved**

Access:	RO
Format:	MBZ

14:12 **log2_sao_offset_scale_luma**

To scale SAO offset values for luma samples

0 to Max(0, BitDepthC10)

Default = 0

Value	Name
0	0 **[Default]**
1	1
2	2

Programming Notes
Decoder only

11:10 **MaxPCMSize**

Format:	U2

Specifies the largest allowed PCM coding block size.

Value	Name
3	Reserved
2	32x32
1	16x16
0	8x8

9:8 **MinPCMSize**

Format:	U2

Specifies the smallest allowed PCM coding block size.

Value	Name
3	Reserved
2	32x32
1	16x16
0	8x8

7:6 **MaxTUSize**

Format:	U2

Specifies the largest allowed transform block size.

HCP_PIC_STATE

Value	Name
3	32x32
2	16x16
1	8x8
0	4x4

Programming Notes
IN VDENC mode this field should always be set to 3.

5:4 **MinTUSize**

Format:	U2

Specifies the smallest allowed transform block size.

Value	Name
3	32x32
2	16x16
1	8x8
0	4x4

Programming Notes
In VDENC mode this bit should be set to 0.

3:2 **CtbSize (LCUSize)**

Format:	U2

Specifies the coding tree block size.

Value	Name	Programming Notes
3	64x64	In VDENC, mode this should be set to 3.
2	32x32	
1	16x16	This can only be used when both picture width and height are fewer than or equal to 4222 pixels.
0	illegal/reserved	

Programming Notes
LCU is restricted based on the picture size. LCU 16x16 can only be used with picture width and height both fewer than or equal to 4222 pixels.

1:0 **MinCUSize**

Format:	U2

Specifies the smallest coding block size.

Value	Name
3	64x64

		2 — 32x32 1 — 16x16 0 — 8x8 **Programming Notes** In VDENC mode, This field should be programmed as 0.
3	31	**sps_range_extension_enable_flag** This flag represents both sps_extension_present_flag and sps_range_extension_flag flags in SPS. If enabled, sps_extension_present_flag=1 and sps_range_extension_flag = 0 or 1. If sps_range_extension_flag = 0, default values are set for range extension parameters. 0 no range extension1 range extension Decoder only feature
	30	**transform_skip_rotation_enabled_flag** 0 no rotation (default)1 apply rotation to the intra 4x4 residual data block coded in transform skip. Decoder only feature
	29	**transform_skip_context_enabled_flag** 0 no context select for parsing sig_coeff_flag with skipped transform (default)1 apply context select. Decoder only feature
	28	**implicit_rdpcm_enabled_flag** 0 the residual modification process is not used for intra blocks in the CVS (default)1 apply this modification process for intra blocks using a transform bypass. Decoder only feature
	27	**explicit_rdpcm_enabled_flag** 0 the residual modification process is not used for inter blocks in the CVS (default)1 apply this modification process for inter blocks using a transform bypass. Decoder only feature
	26	**intra_smoothing_disabled_flag** 0 filtering process of neighbouring samples is not disabled (default)1 filtering process of neighbouring samples is unconditionally disabled for intra prediction. Decoder only feature
	25	**persistent_rice_adaptation_enabled_flag** 0 no previous sub-block state is used in Rice parameter derivation (default)1 use mode dependent statistics accumulated from previous sub-blocks to derive Rice parameter. Decoder only feature
	24	**cabac_bypass_alignment_enabled_flag** 0 no CABAC alignment process is used prior to bypass decoding (default)1 CABAC alignment process is used prior to bypass decoding

	coeff_sign_flag[ ] and coeff_abs_level_remaining[ ]. Decoder only feature
23	**cross_component_prediction_enabled_flag** 0 disable cross component prediction (default)1 Enable cross component prediction. Only for 4:4:4 can it be program to non-zero. Decoder only feature
22:20	**Log2MaxTransformSkipSize** Equal to log2_max_transform_skip_block_size_minus2 + 2 It must less than or equal to Log2MaxTrafoSize. Decoder only feature
19	**High Precision Offsets Enable Flag** Specifies that weighted predicion offset values are signalled using a bit-depth-dependent precision. 0 - Disable1 - Enable **Programming Notes** Decoder only
18	**Reserved** Access: RO Format: MBZ
17:14	**Frame number**
13:8	**Tile number**
7:4	**Reserved** Access: RO Format: MBZ
3	**Reserved** Access: RO Format: MBZ
2	**InsertTestFlag** Format: U1 **Value** / **Name** 0h **[Default]** 1h **Programming Notes** CABAC 0 Word Insertion Test Enable (Encoder Only)This bit will modify CABAC K equation so that a positive K value can be generated easily. This is done for validation purpose only. In normal usage this bit should be set to 0. Regular equation for generating 'K' value when CABAC 0 Word Insertion Test Enable

is set to 0.

```
K = {[((96 * pic_bin_count()) - (RawMinCUBits * PicSizeInMinCUs *3) + 1023) / 1024] - bytes_in_picture} / 3
```

Modified equation when CABAC 0 Word Insertion Test Enable bit set to 1.

```
K = {[((1536 * pic_bin_count()) - (RawMinCUBits * PicSizeInMinCUs *3) + 1023) / 1024] - bytes_in_picture} / 3
```

Encoder only feature.

This bit should be set to 0 in VDENC mode.
This bit should be set to 0 in regular PAK mode.

	1	**CurPicIsI** Format: U1 Specifies that the current picture is comprised solely of I slices and that there are no P or B slices in the picture. **Value** / **Name** 0 – Current picture has at least one P or B slice **Programming Notes** This bit should be set to "0". Note: The value of "1" setting ("Current picture is comprised solely of I slices") is REMOVED. This bit is used for hardware optimization only. There is not enough information to set this bit to "1" correctly. In VDENC mode, this bit should be set to 0.
	0	**ColPicIsI** Format: U1 Specifies that the collocated picture is comprised solely of I slices and that there are no P or B slices in the picture. **Value** / **Name** 0 – Collocated picture has at least one P or B slice **Programming Notes** This bit should be set to "0". Note: The value of "1" setting ("Collocated picture is comprised solely of I slices") is REMOVED. This bit is used for hardware optimization only. There is not enough information to set this bit to "1" correctly.
4	31:28	**Reserved** Access: RO Format: MBZ
	27	**CU packet structure** **Value** / **Name** / **Description** 0 / VME / CU packet structure is in output format of HW VME which is 1/2 CL per CU

1	ExtEnc	CU packet structure is in output format of non HW VME (External Enc/SW) which is 1 CL per CU

Programming Notes
Encoder Only
Value Must be zero only. ExtEnc CU packet structure is not validated.
In VDENC mode, this bit should always be set to 0.

26 **strong_intra_smoothing_enable_flag**

Format:	U1

25 **transquant_bypass_enable_flag**

Format:	Enable

Value	Name	Description
0	Disable	cu_transquant_bypass is not supported
1	Enable	cu_transquant_bypass is supported

24 **Reserved**

Access:	RO
Format:	MBZ

23 **amp_enabled_flag**

Format:	Enable

Value	Name	Description
0	Disable	Asymmetric motion partitions cannot be used in coding tree blocks.
1	Enable	Support asymmetric motion partitions, i.e. PartMode equal to PART_2NxnU, PART_2NxnD, PART_nLx2N, or PART_nRx2N.

Programming Notes
In VDENC mode, this bit should be set to 1.

22 **transform_skip_enabled_flag**

Format:	Enable

Value	Name	Description
0	Disable	transform_skip_flag is not supported in the residual coding
1	Enable	transform_skip_flag is supported

21 **BottomField**

Format:	U1

HCP_PIC_STATE

Value	Name	Description
0	Bottom Field	
1	Top Field	

Programming Notes
Must be zero for encoder only

20 **FieldPic**

Format:	U1

Value	Name	Description
0	Video Frame	
1	Video Field	

Programming Notes
Must be zero for encoder only.
This must be programmed to "0" when pps_curr_pic_ref_enabled_flag (SCC IBC mode) is "1". To handle SCC IBC field mode, each field (top/bottom) should be decoded as separate frame and the two fields should be interlaced as a separate step.

19 **weighted_pred_flag**

Format:	U1

Programming Notes
In VDENC mode, this bit should be set to 1.

18 **weighted_bipred_flag**

Format:	U1

Programming Notes
In VDENC mode, this bit should be set to 1.

17 **tiles_enabled_flag**

Format:	U1

Programming Notes
This is an enable flag to enable tiling.
Only Screen-Extended Main 4:4:4 and Screen-Extended Main 4:4:4 10 allows boththtiles_enabled_flag andentropy_coding_sync_enabled_flag to be ON at the same time. All other current supproted profile only allowed one of them to be ON.

16 **entropy_coding_sync_enabled_flag**

Format:	U1

Not used in encoder mode

Programming Notes
Only Screen-Extended Main 4:4:4 and Screen-Extended Main 4:4:4 10 allows bothtiles_enabled_flag andentropy_coding_sync_enabled_flag to be ON at the same time. All other current supported profile only allowed one of them to be ON.

15 **loop_filter_across_tiles_enabled_flag**

Format:	U1

Programming Notes
For the encoder before, this bit must be set to zero (hardware restriction). Tile support is added, this field can be 0 or 1.
Must be turned off if Partial Frame Update Mode is enabled

14 **Reserved**

Access:	RO
Format:	MBZ

13 **sign_data_hiding_flag**

Format:	Enable

Value	Name	Description
0	Disable	Specifies that sign bit hiding is disabled.
1	Enable	Specifies that sign bit hiding is enabled.

Programming Notes
Currently not supported in encoder, so must be set to 0 for encoding session.

12:10 **log2_parallel_merge_level_minus2**

Format:	U3

Value	Name	Programming Notes
[0,4]	Valid Range	The value of log2_parallel_merge_level_minus2 shall be in the range of 0 to Log2CtbSizeYCtbLog2SizeY - 2, inclusive.

Programming Notes
For encoder, always set to 0 (Intel restriction).

9 **constrained_intra_pred_flag**

Format:	U1

8 **pcm_loop_filter_disable_flag**

Format:	U1

		Programming Notes In VDENC mode, this bit is always set to 1.
	7:6	**diff_cu_qp_delta_depth (or named as max_dqp_depth)** Format: U2 **Programming Notes** cu_qp_delta_enabled_flag = 1 and Max_DQP_Level = 0 or 3is supported in PAK standalone and VDenc modes. cu_qp_delta_enabled_flag/max_dqp_depth: 1/0: has cu qp delta. (cu depth <= max_dqp_depth) will have cu qp delta coded. Only allow QP change across Slices, no change across LCU or CU. In VDEnc mode, Max_DQP_Level can be 0 or 3.
	5	**cu_qp_delta_enabled_flag** Format: U1 Value / Name / Description: 0 / Disable / Does not allow QP change at CU or LCU level, the same QP is used for the entire slice. Max_DQP_Level = 0 (i.e. diff_cu_qp_delta_depath = 0). 1 / Enable / Allow QP change at CU level. MAX_DQP_level can be >0. **Programming Notes** cu_qp_delta_enabled_flag = 1 and Max_DQP_Level = 0 or 3 is supported for PAK standalone andVDEnc modes. In VDENC mode, this field should always be set to 1.
	4	**pcm_enabled_flag** Format: U1 **Programming Notes** In VDENC mode, this bit is always set to zero.
	3	**sample_adaptive_offset_enabled_flag** Format: U1 **Programming Notes** Restriction: HW does not support SAO filtering for LCU size 16x16 Should be set to 0 if slice_sao_luma_flag==0 and slice_sao_chroma_flag==0. It is illegal to program Slice_size_conformance ON when SAO is enabled in both first and second passes HW supports 8bit SAO and source format nv12 only. Must be off for Partial Frame Update Mode in encoder mode Restriction: HW does not support SAO filtering for LCU size 16x16

		Should be set to 0 if slice_sao_luma_flag==0 and slice_sao_chroma_flag==0. HW supports both 8 and 10b SAO in single pass
	2:0	**Reserved** Access: RO Format: MBZ
5	31	**Reserved**
	30	**Reserved** Access: RO Format: MBZ
	29:27	**bit_depth_luma_minus8** Format: U3 This specifies the number of bit allow for Luma pixels. In 8 bit mode, this must be set to 0. Encoder: Supports bit depths 8, 10 and 12 only. Encoder: Does not support 10 or 12 bit Source Pixels and 8bit PAK i.e. the source pixel depth should be less than or equal to PAK bit depth.

Value	Name	Description
0	luma_8bit	
1	luma_9bit	Only HEVC decoder supports 9 bits luma. HEVC encoder does not supports 9 bits luma.
2	luma_10bit	
3	luma_11bit	HEVC SCC does not support 11 bits Luma Also only HEVC decoder (non-SCC) support 11 bits Luma HEVC encoder (non-SCC) does not support 11 bits Luma
4	luma_12bit	HEVC SCC does not support 12bits Luma

Programming Notes
In VDENC mode, This field can only support a value of 0 or 2.

	26:24	**bit_depth_chroma_minus8** Format: U3 This specifies the number of bit allow for Chroma pixels. In 8 bit mode, this must be set to 0. Encoder: Supports bit depths 8, 10 and 12 only. And also it must be same as Luma. Encoder: Does not support 10 or 12 bit Source Pixels and 8bit PAK. i.e. The source pixel depth should be less than or equal to the PAK bit depth.

Value	Name	Description
0	chroma_8bit	
1	chroma_9bit	Only HEVC decoder supports 9 bits chroma. HEVC encoder does not supports 9 bits chroma.
2	chroma_10bit	
3	chroma_11bit	HEVC SCC does not support 11 bits chroma

			Also only HEVC decoder (non-SCC) support 11 bits chroma HEVC encoder (non-SCC) does not support 11 bits chroma
	4	chroma_12bit	HEVC SCC does not support 12bits Luma

Programming Notes
In VDENC mode, this field should match bits 29:27 of this DW.

23:20 **pcm_sample_bit_depth_luma_minus1**

Format:	U4

Description
Encoder: Supports bit depths 8, 10 and 12 for PCM. Must be same as pixel depth. 12bit not validated
Encoder supports 8, 10 and 12 bits. Must be same for luma and Chroma Cb/Cr. Decoder will support upto 12 bits.

19:16 **pcm_sample_bit_depth_chroma_minus1**

Format:	U4

Description
Bit depth must me same as luma for encoder.
Encoder support 8, 10 and 12 bits. Must be the same for luma and chroma Cb and Cr Decode will support upto 12 bits.

15:13 **max_transform_hierarchy_depth_inter(or named as tu_max_depth_inter)**

Format:	U3

Maximum TU split depths for inter blocks

Programming Notes
For encoder, alwasy set to 2 to allow max 2 levels of split. For more splitting rely on CU split to match the content (Intel restriction)

12:10 **max_transform_hierarchy_depth_intra (or named as tu_max_depth_intra)**

Format:	U3

Maximum TU split depth for intra blocks.

Programming Notes
For encoder, always set to 2 to allow max 2 levels of split. For more splitting, rely on CU splitto match the content (Intel restriction).

9:5 **pic_cr_qp_offset**

Format:	S4

Valid range -12 to +12

4:0 **pic_cb_qp_offset**

Format:	S4

Valid range -12 to +12

HCP_PIC_STATE

6

31:30 **Reserved**

Access:	RO
Format:	MBZ

29 **Load Slice Pointer Flag**

Format:	Enable

LoadBitStreamPointerPerSlice (Encoder-only)
To support multiple slice picture and additional header/data insertion before and after an encoded slice. When this field is set to 0, bitstream pointer is only loaded once for the first slice of a frame. For subsequent slices in the frame, bitstream data are stitched together to form a single output data stream. When this field is set to 1, bitstream pointer is loaded for each slice of a frame. Basically bitstream data for different slices of a frame will be written to different memory locations.

Value	Name	Description
0	Disable	Load BitStream Pointer only once for the first slice of a frame.
1	Enable	Load/reload BitStream Pointer only once for the each slice, reload the start location of thebitstream buffer from the Indirect PAK-BSE Object Data Start Address field.

Programming Notes
Must be zero for encoder

28:27 **Reserved**

Access:	RO
Format:	MBZ

26 **FrameSzUnderStatusEn - FrameBitRateMinReportMask**

Format:	Enable

This is a mask bit controlling if the condition of frame level bit count is less than FrameBitRateMin.

Value	Name	Description
0	Disable	Do not update bit 2 (Frame Bit Count Violate -- under run) of HCP_IMAGE_STATUS control register.
1	Enable	Set bit 2 (Frame Bit Count Violate -- under run) of HCP_IMAGE_STATUS control register if the total frame level bit counter is less than or equal to Frame Bit Rate Minimum limit. HW does not use this bit to set the bit in HCP_IMAGE_STATUS_CONTROL register. It's used pass the bit in HCP_IMAGE_STATUS_MASK register

Programming Notes
Encoder Only
In VDENC mode, set this bit to zero always.

HCP_PIC_STATE

25 **FrameSzOverStatusEn - FrameBitRateMaxReportMask**

Format:	Enable

This is a mask bit controlling if the condition of frame level bit count exceeds FrameBitRateMax.

Value	Name	Description
0	Disable	Do not update bit 1 of HCP_IMAGE_STATUS control register.
1	Enable	Set bit 1 of HCP_IMAGE_STATUS control register if the total frame level bit counter isgreater than or equal to Frame Bit Rate Maximum limit. HW does not use this bit to set the bit in HCP_IMAGE_STATUS_CONTROL register. It's used pass the bit inHCP_IMAGE_STATUS_MASK register

Programming Notes
Encoder Only
In VDENC mode, set this bit to zero always.

24 **LCUMaxBitStatusEn - LCUMaxSizeReportMask**

Format:	Enable

This is a mask bit controlling if the condition of any LCU in the frame exceeds LCUMaxSize.

Value	Name	Description
0	Disable	Do not update bit 0 of HCP_IMAGE_STATUS control register.
1	Enable	Set bit 0 of HCP_IMAGE_STATUS control register if the total bit counter for the current LCU is greater than the LCU Conformance Max size limit. HW does not use this bit to set the bit in HCP_IMAGE_STATUS_CONTROL register.

Programming Notes
Encoder Only

23:19 **Reserved**

Access:	RO
Format:	MBZ

18:17 **Reserved**

Access:	RO
Format:	MBZ

16 **NonFirstPassFlag**

Format:	Enable

This signals the current pass is not the first pass. It will imply designate HW behavior.

Value	Name	Description
0	Disable	If it is initial-Pass, this bit is set to 0.
1	Enable	For subsequent passes, this bit is set to 1.

Programming Notes
Encoder Only

15:0 **LCU Max BitSize Allowed**

Format:	U16

This field specifies the Max LCU Bit Size allowed in according to the spec conformance. However, driver can program a different value from the spec for other encoding purpose.

If LCU Limit exceeds, LCUMaxBitStatus in MMIO would be set to 1

If LCU Limit exceeds, LCUMaxBitStatus in MMIO would be set to 1 and also in CU Statistics Streamout, LCU Limit exceed field would be set to 1 at the last CU of a LCU

Programming Notes
Encoder Only For now, only the lower 18 bits are sufficient to support 8 to 12-bit 444. The extra upper bits are reserved for 16-bit pixel depth in future,

7 **Programming Notes:** Encoder Only

31 **FrameBitrateMaxUnit**

Format:	U1

This field is the Frame Bitrate Maximum Limit Units.

Value	Name	Description
0	Byte	32byte unit
1	Kilo Byte	4kbyte unit

Programming Notes
Encoder Only

30:14 **Reserved**

Access:	RO
Format:	MBZ

13:0 **FrameBitRateMax**

Format:	U14

This field is the Frame Bitrate Maximum Limit. This field along with FrameBitrateMaxUnit determines maximum allowed bits in a frame before multi-pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count exceeds this value.

		Value / Name / Description [0-16383] — WhenFrameBitrateMaxUnit =0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-512KBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit. [0-16383] — WhenFrameBitrateMaxUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-64MBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.
8	31	**FrameBitrateMinUnit** Format: U1 This field is the Frame Bitrate Minimum Limit Units. 0 — Byte — 32byte unit 1 — Kilo Byte — 4kbyte unit **Programming Notes** Encoder Only
	30:14	**Reserved** Access: RO Format: MBZ
	13:0	**FrameBitRateMin** Format: U14 This field is the Frame Bitrate Minimum Limit. This field along with FrameBitrateMinUnit determines minimum allowed bits in a frame before multi-pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count happen to be below this value. [0-16383] — WhenFrameBitrateMinUnit=0, this field is measured in unit of 32 bytes.The frame rate in bytes is range from 0-512KBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit. [0-16383] — WhenFrameBitrateMinUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-64MBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit. **Programming Notes** Encoder Only
9	31	**Reserved** Access: RO Format: MBZ

30:16 **FrameBitRateMaxDelta**

Exists If:	//Always
Format:	U15

This field is used to select the slice delta QP when FrameBitRateMax Is exceeded. It shares the sameFrameBitrateMaxUnit.

Value	Name	Description
0	**[Default]**	
[0-32767]		WhenFrameBitrateMaxUnit =0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-1024KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit.
[0-32767]		WhenFrameBitrateMaxUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-128MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit.

Programming Notes
Encoder Only

15 **Reserved**

Access:	RO
Format:	MBZ

14:0 **FrameBitRateMinDelta**

Format:	U15

This field is used to select the slice delta QP when FrameBitRateMin Is exceeded. It shares the sameFrameBitrateMinUnit.

Value	Name	Description
0	**Default]**	
[0-32767]		WhenFrameBitrateMinUnit =0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-1024KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit.
[0-32767]		WhenFrameBitrateMinUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-128MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit.

Programming Notes
HW requires the following condition: FrameBitRateMinDelta <= 2*FrameBitRateMinMust be true, otherwise it may cause unpredicted behavior.
Encoder Only

10..11 63:0 **FrameDeltaQpMax**

Format:	**FrameDeltaQp**

		Range: [0:MAX_QP_DELTA] Frame level delta QP which should be used in case FrameSize - FrameBitRateMax in the range of((DeltaQpMaxRange[n] * FrameBitRateMaxDelta»5)), DeltaQpMaxRange[n+1] * FrameBitRateMaxDelta»5)). **Programming Notes** If n == 7, DeltaQpMaxRange is infinity. Format: 8 bit sign-magnitude, 8bit-> Range: [0: 51] 10bit-> Range: [0:63] 12bit-> Range: [0:75] Encoder Only
12..13	63:0	**FrameDeltaQpMin** Format: FrameDeltaQp Range: [0:MIN_QP_DELTA] Frame level delta QP which should be used in case FrameSize - FrameBitRateMinin the range of ((DeltaQpMinRange[n] * FrameBitRateMinDelta»5)), DeltaQpMinRange[n+1] * FrameBitRateMinDelta»5)). **Programming Notes** If n == 7, DeltaQpMinRange is infinity. Format: 8 bit sign-magnitude 8bit-> Range: [-51: 0] 10bit-> Range: [-63:0] 12bit-> Range:[-75:0] Encoder Only
14..15	63:0	**FrameDeltaQpMaxRange** Format: FrameDeltaQpRange Range: [0:U8_MAX] Condition: FrameDeltaQpMaxRange[n] >= FrameDeltaQpMaxRange[n-1] This field is to calculate ranges for Frame level delta QP, specifically Frame level delta QP[n] and Frame level delta QP[n+1]. **Programming Notes** If n == 0, FrameDeltaQpMaxRange is zero. Encoder Only
16..17	63:0	**FrameDeltaQpMinRange** Format: FrameDeltaQpRange

		Range: [0:U8_MAX] Condition: FrameDeltaQpMinRange[n] >= FrameDeltaQpMinRange[n-1] This field is to calculate ranges for Frame level delta QP, specifically Frame level delta QP[n] and Frame level delta QP[n+1]. **Programming Notes** If n == 0, FrameDeltaQpMinRange is zero. Encoder Only
18	31:30	**MinFrameSizeUnits** Format: U2 This field is the Minimum Frame Size Units **Value / Name / Description** 0 / 4Kb / Minimum Frame Size is in 4Kbytes. 1 / 16Kb / Minimum Frame Size is in 16Kbytes. 2 / Compatibility Mode / Minimum Frame Size is in 4bytes 3 / 16 Bytes / Minimum Frame Size is 16 bytes. **Programming Notes** Encoder Only
	29:16	**Reserved** Access: RO Format: MBZ
	15:0	**MinFrameSize** Default Value: 0 Format: U16 Minimum Frame Size [15:0] (in Word, 16-bit)(Encoder Only) Minimum Frame Size is specified to compensate for intel Rate Control Currently zero fill (no need to perform emulation byte insertion) is done only to the end of the CABAC_ZERO_WORD insertion (if any) at the last slice of a picture. It is needed for CBR. Intel encoder parameter. The caller should always make sure that the value, represented by Minimum Frame Size, is always less than maximum frame size FrameBitRateMax. This field is reserved in Decode mode. **Programming Notes** Programmable range is 0..(2^16-1) * 2^12 when MinFrameSizeUnits is 0. (4KB unit) Programmable range is 0..(2^16-1) * 2^14 when MinFrameSizeUnits is 1. (16KB unit) Encoder Only
19	31	**Temporal MV pred disable** Indicates Temporal MV pred/Surface is disabled

This value should be set to 1 in Partial Frame Update Mode, to disable the Temporal MV prediction.

Programming Notes
This bit must be set to zero in encoder non-Partial Frame Update and decoder mode. SW must disable temporal MV in slice header for Partial Frame update Mode

30 **Partial Frame Update Mode**

Programming Notes
Loop filter across slices and tiles should be turned off SAO filter should be turned off for entire frame Last row of a Tile can't be skip unless the Tile has all skip rows Last row of a Frame can't be skip unless the Frame has all skip rows and Tiling disabled for that frame (This frame level restriction does not apply when Tiling enabled) Temporal MV prediction should be off No Multipass allowed Note: A Tile row can have skip and non-Skip tiles Any Tile row can be Skip in a frame VDenc mode only

29:28 **Reserved**

Access:	RO
Format:	MBZ

27:26 **NumberOfLCUsInNormal Slice size conformance Mode**

Format:	U2

Indicates Max number of LCUs can be issued in Normal Mode of SliceSizeConformance. This is a performance feature to keep HW pipeline full.

Programming Notes
Encoder Only

25 **Reserved**

Access:	RO
Format:	MBZ

24 **SSE Enable**

Format:	Enable

This field indicates if SSE statistics generation is enabled in the PAK. If enabled, the classID in the CU packet is used along with the SSE thresholds in the picture state to generate the SSE frame statistics. When enabled, the PAK will also write and read source pixels (similar to the reference pixels in the LCU ILDB streamout) to generate the SSE distortion.
This parameter is valid to both HEVC and VP9. In HEVC, it is valid for LCU64 and LCU32 modes.
The global streamout flag (PAK Pipeline streamout enable) must be enabled if this bit is

set to 1
Encoder only

Programming Notes
Valid only in VDenc mode. In VDENC mode this bit should be set to one.
In Partial Frame Update mode, This feature can't be enabled in VDENC mode.

23:18 **slice_pic_parameter_set_id**

Format:	U6

This parameter is used in Dynamic slice mode when inserting slice headers. slice_pic_parameter_set_id specifies the value of pps_pic_parameter_set for the PPS in use. The value of slice_pic_parameter_set_id shall be in the range of 0 to 63, inclusive.

Programming Notes
Encoder Only

17 **NalUnitTypeFlag**

Format:	U1

This parameter is used in Dynamic slice mode when inserting slice headers.
(derived as (nal_unit_type >= BLA_W_LP && nal_unit_type <= RSV_IRAP_VCL23)

Programming Notes
Encoder Only

16 **first_slice_segment_in_pic_flag**

Format:	U1

This parameter is used in Dynamic slice mode when inserting slice headers. first_slice_segment_in_pic_flag equal to 1 specifies that the slice segment is the first slice segment of the picture in decoding order. first_slice_segment_in_pic_flag equal to 0 specifies that the slice segment is not the first slice segment of the picture in decoding order.
Note: This bit must be always1 so HW can update in subsequent slices.

Programming Notes
Encoder Only

15 **no_output_of_prior_pics_flag**

Format:	U1

This parameter is used in Dynamic slice mode when inserting slice headers. no_output_of_prior_pics_flag affects the output of previously-decoded pictures in the decoded picture buffer after the decoding of an IDR or a BLA picture that is not the first picture in the bitstream

Programming Notes
Encoder Only

14 **PAK Dynamic Slice Mode Enable**

Format:	Enable

This field controls whether PAK will check will check the Slice Size Threshold in Bytes

parameter for bits overflow and terminate the slice. This mode is called Dynamic Slice Mode.
When this field is disabled, PAK is in Static Slice Mode. It uses the LastCtbOfSlice Flag in the HCP_PAK_OBJECT to terminate the slice.
Note: HW infer "dependent_slice_segments_enabled_flag" to be 0 whenPAK Dynamic Slice Mode Enable = 1

Value	Name
0h	Disable [Static Slice Mode]
1h	Enable [Dynamic Slice Mode]

Programming Notes
The max Tile Size allowed is 5k when Partial Frame Update Mode and Dynamic Slice Modeare enabled This bit can be set to '1' only for HEVC codec. Encoder Only

13:8 **RhoDomainFrameLevelQP**

Format:	U6

This QP is used for RhoDomain Frame level statistics

Programming Notes
Encoder Only

7 **Fractional QP adjustment enable**

Format:	Enable

Programming Notes
Enables Fracitonal QP adjustment feature in PAK. Valid for LCU sizes 64x64 and 32x32 only. Must be disabled for LCU 16x16 If enabled then cu_qp_delta_enable_flag must to be set to 1 Encoder Only
In VDENC mode, set this bit to 1.

6 **RhoDomain Rate Control Enable**

Format:	Enable

This field indicates if RhoDomain statistics generation is enabled in the PAK. The RhoDomain AverageMacroblockQP is used by the PAK along with the CU QP to compute the statistics. The global streamout flag (**PAK Pipeline Streamout Enable**) needs to be enabled when RhoDomain Rate Control is enabled in-order to stream out the rho-domain statistics..
RhoDomain statistics will have non_zero coeffs when Force_zero_coeff feature enabled

		Programming Notes Encoder Only In VDENC mode, set this bit to 1.
	5:3	**Fractional QP Offset** Format: U3 Start segment from the top of the Frame where increased Quantization parameter is added. Within two 64x64 LCUs/four 32x32 LCU rows, the total number LCUs are equally split into 8 segments (i.e.each segment has (FrameWidthInLCU « (4 LCUSize) + 7)»3) LCUs). **Programming Notes** Encoder Only
	2:0	**Fractional QP Input** Format: U3 In a set of 8 segments, Qp is incremented by 1 for F_qp segments. **Programming Notes** Encoder Only
20	31:7	**Reserved** Access: RO Format: MBZ
	6	**IntraTuCountBasedRDOQdisable** **Description** When this field is set (disabled), RDOQ will be disabled based on whether intraTU counting (accumulated at a 4x4 level for all TU sizes -- 4x4/8x8/16x16/32x32) is greater than the threshold.Frame level RDOQ enable flag must be set if this is set. When it is unset (i.e. 0), RDOQ enable/disable is decided based on frame level RDOQ enable flag.
	5:0	**Reserved** Access: RO Format: MBZ
21	31:0	**Slice Size Threshold in Bytes** Format: U32 When a slice exceeds this value in bytes, hardware will end current slice and insert a new slice boundary. **Programming Notes** Slice Size Threshold in Bytes cannot be less than 600 Bytes when Partial Frame Update mode is ON Encoder Only
22	31:0	**Target Slice Size in Bytes** Format: U32

HCP_PIC_STATE

		This field indicates the target slice size in Bytes for slice size conformance feature. When the actual slice size exceeds "Target slice size in Bytes", HW sets "Slice Overflow Occurred" bit in the PAK Frame Statistics. **Programming Notes** Encoder Only
23	31:16	**Class0_SSE_Threshold1** Format: U16 This field specifies the upper bound threshold for Class0 Zone1 to classify the per-4x4sblk SSE statistics. Class0_SSE_Threshold_0 < per-4x4sblk SSE <= Class0_SSE_Threshold_1 fall under Class0 Zone1. Class0_SSE_Threshold_1 < per-4x4sblk SSE fall under Class0 Zone2. **Programming Notes** Encoder Only
	15:0	**Class0_SSE_Threshold0** Format: U16 This field specifies the upper bound threshold for Class0 Zone0 to classify the per-4x4sblk SSE statistics. per-4x4sblk SSE <= Class0_SSE_Threshold_0 fall under Class0 Zone0. **Programming Notes** Encoder Only
24..31	255:0	**SSE thresholds for Class1-8** Format: U256 SSE thresholds for Class 1-8, see DW 21 (SSE Class 0 thresholds) for format. **Programming Notes** Encoder Only
32	31:30	**Reserved** Access: RO Format: MBZ
	29:25	**cb_qp_offset_list[5]** Format: S4 Offsets used in the derivation of Qp'_{Cb} `Sign + 4bit (total 5-bit for each index), range +/- 12 ;` (default 0) **Programming Notes** Decoder Only
	24:20	**cb_qp_offset_list[4]** Format: S4 Offsets used in the derivation of Qp'_{Cb}

		Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0) **Programming Notes** Decoder Only
	19:15	**cb_qp_offset_list[3]** Format: S4 Offsets used in the derivation of Qp'_{Cb}, Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0) **Programming Notes** Decoder Only
	14:10	**cb_qp_offset_list[2]** Format: S4 Offsets used in the derivation of Qp'_{Cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0) **Programming Notes** Decoder Only
	9:5	**cb_qp_offset_list[1]** Format: S4 Offsets used in the derivation of Qp'_{Cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0) **Programming Notes** Decoder Only
	4:0	**cb_qp_offset_list[0]** Format: S4 Offsets used in the derivation of Qp'_{Cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0) **Programming Notes** Decoder Only
33	31:30	**Reserved** Access: RO Format: MBZ
	29:25	**cr_qp_offset_list[5]** Format: S4 Offsets used in the derivation of Qp'_{Cb}

HCP_PIC_STATE

		Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0) **Programming Notes** Decoder Only
	24:20	**cr_qp_offset_list[4]** Format: S4 Offsets used in the derivation of Qp'_{Cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0) **Programming Notes** Decoder Only
	19:15	**cr_qp_offset_list[3]** Format: S4 Offsets used in the derivation of Qp'_{Cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0) **Programming Notes** Decoder Only
	14:10	**cr_qp_offset_list[2]** Format: S4 Offsets used in the derivation of Qp'_{Cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0) **Programming Notes** Decoder Only
	9:5	**cr_qp_offset_list[1]** Format: S4 Offsets used in the derivation of Qp'_{Cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0) **Programming Notes** Decoder Only
	4:0	**cr_qp_offset_list[0]** Format: S4 Offsets used in the derivation of Qp'_{Cb} Sign + 4bit (total 5-bit for each index), range +/- 12 ; (default 0)

HCP_PIC_STATE

		Programming Notes Decoder Only
34	31	**intra_boundary_filtering_disabled_flag** This specifies that the intra boundary filtering process is unconditionally disabled for intra prediction. Decoder only (Encoder default to "0")
	30:29	**motion_vector_resolution_control_idc** This controls the presense and inference of the use_integer_mv_flag that specifies the resolution of motion vectors for inter prediction. Decoder only (Encoder default to "00")

Value	Name	Description
0	No integer MV for the frame	
1	Only integer MV for the frame	
2	Adaptive integer MV for the frame	Slice signal use_inter_mv_flag will indicate if the slice will use interger MV or not
3	Reserved	

	28	**pps_curr_pic_ref_enabled_flag** This bit specifies that a picture referring to the PPS may be included in a reference picture list of a slice of the picture itself. This applies to both decoder and encoder
	27	**residual_adaptive_colour_transform_enabled_flag** This bitspecifies that an adaptive colour transform may be applied to the residual in the decoding process. When ChromaArrayType is not equal to 3, residual_adaptive_colour_transform_enabled_flag shall be equal to 0. Decoder only (Encoder default to "0")
	26	**pps_slice_act_qp_offsets_present_flag** This bitspecifies that slice_act_y_qp_offset, slice_act_cb_qp_offset, slice_act_cr_qp_offset are present in the slice header. Decoder only (Encoder default to "0")
	25:20	**pps_act_y_offset_plus5** This is used to determine the offsets that are applied to the quantization parameter values qP derived in clause8.6.2 for the luma, Cb and Cr components, respectively, when tu_residual_act_flag[xTbY][yTbY] is equal to 1. The variable PpsActQpOffsetY is set equal to pps_act_y_qp_offset_plus55. It is a requirement of bitstream conformance that the values of PpsActQpOffsetYshall be in the range of 12 to +12, inclusive. [Valid Range is from -7 to 17] Decoder only (Encoder default to "0")
	19:14	**pps_act_cb_qp_offset_plus5** This is used to determine the offsets that are applied to the quantization parameter

HCP_PIC_STATE

		values qP derived in clause8.6.2for the luma, Cb and Cr components, respectively, when tu_residual_act_flag[xTbY][yTbY] is equal to 1. The variable PpsActQpOffsetCb is set equal to pps_act_cb_qp_offset_plus55. It is a requirement of bitstream conformance that the values of PpsActQpOffsetCbshall be in the range of 12 to +12, inclusive. [Valid Range is from -7 to 17] Decoder only (Encoder default to "0")
	13:8	**pps_act_cr_qp_offset_plus3** This is used to determine the offsets that are applied to the quantization parameter values qP derived in clause8.6.2for the luma, Cb and Cr components, respectively, when tu_residual_act_flag[xTbY][yTbY] is equal to 1. The variable PpsActQpOffsetCr is set equal to pps_act_cb_qp_offset_plus33. It is a requirement of bitstream conformance that the values of PpsActQpOffsetCrshall be in the range of 12 to +12, inclusive. [Valid Range is from -9to 15] Decoder only (Encoder default to "0")
	7	**pps_deblocking_filter_disabled_flag** pps_deblocking_filter_disabled_flag equal to 1 specifies that the operation of deblocking filter is not applied for slices referring to the PPS in which slice_deblocking_filter_disabled_flag is not present. pps_deblocking_filter_disabled_flag equal to 0 specifies that the operation of the deblocking filter is applied for slices referring to the PPS in which slice_deblocking_filter_disabled_flag is not present.
	6	**deblocking_filter_override_enabled_flag** deblocking_filter_override_enabled_flag equal to 1 specifies the presence of deblocking_filter_override_flag in the slice headers for pictures referring to the PPS. deblocking_filter_override_enabled_flag equal to 0 specifies the absence of deblocking_filter_override_flag in the slice headers for pictures referring to the PPS.
	5:3	**Reserved** Access: RO Format: MBZ
	2:0	**IBC Motion Compensation Buffer Reference IDC** This indicates which one of the eight Motion Compensation buffer is mapped to current picture for IBC. Motion Compensation will use this picture to write and read IBC data This applies to both decoder and encoder
35	31	**palette_mode_enabled_flag** This specifies that the palette mode may be used for intra blocks. This applies to both decoder and encoder
	30	**monochrome_palette_flag** **Description** This specifies that the pictures that refer to this PPS are monochrome. This must be set to 0.

Decoder only [Encoder default to 0]

29:28 **IBC configuration**

IBC configuration is used configure Intra block copy.
- Disable Intra block copy.
- Limit Intra block copy from Left blocks only.
- Allow full range of Intra block copy as specified in spec.

Value	Name	Description	Programming Notes
3		When IBC configuration in VDENC mode is set to 3, Intra block search includes top and left regions. In decoder mode, When SCC is enabled this field should be set to 3.	
2			This is invalid value.
1		When IBC configuration in fixed function encoder (VDENC) mode is set to 1, Intra block search includes only left region.	
0	**[Default]**	When IBC configuration is 0, intra block copy is disabled and it applies for both Fixed function encoder and decoder.	

27:24 **luma_bit_depth_entry_minus8**

This specifies the bit depth of the luma component of the entries of the palette predictor initializer BitDepthEntryY as follows:
BitDepthEntryY=8+luma_bit_depth_entry_minus8
It is a requirement of bitstream conformance that the value of BitDepthEntryY shall be equal to the BitDepthY of the SPS referred to by this PPS.
Valid Range is from 0 to 4.
This applies to both decoder and encoder

23:20 **chroma_bit_depth_entry_minus8**

This specifies the bit depth of the chroma components of the entries of the palette predictor initializer BitDepthEntryC as follows:
BitDepthEntryC=8+chroma_bit_depth_entry_minus8
It is a requirement of bitstream conformance that the value of BitDepthEntryC shall be equal to BitDepthC of the SPS referred to by this PPS.
Valid Range is from 0 to 4.
This applies to both decoder and encoder

19 **IBC Motion Vector Error Handling Disable**

Illegal IBC motion vector (pointing to future) can cause hang to HW. Error handling is needed to avoid hang. This bit disable the error handling if needed

18:17 **Reserved**

Access:	RO
Format:	MBZ

HCP_PIC_STATE

	16:10	**delta_palette_max_predictor_size** This specifies the difference between the maximum allowed palette predictor size and the maximum allowed palette size. It is a requirement of bitstream conformance that the value of delta_palette_max_predictor_size shall be equal to 0 when palette_max_size is equal to 0. Valid value is from 0 to 64 PaletteMaxPredictorSize = palette_max_size + delta_palette_max_predictor_size This applies to both decoder and encoder PaletteMaxPredictorSize will be upto 96 for encoder
	9:7	**Reserved** Access: RO Format: MBZ
	6:0	**palette_max_size** This specifies the maximum allowed palette size. Valid value is from 0 to 64 This applies to both decoder and encoder. Always 64 for encoder
36	31:0	**Reserved** Access: RO Format: MBZ
37	31:16	**Reserved**
	15:0	**RDOQIntraTUThreshold** **Description** This parameter programs the threshold for the Intra Counting. This value is programmed as a multiple of 16. When checking against the IntraTU Count, this value needs to be « 4 (multiplied by 16) and then compared against the IntraTU count to trigger the condition for Intra RDOQ disable. The IntraTU Count is accumulated in units of 4x4.
38	31:16	**RDOQIntra32x32TUThreshold**
	15:0	**RDOQIntra16x16TUThreshold**
39..40	63:0	**SSEThresholds for Class9 ..10** SSE thresholds for Class 9-10, see DW 21 (SSE Class 0 thresholds) for format

HCP_PIPE_BUF_ADDR_STATE

HCP_PIPE_BUF_ADDR_STATE

Source: VideoCS

Length Bias: 2

The HCP is selected with the **Media Instruction Opcode "7h"** for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

This state command provides the memory base addresses for the row store buffer and reconstructed picture output buffers required by the HCP.
This is a picture level state command and is shared by both encoding and decoding processes.

Programming Notes

All pixel surface addresses must be 4K byte aligned. There is a max of 8 Reference Picture BufferAddresses, and all share the same third address DW in specifying 48-bit address.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 2h HCP_PIPE_BUF_ADDR_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value / Name: 66h; 72h

HCP_PIPE_BUF_ADDR_STATE

1..2	63:0	**Decoded Picture** Format: SplitBaseAddress4KByteAligned Frame buffer address for the final decoded picture YUV output.
3	31:0	**Decoded Picture Memory Address Attributes** Format: MemoryAddressAttributes
4..5	63:0	**Deblocking Filter Line Buffer** Format: SplitBaseAddress64ByteAligned Base address of the filter line buffer (read/write) used by the Deblocking Filter.
6	31:0	**Deblocking Filter Line Buffer Memory Address Attributes** Format: MemoryAddressAttributes
7..8	63:0	**Deblocking Filter Tile Line Buffer** Format: SplitBaseAddress64ByteAligned Base address of the tile line buffer (read/write) used by the Deblocking Filter.
9	31:0	**Deblocking Filter Tile Line Buffer Memory Address Attributes** Format: MemoryAddressAttributes
10..11	63:0	**Deblocking Filter Tile Column Buffer** Format: SplitBaseAddress64ByteAligned Base address of the tile column buffer (read/write) used by the Deblocking Filter.
12	31:0	**Deblocking Filter Tile Column Buffer Memory Address Attributes** Format: MemoryAddressAttributes
13..14	63:0	**Metadata Line Buffer** Format: SplitBaseAddress64ByteAligned Base address for the Metadata Line buffer.
15	31:0	**Metadata Line Buffer Memory Address Attributes** Format: MemoryAddressAttributes
16..17	63:0	**Metadata Tile Line Buffer** Format: SplitBaseAddress64ByteAligned Base address for the Metadata Tile Line buffer.
18	31:0	**Metadata Tile Line Buffer Memory Address Attributes** Format: MemoryAddressAttributes
19..20	63:0	**Metadata Tile Column Buffer** Format: SplitBaseAddress64ByteAligned Base address for the Metadata Tile Column buffer.

HCP_PIPE_BUF_ADDR_STATE

21	31:0	**Metadata Tile Column Buffer Memory Address Attributes** Format: **MemoryAddressAttributes**
22..23	63:0	**SAO Line Buffer** Format: **SplitBaseAddress64ByteAligned** Base address for the SAO Line buffer.
24	31:0	**SAO Line Buffer Memory Address Attributes** Format: **MemoryAddressAttributes**
25..26	63:0	**SAO Tile Line Buffer** Format: **SplitBaseAddress64ByteAligned** Base address for the SAO Tile Line buffer.
27	31:0	**SAO Tile Line Buffer Memory Address Attributes** Format: **MemoryAddressAttributes**
28..29	63:0	**SAO Tile Column Buffer** Format: **SplitBaseAddress64ByteAligned** Base address for the SAO Tile Column buffer.
30	31:0	**SAO Tile Column Buffer Memory Address Attributes** Format: **MemoryAddressAttributes**
31..32	63:0	**Current Motion Vector Temporal Buffer** Format: **SplitBaseAddress64ByteAligned** Base address for the Current Motion Vector Temporal buffer.
33	31:0	**Current Motion Vector Temporal Buffer Memory Address Attributes** Format: **MemoryAddressAttributes**
34..35	63:0	**Reserved** Access: RO Format: MBZ
36	31:0	**Reserved** Access: RO Format: MBZ
37..52	511:0	**Reference Picture Base Address (RefAddr[0-7])** Format: **SplitBaseAddress64ByteAligned[8]** Base address of the reference picture buffer. **Programming Notes** This Note is applicable in VDENC mode in IBC enabled case.

HCP_PIPE_BUF_ADDR_STATE

		NumRefIdxL0_minus1 in VDENC defines number of references used for encoding the frame. When IBC is enabled, the base address corresponding to current non-deblocked reconstructed picture should programmed to Reference Picture Base Address[NumRefIdxL0_minus1]
53	31:0	**Reference Picture Base Address Memory Address Attributes** Format: **MemoryAddressAttributes** **Programming Notes** Reference Picture Memory Compression Enables and Types (Media/Render) are moved to HCP_SURFACE_STATE. Separate8 Memory Compression Enables and Types are added in HCP_SURFACE_STATE so each reference picture surfaces have its own separate bits. The memory compression enable and type bit in this DW are not used.
54..55	63:0	**Original Uncompressed Picture Source** Format: **SplitBaseAddress64ByteAligned** Buffer address for fetching YUV pixel data from the original uncompressed input picture for encoding. This value is only valid in **encoding** mode.
56	31:0	**Original Uncompressed Picture Source Memory Address Attributes** Format: **MemoryAddressAttributes**
57..58	63:0	**Streamout Data Destination** Exists If: //Decoder Only Format: **SplitBaseAddress64ByteAligned** **Description** Buffer address for outputting the per-block indirect data to memory when **StreamOutEnable** is set in the HCP_PIPE_MODE_SELECT command. Decoder does not use this buffer. For Encoder: this field is used for dynamic repeat of frame in PAK for Rate Control. Also used for feeding coding information back to the Host, Video Preprocessing Unit and ENC Unit. For Encoder: This surface is used to streamout CU records All data are written in fixed formats, and therefore all record sizes are known in the hardware. Hardware can calculate the offset into this base address for per-block data.
59	31:0	**Streamout Data Destination Memory Address Attributes** Exists If: //Decoder Only Format: **MemoryAddressAttributes**
60..61	63:0	**Decoded Picture Status/Error Buffer Base Address or Encoded slice size streamout Base Address** Format: **SplitBaseAddress64ByteAligned** **Decoder Mode :** Specifies the 64 byte aligned buffer address for writing a single status/error cache-line sized record into memory when the Pic Status/Error Report Enable is set in the

HCP_PIPE_BUF_ADDR_STATE		
		HCP_PIPE_MODE_SELECT command. The pic status/error record is written by hardware after the picture is decoded. The content of this memory location can later be read by the driver only. **Encoder Mode:** This specifies 64 byte aligned buffer address for writing Slice size, when slice size conformance is enabled.
62	31:0	**Decoded Picture Status/Error Buffer Base Address Memory Address Attributes** Format: **MemoryAddressAttributes**
63..64	63:0	**LCU ILDB Streamout Buffer** Format: **SplitBaseAddress64ByteAligned** Buffer address for writing ILDB parameter per LCU to memory when Deblocker Streamout Enable is set in the HCP_PIPE_MODE_SELECT Command. The ILDB MB control parameters are written by HW at the end of each reconstructed LCU. Only edge information is being streamed out.
65	31:0	**LCU ILDB Streamout Buffer Memory Address Attributes** Format: **MemoryAddressAttributes**
66..81	511:0	**Collocated Motion Vector Temporal Buffer[0-7]** Format: **SplitBaseAddress64ByteAligned[8]** Base address for the Collocated Motion Vector Temporal buffer.
82	31:0	**Collocated Motion Vector Temporal Buffer[0-7] Memory Address Attributes** Format: **MemoryAddressAttributes**
83..84	63:0	**VP9 Probability Buffer Read/Write** Format: **SplitBaseAddress64ByteAligned** Specifies the 64 byte aligned buffer address for VP9 Probability Buffer. Hardware reads in the probability for decode and write out the modified probability for future frames. Driver needs to program the Initial VP9 Probability for decoding the current frame. For Key Frame, it should contain the default Key Frame Probability. For non-Key Frame, it could be a default (non-Key) or one of the 8 Reference Buffers Probability. Driver must provide a valid Initial VP9 Probability buffer.
85	31:0	**VP9 Probability Buffer Read/Write Memory Address Attributes** Format: **MemoryAddressAttributes**
86..87	63:0	**VP9 Segment ID Buffer Read/Write** Specifies the 64 byte aligned buffer address for VP9 SegmentID buffer. This should contain the writeout SegmentID from previous frame and will be used to predict SegmentID for the current frame. Hardware will write out SegmentID of the current frame in the same address for the next frame.
88	31:0	**VP9 Segment ID buffer Read/Write Memory Address Attributes** Format: **MemoryAddressAttributes**

HCP_PIPE_BUF_ADDR_STATE

89..90	63:0	**VP9 HVD Line Rowstore Buffer Read/Write** Format: **SplitBaseAddress64ByteAligned** Specifies the 64 byte aligned buffer address for HVD Tile Rowstore Buffer (bitstream decoder).
91	31:0	**VP9 HVD Line Rowstore buffer Read/Write Memory Address Attributes** Format: **MemoryAddressAttributes**
92..93	63:0	**VP9 HVD Tile Rowstore Buffer Read/Write** Format: **SplitBaseAddress64ByteAligned**
94	31:0	**VP9 HVD Tile Rowstore buffer Read/Write Memory Address Attributes** Format: **MemoryAddressAttributes**
95..96	63:0	**SAO Rowstore Buffer Base Address** Specifies the 64 byte aligned buffer address for Rowstoring of SAO parameters in encoder mode
97	31:0	**SAO Rowstore Buffer Read/Write Memory Address Attributes** Format: **MemoryAddressAttributes**
98..99	63:0	**Frame Statistics Streamout Data Destination Buffer Base Address** Format: **SplitBaseAddress64ByteAligned** **Description** Specifies the 64 byte aligned buffer address for outputting the frame statistics data to memory. The statistics are mainly SliceSize conformance, SSE, RhoDomain and CU parameters. Decoder does not use this buffer.
100	31:0	**Frame Statistics Streamout Data Destination buffer (attributes) Read/Write** Format: **MemoryAddressAttributes**
101..102	63:0	**SSE Source Pixel RowStore Buffer Base Address** Format: **SplitBaseAddress64ByteAligned** Specifies the 64 byte aligned buffer address for storing the source pixels for SSE. SSE metrics in the PAK are computed using post loop-filtered pixels or post SAO, if SAO is enabled.
103	31:0	**SSE Source Pixel RowStore buffer (attributes) Read/Write** Format: **MemoryAddressAttributes**
104..105	63:0	**HCP Scalability Slice State Buffer Base Address** Format: **SplitBaseAddress64ByteAligned** Specifies the 64 byte aligned buffer address for storing slice state information on HEVC/VP9 Scalable mode for decode. This is needed since CABAC and BE pass will be separated and BE pass needs to have slice state information as well. This buffer is only used in HEVC Scalable Decode Mode Only (Virtual Tile on both CABAC and Recon Pass)

HCP_PIPE_BUF_ADDR_STATE		
106	31:0	**HCP Scalability Slice State Buffer (attributes) Read/Write** Format: **MemoryAddressAttributes**
107..108	63:0	**HCP Scalability CABAC Decoded Syntax Elements Buffer Base Address** Format: **SplitBaseAddress64ByteAligned** Specifies the 64 byte aligned buffer address for storing CABAC Decoded Syntax Element on HEVC/VP9 Scalable mode for decode
109	31:0	**HCP Scalability CABAC Decoded Syntax Elements Buffer (attributes) Read/Write** Format: **MemoryAddressAttributes**
110..111	63:0	**Motion Vector Upper Right Column Store Buffer Base Address** Format: **SplitBaseAddress64ByteAligned** Specifies the 64 byte aligned buffer address for storing upper right Motion Vector on HEVC /VP9 Scalable mode for decode in BE pass. This buffer is used to pass data across pipes for multiple pipe mode. This buffer is only used on HEVC Scalable Decode Only (Virtual Tile on both CABAC and Recon pass)
112	31:0	**Motion Vector Upper Right Column Store Buffer (attributes) Read/Write** Format: **MemoryAddressAttributes**
113..114	63:0	**Intra Prediction Upper Right Column Store Buffer Base Address** Format: **SplitBaseAddress64ByteAligned** Specifies the 64 byte aligned buffer address for storing upper right Intra Prediction Pixel on HEVC /VP9 Scalable mode for decode in BE pass. This buffer is used to pass data across pipes for multiple pipe mode.
115	31:0	**Intra Prediction Upper Right Column Store Buffer (attributes) Read/Write** Format: **MemoryAddressAttributes**
116..117	63:0	**Intra Prediction Left Recon Column Store Buffer Base Address** Format: **SplitBaseAddress64ByteAligned** Specifies the 64 byte aligned buffer address for storing left column Intra Prediction Pixel on HEVC /VP9 Scalable mode for decode in BE pass. This buffer is used to pass data across pipes for multiple pipe mode.
118	31:0	**Intra Prediction Left Recon Column Store Buffer (attributes) Read/Write** Format: **MemoryAddressAttributes**
119..120	63:0	**HCP Scalability CABAC Decoded Syntax Elements Buffer Max Address** Format: **SplitBaseAddress64ByteAligned** Specifies the 64 byte aligned maximum address for the HCP Scalability CABAC Decoded Syntax Elements Buffer. This address shall either be 0 or larger than HCP Scalability CABAC Decoded Syntax Elements Buffer Base Address. If this address is 0, the upper bound is considered disable and HW will NOT check for upper bound.

HCP_PIPE_BUF_ADDR_STATE		
		Hardware shall only write to memory address less than this address (unless address is 0 which is disabled). Hardware will not write to memory address larger than or equal to address (unless address is 0 which is disabled)

HCP_PIPE_MODE_SELECT

HCP_PIPE_MODE_SELECT

Source: VideoCS

Length Bias: 2

The HCP is selected with the **Media Instruction Opcode "7h"** for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

The workload for the HCP is based upon a single frame decode. There are no states saved between frame decodes in the HCP. Once the bit stream DMA is configured with the HCP_BSD_OBJECT command, and the bitstream is presented to the HCP, the frame decode will begin.
The HCP_PIPE_MODE_SELECT command is responsible for general pipeline level configuration that would normally be set once for a single stream encode or decode and would not be modified on a frame workload basis.
This is a picture level state command and is shared by both encoding and decoding processes.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 0h HCP_PIPE_MODE_SELECT Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Format: =n (Excludes Dwords 0, 1). Value: 5h — Name: Value_5
1	31:24	**Reserved**

HCP_PIPE_MODE_SELECT

23	**Reserved**
22:20	**Reserved** Access: RO Format: MBZ
19	**Reserved** Access: RO Format: MBZ
18	**Prefetch Disable** When memory compression is enabled, we are seeing drop in performance. To compensate loss of performance due to latencies, we are adding pre-fetch. This bit would disable prefetches if they cause unintended behavior.
17	**Tile Based Engine** This bit indicates HW works as a Tile Based Engine(as opposed to Frame based)meaning HW will flush out bitstream and streamout data to memory at the end of each Tile. Tiling must be enabled to set this bit to 1. If this bit set to 1, the Tile row CAN be repeated if needed. Only current tile row is allowed to be repeated otherwise SW has to repeat all Tile Rows starting from the top tile row in a frame.
16:15	**Pipe working Mode** This programs the working mode for HCP pipe.

Value	Name	Description	Programming Notes
00b	Legacy decoder/encoder mode (Single pipe)	This is for single pipe mode standalone mode. It is used by both decoder and encoder.	
01b	CABAC FE only decode mode (Single CABAC pipe)	This is for the single CABAC FE only in decoder mode. This will be only run CABAC and streamout syntax element.	
10b	Decoder BE only or Encoder mode (Scalable Multi-pipe)	This is for multiple-pipe scalable mode. In decoder, it is only on BE reconstruction. In encoder, it is for PAK.	
11b	Decoder Scalable mode with CABAC in real tiles (Scalable Multi-pipe)	This is for multiple-pipe scalable mode decoder mode in real tiles. CABAC and reconstruction will	The real-tile/virtual tile decoding are supported for following features: Virtual tile decoding / Real tile decoding

			run together. Each pipes will run in real tiles vertically.	Rext HEVC (including main, main10)	yes	yes
				SCC HEVC	no	yes
				VP9	yes	no
				AV1	no	yes

14:13 Multi-Engine Mode

This indicates the current pipe is in single pipe mode or if in scalable mode is in left/right/middle pipe in multi-engine mode.

Value	Name	Description
00b	Single Engine Mode or CABAC FE only decode mode	This is for single engine mode (legacy) OR CABAC FE only decode mode During HEVC Decoder Scalability Real Tile Mode, for the last phase, it is possible to have single tile column left. In this case, it should be programmed with pipe as a single engine mode (using this value). For example, for 9 tile column running on 4 pipes. The first two phases will use all 4 pipes and finish 8 tile column. The remaining one column will be processed as last third phase as single tile column.
01b	Pipe is the left engine in a Multi-engine mode	Current pipe is the most left engine while running in scalable multi-engine mode
10b	Pipe is the right engine in a Multi-engine mode	Current pipe is the most right engine while running in scalable multi-engine mode
11b	Pipe is one of the middle engine in a Multi-engine mode	Current pipe is in one of the middle engine while running in scalable multi-engine mode

12 PAK Frame Level StreamOut enable

This bit is valid if global bit PAK Pipeline Streamout Enable is set to 1. This bit is defined to use legacy tests on HW and it's valid for both hevc/vp9.
Frame level streamouts consists of 3 parts:
LCU Streamout (Set PAK Frame level streamout enable and PAK Pipeline Streamout Enable)
SSE Streamout (Set SSE enable and PAK Pipeline Streamout Enable)
RhoDomain Streamout (Set RhoDomain enable and PAK Pipeline Streamout Enable)
If set to 1, HW will output LCU streamouts which are not validated.
By default it should be '0'

HCP_PIPE_MODE_SELECT

11 Reserved

Access:	RO
Format:	MBZ

10 VDEnc_Mode

Format:	Enable

This field indicates if PAK is working in legacy MBEnc mode or the VDEnc mode.

Value	Name	Description
0h	MBEnc mode	PAK is working in legacy mode
1h	VDEnc mode	PAK is working in VDEnc mode

9 Advanced Rate Control Enable

Format:	Enable

Description

It is only defined for encode.

Value	Name	Description
0	Disable	Use the legacy HW generated delta QP for multipass
1	Enable	Use the rate control (HW continues to generate the legacy delta QP and write to MMIO, but do not add to the final QP in the next pass)

HW assistance- HW adds delta QP for every CU in multipass.

VP9: Scalability mode use only Advanced Rate Control for BRC (no HW involvement)
HEVC: Scalability mode use only HW assisted Advanced Rate Control for BRC.

Programming Notes

In VDENC mode, This bit should be set to 1 always.
Only advanced rate control is supported in VDENC mode.

8 Reserved

Access:	RO
Format:	MBZ

7:5 Codec Standard Select

Value	Name
0	HEVC
1	VP9

4 Reserved

Access:	RO
Format:	MBZ

3 Pic Status/Error Report Enable

Format:	Enable

Value	Name	Description
0	Disable	Disable status/error reporting
1	Enable	Status/Error reporting is written out once per picture. The Pic Status/Error Report ID in DWord3along with the status/error status bits are packed into one cache line and written to the Status/Error Buffer address in the HCP_PIPE_BUF_ADDR_STATE command. Must be zero for encoder mode.

2 — PAK Pipeline Streamout Enable

Format:	Enable

Pipeline Streamout Enable is only defined for encode. It is ignored for decode.

Value	Name
0	Disable pipeline states and parameters streamout
1	Enable pipeline states and parameters streamout

Programming Notes
In VDENC mode, this field should be set to 1 always.

1 — Deblocker Streamout Enable

Format:	Enable

Deblocker Streamout Enable not currently supported for Encode or Decode

Value	Name	Description
0	Disable	Disable deblocker-only parameter streamout
1	Enable	Enable deblocker-only parameter streamout

0 — Codec Select

Format:	U1

Value	Name
0	Decode
1	Encode

2 — 31:0 — Media Soft-Reset Counter (per 1000 clocks)

Format:	U32

In decoder modes, this counter value specifies the number of clocks (per 1000) of GAC inactivity before a media soft-reset is applied to the HCP. If counter value is set to 0, the mediasoft-reset feature is disabled and no reset will occur.
In encoder modes, this counter must be set to 0 to disable media soft reset. This feature is not supported for the encoder.

Value	Name
0	Disable

3 — 31:0 — Pic Status/Error Report ID

Format:	U32

The Pic Status/Error Report ID is a unique 32-bit unsigned integer assigned to each picture

status/error output. Must be zero for encoder mode.

Value	Name	Description
0	32-bit unsigned	Unique ID Number
1	Reserved	

Programming Notes
Software must program different Status/Error Buffer addresses between pictures; otherwise the hardware might overwrite previously written data.

4	31:0	**Reserved** Access: RO Format: MBZ
5	31:0	**Reserved** Access: RO Format: MBZ
6	31:8	**Reserved** Access: RO Format: MBZ
	7:4	**Reserved** Access: RO Format: MBZ
	3	**Frame reconstruction disable** This bit disables writing out final reconstructed pixels to memory Normally used for B-frame as it's not used for reference purpose in encoder mode Default value should be '0'
	2	**HEVC Separate Tile Programming** This indicates each tile should be programmed separately in single pipe mode. (Tile can have multiple slices. But in this case, the slice must end at end of tile so it does not affect this bit). If there are multiple tiles in a slice, the slice needs to split into each individual tiles and programmed each tiles separately). This should be set when the following is met: (tiles_enabled_flag =="1") && ((pps_curr_pic_ref_enabled_flag == "1") | (palette_mode_enabled_flag == "1") | (entropy_coding_sync_enabled_flag == 1))
	1:0	**Phase Indicator** This is used to indicate whether this is first, middle or last phase of programming during Real-Tile Decoder Mode. Since HEVC can have up to 20 tile columns, maximum 10 phases are possible during 2 VDbox scalable mode. This is used by hardware to know if the current programming is first or last phases. This field is ignored (programmed to 0) for other modes other than HEVC Real-Tile Decoder Mode.

HCP_PIPE_MODE_SELECT

Value	Name
0	First Phase
1	Middle Phase
2	Last Phase

HCP_QM_STATE

HCP_QM_STATE

Source: VideoCS

Length Bias: 2

The HCP is selected with the **Media Instruction Opcode "7h"** for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

The HCP_QM_STATE command loads the custom HEVC quantization tables into local RAM and may be issued up to20 times: 3x Colour Component plus 2x intra/inter plus 4x SizeID minus 4 for the 32x32 chroma components.
When the scaling_list_enable_flag is set to disable, the scaling matrix is still sent to the decoder, and with all entries programmed to the same value = 16.
This is a picture level state command and is issued in both encoding and decoding processes.

Dwords 2-17 form a table for the DCT coefficients, 4 8-bit coefficients/DWord.

- Size 4x4 for SizeID0, DWords 2-5.
- Size 8x8 for SizeID1/2/3, DWords 2-17.

SizeID 0 (Table 4-10)

4x4	[31:24]	[23:16]	[15:8]	[7:0]
DWord 2	AC(0,3)	AC(0,2)	AC(0,1)	DC
DWord 3	AC(1,3)	AC(1,2)	AC(1,1)	AC(1,0)
DWord 4	AC(2,3)	AC(2,2)	AC(2,1)	AC(2,0)
DWord 5	AC(3,3)	AC(3,2)	AC(3,1)	AC(3,0)

SizeID 1, 2, 3 (Table 4-11)

8x8	[31:24]	[23:16]	[15:8]	[7:0]	[31:24]	[23:16]	[15:8]	[7:0]
DWord 3,2	AC(0,7)	AC(0,6)	AC(0,5)	AC(0,4)	AC(0,3)	AC(0,2)	AC(0,1)	DC
DWord 5,4	AC(1,7)	AC(1,6)	AC(1,5)	AC(1,4)	AC(1,3)	AC(1,2)	AC(1,1)	AC(1,0)
DWord 7,6	AC(2,7)	AC(2,6)	AC(2,5)	AC(2,4)	AC(2,3)	AC(2,2)	AC(2,1)	AC(2,0)
DWord 9,8	AC(3,7)	AC(3,6)	AC(3,5)	AC(3,4)	AC(3,3)	AC(3,2)	AC(3,1)	AC(3,0)
DWord 11,10	AC(4,7)	AC(4,6)	AC(4,5)	AC(4,4)	AC(4,3)	AC(4,2)	AC(4,1)	AC(4,0)
DWord 13,12	AC(5,7)	AC(5,6)	AC(5,5)	AC(5,4)	AC(5,3)	AC(5,2)	AC(5,1)	AC(5,0)
DWord 15,14	AC(6,7)	AC(6,6)	AC(6,5)	AC(6,4)	AC(6,3)	AC(6,2)	AC(6,1)	AC(6,0)
DWord 17,16	AC(7,7)	AC(7,6)	AC(7,5)	AC(7,4)	AC(7,3)	AC(7,2)	AC(7,1)	AC(7,0)

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode

HCP_QM_STATE

	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 4h HCP_QM_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 10h — Name:
1	31:13	**Reserved** Access: RO Format: MBZ
	12:5	**DC Coefficient** Format: U8 Specifies the 8-bit DC coefficient for SizeID 2 and 3. **Programming Notes** The DC Coefficient must be set to zero for SizeID 0 and 1. The DC Coefficient must be set to scaling_list_dc_coef_minus8 + 8 for SizeID 2 and 3.
	4:3	**Color Component** Format: U2 Encoder: When RDOQ is enabled, scaling list for all 3 color components must be same. So this field is set to always 0.

Value	Name
0	Luma
1	Chroma Cb
2	Chroma Cr
3	Reserved

HCP_QM_STATE

	2:1	**SizeID** Format: U2 **Value** / **Name** / **Description** 0 / 4x4 / 1 / 8x8 / 2 / 16x16 / 3 / 32x32 / (Illegal Value for Colour Component Chroma Cr and Cb.)
	0	**Prediction Type** Format: U1 **Value** / **Name** 0 / Intra 1 / Inter
2..17	511:0	**QuantizerMatrix**

HCP_REF_IDX_STATE

HCP_REF_IDX_STATE

Source: VideoCS

Length Bias: 2

The HCP is selected with the **Media Instruction Opcode "7h"** for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

This is a slice level command used in both encoding and decoding processes. For decoder, it is issued with the HCP_BSD_OBJECT command.

Unlike AVC, HEVC allows 16 reference idx entries in each of the L0 and L1 list for a progressive picture. Hence, a max total 32 reference idx in both lists together. The same when the picture is a field picture. Regardless the number of reference idx entries, there are only max 8 reference pictures exist at any one time. Multiple reference idx can point to the same reference picture and can optionally pic a top or bottom field, or frame.

For P-Slice, this command is issued only once, representing L0 list. For B-Slice, this command can be issued up to two times, one for L0 list and one for L1 list.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 12h HCP_REF_IDX_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 10h Name:

HCP_REF_IDX_STATE

1	31:5	**Reserved** Access: RO Format: MBZ
	4:1	**num_ref_idx_l[RefPicListNum]_active_minus1** Format: U4 num_ref_idx_l[*RefPicListNum*]_active_minus1 **Value** [0-14] **Name**
	0	**RefPicListNum** Format: U1 **Value** / **Name** 0 – Reference Picture List 0 1 – Reference Picture List 1
2..17	511:0	**Entries** Format: **HCP_REF_LIST_ENTRY[16]**

HCP_SFC_LOCK

HCP_SFC_LOCK

Source: BSpec

Length Bias: 2

Description
This command is used for VD/VE box to communicate with SFC before the start of any SFC workload. VD/VE uses this command to make sure that it has the ownership of SFC pipe before running workload with SFC since SFC is shared between VD/VE on a frame level. For VD(MFX)-SFC workload, only decoder mode is allowed. Encoder mode cannot use SFC.
For VD(HCP)-SFC workload, only decoder mode is allowed. Encoder mode cannot use SFC

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Format: OpCode Value 9h: Name: Media HCP+SFC Mode **[Default]**; Description: For VD(HCP)+SFC mode, only decoder mode is allowed. Encoder mode cannot use SFC
	22:21	**SubOpcodeA** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcodeB** Default Value: 0h SFC Lock Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2

HCP_SFC_LOCK		
1	31:1	**Reserved** Access: RO Format: MBZ
	0	**HCP SFC pipe select** Default Value: 1

HCP_SFC_STATE

HCP_SFC_STATE

Source: BSpec

Length Bias: 2

This command is sent from VDBOX/HCP/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Default Value: 9h Media HCP+SFC Mode Format: OpCode
	22:21	**SubOpcodeA** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcodeB** Default Value: 1h SFC_State Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 2Bh Excludes DWord (0,1) Format: =n Total Length - 2
1..49	1567:0	**SFC State Body** Format: SFC_STATE_BODY

HCP_SLICE_STATE

HCP_SLICE_STATE

Source: VideoCS

Length Bias: 2

The HCP is selected with the **Media Instruction Opcode "7h"** for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

This is a slice level command used in both encoding and decoding processes. For decoder, it is issued with the HCP_BSD_OBJECT command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 14h HCP_SLICE_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value / Name 9h 7h
1	31:26	**Reserved** Access: RO Format: MBZ

HCP_SLICE_STATE

	25:16	**SliceStartCtbY or (slice_start_lcu_y encoder)** Format: U10 **Description** Specifies the starting row address of the first coding tree block in the current slice. **The following applies to Real Tile Decoding Mode:** **(1) This should be programmed with the starting row address of the current slice segment.** **(2) For slice covering multiple tiles, the slice should be broken up into individual tiles and each tiles to be programmed separately. In this case, this should be programmed with the starting row address of the current tile.**
	15:10	**Reserved** Access: RO Format: MBZ
	9:0	**SliceStartCtbX or (slice_start_lcu_x encoder)** Format: U10 **Description** Specifies the starting column address of the first coding tree block in the current slice. The following applies to Real Tile Decoding Mode: (1) This should be programmed with the starting column address of the current slice segment. (2) For slice covering multiple tiles, the slice should be broken up into individual tiles and each tiles need to be programmed separately. In this case, this should be programmed with the starting column address of the current tile. **Programming Notes** In VDENC mode, this bit should be programmed as zero
2	31:27	**Reserved** Access: RO Format: MBZ
	26:16	**NextSliceStartCtbY or (next_slice_start_lcu_y encoder)** Format: U11 Specifies the starting row address of the first coding tree block in the next slice. Must be set to zero when the current slice is the last slice of a picture. For the single slice per frame case, the only slice is also the last slice, so this parameter should be set to a number larger than the frame height (at least +1). **Programming Notes** The following applies to Real Tile Decoding mode: (1) This should be programmed with the starting row address of the next slice segment. If the current slice is last slice of tile (or single slice tile), this should be programmed to the starting

		row address of the next tile in raster order (tile to the right, except the most right tiles which then will program to the most left tile on the next row). For the very last slice of the frame (last slice of the most bottom right tile [frame bottom-right boundary), this should be programmed to 0. (2) For slice covering multiple tiles, the slice should be broken up into individual tiles and each tiles need to be programmed separately. In this case, this should be programmed with the starting column address of the next tile in raster order.
	15	**Reserved(for Hardware)** Format: MBZ HW uses in Dynamic Slice Mode
	14:10	**Reserved** Access: RO Format: MBZ
	9:0	**NextSliceStartCtbX or (next_slice_start_lcu_x encoder)** Format: U10 Specifies the starting column address of the first coding tree block in the next slice. Must be set to zero when the current slice is the last slice of a picture. For the single slice per frame case, the only slice is also the last slice, so this parameter should be set to a number larger than the frame width (at least +1). **Programming Notes** In VDENC mode, this field should always be programmed as zero The following applies to Real Tile Decoding mode: (1) This should be programmed with the starting column address of the next slice segment. If the current slice is last slice of tile (or single slice tile), this should be programmed to the starting column address of the next tile in raster order (tile to the right, except most right tiles which then will program to the most left tile on the next row). For the very last slice of the frame (last slice of the most bottom right tile [frame bottom-right boundary), this should be programmed to 0. (2) For slice covering multiple tiles, the slice should be broken up into individual tiles and each tiles need to be programmed separately. In this case, this should be programmed with the starting column address of the next tile in raster order.
3	31:26	**Reserved** Access: RO Format: MBZ
	25	**LastSliceOfTileColumn** Format: U1 Indicates Last Slice of a Tile Column **Programming Notes** The following applies to Real Tile Decoding mode: (1) This should onlybe set to "1" on the very last slice of every tile column.

(2) For slice covering multiple tiles, the slice should be broken up into individual tiles and each tiles need to be programmed separately. In this case, this should only be set on the most bottom tile/slice on each tile columns.

24 **LastSliceOfTile**

Format:	U1

Indicates last slice of a Tile

Programming Notes
The following applies to Real Tile Decoding mode: (1) This should only be set to "1" on the last slice of every tile. (2) For slice covering multiple tiles, the slice should be broken up into individual tiles and each tiles need to be programmed separately. In this case, this should be set on every tiles.

23 **cu_chroma_qp_offset_enabled_flag**

Format:	Enable

This specifies that the cu_chroma_qp_offset_flag may be present in the transform unit syntax. chroma_qp_offset_list_enabled_flag equal to 0 specifies that the cu_chroma_qp_offset_flag is not present in the transform unit syntax. When ChromaArrayType is equal to 0, it is a requirement of bitstream conformance that the value of chroma_qp_offset_list_enabled_flag shall be equal to 0.

Programming Notes
Decoder only feature.

22 **Reserved**

Access:	RO
Format:	MBZ

21:17 **slice_cr_qp_offset**

Format:	S4

For deblocking purpose, the pic and slice level cr qp offset must be provided separately.
PAK needs to perform final_chroma_cr_qp_offset = pic_cr_qp_offset + slice_cr_qp_offset.

Value	Name
14h	-12
15h	-11
16h	-10
17h	-9
18h	-8
19h	-7
1Ah	-6
1Bh	-5
1Ch	-4
1Dh	-3

1Eh	-2
1Fh	-1
0h	0
1h	1
2h	2
3h	3
4h	4
5h	5
6h	6
7h	7
8h	8
9h	9
0Ah	10
0Bh	11
0Ch	12

Programming Notes
The valid value is from -12 to 12 (or 14h to 0Ch).

16:12 **slice_cb_qp_offset**

Format:	S4

For deblocking purpose, the pic and slice level cb qp offset must be provided separately.
PAK needs to perform final_chroma_cb_qp_offset = pic_cb_qp_offset + slice_cb_qp_offset.

Value	Name
14h	-12
15h	-11
16h	-10
17h	-9
18h	-8
19h	-7
1Ah	-6
1Bh	-5
1Ch	-4
1Dh	-3
1Eh	-2
1Fh	-1

	0h	0
	1h	1
	2h	2
	3h	3
	4h	4
	5h	5
	6h	6
	7h	7
	8h	8
	9h	9
	0Ah	10
	0Bh	11
	0Ch	12

Programming Notes

The valid value is from -12 to 12 (or 14h to 0Ch).

11:6 **SliceQp**

Format:	U6

Specifies the initial absolute value of QPy quantization parameter for the slice as defined in the Slice Header Semantics section of the HEVC standard.
This signifies only the magnitude of SliceQp. In 8 bit, SliceQp only goes from 0 to 51. But in 10 bit, it needs to go from -12 to 51. There is a sign bit specifies at bit [3] below.

Programming Notes

In 12 bit, this needs to go from -24 to 51

5 **slice_temporal_mvp_enable_flag**

Format:	U1

Programming Notes

Must be same for all the slices within a frame in encoder mode (follow spec)

4 **dependent_slice_flag**

Format:	U1

Decoder only.

3 **SliceQp Sign Flag**

Format:	U1

This specifies the sign bit of SliceQp. This is added for HEVC 10 bit. For 8 bit, SliceQp goes from 0 to 51 so this bit should be zero. In 10 bit, SliceQp goes from -12 to 51 and this bit can be set for negative value.

HCP_SLICE_STATE

	2	**LastSliceofPic** Format: U1 This indicates the current slice is the very last slice of the current picture Value: 0 — Name: Not the last slice of the picture Value: 1 — Name: Last slice of the picture **Programming Notes** The following applies to Real Tile Decoding mode: (1) This should only be set to "1" on the very last slice of most right tile column (frame boundary right-bottom tile). (2) For slice covering multiple tiles, the slice should be broken up into individual tiles and each tiles need to be programmed separately. In this case, this should only be set on the last tile (bottom) of the most right column of the frame.
	1:0	**slice_type** Format: U2 Value: 0 — Name: B-slice Value: 1 — Name: P-slice Value: 2 — Name: I-slice Value: 3 — Name: Illegal/Reserved **Programming Notes** In VDENC mode, for HEVC standard this field can be 0 or 2 only.
4	31:29	**Reserved** Access: RO Format: MBZ
	28:26	**CollocatedRefIDX** Format: U3 Collocated Motion Vector Temporal Buffer Index. **Programming Notes** In VDENC mode, the valid values are 0,1,2.
	25:23	**MaxMergeIDX** Format: U3 MaxNumMergeCand = 5 - five_minus_max_num_merge_cand -1. Value: 0 — Name: 0 Value: 1 — Name: 1

	2	2
	3	3
	4	4

Programming Notes

The valid value is from 0 to 4
(MaxNumMergeCand = 5 - five_minus_max_num_merge_cand -1)

In VDENC mode, this field should be programmed as 4.

22 **cabac_init_flag**

Format:	U1

Programming Notes

In VDEnc Mode, this value should be the same accross all the slices within frame.

21:19 **luma_log2_weight_denom**

Format:	U3

18:16 **ChromaLog2WeightDenom**

Format:	U3

15 **collocated_from_l0_flag**

Format:	U1

14 **isLowDelay**

Format:	U1

If the POCs of all pictures in both lists are less than the current POC, then set to one, else set to zero.

Programming Notes

In VDENC mode, this bit should be set to 1.

13 **mvd_l1_zero_flag**

Format:	U1

Decoder only.

12 **slice_sao_luma_flag**

Format:	U1

Programming Notes

Note: For encoder, all Slices must have same setting within a picture

11 **slice_sao_chroma_flag**

Format:	U1

HCP_SLICE_STATE

		Programming Notes Note: For encoder, all Slices must have same setting within a picture
	10	**slice_loop_filter_across_slices_enabled_flag** Format: U1 **Programming Notes** In VDENC mode, this bit should be set to 1 always. No encoder restriction going forward
	9	**Reserved** Access: RO Format: MBZ
	8:5	**slice_beta_offset_div2 or (final Beta_Offset_div2 Encoder)** Format: S3 Deblocking filter beta offset. Specified in 2's comp. **Value** [1101b,0011b] **Name** [-3,3] **Programming Notes** Note: In VDEnc mode, the value should be the same across all the slices within frame. Valid only in encoder mode
	4:1	**slice_tc_offset_div2 or (final tc_offset_div2 Encoder)** Format: S3 Deblocking filter tc offset. Specified in 2's comp. **Value** [1101b,0011b] **Name** [-3,3] **Programming Notes** Note: In VDEnc mode, the value should be the same across all the slices within frame. Valid only in encoder mode
	0	**slice_header_disable_deblocking_filter_flag** Format: U1 **Programming Notes** In VDENC mode, set this bit to zero always.
5	31:16	**Reserved** Access: RO Format: MBZ

HCP_SLICE_STATE

	15:0	**SliceHeaderLength** Format: U16 Decoder only. Specifies the length in bytes of the slice header including the start code. The starting byte of the slice header in the bit stream buffer is indicated by the Indirect Data Start Address in theHCP_BSD_OBJECT command. The ending byte of the slice header in the same bit stream buffer is indicated by the last byte prior to the slice data (CABAC).
6	31:30	**Reserved** Access: RO Format: MBZ
	29:26	**RoundInter** Format: U4 In VDENC mode, this field is ignored.

Value	Name
0h	+1/32
1h	+2/32
2h	+3/32
3h	+4/32
4h	+5/32 **[Default]**
5h	+6/32
6h	+7/32
7h	+8/32
8h	+9/32
9h	+10/32
Ah	+11/32
Bh	+12/32
Ch	+13/32
Dh	+14/32
Eh	+15/32
Fh	+16/32

Programming Notes
Encoder only feature

	25:24	**Reserved** Access: RO Format: MBZ

	23:20	**RoundIntra** Format: U4 In VDENC mode, this field is ignored. **Value** / **Name** 0h: +1/32 1h: +2/32 2h: +3/32 3h: +4/32 4h: +5/32 **[Default]** 5h: +6/32 6h: +7/32 7h: +8/32 8h: +9/32 9h: +10/32 Ah: +11/32 Bh: +12/32 Ch: +13/32 Dh: +14/32 Eh: +15/32 Fh: +16/32 **Programming Notes** Encoder only feature
	19:0	**Reserved** Access: RO Format: MBZ
7	31:11	**Reserved** Access: RO Format: MBZ
	10	**Header Insertion Enable** Format: U1 Must be followed by the PAK Insertion Object Command to perform the actual insertion. **Value** / **Name** / **Description** 0: No header insertion into the output bitstream buffer, before the current slice encoded bits. 1: Header insertion into the output bitstream buffer is present, and is before the current slice encoded bits.

Programming Notes
Must be always enabled. Encoder Only feature
In VDENC mode, this bit should be set to 1.

9 **SliceData Enable**

Format:	U1

Must always be enabled.
Encoder only feature.

Value	Name	Description
0		No operation; no insertion.
1		Slice Data insertion by PAK Object Commands into the output bitstream buffer.

Programming Notes
In VDENC mode, this bit should be set to 1.

8 **Tail Insertion Enable**

Format:	U1

Must be followed by the PAK Insertion Object Command to perform the actual insertion.

Value	Name	Description
0		No tail insertion into the output bitstream buffer, after the current slice encoded bits.
1		Tail insertion into the output bitstream buffer is present, and is after the current slice encoded bits. Tail insertion is only possible at the end of frame but not in the middle (say slice end)

Programming Notes
Tail Insertion is allowed only at the end of last slice or last tile of a frame but not in the middle of frame. Also, no multiple tail insertions are allowed. Encoder only feature

7:3 **Reserved**

Access:	RO
Format:	MBZ

2 **EmulationByteSliceInsertEnable**

Format:	U1

To have PAK outputting SODB or EBSP to the output bitstream buffer.

Value	Name
0	outputting RBSP
1	outputting EBSP

		Programming Notes Encoder Only feature In VDENC mode this bit should be set to 1.
	1	**CabacZeroWordInsertionEnable** Format: U1 To pad the end of a SliceLayer RBSP to meet the encoded size requirement. **Value** / **Name** / **Description** 0 / / No Cabac_Zero_Word Insertion. 1 / / Allow internal Cabac_Zero_Word generation and append to the end of RBSP (effectively can be used as an indicator for last slice of a picture, if the assumption is only the last slice of a picture needs to insert CABAC_ZERO_WORDs). **Programming Notes** Encoder Only feature
	0	**Dependent Slice due to Tile Split** Format: U1 In real tile decoding mode or SCC Palette mode, if a single slice covers multiple tiles, it is required for SW to split the slice into multiple slices per tile for HW programming. This bit should be set on all slices split out from original slice (except the first slice split out). **Programming Notes** This is not used by HW. This is used for validation only.
8	31:29	**Reserved** Access: RO Format: MBZ
	28:6	**Indirect PAK-BSE Data Start Offset (Write)** Format: U23 This field specifies the memory starting address (offset) to write out the compressed bitstream data from the BSE processing. This pointer is relative to the HCP PAK-BSE Object Base Address. It is a cacheline-aligned address for the HEVC bitstream data. For Write, there is no need to have a data length field. It is assumed the global memory upper bound check specified in the IND_OBJ_BASE_ADDRESS command (Indirect PAK-BSE Object Access Upper Bound) will take care of any illegal write access. **Value** / **Name** [0,524288] /

		Programming Notes Must be zero. Encoder Only feature In VDENC mode, this field should be zero.
	5:0	**Reserved** Access: RO Format: MBZ
9	31	**Force SAO parameters to zero** 1->Force SAO parameters to zero 0->normal compute This bit does not change SAO compute but forces SAO parameters to zero at the end. Default should be0. All slices must be programmed to same value within a frame. This bit is used for rate control purpose on last pass along with delta QP to decrease bitstream size. All previous passes still use deltaQP to decrease/increase bitstream size. HW might not hit the last pass. **Programming Notes** Encoder Only feature
	30:16	**Reserved** Access: RO Format: MBZ
	15:0	**Transformskip_lambda** Lambda value used in transform skip calculation **Programming Notes** Encoder Only feature
10	31:24	**Transformskip_numnonzerocoeffs_factor1** Multiplying factor with number of non-zero coefficients for non-Skip pipe calculation **Programming Notes** Encoder Only feature
	23:16	**Transformskip_numzerocoeffs_factor1** Multiplying factor with number of zero coefficients for non-Skip pipe calculation **Programming Notes** Encoder Only feature
	15:8	**Transformskip_numnonzerocoeffs_factor0** Multiplying factor with number of non-zero coefficients for skip pipe calculation **Programming Notes** Encoder Only
	7:0	**Transformskip_numzerocoeffs_factor0** Multiplying factor with number of zero coefficients for skip pipe calculation

HCP_SLICE_STATE

		Programming Notes Encoder Only feature
11	31:26	**Reserved** Access: RO Format: MBZ
	25:16	**OriginalSliceStartCtbY** Format: U10 This field programs the Slice CTB horizontal starting position of the full slice (independent and dependent slices are counted as single full slice).. If this slice is an independent slice and (if slice spans multiple tiles) first tile of this independent slice, this field should have the same value as SliceStartCtbY (above). Otherwise, this field should have the original SliceStartCtbY of the real slice. This field is only used in decoder scalable mode
	15:10	**Reserved** Access: RO Format: MBZ
	9:0	**OriginalSliceStartCtbX** Format: U10 This field programs the Slice CTB vertical starting position of the full slice (independent and dependent slices are counted as single full slice). If this slice is an independent slice and (if slice spans multiple tiles) first tile of this independent slice, this field should have the same value as SliceStartCtbX (above). Otherwise, this field should have the original SliceStartCtbX of the actual slice. This field is only used in decoder scalable mode
12	31	**use_integer_mv_flag** This specifiesthat the resolution of motion vectors for inter prediction in the current slice is integer. When not present, the value of use_integer_mv_flag is inferred to be equal to motion_vector_resolution_control_idc. Decoder only [Encoder default to 0]
	30:18	**Reserved** Access: RO Format: MBZ
	17:12	**slice_act_y_qp_offset** This specifies offsets to the quantization parameter values qP derived in clause8.6.2for lumacomponents. The values of slice_act_cb_y_offset shall be in the range of 12 to +12, inclusive. The value of PpsActQpOffsetY+ slice_act_y_qp_offset shall be in the range of 12 to +12, inclusive. This applies to both decoder and encoder.

HCP_SLICE_STATE		
	11:6	**slice_act_cb_qp_offset** This specifies offsets to the quantization parameter values qP derived in clause8.6.2 for Cbcomponents. The values of slice_act_cb_qp_offset shall be in the range of 12 to +12, inclusive. The value of PpsActQpOffsetCb+ slice_act_cb_qp_offset shall be in the range of 12 to +12, inclusive. This applies to both decoder and encoder.
	5:0	**slice_act_cr_qp_offset** This specifies offsets to the quantization parameter values qP derived in clause8.6.2 for Crcomponents. The values of slice_act_cr_qp_offset shall be in the range of 12 to +12, inclusive. The value of PpsActQpOffsetCr+ slice_act_cr_qp_offset shall be in the range of 12 to +12, inclusive. This applies to both decoder and encoder.

HCP_SURFACE_STATE

HCP_SURFACE_STATE

Source: VideoCS

Length Bias: 2

The HCP is selected with the **Media Instruction Opcode "7h"** for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

The HCP_SURFACE_STATE command is responsible for defining the frame buffer pitch and the offset of the chroma component.
This is a picture level state command and is shared by both encoding and decoding processes.
Note : When NV12/P010 and Tile Y are being used, full pitch and interleaved UV is always in use. U and V Xoffset must be set to 0; U and V Yoffset must be 4-pixel aligned. For 10-bit pixel, P010 surface definition is being used.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 1h HCP_SURFACE_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 1h — Name:
1	31:28	**Surface Id** Format: U4

Value	Name	Description
0h	HEVC: For current decoded Picture	8-bit uncompressed data
1h	Source Input Picture (encoder)	8-bit uncompressed data
2h	Prev Reference Picture	(VP9 only) Previous Reference
3h	Golden Reference Picture	(VP9 only) Golden Reference
4h	AltRef Reference Picture	(VP9 only) AltRef Reference
5h	HEVC: Reference Pictures	(HEVC only) Reference. Also, this will have separate compressible bits per reference surfaces for HEVC

27:17 **Reserved**

Access:	RO
Format:	MBZ

16:0 **Surface Pitch Minus1**

Format:	U17-1

This field specifies the surface pitch in (#Bytes - 1).

Programming Notes
For TileYF and TileYS surfaces, the range is dependent on the Cu parameter (refer to Memory Data Formats section for the definition of the Cu parameter depending on the case). The range in bytes is [2^{Cu}-1, 131071] -> [(2^{Cu})B, 128KB] = [1 tile, 128KB/($2^{Cu)\ tiles}$] The field specifies the surface pitch in (#Bytes - 1)
For tiled surfaces, the pitch must be a multiple of the tile width (i.e.128 bytes aligned). If Half Pitch for Chroma is set, this field must be a multiple of two tile widths for tiled surfaces, or a multiple of 2 bytes for linear surfaces. For Y-tiled surfaces: Range = [127, 131071] to [128B,128KB] = [1 tile, 1024 tiles]

2 31:27 **Surface Format**

Format:	U5

Specifies the format of the surface.

Value	Name	Description	Programming Notes
0h	YUY2 format		
1h	RGB_8 format		
2h	AYUV4444 format		
3h	P010Variant		P010Variant is a modified P010 format, >8 bit planar 420 with MSB together and LSB at an offset in x direction where the x-offset

				should be 32-bit aligned.
	4h	PLANAR_420_8		
	5h	YCRCB_SwapY format		
	6h	YCRCB_SwapUV format		
	7h	YCRCB_SwapUVY format		
	8h	Y216/Y210 format	Same value is used to represent Y216 and Y210	
	9h	RGB_10 format		
	Ah	Y410 format		
	Bh	NV21 Planar_420_8 Format		
	Ch	Y416 format		
	Dh	P010		
	11h	Y216Variant	Y216Variant is the modified Y210/Y216 format, 8 bit planar 422 with MSB bytes packed together and LSB bytes at an offset in the X-direction where the x-offset is 32-bit aligned. The chroma is UV interleaved with identical MSB and LSB split as luma and is at an offset in the Y-direction (similar to NV12) but is the same height as the luma.	
	12h	Y416Variant	Y416Variant is the modified Y410/Y412/Y416 format,8 bit planar 444 with MSB bytes packed together and LSB bytes at an offset in the X-direction where the x-offset is 32-bit aligned. The U channel is below the luma, has identical MSB and LSB split as luma and is at an offset in the Y-direction (similar to NV12) but is the same height as the luma The V channel is below the U, has identical MSB and LSB split as luma and is at an offset in the Y-direction (similar to NV12) but is the same height as the luma.	
	13h	YUY2Variant	YUY2Variant is the modified YUY2 format, 8 bit planar 422. The chroma is UV interleaved and is at an offset in the Y-	

		direction (similar to NV12) but is the same height as the luma.	
14h	AYUV4444Variant	AYUV4444Variant is the modified AYUV4444 format, 8 bit planar 444 format. The U channel is below the luma and is at an offset in the Y-direction (similar to NV12) but is the same height as the luma. The V channel is below the and is at an offset in the Y-direction (similar to NV12) but is the same height as the luma.	
15h-1Fh	Reserved		

Programming Notes

Programming restriction on HEVC decoder:

1. If both luma_bitdepth_minus8 and chroma_bitdepth_minus 8 are both 0 (8 bits for both luma/chroma), this should be programmed to PLANAR_420_8
2. If either luma_bitdepth_minus8 or chroma_bitdepth_minus 8 is non-zero (9 or 10 bits for either or both luma/chroma), this should be programmed to P010.

26 **Reserved**

Access:	RO
Format:	MBZ

25 **Reserved**

Access:	RO
Format:	MBZ

24:15 **Reserved**

Access:	RO
Format:	MBZ

14:0 **Y Offset for U(Cb) in pixel**

Format:	U15

This field specifies the vertical offset in rows from the **Surface Base Address** to the start(origin) of the U(Cb) plane or the interleaved UV plane if **Interleave Chroma** is enabled. This field is only used for PLANAR surface formats.

Programming Notes

- For PLANAR_420 surface formats, the alignment of this field follows the tile mode described in bits 14:13 of the **Memory Address Attributes** table.
- TileY (legacy 4k) - 8 pixel aligned
- TileYF (New 4k) - 64 pixel aligned

		• TileYS (64k) - 256 pixel aligned
3	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Default Alpha Value**
4	31:21	**Reserved** Access: RO Format: MBZ
	20:16	**Reserved** Access: RO Format: MBZ
	15:8	**Compression Type** This field indicates if the compression type for the reference surface is media or render compressed. In HEVC mode, each bit is used for 1 reference starting with Bit 8for Ref 0 in the ref list and Bit 9for Ref 1 and so on. In VP9 mode, Bit 8is for Previous Reference; Bit 9is for Golden Reference and Bit 10is for Alternate Reference; Bits11-15are unused and should be programmed to 0

Value	Name
0	Media compression Enabled **[Default]**
1	Render Compression Enabled

Programming Notes
In VP9 mode, surface ID 2h, 3h and 4h are for previous, golden and alternate reference (3 surface states are sent per surface since the reference surfaces can be different sizes with different pitch). During all 3 surface states, this field must be programmed the same.

	7:0	**Memory Compression Enable** In HEVC mode, each bit is used for 1 reference starting with Bit 0 for Ref 0 in the ref list and Bit 1 for Ref 1 and so on. In VP9 mode, Bit 0 is for Previous Reference; Bit 1 is for Golden Reference and Bit 2 is for Alternate Reference; Bits 3-7 are unused and should be programmed to 0.

Value	Name
1	Memory Compression Enable
0	Memory Compression Disable

Programming Notes
In VP9 mode, surface ID 2h, 3h and 4h are for previous, golden and alternate reference (3 surface states are sent per surface since the reference surfaces can be different sizes with different pitch). During all 3 surface states, this field must be programmed the same.

HCP_TILE_CODING

HCP_TILE_CODING

Source: BSpec

Length Bias: 2

Programming Notes
This command is used for both HEVC and VP9 codecs

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: Opcode
	28:27	**Pipeline Type** Default Value: 2h Format: Opcode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: Opcode
	22:16	**Media Instruction Command** Default Value: 15h HCP_TILE_CODING Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n **Description**: Excludes Dwords 0 & 1 **Value**: 11h **Name**:
1	31:16	**Reserved**
	15:10	**Reserved MBZ**
	9	**Tile Column store Select** This bit is used for computing Tile Column store write offset and Tile read column store read address. **Programming Notes**

HCP_TILE_CODING

Tile Configuration 1: 3x3 tiles

1	2	3
4	5	6
7	8	9

In the above tiling configuration,
For Tiles 1,4,7,3,6,9Tile Column store select should be programmed as zero.
For Tiles 2,5,8 Tile Row store select should be programmed as 1.

Tile Configuration 1: 3x5 tiles

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

In the above tiling configuration,
For Tiles 1,6,11, 3,8, 13, 5, 10, 15Tile Column store select should be programmed as zero.
For Tiles 2,7, 12, 4, 9, 14Tile Column store select should be programmed as 1.

Tile Configuration 1: 3x5tiles

1	2	3
4	5	6
7	8	9
10	11	12
13	14	15

In the above tiling configuration,
For Tiles 1,4,7,10,13, 3,6, 9, 12, 15 Tile Column store select should be programmed as zero.
For Tiles 2,5,8,11,14Tile Column store select should be programmed as 1.

8 **Tile Row store Select**

This bit is used for computing Tile row store write offset and Tile read row store read address.

Programming Notes

Tile Configuration 1: 3x3 tiles

1	2	3
4	5	6
7	8	9

In the above tiling configuration,
For Tiles 1,2,3,7,8,9 Tile Row store select should be programmed as zero.
For Tiles 4,5,6 Tile Row store select should be programmed as 1.

Tile Configuration 1: 3x5 tiles

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

In the above tiling configuration,

For Tiles 1,2,3,4,5,11,12,13,14,15Tile Row store select should be programmed as zero.
For Tiles 6,7,8,9,10Tile Row store select should be programmed as 1.

Tile Configuration 1: 3x5tiles

1	2	3
4	5	6
7	8	9
10	11	12
13	14	15

In the above tiling configuration,
For Tiles 1,2,3,7,8,9,13,14,15Tile Row store select should be programmed as zero.
For Tiles 4,5,6,10,11,12 Tile Row store select should be programmed as 1.

7:0 Number of Active BE Pipes

Indicates the number of active, consecutive positioned Scalable VDBOXs to be used for the current frame decoding or encoding.
BE Pipe partitioning, SW must guarantee the minimum width is at least two full LCUs for each tiles

This field in general should be smaller or equal to Num of Tile columns in a Frame.
This field is ignored by HW

This field is not used by HW

Value	Comment
0	ignored
1	ignored
2	Supported by Encoder / Decoder.
3	Supported only by Decoder.
4	Supported only by Encoder

2 31 IsLastTileOfColumn

Format:	U1

Indicates if current Tile is last tile of a Column

30 IsLastTileOfRow

Indicates if current Tile is Last Tile of a Row

29:26 Reserved

Access:	RO
Format:	MBZ

25:16 Tile Row Position

Format:	U10

HCP_TILE_CODING

		Ctb row position of tile For VP9: In units of SB64x64
	15:11	**Reserved** Access: RO Format: MBZ
	10	**Non First Pass Tile** Indicates Non first past Tile when Tile Replay Mode is enabled
	9:0	**Tile Column Position** Format: U10 Ctb column position of tile For VP9: In units of SB64x64
3	31	**LastPassOfTile (ValidationOnly)** This bit indicates last pass of a Tile. This is validation use only. HW/SW should not use in design
	30:27	**Reserved** Access: RO Format: MBZ
	26:16	**TileWidthInMinCbMinus1** Specifies Tile width in units of minimum coding block size. The minimal width per tile is at least two full LCUs. In HEVC Encoder mode, the following restrictions apply. Last LCU at frames right edge must align to CU boundary. This applies to all size of LCUs: 16x16, 32x32, 64x64. Kernel does not count/include CUs outside of the picture in PakObj/CU_record respectively. For VP9: In units of 8x8
	15:11	**Reserved** Access: RO Format: MBZ
	10:0	**TileHeightInMinCbMinus1** Specifies Tile Height in units of minimum coding block size In HEVC Encoder mode, the following restrictions apply. Last LCU at frames bottom edge must align to CU boundary. This applies to all size of LCUs: 16x16, 32x32, 64x64. Kernel does not count/include CUs outside of the picture in PakObj/CU_record respectively. For VP9: In units of 8x8
4	31:6	**Bitstream Byte Offset** Offset on top of base address from where the encoded bitstream should be written out for this tile In scalability mode: this offset is valid for every tile and it must be zero for the first tile in a frame.

		Non scalability mode: not valid
	5:1	**Reserved** Access: RO Format: MBZ
	0	**Bitstream Byte Offset Enable** 0-> Disables bitstream byte offset, meaning the encoded bitstream for all TILEs would be contiguous This bit is set to zero.
5	31:6	**PAK Frame Statistics Offset** The frame statistics (SSE, RhoDomain and LCU stats) will be reported per Tile. Valid only in scalability mode
	5:0	**Reserved** Access: RO Format: MBZ
6	31:6	**CU Level Streamout Offset** CU level statistics (see details in streamout section) per tile will be streamed out starting from this offset address This offset is valid for every Tile in scalability mode only.
	5:0	**Reserved** Access: RO Format: MBZ
7	31:6	**Slice Size Streamout Offset** Size of every slice within this tile will be streamed out starting from this offset address. This offset is valid for every Tile in scalability mode only.
	5:0	**Reserved** Access: RO Format: MBZ
8	31:6	**CU record offset** Offset address for CU record for this tile This offset is valid for every Tile in scalability mode only.
	5:0	**Reserved** Access: RO Format: MBZ

HCP_TILE_CODING

9	31:6	**SSE RowStore offset** SSE(Sum Square Error) statistics per tile will be written out at this address This offset is valid for every Tile in scalability mode.
	5:0	**Reserved** Access: RO Format: MBZ
10	31:6	**SAO RowStore offset** SAO Rowstore offset for this tile. This offset is valid for every Tile in scalability mode only.
	5:0	**Reserved** Access: RO Format: MBZ
11	31:6	**Tile Size StreamOut Offset** Tile Size will be written out at this offset This offset is valid for every Tile in scalability mode only.
	5:0	**Reserved** Access: RO Format: MBZ
12	31:6	**VP9 Probability Counter Streamout Offset** Probability counters will be written out starting from this offset address This offset is valid for every Tile in scalability mode only.
	5:0	**Reserved** Access: RO Format: MBZ
13..14	63:0	**HCP Scalability Synchronize Buffer - Base Address** Format: **SplitBaseAddress64ByteAligned** Specifies the 64 byte aligned buffer address used for data synchronization between neighboring pipes in scalable modes. The buffer will be written and read (as a flush mechanism) by hardware to guarantee data made it to memory before neighboring pipe can read the data. Hardware will also write the current row (in LCU) number to indicate which the current processing rows. **Programming Notes** This minimal buffer size (in CLs) should be set to the number of scalable pipes used by this workload. This data should be not be cached.
15	31:0	**HCP Scalability Synchronize Buffer - Attributes** Format: **MemoryAddressAttributes**

HCP_TILE_CODING		
16	31:0	**Reserved** Access: RO Format: MBZ
17	31:18	**Reserved** Access: RO Format: MBZ
	17:14	**Frame Number** Indicates Frame number
	13:8	**Tile number** This field indicates the tile number and used for reporting in Tile bitstream meta data.
	7:0	**Reserved** Access: RO Format: MBZ
18	31:0	**TileMetaData_DW1** First four bytes of Meta data that goes into Tile bitstream metadata.
19	31:0	**TileMetaData_DW2** **Programming Notes** last four bytes of Meta data that goes into Tile bitstream metadata.

HCP_TILE_STATE

HCP_TILE_STATE

Source: VideoCS

Length Bias: 2

The HCP is selected with the **Media Instruction Opcode "7h"** for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

This command is valid for decoder only.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 11h HCP_TILE_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: Fh — Name:
1	31:10	**Reserved** Access: RO Format: MBZ
	9:5	**NumTileColumnsMinus1** Format: U5 Specifies the number of tile columns in Ctbs per picture.

HCP_TILE_STATE		
		Maximum of 20 columns are supported (level 6.2 restriction)
	4:0	**NumTileRowsMinus1** Format: U5 Specifies the number of tile rows in Ctbs per picture. Maximum of 22 rows are supported (level 6.2 restriction)
2..6	159:0	**Ctb column position of tile column** Format: HCP_TILE_POSITION_IN_CTB[5]
7..12	191:0	**Ctb row position of tile row** Format: HCP_TILE_POSITION_IN_CTB[6] Note that there are only 22 rows, so the most significant 16 bits of HCP_TILE_POSITION_IN_CTB[5] (31:16) are reserved
13..14	63:0	**Ctb column position MSB** Format: HCP_TILE_POSITION_IN_CTB_MSB
15..16	63:0	**Ctb row position MSB** Format: HCP_TILE_POSITION_IN_CTB_MSB

HCP_VP9_PAK_OBJECT

HCP_VP9_PAK_OBJECT

Source: VideoCS

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HUC = Bh
	22:16	**Media Instruction Command** Default Value: 35h HCP_VP9_PAK_OBJECT Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 0h — Name:
1	31	**IsLastSB Flag (of Tile)** Format: Enable Value: 0 — Name: False Value: 1 — Name: True
	30	**Reserved** Access: RO Format: MBZ

HCP_VP9_PAK_OBJECT

	29:24	**CU_count_minus1** Format: U6 Number of CUs in the current SB = CU_count_minus1 + 1. Minimum, there must be 1 CU in a SB.
	23:21	**Reserved** Access: RO Format: MBZ
	20:0	**split_coding_unit_flag[x0][y0]** Format: U21 Bit 20: Split_flag_level0 Bit 19:16: Split_flag_level1 [19:16] is in raster order. Bit16 is for partition0 in raster order. Bit 15:12: Split_flag_level2_level1part3 Split flags for bit19 partition. [15:12] is in raster order. Bit12 is for partition0 in raster order. Bit 11:8: Split_flag_level2_level1part2 Split flags for bit18 partition. [11:8] is in raster order. Bit8 is for partition0 in raster order. Bit 7:4: Split_flag_level2_level1part1 Split flags for bit17 partition. [7:4] is in raster order. Bit4 is for partition0 in raster order. Bit 3:0: Split_flag_level2_level1part0 Split flags for bit16 partition. [3:0] is in raster order. Bit0 is for partition0 in raster order.
2	31:16	**Current SB Y Addr)** Format: U16
	15:0	**Current SB X Addr** Format: U16
3	31:0	**Reserved** Access: RO Format: MBZ
4	31:17	**Reserved** Access: RO Format: MBZ
	16	**LcuForceZeroCoeff (Time Budget Overflow Occurred)** Format: Enable The bitstream packer pipeline will force all the coefficients to zero if this bit is set. Also, VDenc Indicates to PAK that time budget overflow has occurred.
	15:12	**SSE ClassID 32x32_3** Format: U4

HCP_VP9_PAK_OBJECT		
		Programming Notes This parameter indicates the SSE classID for the 32x32 block3 of the current LCU. Valid values: 0-8
	11:8	**SSE ClassID 32x32_2** Format: U4 **Programming Notes** This parameter indicates the SSE classID for the 32x32 block2 of the current LCU. Valid values: 0-8
	7:4	**SSE ClassID 32x32_1** Format: U4 **Programming Notes** This parameter indicates the SSE classID for the 32x32 block1 of the current LCU. Valid values: 0-8
	3:0	**SSE ClassID 32x32_0** Format: U4 **Programming Notes** This parameter indicates the SSE classID for the 32x32 block1 of the current LCU. Valid values: 0-8

HCP_VP9_PIC_STATE

HCP_VP9_PIC_STATE

Source: VideoCS

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HUC = Bh
	22:16	**Media Instruction Command** Default Value: 30h HCP_VP9_PIC_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value / Name / Programming Notes: Bh — Decoder DW Length — Only Up to DW12 should be programmed for decoder 1Eh — Encoder DW Length — All DWs should be programmed for encoder
1	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**Frame Height In Pixels Minus 1** Format: U14 Specifies the height of the decoded picture in units of 8 pixels, which is the minimum coding block size. The decoded picture height in units of luma samples

equals
(FrameHeightInMinBlocksMinus1+ 1) *8 1
For Encoder Partial SB:
Kernel, on last SB (at picture edges), splits the SB into as small as possible CUs to have picture boundaries aligned to CU boundary.
Kernel does not count/include CUs outside of the picture in PakObj/CU_record respectively.
Driver sets up a SB aligned (both in X/Y direction) surface.

Value	Name
[4096,16383]	4K_TO_16K

Programming Notes

Decoder supports 16K image and 8K video. Encoder only supports up to 8K.

Value	Description
0-6	Invalid (multiple of 8 pixels)
7-8191	8-8K Pixels (Decoder and Encoder)
8192-16383	8K-16K Pixels (Decoder only)

15:14 **Reserved**

Access:	RO
Format:	MBZ

13:0 **Frame Width In Pixels Minus 1**

Format:	U14

Specifies the width of the decoded picture in units of minimum coding block size. The decoded picture width in units of luma samples equals (FrameWidthInMinBlocksMinus1+ 1) *8 1 This should be programmed to a multiple of 8 pixels minus 1.
For Encoder Partial SB:
Kernel, on last SB (at picture edges), splits the SB into as small as possible CUs to have picture boundaries aligned to CU boundary.
Kernel does not count/include CUs outside of the picture in PakObj/CU_record respectively.
Driver sets up a SB aligned (both in X/Y direction) surface.

Value	Name
[4096,16383]	4K_TO_16K

Programming Notes

Decoder supports 16K image and 8K video. Encoder only supports up to 8K.

Value	Description
0-6	Invalid (multiple of 8 pixels)
7-8191	8-8K Pixels (Decoder and Encoder)

HCP_VP9_PIC_STATE

		8192-16383 | 8K-16K Pixels (Decoder only)
2	31	**Segment ID StreamIn Enable** Format: | Enable Indicates SegmentID from previous frame needs to be streamIn for Segment ID prediction **Value** | **Name** 0 | Disable 1 | Enable **Programming Notes** Deocder Only:SegmentIDStreamInEnable = error_resilient_mode OR intra_only OR VP9_KEY_FRAME
	30	**Segment ID StreamOut Enable** Format: | Enable Indicates SegmentID of current frame needs to be streamOut for next frame **Value** | **Name** 0 | Disable 1 | Enable **Programming Notes** Deocder Only: SegmentIDStreamOutEnable = segementation_enabled AND segmentation_update_map
	29	**Lossless Mode** Format: | Enable This bitSet to indicate lossless coding mode. In encoder mode, software has to set tx_mode to 4x4only and all tu_sizes in CU record as 4x4 for entire frame. Software also has to program such that final_qindex=0 and final_filter_level=0 following the Quant Scale and Filter Level Table in Segmentation State section. Hardware forces Hadamard Tx when this bit is set. When Lossless Mode is on, BRC has to be off. **Value** | **Name** 0 | Normal Mode 1 | Loless Mode
	28	**Segmentation Temporal Update** Format: | Enable Indicates whether segID is decoding from bitstream or predicted from previous

frame.

In encoder Mode it should use either from previous frame or streamIn

Value	Name
0h	Decode segID from bitstream
1h	Get segID either from bitstream or from previous frame

Programming Notes
Decoder Only: For KEY_FRAME or INTRA_ONLY frame, this bit should be set to "0". Note: Driver should override this flag to "0" in KEY_FRAME or INTRA_ONLY frame even if this bit decoded from bitstream is different. This is for hardware optimization. This override does not affect bitstream decoding other than uncompressed header.

27 **Segmentation Update Map**

Format:	Enable

Indicates how hardware determines segmentation ID

Value	Name	Description
0h		Intra block: segment ID is zero Inter block: get segment ID from previous frame (streamIN)
1h		Intra block: decode segment ID from bitstream. Inter block: determines from segmentation_temporal_update setting

26 **Segmentation Enabled**

Format:	Enable

Indicate if segementation is enabled or not

Value	Name
0h	All blocks are implied to belong to segment 0
1h	SegID determination depends on segmentation_update_map setting

25:23 **Sharpness Level**

Format:	U3

Specify the sharpness level, as one of regular deblocking strength control.

Programming Notes
Set to 0 to disable the use of sharpness control

22:17 **Filter Level**

Format:	U6

Specify the Filter level, as one of deblocking strength control

Programming Notes
Set to 0 to disable the use of level control

HCP_VP9_PIC_STATE

16 **Frame Parallel Decoding Mode**

Indicates if parallel decoding mode is enabled. This bit should come from Uncompressed header. Together with Error Resilient mode, they decide the value of AdaptProbabilityFlag.

Value	Name
0	Disable
1	Enable

15 **Error Resilient Mode**

Indicates if error resilient mode is enabled. This bit should come from Uncompressed header. When error resilient is 1, Frame Parallel Decoding Mode will be 1, and Refresh Frame Context will be 0.When error resilient is 0, Frame Parallel Decoding Mode and Refresh Frame Context read from bit stream. Together with Frame Parallel Decoding mode, they decide the value of AdaptProbabilityFlag.

Value	Name
0	Disable
1	Enable

Programming Notes

14 **Refresh Frame Context**

Indicates if Frame Context should be refresh. This bit should come from Uncompressed header

Value	Name
0	Disable
1	Enable

Programming Notes
Decoder Only

13 **Last Frame Type**

It indicates the frame type of previous frame (Key or Non-Key Frame)

Value	Name
0	Key Frame
1	Non Key Frame

Programming Notes
Not used in Encoder Mode

12 **Selectable TX Mode**

Format:	U1

Indicates if tx_mode is selectable

HCP_VP9_PIC_STATE

Value	Name
0	Encoder does not pack tu_size into bitstream. This helps reduce bitstream size further.
1	Encoder packs tu_size into bitstream.

Programming Notes
HW always picks tu_size from CU record of pak_obj. SW responsibility to set tu_size correct.

11 **Hybrid Prediction Mode**

Format:	U1

Indicates if comp_pred_mode is hybrid

Value	Name
0	comp_prediction_mode!=HYBRID, Encoder does not pack comp_pred_mode [interpred_comp in pak_obj] into bitstream.
1	comp_prediction_mode==HYBRID, Encoder packs comp_pred_mode into bitstream. This helps reduce bitstream size further.

10 **Use Prev in Find MV References**

Format:	Enable

0: Temporal MV buffer is not available for MV prediction 1: Temporal MV buffer is available for MV prediction This is set to 0 when: The last picture has a different size Current picture is error-resilient mode Current picture is intra_only, or keyframe Last picture was intra_only or keyframe Last picture was not a displayed picture.

9:7 **Ref Frame Sign Bias[0..2]**

Format:	U3

Reference Frame sign bias (not including intra reference) Bit[7] Sign Bias of Last Frame Bit[8] Sign Bias of Golden Frame Bit[9] Sign Bias of AltRef Frame

6:4 **Mcomp Filter Type**

Format:	U3

Indicate Motion Compensation Filter type.
If set to 4, encoder uses modes in pak_obj command.

Value	Name
0h	Eight-tap
1h	Eight-tap-Smooth
2h	Eight-tap-Sharp
3h	Bilinear

HCP_VP9_PIC_STATE

		4h: Switchable
	3	**Allow Hi Precision MV** Format: Enable Indicate high precision mode for Motion Vector prediction 0h: Normal mode 1h: High Precision mode
	2	**IntraOnly Flag** Format: Enable Indicates intra-only for inter pics. MBZ for keyframes. **Programming Notes** Used for Non-displayable picture; SW responsibility to make sure no Inter block in pak_obj of this frame 0: Inter Frame use both intra/inter-blocks 1: Inter frame use only inta-blocks
	1	**Adapt Probabilities Flag** Format: Enable Indicates that the probabilities used to decode this frame should be adapted 0h: 0: Do not adapt (error resilient or frame_parallel_mode are set) 1h: 1: Adapt (not error resilient and not frame_ parallel_mode) **Programming Notes** Only forward adaptation is supported in encoder mode.
	0	**Frame Type** Format: U1 Specifies the VP9 frame type 0h: Key Frame 1h: Inter Frame
3	31:28	**Profile Level** Format: U4 This indicates VP9 Profile level from bitstream 0: Profile_0: Profile 0 only supports 8 bit 420 only 2: Profile_2: Profile 2 only supports 10 bits 420 only

1	Profile_1	Profile 1 only supports 8 bit 444 only
3	Profile_3	Profile 3 only supports 10-bit 444 only

Programming Notes
Profile 0, 1, 2, and 3 are supported. Profile 0: 8 bit 420 only Profile 1: 8 bit 444 (422 is NOT supported) Profile 2: 10/12 bit 420 Profile 3: 10/12 bit 444 (422 is NOT supported)

27:24 **BitDepthMinus8**

Format:	U4

This indicates the bitdepth (minus 8) of the pixels

Value	Name	Programming Notes
0	Bitdepth_8	It indicates pixel bitdepth is 8. Only profile 0 is allowed in this mode. It indicates pixel bitdepth is 8. Only profile 0 and 1 are allowed in this mode.
2	Bitdepth_10	It indicates pixel bitdepth is 10. Only profile 2 is allowed in this mode. It indicates pixel bitdepth is 10. Only profile 2 and 3 are allowed in this mode.
4	Bitdepth_12	It indicates pixel bitdepth is 12. Only profile 2 is allowed in this mode.

Programming Notes
In profile 0 and 1, only value of 0 (8 bit pixel) is allowed. In profile 2 and 3, only value of 2 and 4 (10 or 12 bit pixel) are allowed.

23:22 **Chroma Sampling Format**

Format:	U2

This indicates the chroma sampling format of the bitstream

Value	Name	Programming Notes
0	Format_420	Chroma Format 420, supported by profile 0 and 2
2	Format_444	Chroma Format 444, suported by Profile 1 and 3

Programming Notes
Currenly only 420 and 444 are supported (in profile 0, 1, 2 and 3). All other modes are not valid. Value 0: Format 420: Profile 0 and 2 only (supported) Value 1: Format 422: Profile 1 and 3 only: Currently NOT supported Value 2: Format 444: rofile 1 and 3 only:(supported)

21 **SSE Enable**

Format:	Enable

This field indicates if SSE statistics generation is enabled in the PAK. If enabled, the classID in the CU packet is used along with the SSE thresholds in the picture state to generate the SSE frame statistics. When enabled, the PAK will also write and read source pixels (similar to the reference pixels in the LCU ILDB streamout) to generate the SSE distortion.
Valid only in VDenc mode.
Encoder Only

20:10 **Reserved**

Access:	RO
Format:	MBZ

9:8 **Log2 Tile Row**

Format:	U2

This indicates the number of tile rows (log2).

Value	Name
0h	1 Tile Row
1h	2 Tile Row
2h	4 Tile Row

Programming Notes
Decoder Only as encoder must use TILE_CODING_COMMAND

7:4 **Reserved**

Access:	RO
Format:	MBZ

3:0 **Log2 Tile Column**

Format:	U4

This indicates the number of tile rows (log2).

Value	Name
0h	1 Tile Column
1h	2 Tile Column
2h	4 Tile Column
3h	8 Tile Column
4h	16 Tile Column
5h	32 Tile Column
6h	64 Tile Column

Programming Notes

HCP_VP9_PIC_STATE

		Decoder only as encoder must use TILE_CODING_COMMAND
4	31:16	**Horizontal Scale Factor for LAST** Format: U2.14 This indicates the scaling factor between current frame and the last reference frame Set to (LastWidth * 2^14)/CurrentWidth
	15:0	**Vertical Scale Factor for LAST** Format: U2.14 This indicates the scaling factor between current frame and the last reference frame Set to (LastHeight * 2^14)/CurrentHeight
5	31:16	**Horizontal Scale Factor for GOLDEN** Format: U2.14 This indicates the scaling factor between current frame and the golden reference frame Set to (LastWidth * 2^14)/CurrentWidth
	15:0	**Vertical Scale Factor for GOLDEN** Format: U2.14 This indicates the scaling factor between current frame and the golden reference frame Set to (LastHeight * 2^14)/CurrentHeight"
6	31:16	**Horizontal Scale Factor for ALTREF** Format: U2.14 This indicates the scaling factor between current frame and the altref reference frame Set to (LastWidth * 2^14)/CurrentWidth
	15:0	**Vertical Scale Factor for ALTREF** Format: U2.14 This indicates the scaling factor between current frame and the altref reference frame Set to (LastHeight * 2^14)/CurrentHeight
7	31	**Reserved**
	30	**Reserved** Access: RO Format: MBZ
	29:16	**Last Frame Hieght In Pixels Minus 1** Format: U14 Specifies the height of the Last picture in units of pixels. The Last picture height in units of luma samples equals (LastFrameHeightInMinBlocksMinus1+ 1) Value | Description

		0-16383 | 1-16K Pixels (encoder only goes to 8K)
	15:14	**Reserved** Access: RO Format: MBZ
	13:0	**Last Frame Width In Pixels Minus 1** Format: U14 Specifies the width of the Last picture in units of pixels. The Last picture width in units of luma samples equals (LastFrameWidthtInMinBlocksMinus1+ 1) Value | Description 0-16383 | 1-16K Pixels (encoder only goes to 8K)
8	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**Golden Frame Height In Pixels Minus 1** Format: U14 Specifies the height of the Last picture in units of pixels. The Golden picture height in units of luma samples equals (LastFrameHeightInMinBlocksMinus1+ 1) Value | Description 0-16383 | 1-16K Pixels (encoder only goes to 8K)
	15:14	**Reserved** Access: RO Format: MBZ
	13:0	**Golden Frame Width In Pixels Minus 1** Format: U14 Specifies the width of the Last picture in units of pixels. The Golden picture width in units of luma samples equals (LastFrameWidthtInMinBlocksMinus1+ 1) Value | Description 0-16383 | 1-16K Pixels (encoder only goes to 8K)
9	31:30	**Reserved** Access: RO Format: MBZ
	29:16	**Altref Frame Height In Pixels Minus 1** Format: U14 Specifies the height of the Last picture in units of pixels. The Altref picture height

HCP_VP9_PIC_STATE

		in units of luma samples equals (LastFrameHeightInMinBlocksMinus1+ 1) Value: 0-16383 — Description: 1-16K Pixels (encoder only goes to 8K)
	15:14	**Reserved** Access: RO Format: MBZ
	13:0	**Altref Frame Width In Pixels Minus 1** Format: U14 Specifies the width of the Last picture in units of pixels. The Altref picture width in units of luma samples equals (LastFrameWidthtInMinBlocksMinus1+ 1) Value: 0-16383 — Description: 1-16K Pixels (encoder only goes to 8K)
10	31:16	**First Partition Size in Bytes [15:0]** Format: U16 Specifies the number of bytes taken up by the first partition size which handle the probability updates **Programming Notes** Only used by Decoder
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**Uncompressed Header Length in Bytes [7:0]** Format: U8 Specifies the number of bytes taken up by the uncompressed frame header. **Programming Notes** this field is used by decoder only
11	31:4	**Reserved** Access: RO Format: MBZ
	3	**Reserved**
	2	**Reserved**
	1	**Motion Comp Scaling Enable Bit** This bit must be set to "1" Value: 1 — Name: Enable **[Default]** — Description: This enables Motion Comp Scaling
	0	**Reserved**

		Access: RO Format: MBZ
12	31:0	**Reserved** Access: RO Format: MBZ
13	31:26	**Reserved** Access: RO Format: MBZ
	25	**Header Insertion Enable** Format: U1 **Value** / **Name** / **Description** 0 / / No header insertion into the output bitstream buffer, before the current slice encoded bits. 1 / / Header insertion into the output bitstream buffer is present, and is before the current slice encoded bits. Must be followed by the PAK Insertion Object Command to perform the actual insertion. For VP9: Header is always present and this bit can never be Zero. As HW does the header back-annotation at the end of frame we currently cannot disable it, if header was not written by HW. Media SDK sends 2 headers. One header has original header and second header has 0s for BRC parameters (LF refDelta, ModeDelta and BaseQindex). Driver needs to pick first header for the first pass, and second header for subsequent passes. **Programming Notes** Encoder Only
	24	**Tail Insertion Enable** Format: U1 **Value** / **Name** / **Description** 0 / / No tail insertion into the output bitstream buffer, after the current slice encoded bits. 1 / / Tail insertion into the output bitstream buffer is present, and is after the current slice encoded bits. Must be followed by the PAK Insertion Object Command to perform the actual insertion. Tail has to be inserted only with the last slice of frame and for VP9 only at the end of Frame. **Programming Notes**

HCP_VP9_PIC_STATE

		Encoder Only
	23:16	**Base Q Index (Same as Luma AC)** Format: U8 Added to Delta Q index of Segment Valid Values : 0..255 **Programming Notes** Encoder Only
	15:0	**Compressed header BIN count** Format: U16 Number of bins compressed header This field is fixed insideHW **Programming Notes** Encoder Only
14	31:21	**Reserved** Access: RO Format: MBZ
	20:16	**Luma DC Q Index Delta** Format: S4 QP delta value for Luma DC Valid Values : -15..15 **Programming Notes** Encoder Only
	15:13	**Reserved** Access: RO Format: MBZ
	12:8	**ChromaDC_QindexDelta** Format: S4 QP Delta Value For Chroma DC Valid Values : -15..15 **Programming Notes** Encoder Only
	7:5	**Reserved** Access: RO Format: MBZ
	4:0	**ChromaAC_QindexDelta** Format: S4 QP Delta Value For Chroma AC Valid Values : -15..15

		Programming Notes Encoder Only
15	31	**Reserved** Access: RO Format: MBZ
	30:24	**LF_ref_delta3** Format: S6 Loop filter level delta3 value; valid range -6363 With this range -63..63, in the case of Base LF Level > 31 (first pass) or (Base LF Level + BRC accumulated LF Delta) > 31 (subsequent pass), the lf_ref_delta0,1,2,3 and lf_mode_delta0,1 in compressed bitstream would be in the range -31..31. **Programming Notes** Encoder Only
	23	**Reserved** Access: RO Format: MBZ
	22:16	**LF_ref_delta2** Format: S6 Loop filter level delta2 value; valid range -6363 With this range -63..63, in the case of Base LF Level > 31 (first pass) or (Base LF Level + BRC accumulated LF Delta) > 31 (subsequent pass), the lf_ref_delta0,1,2,3 and lf_mode_delta0,1 in compressed bitstream would be in the range -31..31. **Programming Notes** Encoder Only
	15	**Reserved** Access: RO Format: MBZ
	14:8	**LF_ref_delta1** Format: S6 Loop filter level delta1 value; valid range -6363 With this range -63..63, in the case of Base LF Level > 31 (first pass) or (Base LF Level + BRC accumulated LF Delta) > 31 (subsequent pass), the lf_ref_delta0,1,2,3 and lf_mode_delta0,1 in compressed bitstream would be in the range -31..31. **Programming Notes** Encoder Only
	7	**Reserved** Access: RO Format: MBZ

HCP_VP9_PIC_STATE

	6:0	**LF_ref_delta0** Format: S6 Loop filter level delta0 value; valid range -6363 With this range -63..63, in the case of Base LF Level > 31 (first pass) or (Base LF Level + BRC accumulated LF Delta) > 31 (subsequent pass), the lf_ref_delta0,1,2,3 and lf_mode_delta0,1 in compressed bitstream would be in the range -31..31. **Programming Notes** Encoder Only
16	31:15	**Reserved** Access: RO Format: MBZ
	14:8	**LF Mode Delta 1** Format: S6 Loop filter level mode1 value; valid range -6363 With this range -63..63, in the case of Base LF Level > 31 (first pass) or (Base LF Level + BRC accumulated LF Delta) > 31 (subsequent pass), the lf_ref_delta0,1,2,3 and lf_mode_delta0,1 in compressed bitstream would be in the range -31..31. **Programming Notes** Encoder Only
	7	**Reserved** Access: RO Format: MBZ
	6:0	**LF Mode Delta 0** Format: S6 Loop filter level mode1 value; valid range -6363 With this range -63..63, in the case of Base LF Level > 31 (first pass) or (Base LF Level + BRC accumulated LF Delta) > 31 (subsequent pass), the lf_ref_delta0,1,2,3 and lf_mode_delta0,1 in compressed bitstream would be in the range -31..31. **Programming Notes** Encoder Only
17	31:16	**BitOffsetForLFModeDelta** Format: U16 Offset from starting position of output bitstream in bits where LFModeDelta should be inserted. In BRC mode, always insert LFModeDelta. {This implies uncompressed header should have: mode_ref_delta_enabled=1 and mode_ref_delta_update=1} and The uncompressed header starting from this offset **BitOffsetForLFModeDelta** has to have the following 16 bits format: {Start here: 1b1, mode_delta_0[6:0], 1b1, mode_delta_1[6:0]} and **BitOffsetForLFModeDelta = BitOffsetForLFRefDelta + 32.**

		Encoder Only
	15:0	**BitOffsetForLFRefDelta** Format: U16 Offset from starting position of output bitstream in bits where LFRefDelta should be inserted. In BRC mode, always insert LFRefDelta. {This implies uncompressed header should have: mode_ref_delta_enabled=1 and mode_ref_delta_update=1} and The uncompressed header starting from this offset **BitOffsetForLFRefDelta** has to have the following 32 bits format: {Start here: 1b1, ref_delta_0[6:0], 1b1, ref_delta_1[6:0], 1b1, ref_delta_2[6:0], 1b1, ref_delta_3[6:0]}. Encoder Only
18	31:16	**BitOffsetForLFLevel** Format: U16 Offset from starting position of output bitstream in bits where LFLevel should be inserted. Encoder Only
	15:0	**BitOffsetForQindex** Format: U16 Offset from starting position of output bitstream in bits where Qindex should be inserted. Encoder Only
19	31:27	**Reserved** Access: RO Format: MBZ
	26	**FrameSzUnderStatusEn - FrameBitRateMinReportMask** Format: U1 This is a mask bit controlling if the condition of frame level bit count is less than FrameBitRateMin. **Value** / **Name** / **Description** 0 / Disable / Do not update bit 2 (Frame Bit Count Violate -- under run) of HCP_VP9_IMAGE_STATUS control register. 1 / Enable / Set bit 2 (Frame Bit Count Violate -- under run) of HCP_VP9_IMAGE_STATUS control register if the total frame level bit counter is less than or equal to Frame Bit Rate Minimum limit.

HCP_VP9_PIC_STATE

		Programming Notes Encoder Only
	25	**FrameSzOverStatusEn - FrameBitRateMaxReportMask** Format: U1 This is a mask bit controlling if the condition of frame level bit count exceeds FrameBitRateMax. **Value** / **Name** / **Description** 0 / Disable / Do not update bit 1 of HCP_VP9_IMAGE_STATUS control register. 1 / Enable / Set bit 1 of HCP_VP9_IMAGE_STATUS control register if the total frame level bit counter is greater than or equal to Frame Bit Rate Maximum limit. **Programming Notes** Encoder Only
	24:18	**Reserved** Access: RO Format: MBZ
	17	**Reserved** Access: RO Format: MBZ
	16	**NonFirstPassFlag** This signals the current pass is not the first pass. It will imply designate HW behavior. **Value** / **Name** / **Description** 0 / Disable / If it is initial-Pass, this bit is set to 0. 1 / Enable / For subsequent passes, this bit is set to 1. **Programming Notes** Encoder Only
	15:0	**Reserved** Access: RO Format: MBZ
20	31	**FrameBitrateMaxUnit** Format: U1 This field is the Frame Bitrate Maximum Limit Units. **Value** / **Name** / **Description** 0 / Byte / 32byte unit 1 / KiloByte / 4Kbyte unit

HCP_VP9_PIC_STATE

		Programming Notes Encoder Only
	30:14	**Reserved** Access: RO Format: MBZ
	13:0	**FrameBitRateMax** Format: U14 This field is the Frame Bitrate Maximum Limit. This field along with FrameBitrateMaxUnit determines maximum allowed bits in a frame before multi-pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count exceeds this value. 0-512KB The programmable range is 0-512KB when FrameBitrateMaxUnit is 0. 0-64MB The programmable range is 0-64Mbyte when FrameBitrateMaxUnit is 1. **Programming Notes** Encoder Only
21	31	**FrameBitrateMinUnit** Format: U1 This field is the Frame Bitrate Maximum Limit Units. 0 – Byte – 32byte unit 1 – KiloByte – 4Kbyte unit **Programming Notes** Encoder Only
	30:14	**Reserved** Access: RO Format: MBZ
	13:0	**FrameBitRateMin** Format: U14 This field is the Frame Bitrate Minimum Limit. This field along with FrameBitrateMinUnit determines maximum allowed bits in a frame before multi-pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count exceeds this value. 0-512KB The programmable range is 0-512KB when FrameBitrateMinUnit is 0. 0-64MB The programmable range is 0-64Mbyte when FrameBitrateMinUnit is 1.

HCP_VP9_PIC_STATE

		Programming Notes Encoder Only
22..23	63:0	**FrameDeltaQindexMax** Frame level delta Qindex which should be used in case FrameSize - FrameBitRateMax in the range of ((FrameDeltaQindexLFMaxRange[n] * FrameBitRateMax»5)), FrameDeltaQindexLFMaxRange[n+1] * FrameBitRateMax»5)). Each DelatQindexMax value is 8-bit with S7M format **Programming Notes** If n == 7, DeltaQpMaxRange is infinity. Encoder Only
24	31:0	**FrameDeltaQindexMin** Frame level delta Qindex which should be used in case FrameSize - FrameBitRateMin in the range of ((FrameDeltaQindexLFMinRange[n] * FrameBitRateMin»5)), FrameDeltaQindexLFMinRange[n+1] * FrameBitRateMin»5)). Each DelatQindexMin value is 8-bit with S7M format **Programming Notes** If n == 3, FrameDeltaQindexLFMaxRange is zero. (n>3 is not supported) Encoder Only
25..26	63:0	**FrameDeltaLFMax** Frame level delta Loop Filter Level which should be used in case FrameSize - FrameBitRateMax in the range of ((FrameDeltaQindexLFMaxRange[n] * FrameBitRateMax»5)), FrameDeltaQindexLFMaxRange[n+1] * FrameBitRateMax»5)). Each delta_lf_max is 7 bits with S6M format **[bits 7, 15, 23, 31,.63 are reserved]** **Programming Notes** If n == 7, FrameDeltaQindexLFMaxRange is infinity. Encoder Only
27	31:0	**FrameDeltaLFMin** Frame level delta Loop Filter Level which should be used in case FrameSize - FrameBitRateMin in the range of ((FrameDeltaQindexLFMinRange[n] * FrameBitRateMin»5)), FrameDeltaQindexLFMinRange[n+1] * FrameBitRateMin»5)). Each delta_lf_min is 7 bits with S6M format **[bits 7, 15, 23, 31 are reserved]** **Programming Notes** If n == 3, FrameDeltaQindexLFMaxRange is zero. (n>3 is not supported) Encoder Only
28..29	63:0	**FrameDeltaQindexLFMaxRange** Condition: FrameDeltaQindexLFMaxRange[n] >= FrameDeltaQindexLFMaxRange[n-1] This field is to calculate ranges for Frame level delta Qindex, specifically Frame level delta Qindex[n] and Frame level delta

Qindex[n+1].

Programming Notes
If n == 0, FrameDeltaQindexLFMaxRange is zero.
Encoder Only

30 — 31:0 — FrameDeltaQindexLFMinRange

Condition: FrameDeltaQindexLFMinRange[n] >= FrameDeltaQindexLFMinRange[n-1]This field is to calculate ranges for Frame level delta Qindex, specifically Frame level delta Qindex[n] and Frame level delta Qindex[n+1].

Programming Notes
If n == 0, FrameDeltaQindexLFMinRange is zero.
Encoder Only

31 — 31:30 — MinFrameSizeUnits

Format:	U2

This field is the Minimum Frame Size Units

Value	Name	Description
0	4Kb	Minimum Frame Size is in 4Kbytes.
1	16Kb	Minimum Frame Size is in 4Kbytes.
2	Compatibility Mode	
3	6 Bytes	

Programming Notes
Encoder Only

29:16 — Reserved

Access:	RO
Format:	MBZ

15:0 — MinFramSize

Format:	U16

Minimum Frame Size [15:0] (in Word, 16-bit)(Encoder Only) Minimum Frame Size is specified to compensate for intel Rate Control Currently zero fill (no need to perform emulation byte insertion) is done at the last slice of a picture. It is needed for CBR. Intel encoder parameter, not part of DXVA. The caller should always make sure that the value, represented by Minimum Frame Size, is always less than maximum frame size FrameBitRateMax. This field is reserved in Decode mode.

Programming Notes
Programmable range is 0..(2^16-1) * 2^12 when MinFrameSizeUnits is 0. (4KB unit)
Programmable range is 0..(2^16-1) * 2^14 when MinFrameSizeUnits is 1. (16KB unit)

HCP_VP9_PIC_STATE

		Encoder Only
32	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**BitOffsetForFirstPartitionSize** Format: U16 Offset from starting position of output bitstream in bits where First Partition Size should be inserted. **Programming Notes** Encoder Only
33	31:16	**Class0_SSE_Threshold1** Format: U16 **Programming Notes** This field specifies the upper bound threshold for Class0 Zone1 to classify the per-4x4sblk SSE statistics. Class0_SSE_Threshold_0 < per-4x4sblk SSE <= Class0_SSE_Threshold_1 fall under Class0 Zone1. Class0_SSE_Threshold_1 < per-4x4sblk SSE fall under Class0 Zone2. Encoder Only
	15:0	**Class0_SSE_Threshold0** Format: U16 **Programming Notes** This field specifies the upper bound threshold for Class0 Zone0 to classify the per-4x4sblk SSE statistics. per-4x4sblk SSE <= Class0_SSE_Threshold_0 fall under Class0 Zone0. Encoder Only
34..41 **Programming Notes:** SSE thresholds for Class 1-8, see DW 33 (SSE Class 0 thresholds) for format. Encoder Only	255:0	**SSE thresholds for Class1-8** Format: U256

HCP_VP9_SEGMENT_STATE

HCP_VP9_SEGMENT_STATE

Source: VideoCS

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HUC = Bh
	22:16	**Media Instruction Command** Default Value: 32h HCP_VP9_SEGMENT_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 6h Name:
1	31:3	**Reserved** Access: RO Format: MBZ
	2:0	**Segment ID** Format: U3 The segment ID of the DWORDS following this one
2	31:4	**Reserved** Access: RO

		Format: MBZ
	3	**Segment Reference Enabled** Format: Enable For encoder: When Segment Reference Enabled is set to 1, Software (Kernel) needs to program all PUs of this segment accordingly. This means: CU record PU refframe0=Segment Reference bit[2:1], refframe1=0, and interpred_comp=single.
	2:1	**Segment Reference** Format: U2 Indicates reference frame for this segment. **Programming Notes** If the current frame is a KEY frame or INTRA_ONLY frame, this field should be set to INTRA for all segments.
	0	**Segment Skipped** Format: Enable Indicates skip mode for this segment. **Programming Notes** If set to 1, all delta coefficients and MVs within CU of this segment should be forced to zero in HW. No block less than 8x8 size allowed segmentSkipped If set to 0, skipcoeff_flag should be coded as normal
3	31:30	**Reserved** Access: RO Format: MBZ
	29:24	**FilterLevelRef1Mode1** Exists If: //Decoder mode only Format: U6 Indicates final filter level for Ref1 (Last Frame) and Mode 1.
	23:22	**Reserved** Access: RO Format: MBZ
	21:16	**FilterLevelRef1Mode0** Exists If: //Decoder mode only Format: U6 Indicates final filter level for Ref1 (Last Frame) and Mode 0.
	15:14	**Reserved** Access: RO Format: MBZ

HCP_VP9_SEGMENT_STATE

	13:8	**FilterLevelRef0Mode1** Exists If: //Decoder mode only Format: U6 Indicates final filter level for Ref0 (Last Frame) and Mode 1.
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**FilterLevelRef0Mode0** Exists If: //Decoder mode only Format: U6 Indicates final filter level for Ref0 (Last Frame) and Mode 0.
4	31:30	**Reserved** Access: RO Format: MBZ
	29:24	**FilterLevelRef3Mode1** Exists If: //Decoder mode only Format: U6 Indicates final filter level for Ref3 (Last Frame) and Mode 1.
	23:22	**Reserved** Access: RO Format: MBZ
	21:16	**FilterLevelRef3Mode0** Exists If: //Decoder mode only Format: U6 Indicates final filter level for Ref3 (Last Frame) and Mode 0.
	15:14	**Reserved** Access: RO Format: MBZ
	13:8	**FilterLevelRef2Mode1** Exists If: //Decoder mode only Format: U6 Indicates final filter level for Ref2 (Last Frame) and Mode 1.

HCP_VP9_SEGMENT_STATE

	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**FilterLevelRef2Mode0** Exists If: //Decoder mode only Format: U6 Indicates final filter level for Ref2 (Last Frame) and Mode 0.
5	31	**Reserved** Access: RO Format: MBZ
	30:16	**Luma AC Quant Scale (Decode mode Only)** Format: U15 Indicates final value of Luma AC Quantized Scale value. Value: [4,29247] — Name: Valid_Range
	15	**Reserved** Access: RO Format: MBZ
	14:0	**Luma DC Quant Scale (Decode mode Only)** Format: U15 Indicates final value of Luma DC Quantized Scale value. Value: [4,21387] — Name: Valid_Range
6	31	**Reserved** Access: RO Format: MBZ
	30:16	**Chroma AC Quant Scale (Decode mode Only)** Format: U15 Indicates final value of Chroma AC Quantized Scale value. Value: [4,29247] — Name: Valid_Range
	15	**Reserved** Access: RO Format: MBZ
	14:0	**Chroma DC Quant Scale (Decode mode Only)** Format: U15

HCP_VP9_SEGMENT_STATE

		Indicates final value of Chroma DC Quantized Scale value. **Value** / **Name** [4,21387] / Valid_Range
7	31:23	**Reserved** Access: RO Format: MBZ
	22:16	**Segment LF Level Delta (Encode mode Only)** Format: S6 Indicates the Loop Filter Delta for this segment. Must be 0 when segmentation_enabled == 0. Range -63..63
	15:9	**Reserved** Access: RO Format: MBZ
	8:0	**Segment QIndex Delta (encode mode only)** Format: S8 Indicates the QIndex delta for this segment. Must be 0 when segmentation_enabled == 0. Range -255..255

HCP_WEIGHTOFFSET_STATE

HCP_WEIGHTOFFSET_STATE

Source: VideoCS

Length Bias: 2

The HCP is selected with the **Media Instruction Opcode "7h"** for all HCP Commands. Each HCP command has assigned a media instruction command as defined in DWord 0, BitField 22:16.

This slice level command is issued in both the encoding and decoding processes, if the weighted_pred_flag orweighted_bipred_flag equals one. If zero, then this command is not issued.
Weight Prediction Values are provided in this command. Only Explicit Weight Prediction is supported inencoder.
For P-Slice, this command is issued only once together with HCP_REF_IDX_STATE Command for L0 list. For B-Slice, this command can be issued up to two times together with HCP_REF_IDX_STATE Command, one forL0 list and one for L1 list.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: 13h HCP_WEIGHTOFFSET_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 28h — Name: 40
1	31:1	**Reserved**

HCP_WEIGHTOFFSET_STATE

		Access: RO Format: MBZ
	0	**RefPicListNum** Format: U1 Value / Name 0 – Reference Picture List 0 1 – Reference Picture List 1
2..17	511:0	**LumaOffsets** Format: HCP_WEIGHTOFFSET_LUMA_ENTRY[16]
18..33	511:0	**ChromaOffsets** Format: HCP_WEIGHTOFFSET_CHROMA_ENTRY[16]
34..41	255:0	**ChromaOffsetsExt** Format: HCP_WEIGHTOFFSET_CHROMA_EXT_ENTRY[8]

HEVC_SFC_AVS_CHROMA_Coeff_Table

HEVC_SFC_AVS_CHROMA_Coeff_Table

Source: BSpec

Length Bias: 2

This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Default Value: 9h Media HEVC+SFC Mode Format: OpCode
	22:21	**SubOpcodeA** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcodeB** Default Value: 6h SFC_AVS CHROMA Coeff_Table Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 3Fh Excludes DWord (0,1) Format: =n Total Length - 2
1..64	2047:0	**AVS CHROMA Coefficient Table Body** Format: SFC_AVS_CHROMA_COEFF_TABLE_BODY

HEVC_SFC_AVS_LUMA_Coeff_Table

HEVC_SFC_AVS_LUMA_Coeff_Table

Source: BSpec

Length Bias: 2

This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Default Value: 9h Media HEVC+SFC Mode Format: OpCode
	22:21	**SubOpcodeA** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcodeB** Default Value: 5h SFC_AVS LUMA Coeff_Table Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 7Fh Excludes DWord (0,1) Format: =n Total Length - 2
1..128	4095:0	**AVS LUMA Coefficient Table Body** Format: SFC_AVS_LUMA_COEFF_TABLE_BODY

HEVC_SFC_AVS_STATE

HEVC_SFC_AVS_STATE

Source: BSpec

Length Bias: 2

This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Format: OpCode Value: 9h — Name: Media HEVC+SFC Mode **[Default]**
	22:21	**SubOpcodeA** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcodeB** Default Value: 2h SFC_AVS_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) Format: =n Total Length - 2
1..3	95:0	**AVS State Body** Format: SFC_AVS_STATE_BODY

HEVC_SFC_FRAME_START

HEVC_SFC_FRAME_START		
Source: BSpec Length Bias: 2		
This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Default Value: 9h Media HEVC+SFC Mode Format: OpCode
	22:21	**SubOpcodeA** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcodeB** Default Value: 4h SFC_FRAME_START Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:0	**Frame Start Body** Format: SFC_FRAME_START_BODY

HEVC_SFC_IEF_STATE

HEVC_SFC_IEF_STATE

Source: BSpec

Length Bias: 2

This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Default Value: 9h Media HEVC+SFC Mode Format: OpCode
	22:21	**SubOpcodeA** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcodeB** Default Value: 3h SFC_IEF_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 16h Excludes DWord (0,1) Format: =n Total Length - 2
1..23	735:0	**SFC IEF State Body** Format: SFC_IEF_STATE_BODY

HEVC_VP9_RDOQ_STATE

HEVC_VP9_RDOQ_STATE

Source: VideoCS

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h Format: Opcode PARALLEL_VIDEO_PIPE
	28:27	**Pipeline** Default Value: 2h Format: Opcode MFX_COMMON
	26:23	**Opcode** Default Value: 7h Format: Opcode Codec/Engine Name = HCP = 7h
	22:21	**SubOpA** Default Value: 0h Format: Opcode
	20:16	**SubOpB** Default Value: 8h Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Total Length - 2 Value: 98h — Name: DWORD_COUNT_n **[Default]** — Description: Excludes DWord (0,1)
1	31	**Disable HTQ performance fix0** set to disable performance optimizations done by doubling LNZ and OSR storage Set to 1, to go back to single LNZ and OSR (no optimization) Set to 0, to use double buffer LNZ and OSR
	30	**Disable HTQ performance fix1**

HEVC_VP9_RDOQ_STATE

		set to disable performance optimization done to save 1clk while switching from EBB to GTRAM storage. This is critical for IntraLoop performance Set to 1, disable optimization Set to 0, enable optimization
	29:0	**Reserved** Access: RO Format: MBZ
2..33	1023:0	**IntraLumaLambda_QP0_63** Format: **HEVC_VP9_RDOQ_LAMBDA_FIELDS[32]**
34..65	1023:0	**IntraChromaLambda_QP0_63** Format: **HEVC_VP9_RDOQ_LAMBDA_FIELDS[32]**
66..97	1023:0	**InterLumaLambda_QP0_63** Format: **HEVC_VP9_RDOQ_LAMBDA_FIELDS[32]**
98..129	1023:0	**InterChromaLambda_QP0_63** Format: **HEVC_VP9_RDOQ_LAMBDA_FIELDS[32]**
130..135	191:0	**IntraLumaLambda_QP64_75** Format: **HEVC_VP9_RDOQ_LAMBDA_FIELDS[6]**
136..141	191:0	**IntraChromaLambda_QP64_75** Format: **HEVC_VP9_RDOQ_LAMBDA_FIELDS[6]**
142..147	191:0	**InterLumaLambda_QP64_75** Format: **HEVC_VP9_RDOQ_LAMBDA_FIELDS[6]**
148..153	191:0	**InterChromaLambda_QP64_75** Format: **HEVC_VP9_RDOQ_LAMBDA_FIELDS[6]**

HI8DS Render Target Write MSD

MSD_RTW_HI8DS - HI8DS Render Target Write MSD

Source: EuSubFunctionRenderDataPort

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Message Precision Subtype** Default Value: 0h Format: Opcode Full precision data message
	29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	**Per-Coarse Pixel PS outputs enable** Format: Enable This bit indicates the render target write is a coarse pixel write. **Programming Notes** This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.

MSD_RTW_HI8DS - HI8DS Render Target Write MSD

	17:14	**Message Type** Default Value: 0Ch Format: Opcode Render Target Write message
	13	**Per-Sample PS Enable** Format: Enable If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot. **Programming Notes** This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.
	12	**Last Render Target Select** Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.
	11	**Slot Group Select** Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 3h Format: Opcode SIMD8 dual source message. Use slots [15:8] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:24] are referenced instead of [15:8].

MSD_RTW_HI8DS - HI8DS Render Target Write MSD		
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

If

if - If

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

An if instruction starts an if/endif or an if/else/endif block of code. It restricts execution within the conditional block to only those channels that were enabled via the predicate control. Each if instruction must have a matching endif instruction and may have up to one matching else instruction before the matching endif. If all channels are inactive (for the if/endif or if/else/endif block), a jump is performed to the instruction referenced by JIP. This jump must be to right after the matching else instruction when present, or otherwise to the matching endif instruction of the conditional block. If SPF is ON, the UIP must be used to update IP; JIP is not used in this case.

The following table describes the 32-bit exit code <JIP> and <UIP>. If <branch_ctrl> is set, then the JIP points to the first join instruction within the if block. If <branch_ctrl> is not set, <JIP> should point to the instruction right after the matching else instruction if it exsits, otherwise <JIP> should point to the endif instruction. <UIP> should always point to the endif instruction. When a jump occurs, this value is added to IP pre-increment. In instruction binary, <JIP> and <UIP> are at location <src0> & <src1> and must be of type D (signed dword integer).

Format:

```
[(pred)] if (exec_size JIP UIP <branch_ctrl>
```

Restriction

The execution size must be the same for the if, else, and endif instructions of the same code block.

Syntax

```
[(pred)] if (exec_size) imm32 imm32 <branch_ctrl>
```

Pseudocode

```
Evaluate(WrEn);
for ( n = 0; n < 32; n++ ) {
    if ( WrEn.channel[n] == 0 ) {
        PcIP[n] = IP + JIP;
    } else {
        PcIP[n] = IP + 1;
    }
}
if ( PcIP != (IP + 1) ) {  // for all channels
    Jump(IP + JIP);
}
```

DWord	Bit	Description

if - If

0..3 — 127:96 — Reserved

Exists If:	([Src0.IsImm]==false)
Format:	MBZ

127:96 — JIP

Exists If:	([Src0.IsImm]==true)
Format:	S31

The byte-aligned jump distance if a jump is taken for the channel.

95:80 — Reserved

Exists If:	([Src0.IsImm]==false)
Format:	MBZ

95:64 — Reserved

Exists If:	([Src0.IsImm]==true) AND ([Src1.IsImm]==false)
Format:	MBZ

95:64 — UIP

Exists If:	([Src0.IsImm]==true) AND ([Src1.IsImm]==true)
Format:	S31

The byte aligned jump distance if a jump is taken for the instruction.

79:66 — Src0.Operand

Exists If:	([Src0.IsImm]==false)
Format:	**DirectOperand**

65:64 — Reserved

Exists If:	([Src0.IsImm]==false)
Format:	MBZ

63:50 — Dst.Operand

Format:	**DirectOperand**

49:48 — Reserved

Access:	RO
Format:	MBZ

47 — Src1.IsImm

This field indicate that Source 1 operand is carrying an immediate value

Value	Name
0	false
1	true

46 — Src0.IsImm

This field indicate that Source 0 operand is carrying an immediate value

if - If

Value	Name
0	false
1	true

45:34 **Reserved**

Access:	RO
Format:	MBZ

33 **BranchCtrl**

This field is used by *goto*, *if*, and *else* instructions to control branching. See the goto instruction description for more information about BranchCtrl.

32 **AtomicCtrl**

Format:	AtomicCtrl

31 **MaskCtrl**

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 **Reserved**

29 **CmptCtrl**

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

if - If

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16 **ExecSize**

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

15:0 **Header**

Format:	Header

Illegal

illegal - Illegal

Source:	EuIsa
Length Bias:	4
Predication:	false
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

The Illegal Opcode Exception Enable flag in cr0.1 is normally set so the normal processing of an illegal opcode is to transfer control to the System Routine. Instruction dispatch treats any unused 8-bit opcode (including bit 7 of the instruction, reserved for future opcode expansion) as if it is the illegal opcode. The illegal opcode is zero because that byte value is more likely than most to be read via a wayward instruction pointer. The illegal instruction is an instruction only in the same way that a NULL pointer in software is a pointer. Both are special values indicating invalid instances.

Format:

```
illegal
```

Restriction

The illegal instruction takes no instruction options.

Syntax

```
illegal
```

Pseudocode

```
{
 Set the Illegal Opcode Exception Status bit in cr0.1.
     if ( Illegal Opcode Exception Enable is set in cr0.1 ) {
         Transfer control to the System Routine (return address to AIP, IP = SIP).
     }
 }
```

DWord	Bit	Description
0..3	127:7	**Reserved** Access: RO Format: MBZ
	6:0	**Opcode** Format: EU_OPCODE

Join

join - Join

Source:	Eulsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

The join instruction makes the inactive channels active at the join IP if those channels are predicated. Any deactivated channels due to a goto instruction match the join IP are activated (qualified with predicates at join). If no IP is matched at this join, the program goes to the next IP with the active channels which followed the program path up to the join instruction. If no active channels are present after executing the join instruction, the program jumps to the offset specified by JIP instead of next IP. The join instruction is used in conjunction with a goto instruction. The join activates channels that are deactivated by the goto instruction. See the goto instruction for the deactivation rules. The goto and join instructions enable unstructured program control flow. These instructions must be used with additional care where dangling channels can result without proper compiler checks, meaning that it is expected that programs will navigate through these paths to reactivate the channels. Hardware does not provide native checks or reconvergence. The following table describes the 32-bit JIP. In instruction binary, JIP is at location src1 and must be of type D (signed DWord integer). JIP must be an immediate operand and is a signed 32-bit number. This value is added to IP pre-increment. If SPF is ON, none of the PcIP are updated.

Format:

```
[(pred)] join (exec_size) JIP
```

Programming Notes

An index of 0 is an infinite loop.

Restriction

JIP must point to a Flow Control Instruction.

Syntax

```
[(pred)] join (exec_size) imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if (WrEn.chan[n] ) {  // for the predicated channels and the remaining channels
         PcIP[n] = IP + 1;
     }
 }
 if ( PcIP != (IP + 1) ) {  // for all channels when no channels are activated and no
other active channels
     Jump(IP + JIP);
```

join - Join

```
}
```

DWord	Bit	Description
0..3	127:96	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	127:96	**JIP** Exists If: ([Src0.IsImm]==true) Format: S31 The byte-aligned jump distance if a jump is taken for the channel
	95:80	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	95:64	**Reserved** Exists If: ([Src0.IsImm]==true) Format: MBZ
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) Format: **DirectOperand**
	65:64	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	63:50	**Dst.Operand** Format: **DirectOperand**
	49:47	**Reserved** Access: RO Format: MBZ
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — Name: false Value: 1 — Name: true
	45:34	**Reserved** Access: RO Format: MBZ
	33	**BranchCtrl**

This field is used by *goto*, *if*, and *else* instructions to control branching. See the **goto** instruction description for more information about BranchCtrl.

32 **AtomicCtrl**

Format:	AtomicCtrl

31 **MaskCtrl**

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 **Reserved**

29 **CmptCtrl**

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the

		content of the selected flag register.
	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Jump Indexed

jmpi - Jump Indexed

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

Description

The jmpi instruction redirects program execution to an index offset relative to the pre-incremented instruction pointer. The index is a signed integer value, with positive or zero integers for forward jumps, and negative integers for backward jumps. In instruction binary, index is carried as src0 register or immediate. The ip register must be put (for example, by the assembler) at the dst. Predication is allowed to provide conditional jump with a scalar condition. As the execution size is 1, the first channel of PMASK (flags post prediction control and negate) is used to determine whether the jump is taken or not. If the condition is false, the jump is not taken and execution continues with the next instruction.

Format:

```
[(pred)] jmpi (1) index {NoMask}
```

Programming Notes

An index argument of 16 would continue to the next instruction (assuming the instruction is encoded as 128b).

An index argument of 0 loops infinitely: all immediate branch arguments, including jmpi now, are relative to the pre-increment IP.

Restriction

The execution size must be 1.

MaskCtrl must be specified.

QtrCtrl must not be used for jmpi instruction.

Jmpi instruction with non-uniform predicates or reg32 source cannot be used when EU Fusion is enabled.

Syntax

```
[(pred)] jmpi (1) reg32 {NoMask}
 [(pred)] jmpi (1) imm32 {NoMask}
```

Pseudocode

```
Evaluate(WrEn);
 if ( WrEn != 0 ) {
     Jump(IP + index );
 }
```

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0..3</td><td>127:96</td><td>Reserved<table><tr><td>Exists If:</td><td>([Src0.IsImm]==false)</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>127:96</td><td>JIP<table><tr><td>Exists If:</td><td>([Src0.IsImm]==true)</td></tr><tr><td>Format:</td><td>S31</td></tr></table>The byte-aligned jump distance if a jump is taken for the channel</td></tr>
<tr><td></td><td>95:80</td><td>Reserved<table><tr><td>Exists If:</td><td>([Src0.IsImm]==false)</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>95:64</td><td>Reserved<table><tr><td>Exists If:</td><td>([Src0.IsImm]==true)</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>79:66</td><td>Src0.Operand<table><tr><td>Exists If:</td><td>([Src0.IsImm]==false)</td></tr><tr><td>Format:</td><td>DirectOperand</td></tr></table></td></tr>
<tr><td></td><td>65:64</td><td>Reserved<table><tr><td>Exists If:</td><td>([Src0.IsImm]==false)</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>63:50</td><td>Dst.Operand<table><tr><td>Format:</td><td>DirectOperand</td></tr></table></td></tr>
<tr><td></td><td>49:47</td><td>Reserved<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>46</td><td>Src0.IsImm
This field indicate that Source 0 operand is carrying an immediate value.<table><tr><th>Value</th><th>Name</th></tr><tr><td>0</td><td>false</td></tr><tr><td>1</td><td>true</td></tr></table></td></tr>
<tr><td></td><td>45:34</td><td>Reserved<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>33</td><td>BranchCtrl
This field is used by goto, if, and else instructions to control branching. See the goto instruction description for more information about BranchCtrl.</td></tr>
</table>

jmpi - Jump Indexed

32 AtomicCtrl

Format:	AtomicCtrl

31 MaskCtrl

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 Reserved

29 CmptCtrl

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 PredInv

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 PredCtrl

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

jmpi - Jump Indexed

	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Leading Zero Detection

lzd - Leading Zero Detection

Source:	Eulsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	true
Source Modifier:	true

The lzd instruction counts component-wise the leading zeros from src0 and stores the resulting counts in dst. If src0 is zero, store 32 in dst.

Format:

```
[(pred)] lzd[.cmod] (exec_size) dst src0
```

Syntax

```
[(pred)] lzd[.cmod] (exec_size) reg reg
 [(pred)] lzd[.cmod] (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         UD udScalar = src0.chan[n];
         UD cnt = 0;
         while ( (udScalar & (1 « 31)) == 0 && cnt != 32 ) {
             cnt ++;
             udScalar = udScalar « 1;
         }
         dst.chan[n] = cnt;
     }
 }
```

Src Types	Dst Types
*B,*W,*D	UD

DWord	Bit	Description
0..3	127:96	**Src0.ImmValue[31:0]** Exists If: ([Src0.IsImm]==true)
	95:92	**CondCtrl** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **FlagModifier**
	95:64	**Src0.ImmValue[63:32]** Exists If: ([Src0.IsImm]==true) AND (([Src0.DataType]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df))

lzd - Leading Zero Detection

	87:84	**Src0.VertStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **VertStride**
	83:81	**Src0.Width** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **Width**
	80	**Src0.AddrMode** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Reserved** Access: RO Format: MBZ
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value.

lzd - Leading Zero Detection

Value	Name
0	false **[Default]**
1	true

45:44 **Src0.Mod**

Format:	**SrcMod**

43:40 **Src0.DataType**

Exists If:	([Src0.IsImm]==false)
Format:	**RegDataType**

43:40 **Src0.DataType**

Exists If:	([Src0.IsImm]==true)
Format:	**ImmDataType**

39:36 **Dst.DataType**

Format:	**RegDataType**

35 **Dst.AddrMode**

Format:	**AddrMode**

34 **Saturate**

Format:	**Saturate**

33 **AccWrCtrl**

Format:	**AccWrCtrl**

32 **AtomicCtrl**

Format:	**AtomicCtrl**

31 **MaskCtrl**

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 **Reserved**

29 **CmptCtrl**

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

lzd - Leading Zero Detection

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. This can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16 **ExecSize**

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

lzd - Leading Zero Detection		
	15:0	**Header** Format: **Header**

LO8DS Render Target Write MSD

MSD_RTW_LO8DS - LO8DS Render Target Write MSD

Source: EuSubFunctionRenderDataPort

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Message Precision Subtype** Default Value: 0h Format: Opcode Full precision data message
	29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	**Per-Coarse Pixel PS outputs enable** Format: Enable This bit indicates the render target write is a coarse pixel write. **Programming Notes** This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.

MSD_RTW_LO8DS - LO8DS Render Target Write MSD

	17:14	**Message Type** Default Value: 0Ch Format: Opcode Render Target Write message
	13	**Per-Sample PS Enable** Format: Enable If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot. **Programming Notes** This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.
	12	**Last Render Target Select** Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.
	11	**Slot Group Select** Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 2h Format: Opcode SIMD8 dual source message. Use slots [7:0] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [23:16] are referenced instead of [7:0].

MSD_RTW_LO8DS - LO8DS Render Target Write MSD		
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Logic And

and - Logic And

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	false
Source Modifier:	true

The and instruction performs component-wise logic AND operation between src0 and src1 and stores the results in dst. Register source operands can use source modifiers: Any source modifier is logical, optionally changing a source value s to ~s (inverting all source bits). This capability allows expressions like a AND (NOT b) to be calculated with one instruction. This operation does not produce sign or overflow conditions. Only the .e/.z or .ne/.nz conditional modifiers should be used.

Format:

```
Source modifier is not allowed if source is an accumulator.
```

Restriction

Source modifier is not allowed if source is an accumulator.

Syntax

```
[(pred)] and[.cmod] (exec_size) reg reg reg
 [(pred)] and[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         dst.chan[n] = src0.chan[n] & src1.chan[n];
     }
 }
```

Src Types	Dst Types
*B,*W,*D	*B,*W,*D

DWord	Bit	Description
0..3	127:126	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	127:96	**Src1.ImmValue[31:0]** Exists If: ([Src1.IsImm]==true)
	125:122	**Reserved** Exists If: ([Src1.IsImm]==false)

and - Logic And

		Format:	MBZ
	121:120	**Src1.Mod**	
		Exists If:	([Src1.IsImm]==false)
		Format:	**SrcMod**
	119:116	**Src1.VertStride**	
		Exists If:	([Src1.IsImm]==false)
		Format:	**VertStride**
	115:113	**Src1.Width**	
		Exists If:	([Src1.IsImm]==false)
		Format:	**Width**
	112	**Src1.AddrMode**	
		Exists If:	([Src1.IsImm]==false)
		Format:	**AddrMode**
	111:98	**Src1.Operand**	
		Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)
		Format:	**IndirectOperand**
	111:98	**Src1.Operand**	
		Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)
		Format:	**DirectOperand**
	97:96	**Src1.HorzStride**	
		Exists If:	([Src1.IsImm]==false)
		Format:	**HorzStride**
	95:92	**CondCtrl**	
		Format:	**FlagModifier**
	91:88	**Src1.DataType**	
		Exists If:	([Src1.IsImm]==true)
		Format:	**ImmDataType**
	91:88	**Src1.DataType**	
		Exists If:	([Src1.IsImm]==false)
		Format:	**RegDataType**
	87:84	**Src0.VertStride**	
		Format:	**VertStride**
	83:81	**Src0.Width**	
		Format:	**Width**
	80	**Src0.AddrMode**	
		Format:	**AddrMode**

and - Logic And

	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value. 0 – false **[Default]** 1 – true
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. 0 – false **[Default]** 1 – true
	45:44	**Src0.Mod** Format: **SrcMod**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: **RegDataType**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: **ImmDataType**
	39:36	**Dst.DataType** Format: **RegDataType**

and - Logic And

35 Dst.AddrMode

Format:	AddrMode

34 Saturate

Format:	Saturate

33 AccWrCtrl

Format:	AccWrCtrl

32 AtomicCtrl

Format:	AtomicCtrl

31 MaskCtrl

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 Reserved

29 CmptCtrl

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 PredInv

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the

		predication mask.
	27:24	**PredCtrl** Format: **PredCtrl** This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.
	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Logic Not

not - Logic Not

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	false
Source Modifier:	true

The not instruction performs logical NOT operation (or one's complement) of src0 and storing the results in dst. This operation does not produce sign or overflow conditions. Only the .e/.z or .ne/.nz conditional modifiers should be used.

A register source operand can use a source modifier: Any source modifier is logical, optionally changing a source value s to ~s (inverting all source bits). Such a source modifier is not particularly useful with the not instruction, as it changes the effect of not to just copying bits.

Format:

```
[(pred)] not[.cmod] (exec_size) dst src0
```

Restriction

Source modifier is not allowed if source is an accumulator.

Syntax

```
[(pred)] not[.cmod] (exec_size) reg reg
 [(pred)] not[.cmod] (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         dst.chan[n] = ~ src0.chan[n];
     }
 }
```

Src Types	Dst Types
*B,*W,*D	*B,*W,*D

DWord	Bit	Description
0..3	127:96	**Src0.ImmValue[31:0]** Exists If: ([Src0.IsImm]==true)
	95:92	**CondCtrl** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **FlagModifier**

not - Logic Not

	95:64	**Src0.ImmValue[63:32]** Exists If: ([Src0.IsImm]==true) AND (([Src0.DataType]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df))
	87:84	**Src0.VertStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **VertStride**
	83:81	**Src0.Width** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **Width**
	80	**Src0.AddrMode** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**

not - Logic Not

47	**Reserved**	
	Access:	RO
	Format:	MBZ
46	**Src0.IsImm**	
	This field indicate that Source 0 operand is carrying an immediate value.	
	Value	**Name**
	0	false **[Default]**
	1	true
45:44	**Src0.Mod**	
	Format:	**SrcMod**
43:40	**Src0.DataType**	
	Exists If:	([Src0.IsImm]==false)
	Format:	**RegDataType**
43:40	**Src0.DataType**	
	Exists If:	([Src0.IsImm]==true)
	Format:	**ImmDataType**
39:36	**Dst.DataType**	
	Format:	**RegDataType**
35	**Dst.AddrMode**	
	Format:	**AddrMode**
34	**Saturate**	
	Format:	**Saturate**
33	**AccWrCtrl**	
	Format:	**AccWrCtrl**
32	**AtomicCtrl**	
	Format:	**AtomicCtrl**
31	**MaskCtrl**	
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value / **Name** / **Description**	
	0 / Normal **[Default]** / Normal. Per channel write enable used for final write enable generation.	
	1 / NoMask / NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.	
30	**Reserved**	
29	**CmptCtrl**	
	Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. This can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

not - Logic Not

	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Logic Or

or - Logic Or

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	false
Source Modifier:	true

The or instruction performs component-wise logic OR operation between src0 and src1 and stores the results in dst. This operation does not produce sign or overflow conditions. Only the .e/.z or .ne/.nz conditional modifiers should be used.

Register source operands can use source modifiers: Any source modifier is logical, optionally changing a source value s to ~s (inverting all source bits). This capability allows expressions like a OR (NOT b) to be calculated with one instruction.

Format:

```
[(pred)] or[.cmod] (exec_size) dst src0 src1
```

Restriction

Source modifier is not allowed if source is an accumulator.

Syntax

```
[(pred)] or[.cmod] (exec_size) reg reg reg
 [(pred)] or[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         dst.chan[n] = src0.chan[n] | src1.chan[n];
     }
 }
```

Src Types	Dst Types
*B,*W,*D	*B,*W,*D

DWord	Bit	Description
0..3	127:126	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	127:96	**Src1.ImmValue[31:0]** Exists If: ([Src1.IsImm]==true)

or - Logic Or

	125:122	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	121:120	**Src1.Mod** Exists If: ([Src1.IsImm]==false) Format: **SrcMod**
	119:116	**Src1.VertStride** Exists If: ([Src1.IsImm]==false) Format: **VertStride**
	115:113	**Src1.Width** Exists If: ([Src1.IsImm]==false) Format: **Width**
	112	**Src1.AddrMode** Exists If: ([Src1.IsImm]==false) Format: **AddrMode**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect) Format: **IndirectOperand**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct) Format: **DirectOperand**
	97:96	**Src1.HorzStride** Exists If: ([Src1.IsImm]==false) Format: **HorzStride**
	95:92	**CondCtrl** Format: **FlagModifier**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==true) Format: **ImmDataType**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==false) Format: **RegDataType**
	87:84	**Src0.VertStride** Format: **VertStride**
	83:81	**Src0.Width** Format: **Width**

or - Logic Or

	80	**Src0.AddrMode** Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
	45:44	**Src0.Mod** Format: **SrcMod**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: **RegDataType**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: **ImmDataType**

or - Logic Or

	39:36	**Dst.DataType** Format: **RegDataType**
	35	**Dst.AddrMode** Format: **AddrMode**
	34	**Saturate** Format: **Saturate**
	33	**AccWrCtrl** Format: **AccWrCtrl**
	32	**AtomicCtrl** Format: **AtomicCtrl**
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". **Value** / **Name** / **Description** 0 / Normal **[Default]** / Normal. Per channel write enable used for final write enable generation. 1 / NoMask / NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	30	**Reserved**
	29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. **Value** / **Name** / **Description** 0 / NoCompaction **[Default]** / No compaction. 128-bit native instruction supporting all instruction options. 1 / Compacted / Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
	28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields **Value** / **Name** / **Description** 0 / Positive / Positive polarity of predication. Use the predication mask produced

	[Default]	by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16 **ExecSize**

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

15:0 **Header**

Format:	Header

Logic Xor

xor - Logic Xor

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	false
Source Modifier:	true

The xor instruction performs component-wise logic XOR operation between src0 and src1 and stores the results in dst. This operation does not produce sign or overflow conditions. Only the .e/.z or .ne/.nz conditional modifiers should be used.

Register source operands can use source modifiers: Any source modifier is logical, optionally changing a source value s to ~s (inverting all source bits). This capability allows expressions like a XOR (NOT b) to be calculated with one instruction.

Format:

```
[(pred)] xor[.cmod] (exec_size) dst src0 src1
```

Restriction

Source modifier is not allowed if source is an accumulator.

Syntax

```
[(pred)] xor[.cmod] (exec_size) reg reg reg
 [(pred)] xor[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         dst.chan[n] = src0.chan[n] ^ src1.chan[n];
     }
 }
```

Src Types	Dst Types
*B,*W,*D	*B,*W,*D

DWord	Bit	Description
0..3	127:126	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	127:96	**Src1.ImmValue[31:0]** Exists If: ([Src1.IsImm]==true)

xor - Logic Xor

	125:122	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	121:120	**Src1.Mod** Exists If: ([Src1.IsImm]==false) Format: **SrcMod**
	119:116	**Src1.VertStride** Exists If: ([Src1.IsImm]==false) Format: **VertStride**
	115:113	**Src1.Width** Exists If: ([Src1.IsImm]==false) Format: **Width**
	112	**Src1.AddrMode** Exists If: ([Src1.IsImm]==false) Format: **AddrMode**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect) Format: **IndirectOperand**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct) Format: **DirectOperand**
	97:96	**Src1.HorzStride** Exists If: ([Src1.IsImm]==false) Format: **HorzStride**
	95:92	**CondCtrl** Format: **FlagModifier**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==true) Format: **ImmDataType**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==false) Format: **RegDataType**
	87:84	**Src0.VertStride** Format: **VertStride**
	83:81	**Src0.Width** Format: **Width**

xor - Logic Xor

	80	**Src0.AddrMode** Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
	45:44	**Src0.Mod** Format: **SrcMod**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: **RegDataType**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: **ImmDataType**

	39:36	**Dst.DataType** Format: **RegDataType**
	35	**Dst.AddrMode** Format: **AddrMode**
	34	**Saturate** Format: **Saturate**
	33	**AccWrCtrl** Format: **AccWrCtrl**
	32	**AtomicCtrl** Format: **AtomicCtrl**
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". **Value** / **Name** / **Description** 0 / Normal **[Default]** / Normal. Per channel write enable used for final write enable generation. 1 / NoMask / NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	30	**Reserved**
	29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. **Value** / **Name** / **Description** 0 / NoCompaction **[Default]** / No compaction. 128-bit native instruction supporting all instruction options. 1 / Compacted / Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
	28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16 **ExecSize**

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

15:0 **Header**

Format:	Header

MEDIA_CURBE_LOAD

MEDIA_CURBE_LOAD

Source: RenderCS

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MEDIA_CURBE_LOAD Format: OpCode
	23:16	**SubOpcode** Default Value: 1h MEDIA_CURBE_LOAD SubOp Format: OpCode
	15:0	**DWord Length** Format: =n Value | Name | Description 2h | DWORD_COUNT_n **[Default]** | Excludes DWord (0,1)
1	31:0	**Reserved** Access: RO Format: MBZ
2	31:17	**Reserved** Access: RO Format: MBZ
	16:0	**CURBE Total Data Length** Format: U17 This field provides the length in bytes of the CURBE data. This field must have the same alignment as the Curbe Object Data Start Address. As the CURBE data are sent directly to ROB, range is limited to CURBE Allocation Size. This field must be 64-byte aligned.

<table>
<tr><th colspan="3">MEDIA_CURBE_LOAD</th></tr>
<tr><td>3</td><td>31:0</td><td>CURBE Data Start Address
<table>
<tr><td>Format:</td><td>DynamicStateOffset[31:0]</td></tr>
</table>
Specifies the 64-byte aligned address of the CURBE data. This pointer is relative to the Dynamics Base Address.
<table>
<tr><th>Value</th><th>Name</th></tr>
<tr><td>[0, FFFFFFFFh]</td><td></td></tr>
</table>
</td></tr>
</table>

MEDIA_INTERFACE_DESCRIPTOR_LOAD

MEDIA_INTERFACE_DESCRIPTOR_LOAD

Source: RenderCS

Length Bias: 2

A Media_State_Flush should be used before this command to ensure that the temporary Interface Descriptor storage is cleared.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MEDIA_INTERFACE_DESCRIPTOR_LOAD Format: OpCode
	23:16	**SubOpcode** Default Value: 2h MEDIA_INTERFACE_DESCRIPTOR_LOAD SubOp Format: OpCode
	15:0	**DWord Length** Format: =n Value / Name / Description 2h / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)
1	31:0	**Reserved** Access: RO Format: MBZ
2	31:17	**Reserved** Access: RO Format: MBZ
	16:0	**Interface Descriptor Total Length** Format: U17 This field provides the length in bytes of the Interface Descriptor data. This field must have the same alignment as the Interface Descriptor Data Start Address. It must be DQWord (32-byte) aligned. As the Interface Descriptor data are sent directly to ROB, range is limited to CURBE Allocation Size.

MEDIA_INTERFACE_DESCRIPTOR_LOAD

		Value / **Name** [32,2048] / [1,64] interface descriptor entries
3	31:0	**Interface Descriptor Data Start Address** Format: DynamicStateOffset[31:0]INTERFACE_DESCRIPTOR_DATA This bit specifies the 64-byte aligned address of the Interface Descriptor data. This pointer is relative to the Dynamics Base Address. **Value** / **Name** [0, FFFFFFFFh] /

MEDIA_OBJECT_GRPID

MEDIA_OBJECT_GRPID

Source: RenderCS

Length Bias: 2

The MEDIA_OBJECT_GRPID command is a variation of MEDIA_OBJECT which includes a group id which is used to allocate and track Barriers and Shared Local Memory. The Interface Descriptor is used to specify how much SLM is needed and how many threads will be reporting to the Barrier. All MEDIA_OBJECT_GRPIDs with the same group id should have the same interface descriptor and be dispatched to the same Tslice the dispatcher will ensure this if Force Destination = 0, but software must ensure this if Force Destination = 1. Software should also ensure that all the threads needed for the Barrier will fit into a Tslice, or the Barrier will never be satisfied. Either SLM or a barrier must be used with MEDIA_OBJECT_GRPID, if neither is needed then a MEDIA_OBJECT must be used instead.

MEDIA_OBJECT_GRPID supports the GPGPU version of payload delivery either indirect or CURBE can be split between the threads in a group (per-thread payload), as well as a section which is sent to all threads (cross-thread payload). See the GPGPU payload section. For indirect, the same pointer must be sent with all the commands associated with the thread group for payload splitting to work properly. Inline data is not split, but the payload attached to each command is sent with that thread. Only one of inline, indirect, or CURBE is allowed, but at least one form of payload must be sent.

MEDIA_STATE_FLUSH with the watermark bit must be placed between groups created by MEDIA_OBJECT_GRPID. The Interface Descriptor associated with the watermark must match the Interface Descriptor used for the following group.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Media Command Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h MEDIA_OBJECT_GRPID Format: OpCode
	23:16	**Media Command Sub-Opcode** Default Value: 6h MEDIA_OBJECT_GRPID SubOp Format: OpCode
	15:0	**DWord Length** Format: =n Excludes DWords 0,1 **Generic Mode:** DWord Length = N+5, where N is in the range of [0,504]. The maximum is 504 DW (equivalent to 63 8-DW registers). When both inline and indirect data are fetched for this command, the total size in 8-DW registers must be less than 112 (with both inline data length N and indirect data length rounded up to 8-DW aligned individually). The

MEDIA_OBJECT_GRPID

		minimal inline data length is 0. Value: 5h — Name: DWORD_COUNT_n **[Default]** Value: [5,112] — Name: Min_Max
1	31:8	**Reserved**
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**Interface Descriptor Offset** Format: U6 This field specifies the offset from the interface descriptor base pointer to the interface descriptor which will be applied to this object. It is specified in units of interface descriptors. Value: [0,30] — Name:
2	31:24	**Reserved** Access: RO Format: MBZ
	23	**End of Thread Group** This bit indicates that this dispatch is the last for the current thread group.
	22:17	**Reserved** Access: RO Format: MBZ
	16:0	**Indirect Data Length** Format: U17 This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address. It must be DQWord (32-byte) aligned. As the indirect data are sent directly to URB, range is limited to(URB Entry Allocation Size)*(Number of URB Entries) in the MEDIA_VFE_STATE command.
3	31:0	**Indirect Data Start Address** Format: GraphicsAddress[31:0] This field specifies the Graphics Memory starting address of the data to be loaded into the kernel for processing. This pointer is relative to the **Indirect Object Base Address**. Hardware ignores this field if indirect data is not present. Alignment of this address depends on the mode of operation. It is the 64-byte aligned address of the indirect data.

MEDIA_OBJECT_GRPID

		Value	Name	Description
		[0-512MB]		Bits 31:29 MBZ
4	31:25	**Reserved** Access: RO Format: MBZ		
	24:16	**Y Position** Format: U9 This field provides the Y position of the block to be dispatched. It is sent the kernel via the R0 header, but is not otherwise used by hardware.		
	15:9	**Reserved** Access: RO Format: MBZ		
	8:0	**X Position** Format: U9 This field provides the X position of the block to be dispatched. It is sent to the kernel via the R0 header, but is not otherwise used by hardware.		
5	31:24	**Reserved** Access: RO Format: MBZ		
	23:16	**Block Color** Format: U8 This field specifies the color value associated with the block to be dispatched. It is sent to the kernel via the R0 header, but is not otherwise used by hardware.		
	15:0	**Reserved** Access: RO Format: MBZ		
6	31:0	**GroupID** A unique identifying number which describes the threads which share a barrier and/or SLM. Reuse of numbers is allowed as long as the old group is not currently running.		
7..n	31:0	**Inline Data** The format of this data is specified by software. Hardware does not interpret this data; it merely passes it to the kernel for processing. The total size for the inline data and indirect data must not exceed 112 registers.		

MEDIA_OBJECT

MEDIA_OBJECT

Source: RenderCS

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Media Command Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h MEDIA_OBJECT Format: OpCode
	23:16	**Media Command Sub-Opcode** Default Value: 0h MEDIA_OBJECT SubOp Format: OpCode
	15	**Reserved** Access: RO Format: MBZ
	14:0	**DWord Length** Default Value: [4,116] Min_Max Format: =n Excludes DWords 0,1 **Generic Mode:** DWord Length = N+4, where N is in the range of [0,504]. The maximum is 504 DW (equivalent to 63 8-DW registers). When both inline and indirect data are fetched for this command, the total size in 8-DW registers must be less than 112 (with both inline data length N and indirect data length rounded up to 8-DW aligned individually). The minimal inline data length is 0.
1	31:8	**Reserved**
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**Interface Descriptor Offset** Format: U6 This field specifies the offset from the interface descriptor base pointer to the interface descriptor which will be applied to this object. It is specified in units of interface descriptors.

MEDIA_OBJECT

2 | 31:27 | Reserved

Access:	RO
Format:	MBZ

26:25 | Slice Destination Select MSBs

These bits are the MSBs of the slice destination select field. The LSB bits are 20:19. Only slices that are available can be selected.

24 | Thread Synchronization

This field when set indicates that the dispatch of the thread originated from this command is based on the "spawn root thread" message.

Value	Name
0	No thread synchronization
1	Thread dispatch is synchronized by the 'spawn root thread' message

23 | Reserved

Access:	RO
Format:	MBZ

22 | Force Destination

If set, bits 20:17 are used to determine the destination of this dispatch, if clear the destination will be chosen based on load.

21 | Reserved

Access:	RO
Format:	MBZ

20:17 | Subslice Destination Select

Format:	U4

This field selects the SubSlice that this thread must be sent to. This configuration has only 1 slice. Ignored if **Force Destination** = 0.

Value	Name	Description
[0-7]	Subslice ID	Valid subslice number in this configuration.
[8-15]	Reserved	Illegal values. Results undefined if used.

16:0 | Indirect Data Length

Format:	U17

This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address. It must be DQWord (32-byte) aligned. As the indirect data are sent directly to URB, range is limited to (URB Entry Allocation Size)*(Number of URB Entries) in the MEDIA_VFE_STATE command. When both inline and indirect data are fetched for this command, enough room should be allowed for at least 1 copy of the indirect and 1 copy of the inline data in the URB.

3 | 31:0 | Indirect Data Start Address

Format:	GraphicsAddress[31:0]

MEDIA_OBJECT

		This field specifies the Graphics Memory starting address of the data to be loaded into the kernel for processing. This pointer is relative to the **Indirect Object Base Address**. Hardware ignores this field if indirect data is not present. Alignment of this address depends on the mode of operation. This field specifies the 64-byte aligned address of the indirect data. **Value** / **Name**: [0,512MB] **Programming Notes**: Bits 31:29 MBZ
4	31:25	**Reserved** Access: RO Format: MBZ
	24:16	**Y Position** Format: U9 This field provides the Y position of the block dispatched by this command. It is sent to the kernel via the R0 header but is not otherwise used by hardware.
	15:9	**Reserved** Access: RO Format: MBZ
	8:0	**X Position** Format: U9 This field provides the X position of the block dispatched by this command. It is sent to the kernel via the R0 header but is not otherwise used by hardware.
5	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**Block Color** Format: U8 This field specifies the color for the block being dispatched. It is sent to the kernel via the R0 header, but is not otherwise used by hardware.
	15:0	**Reserved** Access: RO Format: MBZ

MEDIA_OBJECT		
6..n	31:0	**Inline Data** Generic Mode: The format of this data is specified by software. Hardware does not interpret this data; it merely passes it to the kernel for processing. The total size for the inline data and indirect data must not exceed 112 registers.

MEDIA_OBJECT_WALKER

MEDIA_OBJECT_WALKER

Source: RenderCS

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h MEDIA_OBJECT_WALKER Format: OpCode
	23:16	**SubOpcode** Default Value: 03h MEDIA_OBJECT_WALKER SubOp Format: OpCode
	15	**Reserved** Access: RO Format: MBZ
	14:0	**DWord Length** Default Value: [15,112] Min_Max Format: =n **Note:** If this field is greater than 15, it indicates that inline data is present. If present, inline data is common for all threads generated from this command, If this field is 15, it indicates that inline data is not present. It should be noted that unlike other media object command, inline data is optional for this command.
1	31:8	**Reserved**
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**Interface Descriptor Offset** Format: U6 This field specifies the offset from the interface descriptor base pointer to the interface descriptor which will be applied to this object. It is specified in units of interface descriptors.

MEDIA_OBJECT_WALKER

2 — 31:25 — Reserved

Access:	RO
Format:	MBZ

2 — 24 — Thread Synchronization

This field when set indicates that the dispatch of the thread originated from this command is based on the "spawn root thread" message.

Value	Name
0	No thread synchronization
1	Thread dispatch is synchronized by the 'spawn root thread' message

2 — 23:22 — Masked Dispatch

Format:	U2

Enable the masking of the dispatch of individual threads based on a bitmask read from CURBE, and specifies the pitch of the CURBE surface. If enabled, CURBE will not be used for thread payload.

Value	Name	Description
00b		Masked Dispatch Disabled
01b		Masked Dispatch with 128-bit pitch in CURBE
10b		Masked Dispatch with 256-bit pitch in CURBE
11b		Masked Dispatch with 512-bit pitch in CURBE

2 — 21:17 — Reserved

Access:	RO
Format:	MBZ

2 — 16:0 — Indirect Data Length

Format:	U17

This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address. It must be DQWord (32-byte) aligned. As the indirect data are sent directly to a URB entry, the range is limited to the URB Entry Allocation Size specified in the MEDIA_VFE_STATE command.

3 — 31:0 — Indirect Data Start Address

This field specifies the Graphics Memory starting address of the data to be loaded into the kernel for processing. This pointer is relative to the **Indirect Object Base Address.** Hardware ignores this field if indirect data is not present. Alignment of this address depends on the mode of operation.

It is the 64-byte aligned address of the indirect data

Value	Name	Description
[0 - 512MB]		(Bits 31:29 MBZ)

4 | 31:0 | Reserved

Access:	RO
Format:	MBZ

5 | 31:8 | Group ID Loop Select

This bit field chooses which of the nested loops of the walker are used to identify threads which share a group id and therefore a shared barrier and SLM. The programmer must ensure that each group will fit into a single subslice. When barriers are enabled every group must have the same number of threads matching the number specified in the Interface Descriptor.

Value	Name	Description
0	No_Groups	Groups are not created, barriers and SLM are not allocated
2	InnerLocal_Groups	Each complete iteration of the Inner local loop and Color loop defines a group, the group id is the concatenation of the Outer global loop to the Mid local loop execution counts.
3	MidLocal_Groups	Each complete iteration of the Mid local loop and lower loops defines a group, the group id is the concatenation of the Outer global loop to the Outer local loop execution counts.
4	OuterLocal_Groups	Each complete iteration of the Outer local loop and lower loops defines a group, the group id is the concatenation of the Outer global loop and the Inner global loop execution counts.
5	InnerGlobal_Groups	Each complete iteration of the Inner global loop and lower loops defines a group, the group id is the Outer global loop execution count.
Others	Reserved	

Restriction
When media thread groups are formed (i.e. Group ID Loop Select is non-zero), software must ensure the size of the group does not exceed the maximum number of threads in a DSS (112).

7:0 | Partition Count

Format:	U8-1

This field specifies the number of partitions (minus one) used to select thread destinations. The walk position **Color** modulo the **Partition Count** chooses a non-overlapping set of DSS that the dispatch is directed to.

This field is ignored if **MEDIA_VFE_STATE Dispatch Load Balance** is not set to Color mode.

Value	Name
[0-7]	Supported Partition Counts

Programming Notes
For most efficient use of all the DSS, the **Partition Count** should be an even factor of the HW configuration's number of DSS. Also, the **Color Count** should be a multiple of the **Partition Count**.

MEDIA_OBJECT_WALKER

6	31:24	**Color Count Minus One** Format: U8 This field specifies the number of repeat of the inner most loop of the walker. Each repeated walk position is assigned with an incremental Color number. The Color number together with the X and Y position of the thread is used for dependency scoreboard control. Usage Example: This allows multiple sets of dependency threads to be dispatched.
	23:21	**Reserved** Access: RO Format: MBZ
	20:16	**Middle Loop Extra Steps** Format: U5
	15:14	**Reserved** Access: RO Format: MBZ
	13:12	**Local Mid-Loop Unit Y** Format: S1
	11:10	**Reserved** Access: RO Format: MBZ
	9:8	**Mid-Loop Unit X** Format: S1
	7:0	**Reserved** Access: RO Format: MBZ
7	31:28	**Reserved** Access: RO Format: MBZ
	27:16	**Global Loop Exec Count** Format: U12
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Local Loop Exec Count** Format: U12
8	31:27	**Reserved** Access: RO

MEDIA_OBJECT_WALKER		
		Format: MBZ
	26:16	**Block Resolution Y** Format: U11 Vertical resolution of the local loop.
	15:11	**Reserved** Access: RO Format: MBZ
	10:0	**Block Resolution X** Format: U11 Horizontal resolution of the local loop.
9	31:27	**Reserved** Access: RO Format: MBZ
	26:16	**Local Start Y** Format: U11 Starting vertical position of the local loop.
	15:11	**Reserved** Access: RO Format: MBZ
	10:0	**Local Start X** Format: U11 Starting horizontal position of the local loop.
10	31:0	**Reserved** Access: RO Format: MBZ
11	31:28	**Reserved** Access: RO Format: MBZ
	27:16	**Local Outer Loop Stride Y** Format: S11 Vertical stride of the local outer loop, in 2's complement.

MEDIA_OBJECT_WALKER

	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Local Outer Loop Stride X** Format: S11 Horizontal stride of the local outer loop, in 2's complement.
12	31:28	**Reserved** Access: RO Format: MBZ
	27:16	**Local Inner Loop Unit Y** Format: S11 Vertical stride of the local inner loop, in 2's complement.
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Local Inner Loop Unit X** Format: S11 Horizontal stride of the local inner loop, in 2's complement.
13	31:27	**Reserved** Access: RO Format: MBZ
	26:16	**Global Resolution Y** Format: U11 Vertical resolution of the global loop.
	15:11	**Reserved** Access: RO Format: MBZ
	10:0	**Global Resolution X** Format: U11 Horizontal resolution of the global loop.
14	31:28	**Reserved** Access: RO Format: MBZ

MEDIA_OBJECT_WALKER

	27:16	**Global Start Y** Format: S11 Starting vertical location of the global loop, in 2's complement.
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Global Start X** Format: S11 Starting horizontal location of the global loop, in 2's complement.
15	31:28	**Reserved** Access: RO Format: MBZ
	27:16	**Global Outer Loop Stride Y** Format: S11 Vertical stride of the global outer loop, in 2's complement.
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Global Outer Loop Stride X** Format: S11 Horizontal stride of the global outer loop, in 2's complement.
16	31:28	**Reserved** Access: RO Format: MBZ
	27:16	**Global Inner Loop Unit Y** Format: S11 Vertical stride of the global inner loop, in 2's complement.
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Global Inner Loop Unit X** Format: S11 Horizontal stride of the global inner loop, in 2's complement.
17..n	31:0	**Inline Data**

MEDIA_STATE_FLUSH

MEDIA_STATE_FLUSH

Source: RenderCS

Length Bias: 2

This command updates the Message Gateway state. In particular, it updates the state for a selected Interface Descriptor.
This command can be considered same as a MI_Flush except that only media parser will get flushed instead of the entire 3D/media render pipeline. The command should be programmed prior to new Media state, curbe and/or interface descriptor commands when switching to a new context or programming new state for the same context. With this command, pipelined state change is allowed for the media pipe.
Be cautious when using this command when child_present flag in the media state is enabled. This is because that CURBE state as well as Interface Descriptor state are shared between root threads and child threads. Changing these states while child threads are generated on the fly may cause unexpected behavior. Combining with MI_ARB_ON/OFF command, it is possible to support interruptability with the following command sequence where interrupt may be allowed only when MI_ARB_ON_OFF is ON:

```
MEDIA_STATE_FLUSH VFE_STATE // VFE will hold CS if watermark isn't met MI_ARB_OFF // There must be at least one VFE command before this oneMEDIA_OBJECT ... MI_ARB_ON
```

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MEDIA_STATE_FLUSH Format: OpCode
	23:16	**SubOpcode** Default Value: 4h MEDIA_STATE_FLUSH SubOp Format: OpCode
	15:0	**DWord Length** Format: =n Value / Name / Description 0h / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)
1	31:8	**Reserved** Access: RO Format: MBZ

MEDIA_STATE_FLUSH		
	7	**Flush to GO** Format: Enable This bit indicates that the write data out of this thread group should be flushed to the point where it is visible to following commands.
	6	**Reserved** Access: RO Format: MBZ
	5:0	**Interface Descriptor Offset** Format: U6 This field specifies the offset from the interface descriptor base pointer to the interface descriptor which describes what resources are required to meet the watermark.

MEDIA_VFE_STATE

MEDIA_VFE_STATE

Source: RenderCS

Length Bias: 2

A stalling PIPE_CONTROL is required before MEDIA_VFE_STATE unless the only bits that are changed are scoreboard related: Scoreboard Enable, Scoreboard Type, Scoreboard Mask, Scoreboard * Delta. For these scoreboard related states, a MEDIA_STATE_FLUSH is sufficient.

- MEDIA_STATE_FLUSH (optional, only if barrier dependency is needed)
- MEDIA_INTERFACE_DESCRIPTOR_LOAD (optional)

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MEDIA_VFE_STATE Format: OpCode
	23:16	**SubOpcode** Default Value: 0h MEDIA_VFE_STATE SubOp Format: OpCode
	15:0	**DWord Length** Format: =n Value / Name / Description 07h / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)
1	31:10	**Scratch Space Base Pointer** Format: GeneralStateOffset[31:10] Specifies the 1k-byte aligned address offset to scratch space for use by the kernel. This pointer is relative to the **General State Base Address.**
	9:8	**Reserved** Access: RO Format: MBZ

MEDIA_VFE_STATE

	7:4	**Stack Size** **Value** / **Name** / **Description** [0,11] / / indicating [1KBytes, 2MBytes] **Programming Notes** Since the stack uses the upper portion of the scratch space, **Stack Size =< Per Thread Scratch Space**
	3:0	**Per Thread Scratch Space** Format: U4 Specifies the amount of scratch space allowed to be used by each thread. **Value** / **Name** / **Description** [0,11] / / indicating [1k bytes, 2 Mbytes]: 0=1k, 1=2k, 2=4k, 3=8k 11=2M **Programming Notes** The driver must allocate enough contiguous scratch space, pointed to by the Scratch Space Pointer, to ensure that all threads have their own per-thread scratch space. Scratch Space Size = (MEDIA_VFE_STATE::Per Thread Scratch Space) * (MEDIA_VFE_STATE::Maximum Number Of Threads). If a fused configuration has fewer threads than the native POR configuration, the scratch space allocation is based on the number of threads in the base native POR configuration.
2	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Scratch Space Base Pointer High** Format: GeneralStateOffset[47:32] This field specifies the high 16 bits of starting address of the Scratch Space Base Pointer
3	31:16	**Maximum Number of Threads** Format: U16-1 **Description** Range: [0, 2^16-1], representing [1, 2^16] threads. Normally set to the maximum number of threads: (# EUs) * (# threads/EU). See *Graphics Processing Engine* for listing of #EUs and #threads in each device. See Programming Restrictions here for additional limitations. Restriction : The smallest number of maximum threads supported is 64. The largest number may exceed the number of threads on the GPU, but must be <= (#Slices)*1024. (Range of FFTID per slice is 10 bits.)

Programming Notes
MSB will be zero due to the range limit below.

15:8 **Number of URB Entries**

Format:	U8

Specifies the number of URB entries that are used by the unit.

Value	Name	Description
[1,128]		[1,128] Entries

Programming Notes
Please note that 0 is not allowed for this field.

7 **Reserved**

Access:	RO
Format:	MBZ

6 **Fused EU Dispatch**

Format:	Disable

This field determine if threads will be dispatched in sets to fused EUs if set or if they will be dispatched individually. Depending on the project the set size can be 2 or 4. If dispatched in sets the fused threads will all be part of the same thread group for GPGPU threads or will be part of the same iteration of the inner local loop if media threads.

Value	Name
1h	Legacy Mode, threads are not fused
0h	Fused EU Mode

5:3 **Reserved**

Access:	RO
Format:	MBZ

2 **Dispatch Load Balance**

This bit determines how media threads are dispatched between the various dual subslices. GPGPU threads are not impacted by this bit.

Value	Name	Description
1	Color	When this value is used the threads are split into groups depending on the color value for media threads. The number of groups is specified by **MEDIA_OBJECT_WALKER Partition Count**. Each group is sent to non-overlapping dual subslices. The least loaded active dual subslice available for the group will be selected for all the threads in that group. This allows color to be used to separate workloads with different operations to get better cache coherency.
0	Least Loaded	When this value is used the threads are sent to the least loaded dual subslice of all active dual subslices. If media with groups is being used then each group is kept in the same dual subslice.

MEDIA_VFE_STATE

	1:0	**Reserved**
4	31:8	**Reserved**
	7:0	**Maximum Number of Dual-Subslices** This field determines the maximum number of dual subslices which are allowed to be used. Lowering this value below the maximum number of dual subslices available will reduce the memory footprint for context and scratch space, as well as potentially saving power. A zero in this field indicates that all available dual subslices should be enabled. Setting this field to a value larger or equal to the available dual subslices will also enable all of them.
5	31:16	**URB Entry Allocation Size** Format: U16 Specifies the length of each URB entry used by the unit, in 256-bit (32 byte) register increments. Each URB entry is rounded up to be 64-byte aligned. ROB address for URB starts after CURBE Allocated region. **Programming Notes** When Inline data is used with MEDIA_OBJECT or MEDIA_OBJECT_WALKER, then the URB entry allocation size must match the Inline data size. If Indirect data is being used with MEDIA_OBJECT or GPGPU_WALKER then the allocation size must be sufficient for the Indirect data. If both Inline and Indirect are being used, then the allocation size must match the sum of the Inline and Indirect. The total size of URB used by VFE must be less than number of bytes allocated for the URB in the L3 banks (value in L3CNTLREG / 32 bytes per entry). The total size of URB used by VFE = Number of Interface Descriptors (=64) + **CURBE Allocation Size** + (**Number of URB Entrie**s * (roundup(**URB Entry Allocation Size**, 64))) **Restriction** The total URB allocation size cannot be greater than 32 KB. This ensures that VFE does not overwrite POSH push constants when both POSH and RCS Compute pipes are active. The total URB allocation size for CCS contexts cannot be greater than 32 KB. The total URB allocation size cannot be greater than 1024kB.
	15:0	**CURBE Allocation Size** Format: U16 Specifies the total length allocated for CURBE, in 256-bit (32 byte) register increments. Each URB entry is rounded up to be 64-byte aligned. ROB address for CURBE starts at address 64. **Programming Notes** CURBE Allocation Size should be 0 for GPGPU and Media workloads that use indirect instead of CURBE. See programming notes and restrictions on the **URB Entry Allocation Size** for limitations on the total URB size that apply to CURBE Allocation Size.
6	31:0	**Reserved** Access: RO Format: MBZ

MEDIA_VFE_STATE		
7	31:0	**Reserved** Access: RO Format: MBZ
8	31:0	**Reserved** Access: RO Format: MBZ

Media Block Read MSD

MSD1R_MB - Media Block Read MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 04h Format: Opcode Media Block Read message
	13:11	**Reserved** Access: RO Format: MBZ
	10:8	**Vertical Line Stride Override** Format: MDC_VLSO If enabled, specifies the Vertical Line Stride and Vertical Line Stride Offset override fields.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Media Block Write MSD

MSD1W_MB - Media Block Write MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 0Ah Format: Opcode Media Block Write message
	13:11	**Reserved** Access: RO Format: MBZ
	10:8	**Vertical Line Stride Override** Format: MDC_VLSO If enabled, specifies the Vertical Line Stride and Vertical Line Stride Offset override fields.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Media Transpose Read MSD

MSD1R_TT - Media Transpose Read MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18:14	**Message Type** Default Value: 00h Format: Opcode Transpose Read message
	13:8	**Reserved** Access: RO Format: MBZ
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Memory Fence MSD

MSD_MEMFENCE - Memory Fence MSD

Source: EuSubFunctionDataPort0

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP Indicates that the message requires a header.
	18	**Legacy Message** Default Value: 0h Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 07h Format: Opcode Memory Fence message
	13	**Commit** Format: Enable Specifies whether control is returned to the thread only after the fence has been honored. Value / Name / Description 1 / Enabled **[Default]** / The commit writeback register is always required to guarantee ordering.

MSD_MEMFENCE - Memory Fence MSD

0	Reserved	The commit writeback register is always required to guarantee ordering.

12:9 Reserved

Access:	RO
Format:	MBZ

8 L1 Flush Data

Format:	Enable

When set, invalidate this subslice's L1 read-only data cache.

Restriction
When "L1 Flush Data" is set, the "L3 Flush" field must be set to 0. If both L1 and L3 needs to be invalidated/flushed, SW need to send two separate fence messages.

7:0 Binding Table Index

Format:	MDC_BTS_SLM_A32

Specifies whether global memory or shared local memory (SLM) is fenced with this operation.

Value	Name	Description
0FEh	SLM	Only SLM is fenced with this operation. Global memory is not fenced. Restriction : The L3 Flush and L1 Flush fields are ignored when SLM memory is selected.
00h	Global Memory **[Default]**	Only global memory is fenced with this operation. SLM memory is not fenced. Performance : When a program needs to guarantee that all global memory writes are globally observable before a thread retires, a Memory Fence operation is used immediately before the EOT message.

MFC_AVC_PAK_OBJECT

MFC_AVC_PAK_OBJECT

Source: VideoCS

Length Bias: 2

The MFC_AVC_PAK_OBJECT command is the second primitive command for the AVC Encoding Pipeline. The same command is used for both CABAC and CAVLC modes. The MV Data portion of the bitstream is loaded as indirect data object.Before issuing a MFC_AVC_PAK_OBJECT command, all AVC MFX states need to be valid. Therefore the commands used to set these states need to have been issued prior to the issue of this command.MB record must be consecutive with no gaps, hence we do not need MB(x,y) in each MB command. Internal counter will keep track of the current MB address, starting from the Start_MB_In_Slice loaded at the beginning of each slice. MFC_AVC_PAK_OBJECT command follows the MbType definition like MFD. Many fields in this command are identical to that in VME output. This is intended to reduce software converting overhead from VME to PAK. Encoding statistical data such as the total size of the output bitstream are provided through MMIO registers. Software may access these registers through MI_STORE_REGISTER_MEM command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFC_AVC_PAK_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h AVC_ENC Format: OpCode
	23:21	**SubOpcode A** Default Value: 2h Format: OpCode
	20:16	**SubOpcode B** Default Value: 9h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 000Ah DWORD_COUNT_n Format: =n

MFC_AVC_PAK_OBJECT

1	31:10	**Reserved** Access: RO Format: MBZ
	9:0	**Indirect PAK-MV Data Length** This field provides the length in bytes of the indirect data, which contains all the MVs for the current MB (in any partitioning and subpartitioning form). A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect PAK-MV Data Start Address field is ignored. This field must have the same alignment as the Indirect PAK-MV Data Start Address. This field must be DW aligned (since each MV is 4 bytes in size).Driver has to derived this field from MVsize (MVquantity in DXVA, exact size) *4 bytes per MV.
2	31:29	**Reserved** Access: RO Format: MBZ
	28:0	**Indirect PAK-MV Data Start Address Offset** This field specifies the memory starting address (offset) of the MV data to be fetched into PAK Subsystem for processing. This pointer is relative to the MFC Indirect PAK-MV Object Base Address. Hardware ignores this field if indirect data is not present, i.e. the Indirect PAK-MV Data Length is set to 0. It is a Dword aligned address in all AVC encoding configuration, since each MV is 4 bytes in size. Value: [0,512MB) — Name:
3..10	255:0	**Inline Data** Format: U32[8] All the required MB level controls and parameters for encoding are captured as inline data of the MFC_AVC_PAK_OBJECT command. It has a fixed size of 8 DWs. Its definition is described in the next section.
11	31:0	**Reserved** Access: RO Format: MBZ
12..23	383:0	**VDEnc Mode Inline Data** Format: U32[12] In VDEnc mode, PAK gets inline MVs. These DWs are placed in the PAK Object command in-order to facilitate PAK stand-alone validation mode. Its definition is described in the next section.

MFC_JPEG_HUFF_TABLE_STATE

MFC_JPEG_HUFF_TABLE_STATE

Source: VideoCS

Length Bias: 2

This Huffman table commands contains both DC and AC tables for either luma or chroma. Once a Huffman table has been defined for a particular destination, it replaces the previous tables stored in that destination and shall be used in the remaining Scans of the current image. Two Huffman tables for luma and chroma will be sent to H/W, and chroma table is used for both U and V.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFC_JPEG_HUFF_TABLE_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 7h JPEG Format: OpCode
	23:21	**SubOpcode A** Default Value: 2h Common Format: OpCode
	20:16	**SubOpcode B** Default Value: 3h MEDIA_ Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 0AEh Excludes DWord (0,1) Format: =n
1	31:1	**Reserved** Access: RO Format: MBZ
	0	**Huff Table ID** Format: U1 Huffman table destination identifier will specify one of two destinations at the encoder into which the Huffman table must be stored.

MFC_JPEG_HUFF_TABLE_STATE

		Value	Name	Description
		0		Huffman table 0
		1		Huffman table 1
2..13	383:0	**DC_TABLE** 12 categories with code length and code word. Each run/size has 1-byte code length, and 2-byte code word.		
14..175	5183:0	**AC_TABLE** 162 run/size with code length and code word. Each run/size has 1-byte code length, and 2-byte code word.		

MFC_JPEG_SCAN_OBJECT

MFC_JPEG_SCAN_OBJECT

Source: VideoCS

Length Bias: 2

Encoder Only

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFC_JPEG_SCAN_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 7h JPEG_ENC Format: OpCode
	23:21	**SubOpcode A** Default Value: 2h Format: OpCode
	20:16	**SubOpcode B** Default Value: 9h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 001h Excludes DWord (0,1) Format: =n
1	31:26	**Reserved** Access: RO Format: MBZ
	25:0	**MCU Count** Format: U26 This field indicates the number of MCUs in the Scan.MCU Count = M_x *X* M_y The number of MCUs in a row: M_x= *(X + (*H_1 **8 -1)) / (*H_1 **8)*The number of MCUs in a column: M_y= *(Y + (*V_1**8 -1)) / (*V_1**8)X*: The number of samples per line in Y-image*Y*: The number of lines in Y-image*H1*: Horizontal sampling factor of Y-image in the Frame header*V1*: Vertical sampling factor of Y-image in the Frame header

MFC_JPEG_SCAN_OBJECT

2 — 31:25 Reserved

Access:	RO
Format:	MBZ

24:22 Huffman AC Table

AC Huffman table destination selector specifies one of two possible AC table destinations for each Y, U, V, or R, G, B. The AC Huffman tables must have been loaded in destination 0 and 1 by the time of issuing MFC_JPEG_HUFF_TABLE_STATE Command.

If AC table 0 is used for Y and AC table 1 is used for U and V, it will be set to 110b. If AC table 0 is used for R, G, and B, it will be set to 000b and so on. Refer to the table below for the summary of actions.

Value	Name	Description
0XXb	Bit24 (V0)	The third image component must use the AC table 0.
1XXb	Bit24 (V1)	The third image component must use the AC table 1.
X0Xb	Bit23 (U0)	The second image component must use the AC table 0.
X1Xb	Bit23 (U1)	The second image component must use the AC table 1.
XX0b	Bit22 (Y0)	The first image component must use the AC table 0.
XX1b	Bit22 (Y1)	The first image component must use the AC table 1.

Restriction
When InputSurfaceFormatYUV = RGB, because the order of input image components can be RGB, GBR, BGR,\ or YUV, **Bit22** is used for the first image component, **Bit23** is used for the second image component, and **Bit24** is used for the third image component.

21 Reserved

Access:	RO
Format:	MBZ

20:18 Huffman DC Table

DC Huffman table destination selector specifies one of two possible DC table destinations for each Y, U, V, or R, G, B. The DC Huffman tables shall have been loaded in destination 0 and 1 by the time of issuing MFC_JPEG_HUFF_TABLE_STATE Command.

if DC table 0 is used for Y and DC table 1 is used for U and V, it will be set to 110b. If DC table 0 is used for R, G, and B, it will be set to 000b and so on. Refer to the table below for the summary of actions.

Value	Name	Description
0XXb	Bit20 (V0)	The third image component must use the DC table 0.
1XXb	Bit20 (V1)	The third image component must use the DC table 1.
X0Xb	Bit19 (U0)	The second image component must use the DC table 0.
X1Xb	Bit19 (U1)	The second image component must use the DC table 1.

XX0b	Bit18 (Y0)	The first image component must use the DC table 0.
XX1b	Bit18 (Y1)	The first image component must use the DC table 1.

Restriction
When InputSurfaceFormatYUV = RGB, because the order of input image components can be RGB, GBR, BGR, YUV, **Bit18** is used for the first image component, **Bit19** is used for the second image component, and **Bit20** is used for the third image component.

17 **Head Present Flag**

If this flag is set to 0, then no MFC_JPEG_PAK_INSERT_OBJECT commands will be sent. If this flag is set to 1, then one or more MFC_JPEG_PAK_INSERT_OBJECT commands will be sent after MFC_JPEG_SCAN_OBJECT command.

Value	Name	Description
0		No insertion into the output bitstream buffer before Scan encoded bitstream
1		Headers, tables, App data insertion into the output bitstream buffer. HW will insert the insertion data before the Scan encoded bitstream.

16 **Is Last Scan**

If this flag is set, then HW will insert EOI (0xFFD9) to the end of Scan encoded bitstream.

Value	Name	Description
0		Not the last Scan.
1		Indicates that the current Scan is the last one.

15:0 **Restart Interval**

Format:	U16

Specifies the number of MCUs in an ECS, except for the last ECS. Restart maker is inserted periodically and it separates the two neighboring ECSs.

Value	Name
0-FFFFh	

Programming Notes
A value of '0' implies that the Scan Data has a single ECS.

MFC_MPEG2_PAK_OBJECT

MFC_MPEG2_PAK_OBJECT

Source: VideoCS

Length Bias: 2

The MFC_MPEG2_PAK_OBJECT command is the second primitive command for the MPEG-2 Encoding Pipeline. Different from AVC, the MV Data portion of the bitstream is loaded as part of MB control data.
Before issuing a MFC_MPEG2_PAK_OBJECT command, all MPEG2_MFX states need to be valid. Therefore the commands used to set these states need to have been issued prior to the issue of this command.
MB record must be consecutive with no gaps, hence we do not need MB(x,y) in each MB command. Internal counter will keep track of the current MB address, starting from the Start_MB_In_Slice loaded at the beginning of each slice.
MFC_ MPEG2_PAK_OBJECT command follows the MbType definition like MFD. Many fields in this command are identical to that in VME output. This is intended to reduce software converting overhead from VME to PAK.
Encoding statistical data such as the total size of the output bitstream are provided through MMIO registers. Software may access these registers through MI_STORE_REGISTER_MEM command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFC_AVC_PAK_INSERT_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 3h MPEG2 Format: OpCode
	23:21	**SubOpcode A** Default Value: 2h ENC Format: OpCode
	20:16	**SubOpcode B** Default Value: 9h MEDIA_ Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 0007h Excludes DWord (0,1) Format: =n

MFC_MPEG2_PAK_OBJECT		
1..8	255:0	**Inline Data** Format: U32[8] All the required MB level controls and parameters for encoding are captured as inline data of the MFC_MPEG2_PAK_OBJECT command. It has a fixed size of 8 DWs. Its definition is described in the next section

MFC_MPEG2_SLICEGROUP_STATE

MFC_MPEG2_SLICEGROUP_STATE

Source: VideoCS

Length Bias: 2

This is a slice group level command and can be issued multiple times within a picture that is comprised of multiple slice groups. The same command is used for AVC encoder (PAK mode) and decoder (VLD and IT modes).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MPEG2_SLICEGROUP_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 3h MPEG2 Format: OpCode
	23:21	**SubOpcode A** Default Value: 2h MEDIA_ Format: OpCode
	20:16	**SubOpcode B** Default Value: 3h MEDIA_ Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 6h Excludes DWord (0,1) Format: =n
1	31	**MbRateCtrlFlag- RateControlCounterEnable (Encoder-only)** To enable the accumulation of bit allocation for rate control this field enables hardware Rate Control logic. The rest of the RC control fields are only valid when this field is set to 1. Otherwise, hardware ignores these fields. Note: To reset MB level rate control (QRC), we need to set both bits MbRateCtrlFlag and MbRateCtrlReset to 1 in the new slice Value: 0h — Name: Disable Value: 1h — Name: Enable

MFC_MPEG2_SLICEGROUP_STATE

30 **MbRateCtrlReset- ResetRateControlCounter (Encoder-only)**

To reset the bit allocation accumulation counter to 0 to restart the rate control.

Value	Name	Description
0h	Disable	Not reset
1h	Enable	reset

29:28 **MbRateCtrlMode- RC Triggle Mode (Encoder-only)**

Value	Name	Description
00b		Always Rate Control, whereas RC becomes activeif sum_act > sum_target or sum_act < sum_target
01b		Gentle Rate Control, whereas RC becomes activeif sum_act > upper_midpt or sum_act < lower_midpt
10b		Loose Rate Control, whereas RC becomes activeif sum_act > sum_max or sum_act < sum_min
11b		Reserved

27:24 **MbRateCtrlParam- RC Stable Tolerance (Encoder-only)**

Format:	U4

This field specifies the tolerance required to deactivate RC once it has been triggered.

Value	Name
[0, 15]	

23 **RateCtrlPanicFlag - RC Panic Enable (Encoder-only)**

If this field is set to 1, RC enters panic modewhen sum_act > sum_max. RC Panic Type field controls what type of panicbehavior is invoked.

Value	Name
0	Disable
1	Enable

22 **RateCtrlPanicType - RC Panic Type (Encoder-only)**

This field selects between two RC Panic methods. If it is set to 0, in panic mode, the macroblock QP is maxed out, setting to requested QP + QP_max_pos_mod. If it is set to 1, for an intra macroblock, AC CBPs are set to zero (note that DC CBPs are not modified). For inter macroblocks, AC and DC CBPs are forced to zero.

Value	Name	Description
0h		QP Panic
1h		CBP Panic

21 **Reserved**

Access:	RO
Format:	MBZ

20 **SkipConvDisabled - MB Type Skip Conversion Disable (Encoder-only)**

This field is only valid for a P or B slice. It must be zero for other slice types. Rules are provided in Section 2.3.3.1.6

MFC_MPEG2_SLICEGROUP_STATE

		Value / **Name** / **Description** 0h / Enable / Enable skip type conversion 1h / Disable / Disable skip type conversion
	19	**IsLastSliceGrp** IsLastSliceGrp = 1 if the current slice group is the last slice group of a picture; 0 otherwise. It is used by the zero filling in the Minimum Frame Size test.
	18	**BitstreamOutputFlag - Compressed BitStream Output Disable Flag (Encoder-only)** **Value** / **Name** / **Description** 0h / Enable / enable the writing of the output compressed bitstream 1h / Disable / disable the writing of the output compressed bitstream
	17	**HeaderPresentFlag - Header Insertion Present in Bitstream (Encoder-only)** **Value** / **Name** / **Description** 0h / Disable / no header insertion into the output bitstream buffer, in front of the current slice encoded bits 1h / Enable / header insertion into the output bitstream buffer is present, and is in front of the current slice encoded bits.
	16	**SliceData PresentFlag - SliceData Insertion Present in Bitstream (Encoder-only)** **Value** / **Name** / **Description** 0h / Disable / no Slice Data insertion into the output bitstream buffer 1h / Enable / Slice Data insertion into the output bitstream buffer is present.
	15	**TailPresentFlag - Tail Insertion Present in bitstream (Encoder-only)** **Value** / **Name** / **Description** 0h / / no tail insertion into the output bitstream buffer, after the current slice encoded bits 1h / / tail insertion into the output bitstream buffer is present, and is after the current slice encoded bits.
	14	**FirstSliceHdrDisabled** when this is on, the first slice header of the slice group is expected to be provided by the user via insertion command. PAK HW will skip it.
	13	**IntraSlice** intra slice value included in slice headers, when IntraSliceFlag = 1.
	12	**IntraSliceFlag** intra slice flag included in slice headers
	11:8	**Reserved** Access: RO Format: MBZ
	7:4	**SliceID[3:0] (Encoder-only)** To identify the output data (coding information record) returned for rate control from PAK to ENC and VPP

MFC_MPEG2_SLICEGROUP_STATE

	3:2	**Reserved** Access: RO Format: MBZ
	1:0	**StreamID[1:0] (Encoder-only)** To identify the output data (coding information record) returned for rate control from PAK to ENC and VPP
2	31:24	**NextSgMbYcnt - also NextStartVertPos** Vertical count of the first MB in the next slice group (Encoder-only)Note: This field restricts total number of MB in the Y direction to 255 or less.
	23:16	**NextSgMbXcnt - also NextStartHorzPos** BitFieldDesc
	15:8	**FirstMbYcnt - also CurrStartVertPos** Format: U8 also CurrStartVertPos, Vertical count of the first MB in the current slice group (Encoder-only)
	7:0	**FirstMbXcnt - also CurrStartHorzPos** Format: U8 Horizontal count of the first MB in the current slice group (Encoder-only)
3	31:9	**Reserved** Access: RO Format: MBZ
	8	**SliceGroupSkip** Exists If: //Encoder Only Format: U1 All macroblocks are skipped
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**SliceGroupQp** Exists If: //Encoder Only Format: U6 Initial slice quality parameter
4	31:29	**Reserved** Access: RO Format: MBZ
	28:0	**BitstreamOffset - Indirect PAK-BSE Data Start Address (Write)** Exists If: //Encoder Only This field specifies the memory starting address (offset) to write out the compressed bitstream data from the BSE processing. This pointer is relative to the MFC Indirect PAK-BSE Object Base Address. It is a byte-aligned address for the AVC bitstream data in both CABAC/CAVLC Modes.

MFC_MPEG2_SLICEGROUP_STATE

For Write, there is no need to have a data length field. It is assumed the global memory bound check specified in the IND_OBJ_BASE_ADDRESS command (Indirect PAK-BSE Object Access Upper Bound) will take care of any illegal write access. This field is only valid for AVC encode mode.

Value	Name
[0,512MB)	

5 — 31:24 — MaxQpNegModifier - Magnitude of QP Max Negative Modifier (Encoder-only)

Format:	U8

This field specifies the lower limit of the QP modifier.

Value	Name
[0, 51]	

23:16 — MaxQpPosModifier - Magnitude of QP Max Positive Modifier (Encoder-only)

Format:	U8

This field specifies the upper limit of the QP modifier.

Value	Name
[0, 51]	

15:12 — ShrinkParam - Shrink Resistance (Encoder-only)

Format:	U4

This field specifies the additional points added each time decreased correction is invoked.

Value	Name
[0, 15]	

11:8 — Shrinkaram - Shrink Init (Encoder-only)

Format:	U4

This field specifies the initial points required to trip decreased control.

Value	Name
[0, 15]	

7:4 — GrowParam - Grow Resistance (Encoder-only)

Format:	U4

This field specifies the additional points added each time increased correction is invoked.

Value	Name
[0, 15]	

3:0 — GrowParam - Grow Init (Encoder-only)

Format:	U4

This field specifies the initial points required to trip increased control.

Value	Name
[0, 15]	

MFC_MPEG2_SLICEGROUP_STATE

6	31:24	**Reserved** Access: RO Format: MBZ
	23:20	**CorrectPoints - Correct 6 (Encoder-only)** Format: U4 This field specifies the points used in the lowe rmost RC region when sum_act <= sum_min. Value: [0, 15]
	19:16	**CorrectPoints - Correct 5 (Encoder-only)** Format: U4 This field specifies the points used in the fifth RC region when sum_act > sum_min but <= lower_midpt. Value: [0, 15]
	15:12	**CorrectPoints - Correct 4 (Encoder-only)** Format: U4 This field specifies the points used in the fourth RC region when sum_act > lower_midpt but <= sum_target. Value: [0, 15]
	11:8	**CorrectPoints - Correct 3 (Encoder-only)** Format: U4 This field specifies the points used in the third RC region when sum_act > sum_target but <= upper_midpt. Value: [0, 15]
	7:4	**CorrectPoints - Correct 2 (Encoder-only)** Format: U4 This field specifies the points used in the second RC region when sum_act > upper_midpt but <= sum_max. Value: [0, 15]
	3:0	**CorrectPoints - Correct 1 (Encoder-only)** Format: U4 This field specifies the points used in the top most RC region when sum_act > sum_max Value: [0, 15]

MFC_MPEG2_SLICEGROUP_STATE

7	31:28	**CV7 - Clamp Value 7 (Encoder-only)** Exists If: //Encoder Only
	27:24	**CV6 - Clamp Value 6 (Encoder-only)** Exists If: //Encoder Only Format: U4
	23:20	**CV5 - Clamp Value 5 (Encoder-only)** Exists If: //Encoder Only Format: U4
	19:16	**CV4 - Clamp Value 4 (Encoder-only)** Exists If: //Encoder Only Format: U4
	15:12	**CV3 - Clamp Value 3 (Encoder-only)** Exists If: //Encoder Only Format: U4
	11:8	**CV2 - Clamp Value 2 (Encoder-only)** Exists If: //Encoder Only Format: U4
	7:4	**CV1 - Clamp Value 1 (Encoder-only)** Exists If: //Encoder Only Format: U4
	3:0	**CV0 - Clamp Value 0 (Encoder-only)** If the magnitude of coefficients at locations assigned with CV0 (mapping shown below) exceeds 2CV0-1, they are replaced with2CV0-1. For coefficients at locations marked as 'none', no clamping is performed. The following mappings are only applied to luma and chroma blocks\subblocks containing AC coefficients (blocks\sublocks with only DC coeffs will not be clamped). For 8x8 frame block, each coefficient is mapped to one of the eight CV values as following:

none	none	CV7	CV6	CV5	CV4	CV3	CV3
none	CV7	CV6	CV5	CV4	CV3	CV3	CV2
CV7	CV6	CV5	CV4	CV3	CV3	CV2	CV2
CV6	CV5	CV4	CV3	CV3	CV2	CV2	CV1
CV5	CV4	CV3	CV3	CV2	CV2	CV1	CV1
CV4	CV3	CV3	CV2	CV2	CV1	CV1	CV0
CV3	CV3	CV2	CV2	CV1	CV1	CV0	CV0
CV3	CV2	CV2	CV1	CV1	CV0	CV0	CV0

For 8x8 field block, each coefficient is mapped to one of the eight CV values as following:

none	none	CV6	CV5	CV4	CV3	CV2	CV1
none	CV7	CV6	CV5	CV4	CV3	CV2	CV1

MFC_MPEG2_SLICEGROUP_STATE									
		CV7	CV6	CV5	CV4	CV3	CV3	CV2	CV1
		CV7	CV6	CV5	CV4	CV3	CV2	CV2	CV1
		CV6	CV5	CV4	CV4	CV3	CV2	CV1	CV0
		CV6	CV5	CV4	CV3	CV2	CV2	CV1	CV0
		CV5	CV5	CV4	CV3	CV2	CV1	CV1	CV0
		CV5	CV5	CV4	CV3	CV2	CV1	CV1	CV0

MFD_AVC_BSD_OBJECT

MFD_AVC_BSD_OBJECT

Source: VideoCS

Length Bias: 2

The MFD_AVC_BSD_OBJECT command is the only primitive command for the AVC Decoding Pipeline. The same command is used for both CABAC and CAVLD modes. The Slice Data portion of the bitstream is loaded as indirect data object. Before issuing a MFD_AVC_BSD_OBJECT command, all AVC states of the MFD Engine need to be valid. Therefore the commands used to set these states need to have been issued prior to the issue of a MFD_AVC_BSD_OBJECT command.

Context switch interrupt is not supported by this command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_AVC_BSD_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h AVC_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 8h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 5h Excludes DWord (0,1) = 0005 Format: =n
1	31:0	**Indirect BSD Data Length** Format: U32 This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address. AVC Short Format : It is

MFD_AVC_BSD_OBJECT

		the length in bytes of the bitstream data for the current slice, including Slice Header + Slice Data + Emulation Prevention Bytes + any filling trailing zeros after the last MB. Hardware ignores the contents after the last non-zero byte. Trailing zero is allowed and handled correctly in both CABAC and CAVLC modes.
2	31:29	**Reserved** Access: RO Format: MBZ
	28:0	**Indirect BSD Data Start Address** Format: U29 This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the **MFD Indirect Object Base Address**. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the AVC bitstream data in both CABAC/CAVLD Modes. In implementing a phantom slice at the end of a picture for automatic error concealment, this field should set to 0. It includes the NAL Header (the NAL Header does not need to perform EMU detection). For AVC Base Layer, it is a single byte. But for MVC, the NAL Header is 4 Bytes long. These NAL Header Unit must be passed to HW in the compressed bitstream buffer. Value: [0,512MB) — Name:
3..5	95:0	**Inline Data** Format: **Inline Data Description for MFD_AVC_BSD_Object** All the required Slice Header parameters and error handling settings are captured as InLine Data of the AVC_BSD_OBJECT command. It has a fixed size of 3 DWs. Its definition is described in the following section: Inline Data Description.
6	31:0	**Reserved** Access: RO Format: MBZ

MFD_AVC_DPB_STATE

MFD_AVC_DPB_STATE

Source: VideoCS

Length Bias: 2

This is a frame level state command used only in AVC Short Slice Bitstream Format VLD mode. RefFrameList[16] of interface is replaced with intel Reference Picture Addresses[16] of MFX_PIPE_BUF_ADDR_STATE command. The LongTerm Picture flag indicator of all reference pictures are collected into LongTermPic_Flag[16].FieldOrderCntList[16][2] and CurrFieldOrderCnt[2] of interface are replaced with intel POCList[34] of MFX_AVC_DIRECTMODE_STATE command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MULTI_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h AVC_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 6h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 9h Excludes DWord (0,1) Format: =n
1	31:16	**LongTermFrame_Flag[16][1 bit]** One-to-one correspondence with the entries of the Intel RefFrameList[16]. 1 bit per reference frame. Value: 1 — the picture is a long term reference picture Value: 0 — the picture is a short term reference picture

MFD_AVC_DPB_STATE

	15:0	**Non-ExistingFrame_Flag[16][1 bit]** One-to-one correspondence with the entries of the Intel RefFrameList[16]. 1 bit per reference frame. **Value** / **Name** / **Description** 1 / INVALID / the reference picture in that entry of RefFrameList[] does not exist anymore. 0 / VALID / the reference picture in that entry of RefFrameList[] is a valid reference **Programming Notes** When an element of the list of frames is not relevant (e.g., due to the corresponding reference entry being empty or being marked as "not used for reference"), the value of the corresponding bit of NonExistingFrameFlags shall be set to 0.
2	31:0	**UsedForReference_Flag[16][2 bits]** One-to-one correspondence with the entries of the Intel RefFrameList[16]. 2 bits per reference frame. **Value** / **Name** / **Description** 0 / NOT_REFERENCE / indicates a frame is "not used for reference". 1 / TOP_FIELD / bit[0] indicates that the top field of a frame is marked as "used for reference". 2 / BOTTOM_FIELD / bit[1] indicates that the bottom field of a frame is marked as "used for reference". 3 / FRAME / bit[1:0] indicates that a frame (or field pair) is marked as "used for reference".
3..10	255:0	**LTSTFrameNumList[16][16 bits]** One-to-one correspondence with the entries of the Intel RefFrameList[16]. 16 bits per reference frame.Depending on the corresponding LongTermFrame_Flag[], the content of this field is interpreted differently. **Value** / **Name** / **Description** 1 / LongTermFrame_Flag[i] / LTSTFrameNumList[i] represent LongTermFrameIdx. 0 / ShortTermFrame_Flag[i] / LTSTFrameNumList[i]represent Short Term Picture FrameNum. **Programming Notes** When an element of the list of frames is not relevant (e.g., due to the corresponding reference entry being empty or being marked as "not used for reference"), the value of the LTSTFrameNumList entry shall be set to 0.
11..18	255:0	**ViewIDList[16][16 bits]** One-to-one correspondence with the entries of the Intel RefFrameList[16]. 16 bits per reference frame. The view ids are 10-bits, the upper 6 bits are ignored."000000" & ViewId1[9:0] & "000000" & ViewId0[9:0] **Programming Notes** When an Intel RefFrameList[i] is not an valid entries, Viewid should be set to 0x00

MFD_AVC_DPB_STATE		
19..22	127:0	**ViewOrderListL0[16][8 bits]** One-to-one correspondence with the entries of the Intel RefFrameList[16]. 8 bits per reference frame. The view order need 4-bits, the upper 4 bits are ignored.0000 & ViewOrder3[3:0] & 0000 & ViewOrder2[3:0] & 0000 & ViewOrder1[3:0] & 0000 & ViewOrder0[3:0] **Programming Notes** When the ViewOrderListL0[i] is not an valid inter-view reference, its corresponding View Order should be set to 0xF Since only interview with the same polarity will be used, there is no need to have field bit in this list. Hardware is going to append correct polarity bit as needed.
23..26	127:0	**ViewOrderListL1[16][8 bits]** One-to-one correspondence with the entries of the Intel RefFrameList[16]. 8 bits per reference frame. The view order need 4-bits, the upper 4 bits are ignored.0000 & ViewOrder3[3:0] & 0000 & ViewOrder2[3:0] & 0000 & ViewOrder1[3:0] & 0000 & ViewOrder0[3:0] **Programming Notes** When the ViewOrderListL1[i] is not an valid inter-view reference, its corresponding View Order should be set to 0xF Since only interview with the same polarity will be used, there is no need to have field bit in this list. Hardware is going to append correct polarity bit as needed.

MFD_AVC_PICID_STATE

MFD_AVC_PICID_STATE

Source: VideoCS

Length Bias: 2

This is a frame level state command used for both AVC Long and Short Format in VLD mode. PictureID[16] contains the pictureID of each reference picture (16 maximum) so hardware can uniquely identify the reference picture across frames (this will be used for DMV operation).This command will be needed for both short and long format.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MULTI_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h MFD_AVC_DPB_STATE Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h DEC Format: OpCode
	20:16	**SubOpcode B** Default Value: 5h MEDIA_ Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 0008h Excludes DWord (0,1) Format: =n
1	31:1	**Reserved** Access: RO Format: MBZ
	0	**PictureID Remapping Disable** Value: 0h — Name: AVC decoder will use 16 bits Picture ID to handle DMV and identify the — Description: Desc

MFD_AVC_PICID_STATE

		<table><tr><td></td><td>reference picture</td><td></td></tr><tr><td>1h</td><td>AVC decoder will use 4 bits FrameStoreID (index to RefFrameList) to handle DMV and identify the reference picture</td><td>Desc</td></tr></table> <table><tr><th>Programming Notes</th></tr><tr><td>If Picture ID Remapping Disable is "1", PictureIDList will not be used.</td></tr></table>
2..9	255:0	**PictureIDList[16][16 bits]** One-to-one correspondence with the entries of the Intel RefFrameList[16]. 16 bits per reference frame. PictureID of each Frame uniquely identifies the reference picture across frames. The same number cannot be reused until the reference picture is completely retired(no longer used for reference)When an element of the list of frames is not relevant (e.g., due to the corresponding reference entry being empty or being marked as "not used for reference"), the value of the LTSTFrameNumList entry shall be set to 0.

MFD_AVC_SLICEADDR

MFD_AVC_SLICEADDR

Source: VideoCS

Length Bias: 2

This is a Slice level command used only for AVC Short Slice Bitstream Format VLD mode. When decoding a slice, H/W needs to know the last MB of the slice has reached in order to start decoding the next slice. It also needs to know if a slice is terminated but the last MB has not reached, error concealment should be invoked to generate those missing MBs. For AVC Short Format, the only way to know the last MB position of the current slice, H/W needs to snoop into the next slice's start MB address (a linear address encoded in the Slice Header). Since each BSD Object command can have only one indirect bitstream buffer address, this command is added to help H/W to snoop into the next slice's slice header and retrieve its Start MB Address. This command will take the next slice's bitstream buffer address as input (exactly the same way as a BSD Object command), and parse only the first_mb_in_slice syntax element. The result will stored inside the H/W, and will be used to decode the current slice specified in the BSD Object command. Only the very first few bytes (max 5 bytes for a max 4K picture) of the Slice Header will be decoded, the rest of the bitstream are don't care. This is because the first_mb_in_slice is encoded in Exponential Golomb, and will take 33 bits to represent the max 256 x 256 = 64K-1 value. The indirect data of MFD_AVC_SLICEADDR is a valid BSD object and is decoded as in BSD OBJECT command. The next Slice Start MB Address is also exposed to the MMIO interface. The Slice Start MB Address (first_mb_in_slice) is a linear MB address count; but it is translated into the corresponding 2D MB X and Y raster position, and are stored internally as NextSliceMbY and NextSliceMbX.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_AVC_ SLICEADDR Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h AVC_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 7h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ

MFD_AVC_SLICEADDR

	11:0	**DWord Length** Default Value: 2h Format: =n
1	31:0	**Indirect BSD Data Length** Format: U32 This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. Driver always programs this up to 5 bytes; for bitstream less than 5 bytes, driver program the lesser value. (Emulation Prevention Byte should never happen for the first 5 bytes when the max picture size can only be 4Kx4K)It is the length in bytes of the bitstream data for the current slice, including Slice Header + Slice Data + Emulation Prevention Bytes + any filling trailing zeros after the last MB. Hardware ignores the contents after the last non-zero byte. Trailing zero is allowed and handled correctly in both CABAC and CAVLC modes.
2	31:29	**Reserved** Access: RO Format: MBZ
	28:0	**Indirect BSD Data Start Address** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the AVC bitstream data in both CABAC/CAVLD Modes. In implementing a phantom slice at the end of a picture for automatic error concealment, this field should set to 0.It includes the NAL Header Byte. (but does not perform EMU detection).Must provide a valid MB address, even if error. MB must be clamped to within a pic boundary. **Value** / **Name** [0,512MB)
3	31:13	**Reserved** Access: RO Format: MBZ
	12:9	**Reserved**
	8	**AVC NAL Type First Byte Override Bit** Format: U1 This bit indicates hardware should use the NAL Type (provided below) programmed by driver instead of using the one from bitstream. The NAL byte from bitstream will not be correct. **Value** / **Name** / **Description** 0 / Use Bitstream Decoded NAL Type / NAL Type should come from first byte of decoded bitstream. 1 / Use Driver Programmed NAL Type / NAL Type should come from "Driver Provided NAL Type Values" programmed by driver.

MFD_AVC_SLICEADDR		
	7:0	**Driver Provided NAL Type Value** Format: U8 This will replace the first byte of the NAL unit, containing forbidden_zero_bit, nal_ref_idc, and nal_unit_type. **Programming Notes** This byte should be ignored if AVC NAL Type First Byte Override Bit is programmed to 0

MFD_IT_OBJECT

MFD_IT_OBJECT

Source: VideoCS

Length Bias: 2

All weight mode (default and implicit) are mapped to explicit mode. But the weights come in either as explicit or implicit.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_IT_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MFX_COMMON_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 9h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Format: =n

Value	Name	Description	Exists If
0Ch	AVC	There are total of 7 inline DWs for AVC (in addition to DW0-DW6 here)	//mode=='AVC'
10h	VC1	There are total of 11 inline DWs for VC1 (in addition to DW0-DW6 here)	//mode=='VC1'
0Bh	MPEG2	There are total of 6 inline DWs for AVC (in addition to DW0-DW6 here)	//mode=='MPEG2'

MFD_IT_OBJECT

1	31:10	**Reserved** Access: RO Format: MBZ
	9:0	**Indirect IT-MV Data Length** Format: U10 This field provides the length in bytes of the indirect data, which contains all the MVs for the current MB (in any partitioning and subpartitioning form). A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect IT-MV Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address. AVC-IT Mode: It must be DWord aligned (since each MV is 4bytes in size)Driver has to derived this field from MVsize (MVquantity in DXVA, exact size) *4 bytes per MV. This field is only valid in AVC decoder IT mode (VC1 and MPEG uses inline MV data).
2	31:29	**Reserved** Access: RO Format: MBZ
	28:0	**Indirect IT-MV Data Start Address Offset** This field specifies the memory starting address (offset) of the MV data to be fetched into the IT pipeline for processing. This pointer is relative to the Indirect IT-MV Object Base Address. Hardware ignores this field if indirect data is not present, i.e. the Indirect MV Data Length is set to 0. Alignment of this address depends on the mode of operation. AVC-IT Mode: It must be DWord aligned (since each MV is 4 bytes in size). This field is only valid in AVC decoder IT mode (VC1 and MPEG uses inline MV data). Value: [0,512MB) — Name:
3	31:12	**Reserved** Access: RO Format: MBZ
	11:0	**Indirect IT-COEFF Data Length** This field provides the length in bytes of the indirect data, which contains all the non-zero coefficients for the current MB. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect IT-COEFF Data Start Address field is ignored. Since each IT-COEFF data is 1 DW in size, with 12 bits, this field can be extended to support up to 4:4:4 format.(256 pixel * 3 byte pixel components * 4 bytes per coeff).This field must be integer multiple of 16-bytes for AVC (since each coefficient is 4 bytes in size).This field is only valid in AVC, VC1, MPEG2 decoder IT mode. Value: [0,3072] — Name: In bytes [0, 256*3*4]

MFD_IT_OBJECT

4	31:29	**Reserved** Access: RO Format: MBZ
	28:0	**Indirect IT-COEFF Data Start Address Offset** This field specifies the memory starting address (offset) of the coeff data to be loaded into the IT pipeline for processing. This pointer is relative to the Indirect IT-COEFF Object Base Address. Hardware ignores this field if indirect IT-COEFF data is not present, i.e. the Indirect IT-COEFF Data Length is set to 0.This field must be DW aligned, since each coeff icient is 4 bytes in size. Driver will determine the Num of EOB 4x4/8x8 must match the block cbp flags, if not match, hardware cannot hang - add error handling. This field is only valid in AVC, VC1, MPEG2 decoder IT mode. **Value** / **Name** [0,512MB)
5	31:6	**Reserved** Access: RO Format: MBZ
	5:0	**Indirect IT-DBLK Control Data Length** Format: U6 This field provides the length in bytes of the indirect data, which contains all the deblocker control information for the current MB (in 4x4 sub-block partitioning). A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect IT-DBLK Data Start Address field is ignored. This field must have the same alignment as the Indirect IT-DBLK Data Start Address. It must be DWord aligned. Each Deblock Control Data record is 48 bytes or 12 DWords in size. This field is only valid in AVC decoder IT mode.
6	31:29	**Reserved** Access: RO Format: MBZ
	28:0	**Indirect IT-DBLK Control Data Start Address Offset** Format: IndirectObjectOffset[28:0] This field specifies the memory starting address (offset) of the Deblocker control data to be fetched into the IT Pipeline for processing. This pointer is relative to the Indirect IT-DBLK Object Base Address. Hardware ignores this field if indirect data is not present, ie. The indirect IT-DBLK Control Data Length is set to 0.It must be DWord aligned. Each Deblock Control Data record is 48 bytes or 12 DWords in size. This field is only valid in AVC decoder IT mode. **Value** / **Name** [0,512MB)
7..12 **Exists if:** //mode	191:0	**MPEG2 Inline Data** Union for all 3 codecs. Includes IT, MC, IntraPred inline data as well as Deblocker

MFD_IT_OBJECT		
== 'MPEG2' **Programming Notes:** MPEG2--There are 6 addition DWs so n = 12		control information. AVC-IT Modes: Hardware interprets this data in the specified format. VC1-IT Modes: Hardware interprets this data in the specified format. MV inline. MPEG2-IT Modes: Hardware interprets this data in the specified format. (IS mode) MV inline For AVC there 7 DWords of inline data, hence N is equal to 13.
7..13 **Exists if:** //mode == 'AVC' **Programming Notes:** AVC--There are 7 addition DWs so n = 13	223:0	**AVC Inline Data** Union for all 3 codecs. Includes IT, MC, IntraPred inline data as well as Deblocker control information. AVC-IT Modes: Hardware interprets this data in the specified format. VC1-IT Modes: Hardware interprets this data in the specified format. MV inline MPEG2-IT Modes: Hardware interprets this data in the specified format. (IS mode) MV inline For AVC there 7 DWords of inline data, hence N is equal to 13.
7..17 **Exists if:** //mode == 'VC1' **Programming Notes:** VC1--There are 11 addition DWs so n = 17	351:0	**VC1 Inline Data** Union for all 3 codecs. Includes IT, MC, IntraPred inline data as well as Deblocker control information. AVC-IT Modes: Hardware interprets this data in the specified format. VC1-IT Modes: Hardware interprets this data in the specified format. MV inline. MPEG2-IT Modes: Hardware interprets this data in the specified format. (IS mode) MV inline For AVC there 7 DWords of inline data, hence N is equal to 13.

MFD_JPEG_BSD_OBJECT

MFD_JPEG_BSD_OBJECT

Source: VideoCS
Length Bias: 2
Exists If: //Decoder

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_JPEG_BSD_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 7h JPEG_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 8h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 004h Excludes DWord (0,1) Format: =n
1	31:0	**Indirect Data Length** . It is the length in bytes of the bitstream data for the current Scan. It includes the first byte of the first MCU and the last non-zero byte of the last MCU in the Scan. Specifically, the zero-padding bytes (if present) are excluded. Hardware ignores the contents after the last non-zero byte.
2	31:29	**Reserved** Access: RO Format: MBZ

MFD_JPEG_BSD_OBJECT

	28:0	**Indirect Data Start Address** Format: IndirectObjectOffset[28:0] This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the BSD Indirect Object Base Address.Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the JPEG bitstream data
3	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**Scan Horizontal Position** Format: U13 This field indicates the horizontal position (in block units) of the first MCU in the Scan.
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**Scan Vertical Position** Format: U13 This field indicates the vertical position (in block units) of the first MCU in the Scan.
4	31	**Reserved** Access: RO Format: MBZ
	30	**Interleaved** Value / Name / Description 0 / Non-Interleaved / one component in the Scan 1 / Interleaved / multiple components in the Scan
	29:27	**Scan Components** Bit0: YBit1: UBit2: VFor example, if non-interleaved Y, then it will be set to 001b. If interleaved Y, U, and V, it will be set to 111b.
	26	**Reserved** Access: RO Format: MBZ
	25:0	**MCU Count** Format: U26 This field indicates the number of MCUs in the Scan.

MFD_JPEG_BSD_OBJECT		
5	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**RestartInterval(16 bit)** Format: U16 Specifies the number of MCU in restart interval. Valid values are 1->0xFFFFValue of 0 implies that all the SCAN have only one ECS.

MFD_MPEG2_BSD_OBJECT

MFD_MPEG2_BSD_OBJECT

Source: VideoCS

Length Bias: 2

Different from AVC and VC1, MFD_MPEG2_BSD_OBJECT command is pipelinable. This is for performance purpose as in MPEG2 a slice is defined as a group of MBs of any size that must be within a macroblock row. Slice header parameters are passed in as inline data and the bitstream data for the slice is passed in as indirect data. Of the inline data, slice_horizontal_position and slice_vertical_position determines the location within the destination picture of the first macroblock in the slice. The content in this command is identical to that in the MEDIA_OBJECT command in VLD mode described in the Media Chapter.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_MPEG2_BSD_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 3h MPEG2_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 8h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 0003h Excludes DWord (0,1) Format: =n
1	31:0	**Indirect BSD Data Length** Format: U32 It is the length in bytes of the bitstream data for the current slice. It includes the first byte of the first macroblock and the last non-zero byte of the last macroblock in the slice. Specifically, the zero-padding bytes (if present) and the next start-code are excluded. This field is sized to

<table>
<tr><th colspan="3">MFD_MPEG2_BSD_OBJECT</th></tr>
<tr><td></td><td></td><td>support beyond MPEG-2 MP@HL bitstream (<4K). According to Table 8-6 of ISO/IEC 13818-2, the maximum number of bits per macroblock for 4:2:0 is 4608. So the maximum slice size for 4K x 4K is 4608 * 256 / 8 = 147,456 bytes (0x24000), which requires 18 bits.
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>As MPEG-2 spec does not post any limitation of the size of zero-padding bytes, it is possible to have a slice data with large length (including zero-padding bytes). As the data beyond 0x10E00 would only be zero bytes for a valid slice data</td></tr>
<tr><td>zero-padding restriction is removed</td></tr>
</table></td></tr>
<tr><td rowspan="2">2</td><td>31:29</td><td>Reserved
<table>
<tr><td>Access:</td><td>RO</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td>28:0</td><td>Indirect Data Start Address
<table>
<tr><td>Format:</td><td>IndirectObjectOffset[28:0]</td></tr>
</table>
This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the BSD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the MPEG2 VLD bitstream data This address points to the first byte of the MB layer data, i.e. not including slice header.</td></tr>
<tr><td>3..4</td><td>63:0</td><td>Inline Data
<table>
<tr><td>Format:</td><td>MFD_MPEG2_BSD_OBJECT Inline Data Description</td></tr>
</table>
All the required Slice Header parameters and error handling settings are captured as MPEG2_BSD_OBJECT Inline Data Descriptor structures. It has a fixed size of 2 DWs. Its definition is described in the next section.</td></tr>
</table>

MFD_VC1_BSD_OBJECT

MFD_VC1_BSD_OBJECT

Source: VideoCS

Length Bias: 2

The MFD_VC1_BSD_OBJECT command is the only primitive command for the VC1 Decoding Pipeline. The macroblock data portion of the bitstream is loaded as indirect data object. Before issuing a MFD_VC1_BSD_OBJECT command, all VC1 states of the MFD Engine need to be valid. Therefore the commands used to set these states need to have been issued prior to the issue of a MFD_VC1_BSD_OBJECT command.VC1 deblock filter kernel cross the slice boundary if in the last MB row of a slice, so need to know the last MB row of a slice to disable the edge mask. There is why VC1 BSD hardware need to know the end of MB address for the current slice. As such no more phantom slice is needed for VC1, as long as the driver will program both start MB address in the current slice and the start MB address of the next slice. As a result, we can also support multiple picture state commands in between slices.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MULTI_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 2h VC1_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 8h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 0003h Excludes DWord (0,1) Format: =n

MFD_VC1_BSD_OBJECT

1	31:24	**Reserved** Access: RO Format: MBZ
	23:0	**Indirect BSD Data Length** Format: U24 This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address. Long Format : It is the length in bytes of the bitstream data for the current slice/picture. It includes the first byte of the first macroblock and the last byte of the last macroblock in the slice/picture. Specifically, the zero-padding bytes (if present) and the next start-code are excluded. Hardware ignores the contents after the last non-zero byte (trailing zeros). This field is sized to support VC1 AP@L4 Level bitstream. It includes the byte that contains the First MB Bit Offset Short Format : It is the length in bytes of the bitstream data for the current slice, including Picture/Slice Header + Emulation Prevention Bytes + any filling trailing zeros after the last MB. Hardware ignores the contents after the last non-zero byte. Trailing zero is allowed and handled correctly.
2	31:29	**Reserved** Access: RO Format: MBZ
	28:0	**Indirect Data Start Address** Format: IndirectObjectOffset[28:0] This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This pointer is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VC1 bitstream data. Value: [0,512MB) — Name:
3	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**Slice Start Vertical Position** This field specifies the position in y-direction of the first macroblock in the Slice in unit of macroblocks. For SecondField this value is reset to zero as opposed to the VC1 spec Ref: 9.1.2 Slice Layer. This field is for both Long and Short VC1 Interface Format.
	15:9	**Reserved** Access: RO Format: MBZ
	8:0	**Next Slice Vertical Position** This field specifies the position in y-direction of the first macroblock in the next Slice in unit of macroblocks. This field is primarily used for error concealment. In the case that current slice is the

		last slice, this field should set to the height of picture (since y-direction is zero-based numbering)This field is maintained and provided by the driver for both Long and Short VC1 Interface Format.
4	31:16	**First_MB_Byte_Offset of Slice Data or Slice Header** For DXVA2 VC1 Short Format only, it gives the byte offset to locate the first MB data in the bitstream for a slice, relative to the Indirect BSD Data Start Address.
	15:5	**Reserved** Access: RO Format: MBZ
	4	**Emulation Prevention Byte Present** Value 0h: H/W needs to perform Emulation Byte Removal Value 1h: H/W does not need to perform Emulation Byte Removal
	3	**Reserved** Access: RO Format: MBZ
	2:0	**FirstMbBitOffset (First Macroblock Bit Offset)** Format: U3 This field provides the bit offset of the first macroblock of the Slice in the first byte of the input compressed bitstream. It is used with First_MB_Byte_Offset for non-byte aligned position.

MFD_VC1_LONG_PIC_STATE

MFD_VC1_LONG_PIC_STATE

Source: VideoCS

Length Bias: 2

MFX_VC1_LONG PIC_STATE command encapsulates the decoding parameters that are read or derived from bitstream syntax elements above (inclusive) picture header layer. These parameters are static for a picture and when slice structure is present, these parameters are not changed from slice to slice of the same picture. Hence, this command is only issued at the beginning of processing a new picture and prior to the VC1_*_OBJECT command. The values set for these state variables are retained internally across slices. Only the parameters needed by hardware (BSD unit) to decode bit sequence for the macroblocks in a picture layer or a slice layer are presented in this command. Other parameters such as the ones used for inverse transform or motion compensation are provided in MFX_VC1_PRED_PIPE_STATE command. This Long interface format is intel proprietary interface. Driver will need to perform addition operations to generate all the fields in this command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_VC1_LONG_PIC_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 2h VC1_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 1h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 0004h Excludes DWord (0,1) Format: =n

MFD_VC1_LONG_PIC_STATE

1 — 31:24 — Reserved

Access:	RO
Format:	MBZ

1 — 23:16 — PictureHeightInMBsMinus1 (Picture Height Minus 1 in Macroblocks)

Format:	U8

This field indicates the height of the picture in unit of macroblocks. For example, for a 1920x1080 frame picture, PictureHeightInMBs equals 68 (1080 divided by 16, and rounded up, i.e. effectively specified as 1088 instead).This field is used in VLD and IT modes.

Value	Name	Description
[0,255]	ValueHeight	Represents 1 MB to 256 MB

Programming Notes

Note: Even though the Advanced Profile allows frame dimensions (width, height) to not be aligned to macroblock boundary, it doesn't affect the bitstream decoding. And it is preferable to use 'intermediate buffer' that is macroblock aligned for decoding. In order to simplify the out-of-bound reference pixel access, the out-of-bound extrapolation rule in VC1 spec can be used to expand the expected decoded frame to the intermediate buffer dimension.

1 — 15:8 — Reserved

Access:	RO
Format:	MBZ

1 — 7:0 — PictureWidthInMBsMinus1 (Picture Width Minus 1 in Macroblocks)

Format:	U8-1

This field indicates the width of the picture in unit of macroblocks. For example, for a 1920x1080 frame picture, PictureWidthInMBs equals 120 (1920 divided by 16).This field is used in VLD and IT modes

Value	Name	Description
[0,255]	ValidWidth	Represents 1 MB to 256 MB

2 — 31:24 — Bitplane Buffer Pitch Minus 1

Format:	U8-1

Specifies the bitplane buffer pitch in (#Bytes - 1). Bitplane buffer is a linear buffer. It is needed only when the bitplane is not encoded as raw, and therefore is present in the header explicitly. In VC1 Long Format, it is written by an application and later read by the HW. But in VC1 Short Format, it is written and read by H/W only. This field is specified for better performance

Value	Name
[0h, FFh]	

Programming Notes

The pitch must be equal to PictureWidthInMBs/2.For VC1 Long Format : The pitch must be equal to PictureWidthInMBs/2.For VC1 Short Format : If Pic Width is less than or equal to 2K pixels, bitplane pitch is set to 64 (one cacheline; programmed as 63) bytes per MB row. If Pic

		Width is greater than 2K pixels, bitplane pitch is set to 128 (two cachelines; programmed as 127) bytes per MB row.This field is not used in IT mode, used in VLD mode only. For VC1 Short Format, the bitplane specification is between H/W and Driver only. For Long Format, application is responsible for allocation with the driver.
	23:16	**Reserved** Access: RO Format: MBZ
	15	**DmvSurfaceValid** Indicated when the DMV read surface is valid. This surface stored the direct motion vectors and Mb type. This field is set for B pictures that can refer to a previous P picture for DMV. If there is an I-picture before a B (in decoding order) then this field is not set (as a result, zero's DMV's will be assumed while decoding the B picture. That is, there is no explicit DMV buffer for an I-picture).When the current picture being decoded is an I, P or BI, this bit is set to 0, since there is no DMV read in these picture decoding process. This field is not used in IT mode, used in VLD mode only.
	14	**ImplicitQuantizer** Derived by driver from QUANTIZER. This field is used in intel VC1 VLD Long Format only, not used in IT and VC1. This bit is set to 1 when syntax element QUANTIZER=0, else its set to 0
	13	**Interpolation Rounder Contro** Used only in MC operation. This field specifies the rounding control value used in interpolation operation of motion prediction process. This field is used in VLD and IT modes. **Programming Notes** This bit field is taken from bRcontrol in PictureParameters data structure
	12	**SyncMarker** Indicates whether sync markers are enabled/disabled. If enable, sync markers "may be" present in the current video sequence being decoded. It is a sequence level syntax element and is valid only for Simple and Main Profiles. **Value** / **Name** / **Description** 0h / Not Present / Sync Marker is not present in the bitstream 1h / Maybe present / Sync Marker maybe present in the bitstream **Programming Notes** This field is only valid in VLD mode .For Simple Profile, SyncMarker must set to 0.For Main Profile, SyncMarker can be set to 0 or 1.This field is used in both intel and MS VLD interface, but not used in IT mode.
	11:8	**Motion Vector Mode** This field indicates one of the following motion compensation interpolation modes for P and B pictures. The MC interpolation modes apply to prediction values of luminance blocks and are always in quarter-sample. For chrominance blocks, it always performs bilinear interpolation with either half-pel or quarter-pel precision. Before the polarity of Chroma Half-pel or Q-pel is reversed from Spec, now I have fixed it to match with VC1 Spec.

Value	Name	Description
0XX0b		Chroma Quarter -pel + Luma bicubic. (can only be 1MV)
0XX1b		Chroma Half-pel + Luma bicubic. (can be 1MV or 4MV)
1XX0b		Chroma Quarter -pel + Luma bilinear. (can only be 1MV)
1XX1b		Chroma Half-pel + Luma bilinear

Programming Notes
Bits 11:8 are taken from bMVprecisionAndChromaRelation in PictureParameters data structure.Bit 11 of Motion Vector Mode = 1 for Luma Bilinear MC; = 0 for Luma Bicubic MCBit 8 of Motion Vector Mode = 1 for half-sample Chroma motion = 0 for quarter-sample Chroma motion.This field is used in both VLD and IT modes.

7

RangeReductionScale

This field specifies whether the reference picture pixel values should be scaled up or scaled down on-the-fly, if RangeReduction is Enabled.

Value	Name	Description
0h		Scale down reference picture by factor of 2
1h		Scale up reference picture by factor of 2

Programming Notes
This bit is derived by driver for Main Profile only. Ignored in Simple and Advanced Profiles. This field is used in both VLD and ITmodes. This is derived by driver from the history of RANGERED and RANGEREDFRM syntax elements (i.e. of forward/preceding reference picture) and those of the current picture. RANGERED is the same as (bPicOverflowBlocks » 3) &1. RANGEREDFRM is the same as (bPicDeblocked » 5) & 1. For thecurrent picture is a B picture, this field represents the state of theforward/preceding reference picture onlyDriver is responsible to keep RangeReductionScale, RangeReduction Enable and RANGERED Present Flag of current picture coherent.

6

RangeReduction Enable

This field specifies whether on-the-fly pixel value range reduction should be performed for the preceding (or forward) reference picture. Along with RangeReductionScale to specify whether scale up or down should be performed. It is not the same value as RANGEREDFRM Syntax Element (PictureParameters bPicDeblocked bit 5) in the Picture Header.

Value	Name	Description
0h	Disable	Range reduction is not performed
1h	Enable	Range reduction is performed

Programming Notes
This field is for Main Profile only. Simple Profile is always disable, and not applicable to Advanced Profile. This field is used in both VLD and IT modes. This is derived by driver from the history of RANGERED and RANGEREDFRM syntax elements (i.e. of forward/preceding reference picture) andthose of the current picture.RANGERED is the same as

MFD_VC1_LONG_PIC_STATE

		(bPicOverflowBlocks» 3) & 1. RANGEREDFRM is the same as (bPicDeblocked » 5) &1.For the current picture is a B picture, this field represents the state of the forward/preceding reference picture only Driver is responsible to keep RangeReductionScale, RangeReduction Enable and RANGERED Present Flag of current picture coherent.
	5	**Reserved**
	4	**Overlap Smoothing Enable Flag** This field is the decoded syntax element OVERLAP in bitstreamIndicates if Overlap smoothing is ON at the picture level This field is used in both VLD and IT modes. **Value** / **Name** / **Description** 0h / Disable / to disable overlap smoothing filter 1h / Enable / to enable overlap smoothing filter
	3	**Secondfield** This flag is set for the second field in field pictures. This field is used in both VLD and IT modes.
	2:1	**Reserved** Access: RO Format: MBZ
	0	**VC1 Profile** specifies the bitstream profile. This field is used in both VLD and IT modes. **Value** / **Name** / **Description** 0h / Disable / current picture is in Simple or Main Profile (No need to distinguish Simple and Main Profile) 1h / Enable / current picture is in Advanced Profile **Programming Notes** This is required because 128 is added for intra blocks post inverse transform in advanced profile and also to find out if Motion vectors are adjusted or not.
3	31	**Reserved** Access: RO Format: MBZ
	30:29	**CondOver** This field is the decoded syntax element CONDOVER in a bitstream of advanced profile. It controls the overlap smoothing filter operation for an I frame or an BI frame when the picture level qualization step size PQUANT is 8 or lower. This field is used in intel VC1 VLD mode only, not in VC1 and IT modes. **Value** / **Name** / **Description** 00b / / No overlap smoothing 01b / / Reserved 10b / / Always perform overlap smoothing filter

11b		Overlap smoothing on a per macroblock basis based on OVERFLAGS

28:26 PicType (Picture Type)

This field specifies the coding type of the picture according to the Frame Coding Mode. When FCM = 00 | 01 (a Progressive or Interlaced Frame Picture):000 = I001 = P010 = B011 = BI100 = Skipped Other encodings are reserved When FCM = 10 | 11 (a Field Picture)000 = I/I001 = I/P010 = P/I011 = P/P100 = B/B101 = B/BI110 = BI/B111 = BI/BIAlthough, for a field picture, it is set for a field-pair, but HW will only look at one field state only, and the other field state is don't care. This field is read and qualified with the SecondField flag internally. This field is unique to intel VC1 VLD Long format, and is used in IT mode as well. For VC1 IT mode, driver needs to convert the interface to intel HW VLD Long Format interface.

25:24 FCM (Frame Coding Mode)

This is the same as the variable FCM defined in VC1.This field must be set to 0 for Simple and Main Profiles This field is unique to intel VC1 VLD Long format, and is used in IT mode as well. For VC1 IT mode, driver needs to convert the interface to intel HW VLD Long Format interface.

Value	Name	Description
00b	Disable	Progressive Frame Picture
01b	Enable	Interlaced Frame Picture
10b		Field Picture with Top Field First
11b		Field Picture with Bottom Field First

23:21 Reserved

Access:	RO
Format:	MBZ

20:16 AltPQuant (Alternative Picture Quantization Value)

This field is identical to the variable ALTPQUANT which is derived from VOPDQUANT configuration in the VC1 standard. This field must be set to 0 for Simple/Main I and BI pictures as VOPDQUANT is not present. This field is used in intel VC1 VLD Long Format mode only, not used in VC1 VLD and IT modes.

15:13 Reserved

Access:	RO
Format:	MBZ

12:8 PQuant (Picture Quantization Value)

Format:	U5

This is the same as the calculated variable PQUANTin VC1 standard where PQuant = PQINDEX, except when QUANTIZER = 0 and PQINDEX> 8, it is given as PQuant = (PQINDEX < 29) ? PQINDEX - 3 : PQINDEX*2 -31This field is used in all picture types (I, P, B and BI) and all operatingmodes (IT mode and intel and VLD modes).

7:0 BScaleFactor

BScaleFactor This field is the scale factor for computing Direct-mode motion vectors. It is derived from the variable BFRACTION in the VC1 standard, section 8.4.5.4.There are only 21 valid values corresponding to the 21 encodings of BFRACTION as shown in the table here.

Other values are reserved. MSB of this field can be used to determine if BFRACTION is greater than or equal to 1/2, which is used to determine Motion Prediction Type for B pictures. Effectively, condition "BFRACTION >= 1/2" is equivalent to condition "BScaleFactor >= 128".This field is only valid for B pictures. This field is used only in intel VC1 VLD Long format mode, it is not used in VC1 VLD and IT modes.
BFRACTIONVLCBFRACTIONBScaleFactor0001/21280011/3850102/31700111/4641003/41921011/5511102/510211100003/515311100014/520411100101/64311100115/621511101001/73711101012/77411101103/711111101114/714811110005/718511110016/722211110101/83211110113/89611111005/816011111017/8224

4 — 31:30 — Reserved

Access:	RO
Format:	MBZ

29:28 — UnifiedMvMode (Unified Motion Vector Mode)

This field is a combination of the variables MVMODE and MVMODE2 in the VC1 standard, for parsing Luma MVD from the bitstream. This field is used to signal 1MV vs 4MVallowed (Mixed Mode). This field is also used to signal Q-pel or Half-pel MVD read from the bitstream. The bicubic or bilinear Luma MC interpolation mode is duplicate information from Motion Vector Mode field, and is ignored here. This field is used in intel VC1 VLD Long Format mode only, it is not used in VC1 VLD and IT modes.

Value	Name	Description
00b		Mixed MV, Q-pel bicubic
01b		1-MV, Q-pel bicubic
10b		1-MV half-pel bicubic
11b		1-MV half-pel bilinear

27 — FourMvSwitch (Four Motion Vector Switch)

This field indicates if 4-MV is present for an interlaced frame P picture. It is identical to the variable 4MVSWITCH (4 Motion Vector Switch) in VC1 standard. This field is used in intel VC1 VLD Long Format mode only, it is not used in VC1 VLD and IT modes.

Value	Name	Description
0h	Disable	only 1-MV
1h	Enable	1, 2, or 4 MVs

26 — FastUVMCFlag (Fast UV Motion Compensation Flag)

This field specifies whether the motion vectors for UV is rounded to half or full pel position. It is identical to the variable FASTUVMC in VC1 standard. This field is used in both VLD and IT modes .It is derived from FASTUVMC = (bPicSpatialResid8 » 4) & 1 in both VLD and IT modes, and should have the same value as Motion Vector ModeLSBit.

Value	Name	Description
0h		no rounding
1h		quarter-pel offsets to half/full pel positions

25 — RefFieldPicPolarity (Reference Field Picture Polarity)

This field specifies the polarity of the one reference field picture used for a field P picture. It is

derived from the variable REFFIELD defined in VC1 standard and is only valid when one field is referenced (NUMREF = 0) for a field P picture. When NUMREF = 0 and REFFIELD = 0, this field is the polarity of the reference I/P field that is temporally closest; When NUMREF = 0 and REFFIELD = 1, this field is the polarity of the reference I/P field that is the second most temporally closest. The distance is measured based on display order but ignoring the repeated field if present (due to RFF = 1).This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

Value	Name	Description
0h		Top (even) field
1h		Bottom (odd) field

24

NumRef (Number of References)

This field indicates how many reference fields are referenced by the current (field) picture. It is identical to the variable NUMREF in the VC1 standard. This field is only valid for field P picture (FCM = 10 | 11).This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

Value	Name	Description
0h		One field referenced
1h		Two fields referenced

23:20

BwdRefDist (Reference Distance)

This field is valid only in B field pictures giving the value of BRFD. The field is ignored in P Picture. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

19:16

FwdRefDist (Reference Distance)

Format:	U4

This field is the number of frames between the current frame and its reference frame. It is derived from the syntax element REFDIST (P Reference Distance) in the VC1 standard. 0 means that the previous frame is the reference frame. It has the same value as of FRFD for both P and B field pictures. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

Value	Name
[0, 15]	

15:12

Reserved

Access:	RO
Format:	MBZ

11:10

ExtendedDMVRange (Extended Differential Motion Vector Range Flag)

This field specifies the differential motion vector range in interlaced pictures. It is equivalent to the variable DMVRANGE in the VC1 standard. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

Value	Name	Description
00b		No extended range
01b		Extended horizontally

10b		Extended vertically
11b		Extended in both directions

9:8 ExtendedMVRange (Extended Motion Vector Range Flag)

This field specifies the motion vector range in quarter-pel or half-pel modes. It is equivalent to the variable MVRANGE in the VC1 standard. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes

Value	Name	Description
00b		[-256, 255] x [-128, 127]
01b		512, 511] x [-256, 255]
10b		[-2048, 2047] x [-1024, 1023]
11b		[-4096, 4095] x [-2048, 2047]

7:4 AltPQuantEdgeMask (Alternative Picture Quantization Edge Mask)

This field is a bit mask for the four edges in clock-wise order, indicating whether AltPQuant is used for the edge macroblocks.It is derived based on the following variables DQUANT, DQUANTFRM, DQPROFILE, DQSBEDGE, DQDBEDGE, and DQBILEVEL defined in the VC1 standard, as shown in Error! Reference source not found. This field is valid only if AltPQuantConfig is 01. Bit 0: Left picture edge macroblocksBit 1: Top picture edge macroblocksBit 2: Right picture edge macroblocksBit 3: Bottom picture edge macroblocks This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

3:2 AltPQuantConfig (Alternative Picture Quantization Configuration)

This field specifies the way AltPQuant is used in the picture. It determines how to compute the macroblock quantizer step size, MQUANT. It is derived based on the following variables DQUANT, DQUANTFRM, DQPROFILE, DQSBEDGE, DQDBEDGE, and DQBILEVEL defined in the VC1 standard, as shown in Error! Reference source not found. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and modes.

Value	Name	Description
00b		AltPQuant not used
01b		AltPQuant is used and applied to edge macroblocks only
10b		MQUANT is encoded in macroblock layer
11b		AltPQuant and PQuant are selected on macroblock basis

1 HalfQP

This field is used for inverse quantization of AC coefficients. It is valid only when PQuant is used. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

0 PQuantUniform

Indicating if uniform quantization applies to the picture. It is used for inverse quantization of the AC coefficients.QUANTIZER001123PQUANTIZER --01--PQINDEX>=9<=8----PQuantUniform010201ImplicitQuantizer = 0, and PQuantUniform = 0 is used to represent 2 cases : 1) QUANTIZER=01 and PQUANTIZER=0; and 2) QUANTIZER = 10b.ImplicitQuantizer = 0, and PQuantUniform =1 is used to represent 2 cases : 1) QUANTIZER=01 and PQUANTIZER=1; and 2)QUANTIZER = 11bThis field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

Value	Name	Description
0h		Non-uniform
1h		Uniform

5	31	**BitplanePresentFlag (Bitplane Buffer Present Flag)** This field indicates whether the bitplane buffer is present for the picture. If set, at least one of the fields listed in bits 22:16 is coded in non-raw mode, and Bitplane Buffer Base Address field in the VC1_BSD_BUF_BASE_STATE command points to the bitplane buffer. Otherwise, all the fields that are applicable for the current picture in bits 22:16 must be coded in raw mode. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

Value	Name	Description
0h		bitplane buffer is not present
1h		bitplane buffer is present

	30	**ForwardMbRaw** This field indicates whether the FORWARDMB field is coded in raw or non-raw mode. This field is only valid when PictureType is B. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

Value	Name	Description
0h		non-raw mode
1h		raw mode

	29	**MvTypeMbRaw** This field indicates whether the MVTYPREMB field is coded in raw or non-raw mode. This field is only valid when PictureType is P. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

Value	Name	Description
0h		Non-Raw Mode
1h		Raw Mode

	28	**SkipMbRaw** This field indicates whether the SKIPMB field is coded in raw or non-raw mode. This field is only valid when PictureType is P or B.0 = non-raw mode1 = raw mode This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

Value	Name	Description
0h	Disable	Non-Raw Mode
1h	Enable	Raw Mode

	27	**DirectMbRaw** This field indicates whether the DIRECTMB field is coded in raw or non-raw mode. This field is only valid when PictureType is P or B. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

Value	Name	Description
0h		Non-Raw Mode

1h		Raw Mode

26 OverflagsRaw

This field indicates whether the OVERFLAGS field is coded in raw or non-raw mode. This field is only valid when PictureType is I or BI. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

Value	Name	Description
0h		Non-Raw Mode
1h		Raw Mode

25 AcPredRaw

This field indicates whether the ACPRED field is coded in raw or non-raw mode. This field is only valid when PictureType is I or BI. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

Value	Name	Description
0h	Disable	Non-Raw Mode
1h	Enable	Raw Mode

24 FieldTxRaw

This field indicates whether the FIELDTX field is coded in raw or non-raw mode. This field is only valid when PictureType is I or BI. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

Value	Name	Description
0h	Disable	Non-Raw Mode
1h	Enable	Raw Mode

23 Reserved

Access:	RO
Format:	MBZ

22:20 MvTab (Motion Vector Table)

Format:	U3

This field specifies which motion vector table(s) is (are) used for motion vector (differential) decoding in a P or B picture. This field is the combination of the variables MVTAB and IMVTAB in the VC1 standard. Two bits are defined for progressive frame pictures; And two or three bits are defined for interlaced field/frame pictures depending on NUMREF and P or B picture types. This field is valid for P and B pictures. It is not valid for I pictures. For P or B progressive frame pictures0 = Motion Vector Differential VLD Table 01 = Motion Vector Differential VLD Table 12 = Motion Vector Differential VLD Table 23 = Motion Vector Differential VLD Table 3The other encodings are reserved For P interlace field pictures with NUMREF = 0 or P/B interlace frame pictures0 = 1-Reference Table 01 = 1-Reference Table 12 = 1-Reference Table 23 = 1-Reference Table 3The other encodings are reserved For P interlace field picture with NUMREF = 1 or B interlaced field pictures0 = 2-Reference Table 01 = 2-Reference Table 12 = 2-Reference Table 23 = 2-Reference Table 34 = 2-Reference Table 45 = 2-Reference Table 56 = 2-Reference Table 67 = 2-Reference Table 7The other encodings are reserved This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

	19:18	**FourMvBpTab (4-MV Block Pattern Table)** This field specifies which table is used to decode the 4-MV block pattern (4MVBP) syntax element in 4-MV macroblocks. It is identical to the variables 4MVBPTAB in the VC1 standard, section 9.1.1.37. This field is valid only in interlace frame P, B pictures, or interlace field P, B pictures. It is not valid for I picture. For interlace field P and B pictures, it is only valid if UnifiedMvMode is equal to Mixed-MV Type. For interlace frame P picture, it is only valid if FourMvSwitch is 1.For interlace frame B picture, it is always valid.0 = 4MVBP Table 01 = 4MVBP Table 12 = 4MVBP Table 23 = 4MVBP Table 3This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.
	17:16	**TwoMvBpTab (2MV Block Pattern Table)** This field specifies which table is used to decode the 2MV block pattern (2MVBP) syntax element in 2MV field macroblocks. It is identical to the variables 2MVBPTAB in the VC1 standard, section 9.1.1.36. This field is valid only in interlace frame P/B pictures. It is not valid for I picture, nor for interlace field P or B pictures.0 = 2MVBP Table 01 = 2MVBP Table 12 = 2MVBP Table 23 = 2MVBP Table 3This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.
	15:14	**Reserved** Access: RO Format: MBZ
	13:12	**TransType (Picture-level Transform Type)** Format: U2 This field specifies the Transform Type at picture level. It is identical to the variable TTFRM in the VC1 standard, section 7.1.1.41.This field is only valid when TransTypeMbFlag is 1. Otherwise, it is reserved and MBZ. This field is set to 00 when VSTRANSFORM is 0 in the entry point layer.00 = 8x8 Transform01 = 8x4 Transform10 = 4x8 Transform11 = 4x4 Transform This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.
	11	**TransTypeMbFlag (Macroblock Transform Type Flag)** This field indicates whether Transform Type is fixed at picture level or variable at macroblock level. It is identical to the variable TTMBF in the VC1 standard, section 7.1.1.40.This field is set to 1 when VSTRANSFORM is 0 in the entry point layer. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes. **Value** / **Name** / **Description** 0h / / variable transform type in macroblock layer 1h / / use picture level transform type TransType
	10:8	**MbModeTab (Macroblock Mode Table)** This field signals which code table is used to decode the macroblock mode syntax element (MBMODE) in the macroblock layer in a P or B picture. This field is identical to the variables MBMODETAB in the VC1 standard, section 9.1.1.33. This field is valid for interlace frame P, B picture and interlace field P, B picture. It is not valid for I picture, nor progressive frame P, B pictures. Two bits are defined for interlace frame P, B pictures; And three bits are defined for interlaced field P, B pictures. Two bits are defined for interlace frame P, B pictures. There are

		two set of code tables selected based on if UnifiedMvMode is equal to 4-MV Type or not. 0 = Code Table 01 = Code Table 12 = Code Table 23 = Code Table 3Other encodings are invalid Three bits are defined for interlace field P, B pictures. There are two set of code tables selected based on if UnifiedMvMode is equal to Mixed-MV Type or not. 0 = Code Table 01 = Code Table 12 = Code Table 23 = Code Table 34 = Code Table 45 = Code Table 56 = Code Table 67 = Code Table 7This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.
	7:6	**TransAcY (Picture-level Transform Luma AC Coding Set Index, TRANSACTABLE2** BitFieldDesc
	5:4	**TransAcUV (Picture-level Transform Chroma AC Coding Set Index, TRANSACTABLE)** This field, together with PQINDEX, specifies which intra AC coding set to be used for decoding the non-zero AC coefficients in a coded luma (Y) block. This field is the combination of the variables TRANSACFRM and TRANSACFRM2 in the VC1 standard. For I pictures, TransAcY is the same as TRANSACFRM2. For other pictures, it is the same as TRANSACFRM, and therefore must be programmed to be the same as TransAcUV. This field is valid for all picture types.0 = Coding set index 01 = Coding set index 12 = Coding set index 23 is invalid. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.
	3	**TransDcTab (Intra Transform DC Table)** This field specifies whether the low motion tables or the high motion tables are used to decode the Transform DC coefficients in intra-coded blocks. This field is identical to the variable TRANSDCTAB in the VC1 standard, section 8.1.1.2.This field is valid for all picture types. This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.
	2:0	**CbpTab (Coded Block Pattern Table)** This field specifies the table used to decode the CBPCY syntax element for each coded macroblock in P and B pictures. This field is combination of the variable CBPTAB for P and B frame pictures and the variable ICBPTAB in interlace field P, B pictures and interlace frame P, B pictures in the VC1 standard (Table 52 and Table 102).This field is reserved and MBZ for I or BI pictures as I only has a fixed table.000 = Table 0 (Table 169 for P, B frames or Table 124 otherwise)001 = Table 1 (Table 170 for P, B frames or Table 125 otherwise)010 = Table 2 (Table 171 for P, B frames or Table 126 otherwise)011 = Table 3 (Table 172 for P, B frames or Table 127 otherwise)100 = Table 4 (Table 128 for interlace field/frame P, B pictures)101 = Table 5 (Table 129 for interlace field/frame P, B pictures)110 = Table 6 (Table 130 for interlace field/frame P, B pictures)111 = Table 7 (Table 131 for interlace field/frame P, B pictures)This field is unique to intel VC1 VLD Long format mode, and is not used in IT and VC1 modes.

Value	Name	Description
0h		The high motion tables
1h		The low motion tables

MFD_VC1_SHORT_PIC_STATE

MFD_VC1_SHORT_PIC_STATE

Source: VideoCS

Length Bias: 2

D Word	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_VC1_SHORT_PIC_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 2h VC1_DEC Format: OpCode
	23:21	**SubOpcode A** Default Value: 1h Format: OpCode
	20:16	**SubOpcode B** Default Value: 0h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 0003h Excludes DWord (0,1) Format: =n
1	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**Picture Height** Format: U8-1 This field indicates the height of the picture in unit of macroblocks. For example, for a 1920x1080 frame picture, PictureHeightInMBs equals 68 (1080 divided by 16, and rounded up, i.e. effectively specified as 1088 instead).This field is used in VLD and IT modes. Note: Even though the Advanced Profile allows frame dimensions (width, height) to not be aligned to macroblock boundary, it doesn't affect the bitstream decoding. And it is preferable to use 'intermediate buffer' that is macroblock

MFD_VC1_SHORT_PIC_STATE

Source: VideoCS

Length Bias: 2

D Word	Bit	Description
		aligned for decoding. In order to simplify the out-of-bound reference pixel access, the out-of-bound extrapolation rule in VC1 spec can be used to expand the expected decoded frame to the intermediate buffer dimension.
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**Picture Width** Format: U8-1 This field indicates the width of the picture in unit of macroblocks. For example, for a 1920x1080 frame picture, PictureWidthInMBs equals 120 (1920 divided by 16).This field is used in VLD and IT modes.
2	31:24	**Bitplane Buffer Pitch Minus 1** Format: U8-1 Specifies the bitplane buffer pitch in (#Bytes - 1). Bitplane buffer is a linear buffer. It is needed only when the bitplane is not encoded as raw, and therefore is present in the header explicitly. In VC1 Long Format, it is written by an application and later read by the HW. In VC1 Long Format, it is written by an application, and later read by the HW. But in VC1 Short Format, it is written and read by H/W only. This field is specified for better performance. The pitch must be equal to PictureWidthInMBs/2. For VC1 Long Format : The pitch must be equal to PictureWidthInMBs/2. For VC1 Short Format : If Pic Width is less than or equal to 2K pixels, bitplane pitch is set to 64 (one cacheline; programmed as 63) bytes per MB row. If Pic Width is greater than 2K pixels, bitplane pitch is set to 128 (two cachelines; programmed as 127) bytes per MB row. This field is not used in IT mode, used in VLD mode only. For VC1 Short Format, the bitplane specification is between H/W and Driver only. For Long Format, application is responsible for allocation with the driver.
	23	**Interpolation Rounder Control** Used only in MC operation. This field specifies the rounding control value used in interpolation operation of motion prediction process. Note: This bit field is taken from bRcontrol in PictureParameters data structure This field is used in VLD and IT modes.
	22:20	**Reserved** Access: RO Format: MBZ
	19:16	**Motion Vector Mode** This field indicates one of the following motion compensation interpolation modes for P and B pictures. The MC interpolation modes apply to prediction values of luminance blocks and are always in quarter-sample. For chrominance blocks, it always performs bilinear interpolation with either half-

MFD_VC1_SHORT_PIC_STATE

Source: VideoCS

Length Bias: 2

<table>
<tr><th>D Word</th><th>Bit</th><th>Description</th></tr>
<tr><td></td><td></td><td>pel or quarter-pel precision.0XX0 = Chroma Quarter -pel + Luma bicubic. (can only be 1MV)0XX1 = Chroma Half-pel + Luma bicubic. (can be 1MV or 4MV)1XX0 = Chroma Quarter -pel + Luma bilinear. (can only be 1MV)1XX1 = Chroma Half-pel + Luma bilinearNote: Bits 19:16 are taken from bMVprecisionAndChromaRelation in PictureParameters data structure.Bit 19 of Motion Vector Mode = 1 for Luma Bilinear MC; = 0 for Luma Bicubic MCBit 16 of Motion Vector Mode = 1 for half-sample Chroma motion = 0 for quarter-sample Chroma motion. This field is used in both VLD and IT modes. Before the polarity of Chroma Half-pel or Q-pel is reversed from Spec, now I have fixed it to match with VC1 Spec.</td></tr>
<tr><td></td><td>15</td><td>DmvSurfaceValid
Indicated when the DMV read surface is valid. This surface stored the direct motion vectors. This field is set fo B pictures that can refer to a previous P picture for DMV. If there is an I-picture before a B (in decoding order) then this field is not set (as a result, zero's DMV's will be assumed while decoding the B picture. That is, there is no explicit DMV buffer for an I-picture).This field is not used in IT mode, used in VLD mode only.</td></tr>
<tr><td></td><td>14:12</td><td>Reserved
<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>11</td><td>VC1 Profile
specifies the bitstream profile. Note: This is required because 128 is added for intra blocks post inverse transform in advanced profile and also to find out if Motion vectors are adjusted or not. This field is used in both VLD and IT modes.
<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0h</td><td>[Default]</td><td>current picture is in Simple or Main Profile (No need to distinguish Simple and Main Profile)</td></tr><tr><td>1h</td><td></td><td>current picture is in Advanced Profile</td></tr></table></td></tr>
<tr><td></td><td>10:6</td><td>Reserved
<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td></td><td>5</td><td>Backward Prediction Present Flag
Note : a B picture that only uses forward prediction may have this flag set to 1 as well. Driver may still need to provide a valid reference picture index. This field is used in both VC1 VLD mode and IT mode. It is the same parameter as bPicBackwardPrediction in VC1 spec. The Intra Picture Flag, Backward Prediction Present Flag and RefPicFlag are used to derive the picture type, as specified in PTYPE for a frame, and in FPTYPE for a field, in VC1 VLD and IT mode.</td></tr>
<tr><td></td><td>4</td><td>Intra Picture Flag
This field is used in both VC1 VLD mode and IT mode. It is the same parameter as bPicIntra in VC1 spec. The Intra Picture Flag, Backward Prediction Present Flag and RefPicFlag are used to derive the</td></tr>
</table>

MFD_VC1_SHORT_PIC_STATE

Source: VideoCS

Length Bias: 2

<table>
<tr><th>D Word</th><th>Bit</th><th>Description</th></tr>
<tr><td></td><td></td><td>picture type, as specified in PTYPE for a frame, and in FPTYPE for a field, in VC1 VLD and IT mode.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td></td><td>entire picture can have a mixture of intra and inter MB type or just inter MB type.</td></tr>
<tr><td>1h</td><td></td><td>entire picture is coded in intra MB type</td></tr>
</table></td></tr>
<tr><td></td><td>3</td><td>SecondField
This flag is set for the second field in field pictures. This field is used in both VLD and IT modes.</td></tr>
<tr><td></td><td>2</td><td>Reserved
<table>
<tr><td>Access:</td><td>RO</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>1:0</td><td>Picture Structure
This field is used in both VC1 VLD mode and IT mode. It is the same parameter as bPicStructure in VC1 spec.The Picture Structure and Progressive Pic Type are used to derive the picture structure as specified in FCM, in VC1 VLD and IT mode.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>01b</td><td></td><td>top field (bit 0)</td></tr>
<tr><td>10b</td><td></td><td>bottom field (bit 1)</td></tr>
<tr><td>11b</td><td></td><td>frame (both fields are present)</td></tr>
<tr><td>00b</td><td></td><td>illegal</td></tr>
</table></td></tr>
<tr><td>3</td><td>31</td><td>Reserved
<table>
<tr><td>Access:</td><td>RO</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>30</td><td>Overlap Smoothing Enable Flag
This field is the decoded syntax element OVERLAP in bitstreamIndicates if Overlap smoothing is ON at the picture level This field is used in both VLD and IT modes
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Disable</td><td>to disable overlap smoothing filter</td></tr>
<tr><td>1h</td><td>Enable</td><td>to enable overlap smoothing filter</td></tr>
</table></td></tr>
<tr><td></td><td>29</td><td>Range Reduction Scale
<table>
<tr><td>Access:</td><td>None</td></tr>
</table>
This field specifies whether the reference picture pixel values should be scaled up or scaled down on-the-fly, if RangeReduction is Enabled.NOTE: This bit is derived by driver for Main Profile only. Ignored in Simple and Advanced Profiles. This field is used in both VLD and IT modes. This is derived by driver from the history of RANGERED and RANGEREDFRM syntax elements (i.e. of forward/preceding reference picture) and those of the currentpicture. RANGERED is the same as (bPicOverflowBlocks » 3) & 1.RANGEREDFRM is the same as (bPicDeblocked » 5) & 1. For the current picture is a B picture, this</td></tr>
</table>

MFD_VC1_SHORT_PIC_STATE

Source: VideoCS

Length Bias: 2

D Word	Bit	Description
		field represents the state of the forward/preceding reference picture only Driver is responsible to keep RangeReductionScale, RangeReduction Enable and RANGERED Present Flag of current picture coherent.

Value	Name	Description
0h	Disable **[Default]**	Scale down reference picture by factor of 2
1h	Enable	Scale up reference picture by factor of 2

28 — **Range Reduction Enable**

This field specifies whether on-the-fly pixel value range reduction should be performed for the preceding (or forward) reference picture. Along with RangeReductionScale to specify whether scale up or down should be performed. It is not the same value as RANGEREDFRM Syntax Element(PictureParameters bPicDeblocked bit 5) in the Picture Header. This field is for Main Profile only. Simple Profile is always disable, and not applicable to Advanced Profile. This field is used in both VLD and IT modes. This is derived by driver from the history of RANGERED and RANGEREDFRM syntax elements (i.e. offorward/preceding reference picture) and those of the current picture. RANGEREDis the same as (bPicOverflowBlocks » 3) & 1. RANGEREDFRM is the sameas (bPicDeblocked » 5) & 1.For the current picture is a B picture, this field represents the state of the forward/preceding reference picture only Driver is responsible to keep RangeReductionScale, RangeReduction Enable and RANGERED Present Flag of current picture coherent.

Value	Name	Description
0h	Disable **[Default]**	Range reduction is not performed
1h	Enable	Range reduction is performed

27:24 — **Reserved**

Access:	RO
Format:	MBZ

23:22 — **Progressive Pic Type**

This field is used in both VC1 VLD mode and IT mode. It is the same parameter as bPicExtrapolation in VC1 spec. The Picture Structure and Progressive Pic Type are used to derive the picture structure as specified in FCM, in VC1 VLD and IT mode.

Value	Name	Description
0		progressive only picture
1		progressive only picture
2		interlace picture (frame-interlace or field-interlace)
3		illegal

21 — **Reserved**

Access:	RO
Format:	MBZ

MFD_VC1_SHORT_PIC_STATE

Source: VideoCS

Length Bias: 2

D Word	Bit	Description
	20:16	**P-Pic Ref Distance** Access: None This element defines the number of frames between the current frame and the reference frame. It is the same as the REFDIST SE in VC1 interlaced field picture header. It is present if the entry-level flag REFDIST_FLAG == 1, and if the picture type is not one of the following types: B/B, B/BI, BI/B, BI/BI. If the entry level flag REFDIST_FLAG == 0, REFDIST shall be set to the default value of 0.This field is used in VC1 VLD mode only, not used in IT and intel VC1 VLD Long Format modes. Value: 0-16 — unsigned integer Value: 0h — **[Default]**
	15:14	**QUANTIZER** 00b: implicit quantizer at frame leve 01b: explicit quantizer at frame level, and use PQUANTIZER SE to specify uniform or non-uniform 10b: explicit quantizer, and non-uniform quantizer for all frames 11b: explicit quantizer, and uniform quantizer for all frames
	13	**MULTIRES Present Flag (for Simple/Main Profile only)** 0h: RESPIC Parameter is present in the picture header 1h: RESPIC Parameter is present in the picture header
	12	**SYNCMARKER Present Flag (for Simple/Main Profile only)** 0: Bitstream for Simple and Main Profile has no sync marker 1: Bitstream for Simple and Main Profile may have sync marker(s)
	11	**RANGERED Present Flag (for Simple/Main Profile only)** It is needed for Picture Header Parsing. Driver is responsible to keep RangeReductionScale, RangeReduction Enable and RANGERED Present Flag of current picture coherent. 0: Range Reduction Parameter (RANGEREDFRM) is not present in the picture header 1: Range Reduction Parameter (RANGEREDFRM) is present in the picture header.
	10:8	**MAXBFRAMES** Number of consecutive B Frames.

MFD_VC1_SHORT_PIC_STATE

Source: VideoCS

Length Bias: 2

<table>
<tr><th>D Word</th><th>Bit</th><th>Description</th></tr>
<tr><td></td><td>7</td><td>PANSCAN Present Flag
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td></td><td>Pan Scan Parameters are not present in the picture header</td></tr>
<tr><td>1</td><td></td><td>Pan Scan Parameters are present in the picture header</td></tr>
</table></td></tr>
<tr><td></td><td>6</td><td>REFDIST_FLAG
For header parsing REFDIST. This is used in VC1 VLD mode only, not used in IT and intel VC1 VLD modes.</td></tr>
<tr><td></td><td>5</td><td>Reserved</td></tr>
<tr><td></td><td>4</td><td>FastUVMCFlag (Fast UV Motion Compensation Flag)
This field specifies whether the motion vectors for UV is rounded to half or full pel position. It is identical to the variableFASTUVMC in VC1 standard .This field is used in both VLD and IT modes. It is derived from FASTUVMC = (bPicSpatialResid8 » 4) & 1 in both VLD and IT modes, and should have the same value as Motion Vector ModeLSBit.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td></td><td>no rounding</td></tr>
<tr><td>1h</td><td></td><td>quarter-pel offsets to half/full pel positions</td></tr>
</table></td></tr>
<tr><td></td><td>3</td><td>EXTENDED_MV Present Flag
BitFieldDesc
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td></td><td>Extended_MV is not present in the picture header</td></tr>
<tr><td>1h</td><td></td><td>Extended_MV is present in the picture header</td></tr>
</table></td></tr>
<tr><td></td><td>2:1</td><td>DQUANT
<table>
<tr><td>Access:</td><td>None</td></tr>
<tr><td>Format:</td><td>U2</td></tr>
</table>
Use for Picture Header Parsing of VOPDUANT elements
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>[Default]</td><td></td></tr>
<tr><td>00b</td><td></td><td>no VOPDQUANT elements; Quantizer cannot vary in frame, same quantization step size PQUANT is used for all MBs in the frame</td></tr>
<tr><td>01b</td><td></td><td>refer to VC1 Spec. for all the MB position dependent quantizer selection</td></tr>
<tr><td>10b</td><td></td><td>The macroblocks located on the picture edge boundary shall be quantized with ALTPQUANT while the rest of the macroblocks shall be quantized with PQUANT.</td></tr>
</table></td></tr>
</table>

MFD_VC1_SHORT_PIC_STATE

Source: VideoCS

Length Bias: 2

D Word	Bit	Description
		11b Reserved
	0	**VSTRANSFORM flag** Value 0h: Disable — variable-sized transform coding is not enabled Value 1h: Enable — variable-sized transform coding is enabled
4	31:29	**Reserved** Access: RO Format: MBZ
	28:24	**BFraction Enumeration** This field is the scale factor for computing Direct-mode motion vectors. It is derived from the variable BFRACTION in the VC1standard, section 8.4.5.4.There are only 21 valid values corresponding to the 21encodings of BFRACTION as shown in the table here. Other values are reserved. The VLD decoded value of BFRACTION (from the picture header) is mapped into an enum value from 0 to 20.(??? MSB of this field can be used to determine if BFRACTION is greater than or equal to 1/2, which is used to determine MotionPrediction Type for B pictures. Effectively, condition "BFRACTION >= 1/2" is equivalent to condition "ScaleFactor >= 128". ??? How can the enum replicate this feature ???)This field is only valid for B pictures. This field is usedonly in VC1 VLD Long Format mode and IT mode.BFRACTIONVLCBFRACTIONEnum0001/200011/310102/320111/431003/441011/551102/5611100003/5711100014/5811100101/6911100115/61011101001/71111101012/71211101103/71311101114/71411110005/71511110016/71611110101/81711110113/81811111005/81911111017/8201111111BIPic Indicator31 (optional)
	23:9	**Reserved** Access: RO Format: MBZ
	8	**4MV Allowed Flag**
	7	**POSTPROC Flag**
	6	**PULLDOWN**
	5	**INTERLACE**
	4	**TFCNTRFLAG**
	3	**FINTERFLAG**
	2	**REFPIC Flag** For a BI picture, REFPIC flag must set to 0For I and P picture, REFPIC flag must set to 0.For a B picture, REFPIC flag must set to 0, except for a B-field in interlaced field mode which can be 0 or 1 (e.g. the top B field can be used as a reference for decoding its corresponding bottom B-field in a field pair).In VLD mode, this flag cannot be used as an optimization signaling for an I or P picture that is not used as a

MFD_VC1_SHORT_PIC_STATE

Source: VideoCS

Length Bias: 2

<table>
<tr><th>D Word</th><th>Bit</th><th>Description</th></tr>
<tr><td></td><td></td><td>reference picture. This field is used in both VC1 VLD mode and IT mode. It is the same parameter as bPicDeblockConfined[bit2] in VC1 spec. The Intra Picture Flag, Backward Prediction Present Flag and RefPicFlag are used to derive the picture type, as specified in PTYPE for a frame, and in FPTYPE for a field, in VC1 VLD and IT mode.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td></td><td>the current picture after decoded, will never used as a reference picture</td></tr>
<tr><td>1h</td><td></td><td>the current picture after decoded, will be used as a reference picture later</td></tr>
</table></td></tr>
<tr><td></td><td>1</td><td>PSF</td></tr>
<tr><td></td><td>0</td><td>EXTENDED_DMV Present Flag
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>[Default]</td><td>Extended_DMV is not present in the picture header</td></tr>
<tr><td>1h</td><td></td><td>Extended_DMV is present in the picture header</td></tr>
</table></td></tr>
</table>

MFD_VP8_BSD_OBJECT

MFD_VP8_BSD_OBJECT

Source: VideoCS

Length Bias: 2

The MFD_VP8_BSD_OBJECT command is the only primitive command for the VP8 Decoding Pipeline. The Partitions of the bitstream is loaded as indirect data object. Before issuing a MFD_VP8_BSD_OBJECT command, all VP8 frame level states of the MFD Engine need to be valid. Therefore the commands used to set these states need to have been issued prior to the issue of a MFD_VP8_BSD_OBJECT command. Context switch interrupt is not supported by this command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFD_VP8_BSD_OBJECT Format: OpCode
	26:24	**Media Command OpCode** Default Value: 4h VP8_DEC Format: OpCode
	23:21	**subOpcodeA** Default Value: 1h Format: OpCode
	20:16	**subOpcodeB** Default Value: 8h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 14h Excludes DWord (0,1) Format: =n
1	31:21	**Reserved** Access: RO Format: MBZ
	20:16	**Partition0 CPBAC Entropy Count** Pass the Partition0 CPBAC State to HW.Max value is 24.

MFD_VP8_BSD_OBJECT

	15:8	**Partition0 CPBAC Entropy Range** Pass the Partition0 CPBAC State to HW.
	7:6	**Reserved** Access: RO Format: MBZ
	5:4	**Coded Num of Coeff Token Partitions** Num of Partitions = 2^CodedNumCoeffTokenParititons.0 = 1 Partition only1 = 2 Partitions2 = 4 Partitions3 = 8 Partitions are present in the bitstream.
	3	**Reserved** Access: RO Format: MBZ
	2:0	**Partition0 FirstMBBitOffset from Frame Header** Allow HW to jump to the location in the bitstream where per MB information starts in the Partition0.
2	31:24	**Partition0 CPBAC Entropy Value** Pass the Partition0 CPBAC State to HW.
	23:0	**Reserved** Access: RO Format: MBZ
3	31:24	**Reserved** Access: RO Format: MBZ
	23:0	**Indirect Partition0 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends.It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. **Programming Notes** This needs to be set to the (actual Partition 0 length + 1) in bytes
4	31:0	**Indirect Partition0 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
5	31:24	**Reserved** Access: RO Format: MBZ

MFD_VP8_BSD_OBJECT

	23:0	**Indirect Partition1 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. **Programming Notes** This needs to be set to the (actual Partition 1 length + 1) in bytes
6	31:0	**Indirect Partition1 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
7	31:24	**Reserved** Access: RO Format: MBZ
	23:0	**Indirect Partition2 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. **Programming Notes** This needs to be set to the (actual Partition 2 length + 1) in bytes
8	31:0	**Indirect Partition2 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
9	31:24	**Reserved** Access: RO Format: MBZ
	23:0	**Indirect Partition3 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. **Programming Notes** This needs to be set to the (actual Partition 3 length + 1) in bytes

MFD_VP8_BSD_OBJECT

10	31:0	**Indirect Partition3 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
11	31:24	**Reserved** Access: RO Format: MBZ
	23:0	**Indirect Partition4 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. **Programming Notes** This needs to be set to the (actual Partition 4 length + 1) in bytes
12	31:0	**Indirect Partition4 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
13	31:24	**Reserved** Access: RO Format: MBZ
	23:0	**Indirect Partition5 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. **Programming Notes** This needs to be set to the (actual Partition 5 length + 1) in bytes
14	31:0	**Indirect Partition5 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
15	31:24	**Reserved** Access: RO Format: MBZ

MFD_VP8_BSD_OBJECT

	23:0	**Indirect Partition6 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. **Programming Notes** This needs to be set to the (actual Partition 6 length + 1) in bytes
16	31:0	**Indirect Partition6 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
17	31:24	**Reserved** Access: RO Format: MBZ
	23:0	**Indirect Partition7 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends.It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. **Programming Notes** This needs to be set to the (actual Partition 7 length + 1) in bytes
18	31:0	**Indirect Partition7 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
19	31:24	**Reserved** Access: RO Format: MBZ
	23:0	**Indirect Partition8 Data Length** This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Partition Start Offset field is ignored. The Partition is byte aligned in both ends. It is the length in bytes of the bitstream data for the current partition. It includes the first byte of the first macroblock and the last byte of the last macroblock in the partition. **Programming Notes** This needs to be set to the (actual Partition 8 length + 1) in bytes

20	31:0	**Indirect Partition8 Data Start Offset** This field specifies the Graphics Memory starting address of the data to be fetched into BSD Unit for processing. This offset is relative to the MFD Indirect Object Base Address. Hardware ignores this field if indirect data is not present. It is a byte-aligned address for the VP8 bitstream data in each partition.
21	31	**Concealment Method** This field specifies the method used for concealment when error is detected. **Value** / **Name** / **Description** 0 / Intra 16x16 Prediction / A copy from the current picture is performed using Intra 16x16 Prediction method. 1 / Inter P Copy / A copy from collocated macroblock location is performed from the concealment reference indicated by the ConCeal_Pic_Id field.
	30:18	**Reserved** Access: RO Format: MBZ
	17:16	**Conceal_Pic_Id (Concealment Picture ID)** Exists If: [Concealment Method] == 1 This field identifies the picture in the reference list to be used for concealment. This field is only valid if Concealment Method is Inter P Copy.00 - Last Decoded Picture01 - Golden Reference Picture02 - Alternate Reference Picture03 - User provided Reference Picture
	15	**Reserved** Access: RO Format: MBZ
	14	**BSDPrematureComplete Error Handling** It occurs in situation where the decode is completed but there are still data in the bitstream. **Value** / **Name** 0 / Ignore the error and continue (masked the interrupt), assume the hardware automatically perform the error handling 1 / Set the interrupt to the driver (provide MMIO registers for MB address R/W)
	13	**Reserved** Access: RO Format: MBZ
	12	**MPR Error (MV out of range) Handling** **Value** / **Name** 0 / Ignore the error and continue (masked the interrupt), assume the hardware automatically perform the error handling 1 / Set the interrupt to the driver (provide MMIO registers for MB address R/W)

MFD_VP8_BSD_OBJECT

	11	**Reserved** Access: RO Format: MBZ
	10	**Entropy Error Handling** 0 – Ignore the error and continue (masked the interrupt), assume the hardware automatically perform the error handling 1 – Set the interrupt to the driver (provide MMIO registers for MB address R/W)
	9	**Reserved** Access: RO Format: MBZ
	8	**MB Header Error Handling** 0 – Ignore the error and continue (masked the interrupt), assume the hardware automatically perform the error handling 1 – Set the interrupt to the driver (provide MMIO registers for MB address R/W)
	7:0	**Reserved** Access: RO Format: MBZ

MFX_AVC_DIRECTMODE_STATE

MFX_AVC_DIRECTMODE_STATE

Source: VideoCS

Length Bias: 2

This is a picture level command and is issued once per picture. All DMV buffers are treated as standard media surfaces, in which the lower 6 bits are used for conveying surface states. Current Pic POC number is assumed to be available in POCList[32 and 33] of the MFX_AVC_DIRECTMODE_STATE Command. This command is only valid in the AVC decoding in VLD and IT modes, and AVC encoder mode. The same command supports both Long and Short AVC Interface. The DMV buffers are not required to be programmed for encoder mode.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_SINGLE_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h AVC_COMMON Format: OpCode
	23:21	**SubOpcodeA** Default Value: 0h Format: OpCode
	20:16	**SubOpcodeB** Default Value: 2h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 0045h Excludes DWord (0,1) Format: =n
1..32	1023:0	**Direct MV Buffer for Reference Frame 0 to 15 - Base Address** Format: **SplitBaseAddress64ByteAligned[16]** This field is for the Pre-Deblocking Destination Address, and provides the base address of the DMV buffer for reference frames 2 to 31. They are needed if the current B-Picture has MBs coded in direct mode. This is a private buffer used by the MPR hardware only. Its content is not accessed by software. All these buffers must be 64-byte cacheline aligned. There are a total of

MFX_AVC_DIRECTMODE_STATE

		32 possible Direct MV Read Buffers (not including the current write buffer of the current picture) to read in the corresponding DMV. Each read buffer size is 557,056 bytes for 1 frame (the selected colPic). Scalable with frame height, but do not scale with frame width as the hardware assumes frame width (in MBs) fixed at 128 (smallest power of 2 value larger than 120 - 1920x1088 screen resolution). The adjacent DMV buffers are paired ([2 and 3], [4 and 5], [N and N+1], ..[30 and 31]). This field is changed to one per frame: both top and bottom field share the same Direct MV Buffer Base Address. **Programming Notes** This field is ignored if PreDeblockOutEnable is set to 0 (disable).
33	31:0	**Direct MV Buffer for Reference Frame 0 to 15 - Attributes** Format: **MemoryAddressAttributes**
34..35	63:0	**Direct MV Buffer for Write - Base Address** Format: **SplitBaseAddress64ByteAligned** This field provides the base address of the DMV write-only buffer for the current decoding frame/field. It is a private buffer used by the MPR hardware only. Its content is not accessed by software. All these buffers must be 64-byte cacheline aligned, i.e. the same as the above DMV read/write buffers. These 2 buffers can only be addressed by [img_dec_fs_idc[4:0]«1 + img_structure[1]] for the current picture being decoded. Each write buffer size is 557,056 bytes for 1 frame (the selected colPic). Scalable with frame height, but do not scale with frame width as the hardware assumes frame width (in MBs) fixed at 128 (smallest power of 2 value larger than 120 - 1920x1088 screen resolution). DMV write buffer 32 is valid only if the current picture is a progressive frame, MbAff frame, or a top field. DMV write buffer 33 is valid only if the current picture is a bottom field. GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and MBZ. Some GraphicsAddress fields only specify the upper address bits. For example, GraphicsAddress[47:12] is a 4KB page address.
36	31:0	**Direct MV Buffer for Write - Attributes** Format: **MemoryAddressAttributes**
37..70	1087:0	**POCList[34][31:0]** Each POC value is a signed 32-bit number.One-to-one correspondence with the 34 Direct MV Buffer Address for Reference and Currrent Frames/Fields There are 34 POC entries in the list. For reference picture, only the lower 32 POC [0-31] entries can be used, and POCList[ ] is indexed by the frame_store_ID[4:0], which is obtained from RefPicList L0/L1[RefPicIdx]. frame_Store_IDbit[0] (indicator for Top/Bottiom Field).For current picture, all 34 POC entries [0-33] can be addressed by POCList[img_dec_fs_idc[4:0]«1 + img_structure[1]].For frame-only mode, every other entry is skipped. For MBAFF and field-only picture, each entry is a field POC, and every two entries are paired.

MFX_AVC_IMG_STATE

MFX_AVC_IMG_STATE

Source: VideoCS

Length Bias: 2

This must be the very first command to issue after the surface state, the pipe select and base address setting commands. This command supports both Long and Short VLD and IT AVC Decoding Interface.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_AVC_IMG_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h AVC_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 0h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 0Ch Excludes DWord (0,1) Format: =n
1	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Frame Size** Format: U16-1 The value for FrameSizeInMBs must match the product of FrameWidthInMBs and FrameHeightInMBs, Max. Screen resolution is therefore limited to 256 x 256 in MB unit. It is enough to cover all the Profile-Level specified in the current HD-DVD specification. E.g.. for 1920x1080,

FrameSizeInMBs[15:0] = 8160 (1920/16 * 1088/16; rounded up 1080.) This parameter is specified for Intel interface only.

Value	Name	Description
[0,16383]		representing Number of MBs [1,16384]

2 | 31:24 | Reserved

Access:	RO
Format:	MBZ

23:16 | Frame Height

Format:	U8-1

It is set to the value of (FrameHeightInMBsMinus1+ 1). Since the max value for FrameHeightInMBs is 255, the max allowed value for FrameHeightInMBsMinus1 is only 254. The min value for FrameHeightInMBs is 1.Although the max. value that can be specified for FrameHeightInMBs is 255 (in the current implementation), FrameWidthInMBs * FrameHeightInMBs must not exceed the max value of FrameSizeInMBs[14:0].e.g. for 1920x1080, FrameHeightInMBs[7:0] is equal to 68 (1080 divided by 16, and rounded up, i.e. effectively specified as 1088 instead).It is derived from FrameHeightInMbs = (2 - frame_mbs_only_flag) * PicHeightInMapUnits and PicHeightInMbs = FrameHeightInMbs / (1 + field_pic_flag) internally done. For MBAFF, PicHeightInMapUnits is in MB pair unit, so the bitstream sends only half frame height.

Value	Name	Description
[0,255]		representing height [1,256]

15:8 | Reserved

Access:	RO
Format:	MBZ

7:0 | Frame Width

Format:	U8-1

It is set to the value of (FrameWidthInMBsMinus1+ 1). Since the max value for FrameWidthInMBs is 255, the max allowed value for FrameWidthInMBsMinus1 is only 254. The min value for FrameWidthInMBs is 1.Although the max. value that can be specified for FrameWidthInMBs is 255 (in the current implementation), FrameWidthInMBs * FrameWidthInMBs must not exceed the max value of FrameSizeInMBs[14:0].e.g. for 1920x1080, FrameHeightInMBs[7:0] is equal to 68 (1080 divided by 16, and rounded up, i.e. effectively specified as 1088 instead).It is derived from FrameWidthInMbs = (2 - frame_mbs_only_flag) * PicWidthInMapUnits and PicWidthInMbs = FrameWidthInMbs / (1 + field_pic_flag) internally done. For MBAFF, PicWidthInMapUnits is in MB pair unit, so the bitstream sends only half frame width.

Value	Name	Description
[0,255]		representing width [1,256]

3 | 31:29 | Reserved

Access:	RO
Format:	MBZ

	28:24	**Second Chroma QP Offset** Signed integer value. It should be in the range of -12 to +12 (according to AVC spec).It specifies the offset for determining QP Cr from QP Y. It is set to the upper 5 bits of the value of the syntax element (Chroma_qp_offset[9:0]) read from the current active PPS.Chroma_qp_offset [4:0] - chroma_qp_offset_bits (from the current active PPS)Chroma_qp_offset [9:5] - second_chroma_qp_offset_bits
	23:21	**Reserved** Access: RO Format: MBZ
	20:16	**First Chroma QP Offset** Signed integer value. It should be in the range of -12 to +12 (according to AVC spec).It specifies the offset for determining QP Cb from QP Y. It is set to the lower 5 bits of the value of the syntax element (Chroma_qp_offset[9:0]) read from the current active PPS.Chroma_qp_offset [4:0] - chroma_qp_offset_bits (from the current active PPS)Chroma_qp_offset [9:5] - second_chroma_qp_offset_bits
	15:14	**Reserved** Access: RO Format: MBZ
	13	**RhoDomain Rate Control Enable** Format: Enable This field indicates if RhoDomain related parameters are present in the MFX_AVC_IMAGE_STATE. (AverageMacroblockQP). It enables the Rho Domain statistics collection. **Value** / **Name** / **Description** 0 / Disable / RhoDomain rate control parameters are not present in MFX_AVC_IMAGE_STATE 1 / Enable / RhoDomain rate control parameters are present in MFX_AVC_IMAGE_STATE. **Programming Notes** This field must set to '0' for B pictures.
	12	**Weighted_Pred_Flag** Format: Enable (follows strictly AVC interface.) **Value** / **Name** / **Description** 0 / Disable **[Default]** / specifies that weighted prediction is not used for P and SP slices 1 / Enable / specifies that weighted prediction is used for P and SP slices

MFX_AVC_IMG_STATE

		Programming Notes This field must set to '0' for B and I pictures.
	11:10	**Weighted_BiPred_Idc** (follows strictly AVC interface.) **Value** / **Name** / **Description** 0 / DEFAULT **[Default]** / Specifies that the default weighted prediction is used for B slices 1 / EXPLICIT / Specifies that explicit weighted prediction is used for B slices 2 / IMPLICIT / Specifies that implicit weighted prediction is used for B slices. 3 / Reserved / Illegal value **Programming Notes** This field must set to 0 for P and I pictures.
	9:8	**ImgStruct - Image Structure, img_structure[1:0]** The current encoding picture structure can only takes on 3 possible values **Value** / **Name** 00b / Frame Picture 01b / Top Field Picture 11b / Bottom Field Picture 10b / Invalid, not allowed. **Programming Notes** img_structure[0] can be used as a flag to distinguish between frame and field structure. It must be consistent with the field_pic_flag setting in the Slice Header. This parameter is specified for Intel interface only, as a separate state (instead the img_structure[1] is embedded inside the picture definition).
	7:0	**Reserved** Access: RO Format: MBZ
4	31:16	**MinFrameWSize** Default Value: 0h Format: U16 **Minimum Frame Size [15:0] (in Word, 16-bit)(Encoder Only)** Minimum Frame Size is specified to compensate for intel Rate Control Currently zero fill (no need to perform emulation byte insertion) is done only to the end of the CABAC_ZERO_WORD insertion (if any) at the last slice of a picture. Intel encoder parameter. The caller should always make sure that the value, represented by Minimum Frame Size, is always less than maximum frame size **FrameBitRateMax (DWORD 10 bits** 29:16). This field is reserved

		in Decode mode. The programmable range 02^18-1 When MinFrameWSizeUnits is 00. Programmable range is 02^20-1 when MinFrameWSizeUnits is 01. Programmable range is 02^26-1 when MinFrameWSizeUnits is 10. Programmable range is 02^32-1 when MinFrameWSizeUnits is 11.
	15	**MbStatEnabled** Format: Enable Enable reading in MB status buffer (a.k.a. encoding stream-out buffer) Note: For multi-pass encoder, all passes except the first one need to set this value to 1. By setting the first pass to 0, it does save some memory bandwidth. In VDenc mode this must be set to zero as no MB level rate control is used. **Value / Name / Description** 0 / Disable / Disable Reading of Macroblock Status Buffer 1 / Enable / Enable Reading of Macroblock Status Buffer
	14	**LoadSlicePointerFlag** Format: Enable LoadBitStreamPointerPerSlice (Encoder-only)To support multiple slice picture and additional header/data insertion before and after an encoded slice. When this field is set to 0, bitstream pointer is only loaded once for the first slice of a frame. For subsequent slices in the frame, bitstream data are stitched together to form a single output data stream. When this field is set to 1, bitstream pointer is loaded for each slice of a frame. Basically bitstream data for different slices of a frame will be written to different memory locations. **Value / Name / Description** 0 / Disable / Load BitStream Pointer only once for the first slice of a frame 1 / Enable / Load/reload BitStream Pointer only once for the each slice, reload the start location of the bitstream buffer from the Indirect PAK-BSE Object Data Start Address field
	13	**Reserved**
	12	**MvUnpackedFlag** MVUnPackedEnable (Encoder Only)This field is reserved in Decode mode. **Value / Name / Description** 0 / PACKED / use packed MV format 1 / UNPACKED / use unpacked 8MV/32MV format only
	11:10	**ChromaFormatIdc** Chroma Format IDC, ChromaFormatIdc[1:0]It specifies the sampling of chroma component (Cb, Cr) in the current picture as follows : **Value / Name / Description** 00b / monochrome picture / Desc

01b	4:2:0 picture	Desc
10b	4:2:2 picture (not supported)	
11b	4:4:4 picture (not supported)	

Programming Notes
It is set to the value of the syntax element read from the current active SPS. The corresponding Monochrome Flag (monochrome_flag) can be derived from this field.

9 **Reserved**

Access:	RO
Format:	MBZ

8 **MbMvFormatFlag**

Use MB level MvFormat flag (Encoder Only)

Value	Name	Description
0	IGNORE	HW PAK ignore MvFormat in the MB data. When bit 12 == 0, all MBs use packed MV format When bit 12 == 1, each MB data must use unpacked MV format, 8MV when there is no minor MV involved, and 32MV if there are some minor MVs.
1	FOLLOW	HW PAK will follow MvFormat value set within each MB data.

Programming Notes
They must take one of the two values: the 8MV unpacked format (MvFormat =101b), and the 32MV unpacked format (MvFormat =110b).This bit can be set only when MvUnpackedFlag (bit 12 of this register) is set otherwise system could hang.

7 **EntropyCodingFlag**

Entropy Coding Flag, entropy_coding_flag

Value	Name	Description
0	CAVLC bit-serial encoding mode	Desc
1	CABAC bit-serial encoding mode.	Desc

Programming Notes
It specifies one of the two possible bit stream encoding modes in the AVC. It is set to the value of the syntax element read from the current active PPS.

6 **ImgDisposableFlag**

Current Img Disposable Flag or Non-Reference Picture Flag

Value	Name	Description
0	REFERENCE	the current decoding picture may be used as a reference picture for others
1	DISPOSABLE	the current decoding picture is not used as a reference picture (e.g. a B-picture cannot be a reference picture for any subsequent decoding)

Programming Notes
It is derived from ImgDisposableFlag = (nal_ref_idc == 0). nal_ref_idc is a syntax element from a NAL unit. When this flag is set, no reference picture and DMV are written out.This field is only valid for VLD decoding mode.

5 **ConstrainedIPredFlag**

Constrained Intra Prediction Flag, constrained_ipred_flag It is set to the value of the syntax element in the current active PPS.

Value	Name	Description
0	INTRA_AND_INTER	allows both intra and inter neighboring MB to be used in the intra-prediction encoding of the current MB.
1	INTRA_ONLY	allows only to use neighboring Intra MBs in the intra-prediction encoding of the current MB. If the neighbor is an inter MB, it is considered as not available.

4 **Direct8x8InfFlag**

Direct 8x8 Inference Flag, direct_8x8_inference_flagIt is set to the value of the syntax element in the current active SPS. It specifies the derivation process for luma motion vectors in the Direct MV coding modes (B_Skip, B_Direct_16x16 and B_Direct_8x8). When frame_mbs_only_flag is equal to 0, direct_8x8_inference_flag shall be equal to 1.It must be consistent with the frame_mbs_only_flag and transform_8x8_mode_flag settings.

Value	Name	Description
0	SUBBLOCK	allows subpartitioning to go below 8x8 block size (i.e. 4x4, 8x4 or 4x8)
1	BLOCK	allows processing only at 8x8 block size. MB Info is stored for 8x8 block size.

3 **Transform8x8Flag**

8x8 IDCT Transform Mode Flag, trans8x8_mode_flagSpecifies 8x8 IDCT transform may be used in this picture It is set to the value of the syntax element in the current active PPS.

Value	Name	Description
0	4x4	no 8x8 IDCT Transform, only 4x4 IDCT transform blocks are present
1	8x8	8x8 Transform is allowed

2 **FrameMbOnlyFlag**

Frame MB only flag, frame_mbs_only_flagIt is set to the value of the syntax element in the current active SPS.

Value	Name	Description
0	FALSE	not true ; effectively enables the possibility of MBAFF mode.
1	TRUE	true, only frame MBs can occur in this sequence, hence disallows the MBAFF mode and field picture.

MFX_AVC_IMG_STATE

	1	**MbaffFlameFlag** MBAFF mode is active, mbaff_frame_flag.It is derived from MbaffFrameFlag = (mb_adaptive_frame_field_flag && ! field_pic_flag). mb_adaptive_frame_field_flag is a syntax element in the current active SPS and field_pic_flag is a syntax element in the current Slice Header. They both are present only if frame_mbs_only_flag is 0. Although mbaff_frame_flag is a Slice Header parameter, its value is expected to be the same for all the slices of a picture.It must be consistent with the mb_adaptive_frame_field_flag, the field_pic_flag and the frame_mbs_only_flag settings. This bit is valid only when the img_structure[1:0] indicates the current picture is a frame. **Value** / **Name** / **Description** 0 / FALSE / not in MBAFF mode 1 / TRUE / in MBAFF mode
	0	**FieldPicFlag** Field picture flag, field_pic_flag, specifies the current slice is a coded field or not. It is set to the same value as the syntax element in the Slice Header. It must be consistent with the img_structure[1:0] and the frame_mbs_only_flag settings. Although field_pic_flag is a Slice Header parameter, its value is expected to be the same for all the slices of a picture. **Value** / **Name** / **Description** 0h / FRAME / a slice of a coded frame 1h / FIELD / a slice of a coded field
5 [ExistsIf]Encode Only	31	**Trellis Quantization Enabled (TQEnb)** Format: Enable The TQ improves output video quality of AVC CABAC encoder by selecting quantized values for each non-zero coefficient so as to minimize the total R-D cost. This flag is only valid AVC CABAC mode. Otherwise, this flag should be disabled. **Value** / **Name** / **Description** 0h / Disable / Use Normal 1h / Enable / Use Trellis quantization In VDENC mode, Enable Trellis Quantization in MFX. VDENC can control trellis Quantization based on it's programming.
	30:28	**Trellis Quantization Rounding (TQR)** This rounding scheme is only applied to the quantized coefficients ranging from 0 to 1 when TQEnb is set to 1 in AVC CABAC mode. One of the following values is added to quantized coefficients before truncating fractional part. **Value** / **Name** / **Description** 000b / / Add 1/8 001b / / Add 2/8 010b / / Add 3/8 011b / **[Default]** / Add 4/8 (rounding 0.5) 100b / / Add 5/8

101b		Add 6/8
110b	Default	Add 7/8 (Default rounding 0.875)

27 **Trellis Quantization Chroma Disable (TQChromaDisable)**

This signal is used to disable chroma TQ. To enable TQ for both luma and chroma, TQEnb=1, TQChromaDisable=0. To enable TQ only for luma, TQEnb=1, TQChromaDisable=1.

Value	Name	Description
0h		Enable Trellis Quantization chroma
1h	Default	Disable Trellis Quantization chroma

26:17 **Reserved**

Access:	RO
Format:	MBZ

16 **NonFirstPassFlag**

This signals the current pass is not the first pass. It will imply designate HW behavior: e.g

Value	Name	Description
0h	Disable	Always use the MbQpY from initial PAK inline object for all passes of PAK
1h	Enable	Use MbQpY from stream-out buffer if MbRateCtrlFlag is set to 1

15:12 **Reserved**

Access:	RO
Format:	MBZ

11:10 **MinFrameWSizeUnits**

This field is the Minimum Frame Size Units

Value	Name	Description
00b	compatibility mode	Minimum Frame Size is in old mode (words, 2bytes)
01b	16 byte	Minimum Frame Size is in 16bytes
10b	4Kb	Minimum Frame Size is in 4Kbytes
11b	16Kb	Minimum Frame Size is in 16Kbytes

9 **MbRateCtrlFlag - MB level Rate Control Enabling Flag**

MB Rate Control conformance mask

In VDenc mode, this field must be zero as frame level rate control is used.

Value	Name	Description
0h	Disable	Apply accumulative delta QP for consecutive passes on top of the macroblock QP values in inline data
1h	Enable	Apply RC QP delta to suggested QP values in Macroblock Status Buffer except the first pass.

		Programming Notes This field is ignored when MacroblockStatEnable is disabled or MB level Rate control flag for the current MB is disable in Macroblock Status Buffer.
	8	**Reserved** Access: RO Format: MBZ
	7	**Intra/InterMbIpcmFlag - ForceIPCMControlMask** This field is to Force **IPCM** for Intra or Inter Macroblock size conformance mask. **Value** / **Name** / **Description** 0h / Disable / Do not change intra or Inter macroblocks even 1h / Enable / Change intra or Inter macroblocks MB_type to IPCM **Programming Notes** This field is ignored when MacroblockStatEnable is disabled or MB level Intra MB conformance flag for the current MB is disable in Macroblock Status Buffer.
	6:4	**Reserved** Access: RO Format: MBZ
	3	**FrameSzUnderFlag - FrameBitRateMinReportMask** This is a mask bit controlling if the condition of frame level bit count is less than FrameBitRateMin **Value** / **Name** / **Description** 0h / Disable / Do not update bit0 of MFC_IMAGE_STATUS control register. 1h / Enable / set bit0 and bit 1of MFC_IMAGE_STATUS control register if the total frame level bit counter is less than or equal to Frame Bit rate Minimum limit.
	2	**FrameSzOverFlag - FrameBitRateMaxReportMask** This is a mask bit controlling if the condition of frame level bit count exceeds FrameBitRateMax. **Value** / **Name** / **Description** 0 / Disable / Do not update bit0 of MFC_IMAGE_STATUS control register. 1 / Enable / Set bit0 and bit 1 of MFC_IMAGE_STATUS control register if the total frame level bit counter is greater than or equal to Frame Bit rate Maximum limit.
	1	**InterMbMaxBitFlag - InterMBMaxSizeReportMask** This is a mask bit controlling if the condition of any inter MB in the frame exceeds InterMBMaxSize. **Value** / **Name** / **Description** 0 / Disable / Do not update bit0 of MFC_IMAGE_STATUS control register.

		1 | Enable | Set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Inter MB Conformance Max size limit.
	0	**IntraMbMaxBitFlag - IntraMBMaxSizeReportMask** This is a mask bit controlling if the condition of any intra MB in the frame exceeds IntraMBMaxSize. Value | Name | Description 0h | Disable | Do not update bit0 of MFC_IMAGE_STATUS control register. 1 | Enable | set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Intra MB Conformance Max size limit.
6 [ExistsIf]Encode Only	31:28	**Reserved** Access: RO Format: MBZ
	27:16	**InterMbMaxSz** Format: U12 This field, Inter MB Conformance Max size limit, indicates the allowed max bit count size for Inter MB
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**IntraMbMaxSz** Exists If: //Intra Only Format: U12 This field, Intra MB Conformance Max size limit, indicates the allowed max bit count size for Intra MB All IPCM MBs should ignore this Max size limit.
7	31:0	**Reserved** Access: RO Format: MBZ
8 [ExistsIf]Encode Only	31:24	**SliceDeltaQpMax[3]** Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for total bit-count above FrameBitRateMax - first 1/8 region This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register whentotal bit count for the entire frame exceeds

		FrameBitRateMax but is within 1/8of FrameBitRateMaxDelta above FrameBitRateMax, i.e., in the range of(FrameBitRateMax, (FrameBitRateMax+FrameBitRateMaxDelta»3).
	23:16	**SliceDeltaQpMax[2]** Format: U8 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP forbit-count above FrameBitRateMax - above 1/8 and below 1/ 4 This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between 1/8 and ofFrameBitRateMaxDelta above FrameBitRateMax, i.e., in the range of((FrameBitRateMax+ FrameBitRateMaxDelta»3), (FrameBitRateMax+FrameBitRateMaxDelta»2).
	15:8	**SliceDeltaQpMax[1]** Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP forbit-count above FrameBitRateMax - above1/ 4 and below 1/2 This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between and of FrameBitRateMaxDeltaabove FrameBitRateMax, i.e., in the range of ((FrameBitRateMax+FrameBitRateMaxDelta»2), (FrameBitRateMax+FrameBitRateMaxDelta»1).
	7:0	**SliceDeltaQpPMax[0]** Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP forbit-count above FrameBitRateMax - above 1/ 2This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is above FrameBitRateMax by more than half the distance of FrameBitRateMaxDelta , i.e., in the range of ((FrameBitRateMax+FrameBitRateMaxDelta»1), infinite).
9 [ExistsIf]Encode Only	31:24	**SliceDeltaQpMin[3]** Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP for total bit-count below FrameBitRateMin - first 1/8 region This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is less than FrameBitRateMin and greater than or equal to 1/8 the distance of FrameBitRateMinDelta from FrameBitRateMin,i.e., in the range of [(FrameBitRateMin-FrameBitRateMinDelta»3), FrameBitRateMin).

	23:16	**SliceDeltaQpMin[2]** Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP forbit-count below FrameBitRateMin - below 1/ 8 and above 1/ 4This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between one-eighth and quarter the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of[(FrameBitRateMin- FrameBitRateMinDelta»2), (FrameBitRateMin-FrameBitRateMinDelta»3)).
	15:8	**SliceDeltaQpMin[1]** Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice level delta QP forbit-count below FrameBitRateMin- below 1/4 and above 1/ 2This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between quarter and half the distance ofFrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of[(FrameBitRateMin- FrameBitRateMinDelta»1), (FrameBitRateMin-FrameBitRateMinDelta»2)).
	7:0	**SliceDeltaQpMin[0]** Format: S7 Range: [0:MAX_QP_DELTA] This field is the Slice Level Delta QP forbit-count below FrameBitRateMin - below 1/ 2This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bitcount for the entire frame is below FrameBitRateMin by more than half the distance of FrameBitRateMinDelta , i.e., in the range of [0, (FrameBitRateMin-FrameBitRateMinDelta»1).
10 [ExistsIf]Encode Only	31	**FrameBitrateMaxUnit** This field is the Frame Bitrate Maximum Limit Units. Value 0 / Name Byte / Description: FrameBitRateMax is in units of 32 Bytes when FrameBitrateMaxUnitMode is 1 and in units of 128 Bytes if FrameBitrateMaxUnitMode is 0 Value 1 / Name Kilo Byte / Description: FrameBitRateMax is in units of 4KBytes Bytes when FrameBitrateMaxUnitMode is 1 and in units of 16KBytes if FrameBitrateMaxUnitMode is 0
	30	**FrameBitrateMaxUnitMode** This field is the Frame Bitrate Maximum Limit Units.

Value	Name	Description
0h	compatibility mode	FrameBitRateMaxUnit is in old mode (128b/16Kb)
1h	New mode	FrameBitRateMaxUnit is in new mode (32byte/4Kb)

29:16 **FrameBitRateMax**

Format:	U14

This field is the Frame Bitrate Maximum Limit. This field along with FrameBitrateMaxUnit determines maximum allowed bits in a frame before multi-pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count exceeds this value. When FrameBitrateMaxUnitMode is 0(compatibility mode) bits 16:27 should be used, bits 28 and 29 should be 0..

Value	Name	Description
[0-16383]		WhenFrameBitrateMaxUnit =0, this field is measured in unit of 32 bytes.The frame rate in bytes is range from 0-512KBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.
[0-16383]		WhenFrameBitrateMaxUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-64MBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.

15 **FrameBitrateMinUnit**

This field is the Frame Bitrate Minimum Limit Units.

Value	Name	Description
0	Byte	FrameBitRateMax is in units of 32 Bytes when FrameBitrateMinUnitMode is 1 and in units of 128 Bytes if FrameBitrateMinUnitMode is 0
1	Kilo Byte	FrameBitRateMax is in units of 4KBytes Bytes when FrameBitrateMaxUnitMode is 1 and in units of 16KBytes if FrameBitrateMaxUnitMode is 0

14 **FrameBitrateMinUnitMode**

This field is the Frame Bitrate Minimum Limit Units.

Value	Name	Description
0h	Compatibility mode	FrameBitRateMaxUnit is in old mode (128b/16Kb)
1h	New mode	FrameBitRateMaxUnit is in new mode (32byte/4Kb)

13:0 **FrameBitRateMin**

This field is the Frame Bitrate Minimum Limit. This field along with FrameBitrateMinUnit determines minimum allowed bits in a frame before multi-pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count happen to be below this value.

It takes on a value in the range of [0-16383].

When FrameBitrateMinUnitMode is 0 (compatibility mode) bits 0:11 should be used, bits 12 and 13 should be 0.

WhenFrameBitrateMinUnit=0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-512KBytes, hence the .this field is programmed from 0 to

		16,384 (14-bits) unit. WhenFrameBitrateMinUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-64MBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.
11 [ExistsIf]Encode Only	31	**Slice Stats Streamout Enable** Enable signal for Slice size streamout. When this bit is set, the slice size is written to SliceSize StreamOut Data (MFX_PIPE_BUF_ADDR_STATE DW 65,66,67)using one word. The slice size is in units of Bytes.
	30:16	**FrameBitRateMaxDelta** Format: U15 This field is used to select the slice delta QP when FrameBitRateMax Is exceeded. It shares the same FrameBitrateMaxUnit. When FrameBitrateMaxUnitMode is 0(compatibility mode), only bits 16:27 should be used, bits 28, 29 and 30 should be 0. **Value** / **Name** / **Description** [0-32767] / / WhenFrameBitrateMaxUnit =0, this field is measured in unit of 32 bytes.The frame rate in bytes is range from 0-1024KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit. [0-32767] / / WhenFrameBitrateMaxUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-128MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit. 0h / **[Default]** /
	15	**Reserved** Access: RO Format: MBZ
	14:0	**FrameBitRateMinDelta** Range: [0-32767]. This field is used to select the slice delta QPwhen FrameBitRateMin Is exceeded. It shares the same FrameBitrateMinUnit. WhenFrameBitrateMinUnitMode is 0(compatibility mode) bits 0:11 should be used, bits12, 13 and 14 should be 0.Note: HW requires the following conditionFrameBitRateMinDelta <= 2*FrameBitRateMinMust be true, otherwise it may causeunpredicted behavior. WhenFrameBitrateMinUnit =0, this field is measured in unit of 32 bytes .The frame rate in bytes is range from 0-1024KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit. WhenFrameBitrateMinUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-128MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit.
12	31:0	**Reserved** Access: RO Format: MBZ

MFX_AVC_IMG_STATE

13	31:30	**Reserved** Access: RO Format: MBZ
	29	**Current Picture Has Performed MMCO5** Set to 1 if the current Pic has performed the memory_management_control_operation = = 5.
	28:24	**Number of Reference Frames** Format: U5 Range: Range 0 to MaxDpbSize (=16 for Level 4.1) Specifies the maximum number of reference frames (frames, field pairs, unpaired field) existed in the current DBP for decoding the current picture.
	23:22	**Reserved** Access: RO Format: MBZ
	21:16	**Number of Active Reference Pictures from L1** Format: U6-1 Specifies the initial maximum reference index value minus 1 to access the L1 Reference List. It is extracted from PPS. It corresponds to the number of active reference pictures from L1 to decode the current picture. It can be modified by the slice header if num_ref_idx_active_override_flag is set. Only valid for B picture. Value: [0,31] — Name:
	15:14	**Reserved** Access: RO Format: MBZ
	13:8	**Number of Active Reference Pictures from L0** Format: U6-1 Specifies the initial maximum reference index value minus 1 to access the L0 Reference List. It is extracted from PPS. It corresponds to the number of active reference pictures from L0 to decode the current picture. It can be modified by the slice header if num_ref_idx_active_override_flag is set. Valid for both P and B pictures. Value: [0,31] — Name:
	7:0	**Initial QP Value** Format: S7 Range: [-26,25] Initial QP value for a Slice, extracted from PPS. It may further get modified by

		slice_qp_delta in slice header and mb_qp_delta in MB header.
14 [ExistsIf] Short Format only	31:24	**Log2_max_pic_order_cnt_lsb_minus4** Exists If: //Short Format Only It is a SPS syntax element, used to determine how many bits in the bitstream are used to represent pic_order_cnt_lsb syntax element in the slice header. Unsigned
	23:16	**Log2_max_frame_num_minus4** Exists If: //Short Format Only It is a SPS syntax element, used to determine how many bits in the bitstream are used to represent frame_num syntax element in the slice header. Unsigned.
	15	**deblocking_filter_control_present_flag** Exists If: //Short Format Only It is a PPS syntax element, indicates if more deblocking filter control syntax elements are present in the slice header.
	14:12	**num_slice_groups_minus1** Exists If: //Short Format Only BitField It is a PPS syntax element. Use for Slice Header parsing only, to read in slice_group_change_cycle, if any, but is not used by H/W, i.e. no slice group support.Desc
	11	**redundant_pic_cnt_present_flag** Exists If: //Short Format Only It is a PPS syntax element.Use for Slice Header parsing only, to read-in redundant_pic_cnt, if any, but is not used by H/W, i.e. no support for redundant slice processing.
	10:8	**slice_group_map_type** Exists If: //Short Format Only It is a PPS syntax element. Use for Slice Header parsing only, to read in slice_group_change_cycle, if any, but is not used by H/W, i.e. no slice group support.
	7:4	**Reserved** Access: RO Format: MBZ
	3:2	**Pic_order_cnt_type** Exists If: //Short Format Only It is a SPS syntax element. Use for Slice Header parsing only.

	1	**Delta_pic_order_always_zero_flag** Exists If: //Short Format Only It is a SPS syntax element. Use for Slice Header parsing only.
	0	**Pic_order_present_flag** Exists If: //Short Format Only It is a PPS syntax element. Use for Slice Header parsing only.
15 [ExistsIf] Short Format only	31:16	**Curr Pic Frame Num** Exists If: //Short Format Only Format: U16 Derived from Slice Header syntax element
	15:0	**Slice Group Change Rate** Exists If: //Short Format Only Format: U16-1 It is a PPS syntax element Use for Slice Header parsing only, to read in slice_group_change_cycle, if any, but is not used by H/W, i.e. no slice group support.
16 [ExistsIf]: Short Format only	31	**Inter View Order Disable** Exists If: //Short Format Only It indicates how to append inter-view picture into initial sorted reference list. (due to ambiguity in the MVC Spec) Value / Name / Description: 0h – Default **[Default]** – View Order Ascending 1h – Disable – View ID Ascending
	30:22	**Reserved** Access: RO Format: MBZ
	21:18	**Max View IDXL1** Exists If: //Short Format Only It is a PPS syntax element corresponding to Anchor/Non-Anchor Reference ListL1It indicates the maximum number of inter-view picture for Reference List L1
	17:16	**Reserved** Access: RO Format: MBZ
	15:12	**Max View IDXL0** Exists If: //Short Format Only

		Reference ListL0It indicates the maximum number of inter-view picture for Reference List L0
	11:10	**Reserved** Access: RO Format: MBZ
	9:0	**Current Frame View ID** Exists If: //Short Format Only It indicates the View ID of the current decoding frame
17	31:22	**Reserved** Access: RO Format: MBZ
	21:16	**RhoDomain AverageMacroblockQP** Exists If: //[RhoDomain Rate Control] == 1 Format: U6
	15:9	**Reserved** Access: RO Format: MBZ
	8	**Extended RhoDomain Statistics Enable** This parameter enables PAK to generate RhoDomain statistics from marcoblock QP and fractional QP computation. **Value** / **Name** / **Description** 0 / Disable / RhoDomain statistics generated from Frame Qp and no fractional QP computation. 1 / Enable / Enable MB QP based RhoDomain statistics and fractional QP computation. Note: By default, this is always enabled. Only the enable case will be validated.
	7:6	**Reserved** Access: RO Format: MBZ
	5:3	**Fractional QP offset** Format: U3 Start position from the top of the Frame where increased Quantization parameter is added.
	2:0	**Fractional QP input** In a set of 8 rows, Qp is incremented by 1 for F_qp rows.

MFX_AVC_IMG_STATE		
18	31:0	**Reserved** Access: RO Format: MBZ
19	31:0	**Threshold Size in Bytes** Format: U32 When a slice exceeds this value in bytes, hardware will end current slice as soon as possible and insert a new slice boundary. Note there is no guarantee that the actual slice size will meet this value, it is a hint to the HW to end the slice as soon as possible (which could be 2-5 macroblocks in the future from this detection point).
20	31:0	**Target Slice Size in Bytes** Format: U32 In VDENC mode, this field indicates the target slice size in Bytes for slice size conformance feature. When the actual slice size exceeds "Target slice size in Bytes", HW sets "VDENC Slice Overflow Error" bit in MFC_IMAGE_STATUS_CONTROL register**.**

MFX_AVC_REF_IDX_STATE

MFX_AVC_REF_IDX_STATE

Source: VideoCS

Length Bias: 2

This is a slice level command and can be issued multiple times within a picture that is comprised of multiple slices. The same command is used for AVC encoder (PAK mode) and decoder (VLD mode); it is not need in decoder IT mode.
The inline data of this command is interpreted differently for encoder as for decoder. For decoder, it is interpreted as RefIdx List L0/L1 as in AVC spec., and it matches with the AVC API data structure for decoder in VLD mode : RefPicList[2][32] (L0:L1, 0:31 RefPic). But for encoder, it is interpreted as a Reference Index Mapping Table for L0 and L1 reference pictures. For packing the bits at the output of PAK, the syntax elements must follow the definition of RefIdxL0/L1 list according to the AVC spec. However, the decoder pipeline was designed to use a variation of that standard definition, as such a conversion (mapping) is needed to support the hardware design. The Reference lists are needed in processing both P and B slice in AVC codec. For P-MB, only L0 list is used; for B-MB both L0 and L1 lists are needed. For a B-MB that is coded in L1-only Prediction, only L1 list is used.

Programming Notes

An application will create the RefPicList L0 and L1 and pass onto the driver. The content of each entry of RefPicList L0/L1[ ] is a 7-bit picture index. This picture index is the same as that of RefFrameList[ ] content. This picture index, however, is not defined the same as the frame store ID (0 to 16, 5-bits) we have implemented in H/W. Hence, driver is required to manage a table to convert between picture index and intel frame store ID. As such, the final RefPicList L0/L1[ ] that the driver passes onto the H/W is not the same as that defined.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_AVC_REF_IDX_STATE Format: OpCode
	26:24	**Command Opcode** Default Value: 1h AVC Format: OpCode
	23:21	**SubOpcodeA** Default Value: 0h MFX_AVC_REF_IDX_STATE Format: OpCode
	20:16	**SubOpcodeB** Default Value: 4h MFX_AVC_REF_IDX_STATE Format: OpCode

MFX_AVC_REF_IDX_STATE

	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 0008h Format: =n Excludes DWords 0,1
1	31:1	**Reserved** Access: RO Format: MBZ
	0	**RefPicList Select** Num_ref_idx_l1_active is resulted from the specifications in both PPS and Slice Header for the current slice. However, since the full reference list L0 and/or L1 are always sent, only present flags are specified instead. This parameter is specified for Intel interface only. Value: 0 — Name: RefPicList 0 — Description: The list that followed represents RefList L0 (Decoder VLD mode) or Ref Idx Mapping Table L0 (Encoder PAK mode) Value: 1 — Name: RefPicList1 — Description: The list that followed represents RefList L1 (Decoder VLD mode) or Ref Idx Mapping Table L1 (Encoder PAK mode)
2..9 **Programming Notes:** In VDENC mode, we support only three forward reference pictures. HW supports only 1:1 reference index to reference picture mapping. Reference index 0 should point to Reference picture 0, whose address is specified inMFX_PIPE_BUF_ADDR DW 19,20 (The reference picture numbers may be different from bit-stream reference picture) Reference index1 should point to Reference picture2, whose address is specified inMFX_PIPE_BUF_ADDR 23, 24 (The reference picture numbers may be different from bit-stream reference picture) Reference index2 should point to	255:0	**Reference List Entry** This set of fields is always present whenever this command is issued. It always specifies the full 32 reference pictures in the selected list, regardless they are "existing picture" or not. If a picture is non-existing, the corresponding entry should be set to all ones. Each list entry is 1 byte. A 32-bit DW can hold 4 list entries in the following format • 31:24 entry X+3 (e.g. listY_3) • 23:16 entry X+2 (e.g. listY_2) • 15:8 entry X+1 (e.g. listY_1) • 7:0 entry X (e.g. listY_0) X is replaced by the paddr[2:0] * 4 ; paddr[5:0] with 0x20 and 0x27, and Y is replaced by 0 or 1.The byte definition for a reference picture : • Bit 7 : Non-Existing - indicates that frame store index that should have been at this entry did not exist and was

MFX_AVC_REF_IDX_STATE		
Reference picture4, whose address is specified inMFX_PIPE_BUF_ADDR 27,28 (The reference picture numbers may be different from bit-stream reference picture)		replaced by an index 0 (a valid entry) for error concealment • Bit 6 : Long term bit - set this reference picture to be used as long term reference • Bit 5 : Field picture flag - indicates frame/field • Bit 4:0 : Frame store index or Frame Store ID (Bit 4:1 is used to form the binding table index in intel implementation) This is the final Reference List L0 or L1 after any reordering specified in the Slice Header as well as modified by the driver, and its indices values are all translated to the intel specification. If the reference picture is a frame (Bit5 = 1), frame store ID is always an even number. This list is used in outputting MV information by the BSD unit in VLD mode. DMV access also reads and writes Mvlist0 using this frame store ID. If this set of fields is interpreted as Reference Index Mapping Table L0/L1, the same field alignment is followed, i.e. 4 mapping entries per DW. Each mapping entry is one byte in size, but only the least significant 5 bits [4:0] is relevant. Driver should zero all the upper bits [7:5] for each entry.

MFX_AVC_SLICE_STATE

MFX_AVC_SLICE_STATE

Source: VideoCS

Length Bias: 2

This is a slice level command and can be issued multiple times within a picture that is comprised of multiple slices. The same command is used for AVC encoder (PAK mode) and decoder (VLD and IT modes).

In VDEnc mode, this command is programmed for every super-slice. However not all parameters are allowed to change across super-slices.

Programming Notes

MFX_AVC_SLICE_STATE command is not issued for AVC Short Format Bitstream decode, instead MFD_AVC_SLICEADDR command is executed to retrieve the next slice MB Start Address X and Y by H/W itself.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_AVC_SLICE_STATE Format: OpCode
	26:24	**Command Opcode** Default Value: 1h AVC Format: OpCode
	23:21	**SubOpcodeA** Default Value: 0h MFX_AVC_SLICE_STATE Format: OpCode
	20:16	**Command SubOpcodeB** Default Value: 3h MFX_AVC_SLICE_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 8h DWORD_COUNT_n Format: =n Excludes DWords 0,1

MFX_AVC_SLICE_STATE

1 — 31:4 — Reserved

Access:	RO
Format:	MBZ

1 — 3:0 — Slice Type

It is set to the value of the syntax element read from the Slice Header.

Value	Name
0000b	P Slice
0001b	B Slice
0010b	I Slice
0011b-1111b	Reserved

Programming Notes
Bits[3:2] must be 0

2 — 31:30 — Reserved

Access:	RO
Format:	MBZ

2 — 29:24 — Number of Reference Pictures in Inter-prediction List 1

Format:	U6

This field is valid only for encoding a B Slice, for which it is expected to have at least one entry in the reference list L1; otherwise (if Slice Type is not a B Slice), this field must be set to 0.This field can be derived for a B Slice from the Slice Header syntax element NumRefIdxActiveMinus1 as, Num_Ref_Idx_L1 = NumRefIdxActiveMinus1[1] + 1.

Value	Name
0-32	

2 — 23:22 — Reserved

Access:	RO
Format:	MBZ

2 — 21:16 — Number of Reference Pictures in Inter-prediction List 0

Format:	U6

This field is valid for encoding a P or B Slice, for which it is expected to have at least one entry in the reference list L0; otherwise (if Slice Type is not a P or B Slice), this field must be set to 0.This field can be derived for a P or B Slice from the Slice Header syntax element NumRefIdxActiveMinus1 as, Num_Ref_Idx_L0 = NumRefIdxActiveMinus1[0] + 1.

Value	Name
0-32	

2 — 15:11 — Reserved

Access:	RO
Format:	MBZ

MFX_AVC_SLICE_STATE

	10:8	**Log 2 Weight Denom Chroma** Format: U3 Value: 0-7
	7:3	**Reserved** Access: RO Format: MBZ
	2:0	**Log 2 Weight Denom Luma** Format: U3 It is the base 2 logarithm of the denominator for all Luma weighting factors. It is set to the value of the syntax element read from the Slice Header Pred_Weight_Table(). In VDEnc Mode, this value is programmed per super slice. Value: 0-7
3	31:30	**Weighted Prediction Indicator** This field indicates the Weighted Prediction mode for a P or B Slice. It is a combined field corresponding to the syntax element WeightedBiPredIdc or WeightedPredFlag read from the current active PPS. • If it is a B-Slice, these bits are interpreted as: 00b - Specifies the default weighted inter-prediction to be applied01b - Specifies the explicit weighted inter-prediction to be applied10b - Specifies the implicit weighted inter-prediction to be applied11b - Reserved (not allowed) • If it is a P Slice, these bits are interpreted as: 00b - Disables weighted inter-prediction (Default weighted)01b - Enables weighted inter-prediction (Explicit weighted)10b - 11b - Reserved **Programming Notes** Only when in B Slice with Weighted_Pred_Idc = 1 (explicit weighted prediction), will there be a L1 and/or a L0 weight+offset tables being sent to the BSD unit through the Slice_State command. Only when in P Slice with Weighted_Pred_Idc = 1, will there be a L0 weight+offset table being sent to the BSD. If Weighted_Pred_Idc != 1 for B Slice or Weighted_Pred_Idc =0 for P Slice, no Slice_State command should be issued to send these tables. If still being issued, the data is read but ignored. DXVA specifies Weighted_Bipred and Weighted_Pred in frame-level state. However, these two flags are combined and specified in slice level for both P and B slice type.
	29	**Direct Prediction Type** Type of direct prediction used for B Slices. This field is valid only for Slice_Type = B Slice;

otherwise, it must be set to 0.

Value	Name
0	Temporal
1	Spatial

28:27 **Disable Deblocking Filter Indicator**

Value	Name	Description
00b		FilterInternalEdgesFlag is set equal to 1
01b		Disable all deblocking operation, no deblocking parameter syntax element is read; filterInternalEdgesFlag is set equal to 0
10b		Macroblocks in different slices are considered not available; filterInternalEdgesFlag is set equal to 1
11b	Reserved	Not defined in AVC

26 **Reserved**

Access:	RO
Format:	MBZ

25:24 **Cabac Init Idc[1:0]**

Specifies the index for determining the initialization table used in the context variable initialization process.

Value	Name
0-2	

Programming Notes
Cabac initialization is also dependent on the field/frame picture type, Slice type, and the current SliceQP value.

23:22 **Reserved**

Access:	RO
Format:	MBZ

21:16 **Slice Quantization Parameter**

Quantization Parameter for current slice. Derived from PPS and slice_delta_qp syntax element in Slice Header. It is needed for CABAC context initialization and deblocking filter control. And it is also used as the starting QP value in the very first MB of a slice. It is in the range of unsigned integer 0 to 51, for 8-bit pixel bit-depth.

15:12 **Reserved**

Access:	RO
Format:	MBZ

11:8 **Slice Beta Offset Div2**

Format:	S3

Range: [-6, 6] Inclusive

		Specifies the offset used in accessing the deblocking filter strength tables.
	7:4	**Reserved** Access: RO Format: MBZ
	3:0	**Slice Alpha C0 Offset Div2** Format: S3 Range: [-6, 6] Inclusive Specifies the offset used in accessing the deblocking filter strength tables.
4	31:24	**Slice Vertical Position** This field specifies the position in y-direction of the first macroblock in the Slice in unit of macroblocks. The fields (Slice_MB_Start_Hor_Pos, Slice_MB_Start_Vert_Pos) are valid in VLD (decoding) mode only. They are ignored by hardware in decoding IT mode and encoding mode (whereas the position is provided by the per-macroblock object command).Derived **Programming Notes** Error Handling: Driver needs to check if FirstMbY starts at 0 on the first slice of frame. If not, driver needs to add a phantom slice with FirstMbX and FirstMbY set to 0.
	23:16	**Slice Horizontal Position** This field specifies the position in x-direction of the first macroblock in the Slice in unit of macroblocks. Derived **Programming Notes** Error Handling: Driver needs to check if FirstMbY starts at 0 on the first slice of frame. If not, driver needs to add a phantom slice with FirstMbX and FirstMbY set to 0.
	15	**Reserved** Access: RO Format: MBZ
	14:0	**Slice Start Mb Num** Exists If: //Decoder Only The MB number (linear MB address in a picture) at the start of a Slice, it must match with the Slice Horizontal Position (Slice_MB_Start_Hor_Pos) and Vertical Position (Slice_MB_Start_Vert_Pos) in the picture. **Programming Notes** In creating the Phantom Slice for error concealment, this field should set to the total number of MB in the current picture + 1.
5	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**Next Slice Vertical Position** This field specifies the position in y-direction of the first macroblock in the next Slice in unit of

macroblocks. This field is primarily used for error concealment. In the case that current slice is the last slice, this field should set to the height of picture (since y-direction is zero-based numbering).

15:8 Reserved

Access:	RO
Format:	MBZ

7:0 Next Slice Horizontal Position

This field specifies the position in x-direction of the first macroblock in the next Slice in unit of macroblocks. This field is primarily used for error concealment. In the case that current slice is the last slice, this field should set to 0.

6 Encoder Only

31 Rate Control Counter Enable

To enable the accumulation of bit allocation for rate control This field enables hardware Rate Control logic. The rest of the RC control fields are only valid when this field is set to 1. Otherwise, hardware ignores these fields.

Value	Name
0	Disable
1	Enable

30 ResetRateControlCounter

To reset the bit allocation accumulation counter to 0 to restart the rate control.

Value	Name
0	Not Reset
1	Reset

29:28 RC Triggle Mode

Value	Name	Description
00b	Always Rate Control	Whereas RC becomes active if sum_act > sum_target or sum_act < sum_target
01b	Gentle Rate Control	whereas RC becomes active if sum_act > upper_midpt or sum_act < lower_midpt
10b	Loose Rate Control	whereas RC becomes active if sum_act > sum_max or sum_act < sum_min
11b	Reserved	

27:24 RC Stable Tolerance

Format:	U4

This field specifies the tolerance required to deactivate RC once it has been triggered.

Value	Name
0-15	

23 RC Panic Enable

If this field is set to 1, RC enters panic mode when sum_act > sum_max. RC Panic Type field controls what type of panic behavior is invoked.

Value	Name
0	Disable
1	Enable

22 **RC Panic Type**

This field selects between two RC Panic methods

Value	Name
0	QP Panic
1	CBP Panic

Programming Notes
If it is set to 0, in panic mode, the macroblock QP is maxed out, setting to requested QP + QP_max_pos_mod. If it is set to 1, for an intra macroblock, AC CBPs are set to zero (note that DC CBPs are not modified). For inter macroblocks, AC and DC CBPs are forced to zero.

21 **MB Type Direct Conversion Disable**

Exists If:	//B-Slice

For all Macroblock type conversions in different slices, refer to Section "Macroblock Type Conversion Rules" in the same volume.

Value	Name
0	Enable direct mode conversion
1	Disable direct mode conversion

Programming Notes
This field is zero for all other slices other than B-Slice.

20 **MB Type Skip Conversion Disable**

Exists If:	//P-Slice or B-Slice

For all Macroblock type conversions in different slices, refer to Section "Macroblock Type Conversion Rules" in the same volume.

Value	Name
0	Enable skip type conversion
1	Disable skip type conversion

Programming Notes
This field is zero for all other slices other than P_Slice or B-Slice. \

19 **Is Last Slice**

It is used by the zero filling in the Minimum Frame Size test.
Set this only for the last slice group

Value	Name	Description
1		Current slice is the last slice of a picture

	0		Current slice is NOT the last slice of a picture

18 **Reserved**

17 **Header Insertion Present in Bitstream**

Value	Name	Description
0		No header insertion into the output bitstream buffer, in front of the current slice encoded bits.
1		Header insertion into the output bitstream buffer is present, and is in front of the current slice encoded bits.

Programming Notes
Note: In VDEnc mode, the slice header PAK object maximum size is 25 DWs.
This need to be set only for super slice0.

16 **SliceData Insertion Present in Bitstream**

Value	Name	Description
0		No Slice Data insertion into the output bitstream buffer
1		Slice Data insertion into the output bitstream buffer is present.

Programming Notes
This bit should be set for all super-slices.

15 **Tail Insertion Present in bitstream**

Value	Name	Description
0		No tail insertion into the output bitstream buffer, after the current slice encoded bits
1		Tail insertion into the output bitstream buffer is present, and is after the current slice encoded bits.

Programming Notes
This bit should only be set for the last super slice. In VDENC mode, SW should insert 1000 **VD_PIPELINE_FLUSH commands with VDENC_pipeline_Done set to 1before inserting tail command. This is for delaying the tail insertion in HW. The HW recommendation is to insert tail only at the end of sequence to avoid performance loss since this restriction potentially cause performance degradation.**

14 **Reserved**

Access:	RO
Format:	MBZ

13 **EmulationByteSliceInsertEnable**

To have PAK outputting SODB or EBSP to the output bitstream buffer

Value	Name	Description

0		outputting RBSP
1		outputting EBSP

12 CabacZeroWordInsertionEnable

To pad the end of a SliceLayer RBSP to meet the encoded size requirement.

Value	Name	Description
0		No Cabac_Zero_Word Insertion
1		Allow internal Cabac_Zero_Word generation and append to the end of RBSP(effectively can be used as an indicator for last slice of a picture, if the assumption is only the last slice of a picture needs to insert CABAC_ZERO_WORDs.

11:8 Reserved

Access:	RO
Format:	MBZ

7:4 Slice ID [3:0]

To identify the output data (coding information record) returned for rate control from PAK to ENC and VPP.

3:2 Reserved

Access:	RO
Format:	MBZ

1:0 Stream ID [1:0]

To identify the output data (coding information record) returned for rate control from PAK to ENC and VPP.

7 Encoder Only — 31:29 Reserved

Access:	RO
Format:	MBZ

28:0 Indirect PAK-BSE Data Start Address (Write)

Exists If:	//AVC Encode Mode

This field specifies the memory starting address (offset) to write out the compressed bitstream data from the BSE processing. This pointer is relative to the MFC Indirect PAK-BSE Object Base Address. It is a byte-aligned address for the AVC bitstream data in both CABAC/CAVLC Modes. For Write, there is no need to have a data length field. It is assumed the global memory bound check specified in the IND_OBJ_BASE_ADDRESS command (Indirect PAK-BSE Object Access Upper Bound) will take care of any illegal write access.

Value	Name
[0h,1FFFFFFFh]	

8 Encoder Only — 31:24 Magnitude of QP Max Negative Modifier

Format:	U8

This field specifies the lower limit of the QP modifier.

Value	Name

		0-51
	23:16	**Magnitude of QP Max Positive Modifier** Format: U8 This field specifies the upper limit of the QP modifier. Value: 0 - 15
	15:12	**Shrink Param - Shrink Resistance** Format: U4 This field specifies the additional points added each time decreased correction is invoked. Value: 0 - 15
	11:8	**Shrink Param - Shrink Init** Format: U4 This field specifies the initial points required to trip decreased control. Value: 0 - 15
	7:4	**Grow Param - Grow Resistance** Format: U4 This field specifies the additional points added each time increased correction is invoked. Value: 0 - 15
	3:0	**Grow Param - Grow Init** Format: U4 This field specifies the initial points required to trip increased control. Value: 0 - 15
9 Encoder Only	31	**RoundInterEnable** Format: Enable When this bit is not set, RoundInter defaults to 2.
	30:28	**RoundInter** Format: U3 Rounding precision for Inter quantized coefficients 000b: +1/16 **[Default]** 001b: +2/16 010b: +3/16

011b	+4/16
100b	+5/16
101b	+6/16
110b	+7/16
111b	+8/16

27 **RoundIntraEnable**

Format:	Enable

When this bit is not set, RoundIntra defaults to 4.

26:24 **RoundIntra**

Format:	U3

Rounding precision for Intra quantized coefficients

Value	Name
000b	+1/16 **[Default]**
001b	+2/16
010b	+3/16
011b	+4/16
100b	+5/16
101b	+6/16
110b	+7/16
111b	+8/16

23:20 **Correct 6**

Format:	U4

This field specifies the points used in the lowermost RC region when sum_act <= sum_min.

Value	Name
0 - 15	

19:16 **Correct 5**

Format:	U4

This field specifies the points used in the fifth RC region when sum_act > sum_min but <= lower_midpt.

Value	Name
0 - 15	

15:12 **Correct 4**

Format:	U4

This field specifies the points used in the fourth RC region when sum_act > lower_midpt but <= sum_target.

Value	Name
0 - 15	

	11:8	**Correct 3** Format: U4 This field specifies the points used in the third RC region when sum_act > sum_target but <= upper_midpt. Value: 0 - 15
	7:4	**Correct 2** Format: U4 This field specifies the points used in the second RC region when sum_act > upper_midpt but <= sum_max. Value: 0 - 15
	3:0	**Correct 1** Format: U4 This field specifies the points used in the topmost RC region when sum_act > sum_max. Value: 0 - 15
10 Encoder Only	31:28	**ClampValues - CV7**
	27:24	**CV6**
	23:20	**CV5**
	19:16	**CV4**
	15:12	**CV3**
	11:8	**CV2**
	7:4	**CV1**
	3:0	**CV0 - Clamp Value 0** Format: U4 If the magnitude of coefficients at locations assigned with CV0 (mapping shown below) exceeds 2CV0-1, they are replaced with 2CV0-1. For coefficients at locations marked as 'none', no clamping is performed. The following mappings are only applied to luma and chroma blocks\subblocks containing AC coefficiencts (blocks\sublocks with only DC coeffs will not be clamped). **For 4x4 frame block, each coefficient is mapped to one of the eight CV values as following:** none CV7 CV5 CV4 CV7 CV6 CV4 CV3 CV5 CV4 CV2 CV1 CV4 CV3 CV1 CV0 **For 8x8 frame block, each coefficient is mapped to one of the eight CV values as following:**

none	none	CV7	CV6	CV5	CV4	CV3	CV3
none	CV7	CV6	CV5	CV4	CV3	CV3	CV2
CV7	CV6	CV5	CV4	CV3	CV3	CV2	CV2
CV6	CV5	CV4	CV3	CV3	CV2	CV2	CV1
CV5	CV4	CV3	CV3	CV2	CV2	CV1	CV1
CV4	CV3	CV3	CV2	CV2	CV1	CV1	CV0
CV3	CV3	CV2	CV2	CV1	CV1	CV0	CV0
CV3	CV2	CV2	CV1	CV1	CV0	CV0	CV0

For 4x4 field block, each coefficient is mapped to one of the eight CV values as following:

none	CV6	CV3	CV1
CV7	CV6	CV3	CV1
CV5	CV4	CV2	CV0
CV5	CV4	CV2	CV0

For 8x8 field block, each coefficient is mapped to one of the eight CV values as following:

none	none	CV6	CV5	CV4	CV3	CV2	CV1
none	CV7	CV6	CV5	CV4	CV3	CV2	CV1
CV7	CV6	CV5	CV4	CV3	CV3	CV2	CV1
CV7	CV6	CV5	CV4	CV3	CV2	CV2	CV1
CV6	CV5	CV4	CV4	CV3	CV2	CV1	CV0
CV6	CV5	CV4	CV3	CV2	CV2	CV1	CV0
CV5	CV5	CV4	CV3	CV2	CV1	CV1	CV0
CV5	CV5	CV4	CV3	CV2	CV1	CV1	CV0

Value	Name
0 - 15	

MFX_AVC_WEIGHTOFFSET_STATE

MFX_AVC_WEIGHTOFFSET_STATE

Source: VideoCS

Length Bias: 2

This is a slice level command and can be issued multiple times within a picture that is comprised of multiple slices. The same command is used for AVC encoder (PAK mode) and decoder (VLD and IT modes). However, since for AVC decoder VLD and IT modes, and AVC encoder mode, the implicit weights are computed in hardware, this command is not issued. For encoder, regardless of the type of weight calculation is active for the current slice (default, implicit or explicit), they are all sent to the PAK as if they were all in explicit mode. However, for implicit weight and offset, each entry contains only a 16-bit weight and no offset (offset = 0 always in implicit mode and can be hard-coded inside the hardware).The weights (and offsets) are needed in processing both P and B slice in AVC codec. For P-MB, at most only L0 list is used; for B-MB both L0 and L1 lists may be needed. For a B-MB that is coded in L1-only Prediction, only L1 list is sent. The content of this command matches with the AVC API data structure for explicit prediction mode only : Weights[2][32][3][2] (L0:L1, 0:31 RefPic, Y:Cb:Cr, W:0)

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_ AVC_ WEIGHTOFFSET_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 1h AVC_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 5h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 60h Excludes DWord (0,1) Format: =n
1	31:1	**Reserved**

MFX_AVC_WEIGHTOFFSET_STATE

		Access: RO
		Format: MBZ
	0	**Weight and Offset Select** It must be set in consistent with the WeightedPredFlag and WeightedBiPredIdc in the Img_State command.This parameter is specified for Intel interface only. For implicit even though only one entry may be used, still loading the whole 32-entry table.

Value	Name	Description
0	Weight and Offset L0 table	The list that followed is associated with the weight and offset for RefPicList L0
1	Weight and Offset L1 table	The list that followed is associated with the weight and offset for RefPicList L1

2..97	3071:0	**WeightOffset** WeightOffset[L=L0=0 or L1=1][i=0 to 31][Y=0/Cb=1/Cr=2][weight=0/offset=1]WeightOffset[L][i=0][Y=0][Weight=0], WeightOffset[L][i=0][Y=0][Offset=1]WeightOffset[L][i=0][Cb=1][Weight=0], WeightOffset[L][i=0][Cb=1][Offset=1]WeightOffset[L][i=0][Cr=2][Weight=0], WeightOffset[L][i=0][Cr=2][Offset=1]:WeightOffset[L][i=31][Y=0][Weight=0], WeightOffset[L][i=31][Y=0][Offset=1]WeightOffset[L][i=31][Cb=1][Weight=0], WeightOffset[L][i=31][Cb=1][Offset=1]WeightOffset[L][i=31][Cr=2][Weight=0], WeightOffset[L][i=31][Cr=2][Offset=1] Format for explicit: Both Weight and Offset are S15 in two's compliment, with a valid range from -128 to 128Format for implicit: S15 This set of fields is always present whenever this command is issued. The full table, one entry for each reference picture, is always specified. Any reference list L0/L1[i] that does not exist, the corresponding weight and offset are set to 0.Weight and Offset are 2 byte each. Apair of Weight and Offset forms a dword, with Weight in the LOWER word and Offset in the HIGHER word.WeightOffset[L0=0][i=0 to 31][Y=0] (i.e. luma_weight_l0[i]) are specified for the weighting and offset factors applied to the luma prediction value for list 0 prediction using RefPicList0[i] (one-to-one correspondence in i). When luma_weight_l0_flag (Slice Header syntax element) is equal to 1, the value of luma_weight_l0[i] shall be in the range of -128 to 127. When luma_weight_l0_flag is equal to 0, luma_weight_l0[i] shall be inferred to be equal to 2luma_log2_weight_denom for RefPicList0[i]. luma_log2_weight_denom is a Slice Header syntax element.WeightOffset[L0=0][i=0 to 31][Cb=1] (i.e. chromaCb_weight_l0[i]) are specified for the weighting and offset factors applied to the chroma Cb prediction values for list 0 prediction using RefPicList0[i] (one-to-one correspondence in i). When chroma_weight_l0_flag (Slice Header syntax element) is equal to 1, the value of chromaCb_weight_l0[i] shall be in the range of -128 to 127. When chroma_weight_l0_flag is equal to 0, chromaCb_weight_l0[i] shall be inferred to be equal to 2chroma_log2_weight_denom for RefPicList0[i]. chroma_log2_weight_denom is a Slice Header syntax element.WeightOffset[L0=0][i=0 to 31][Cr=2] (i.e. chromaCr_weight_l0[i]) are specified for the weighting and offset factors applied to the chroma Cr prediction values for list 0 prediction using RefPicList0[i] (one-to-one correspondence in i). When chroma_weight_l0_flag (Slice Header syntax element) is equal to 1, the value of chromaCr_weight_l0[i] shall be in the range of -128 to 127. When chroma_weight_l0_flag is

MFX_AVC_WEIGHTOFFSET_STATE		
		equal to 0, chromaCr_weight_l0[i] shall be inferred to be equal to 2chroma_log2_weight_denom for RefPicList0[i].

MFX_BSP_BUF_BASE_ADDR_STATE

MFX_BSP_BUF_BASE_ADDR_STATE

Source: VideoCS

Length Bias: 2

This frame-level state command is used to specify all the buffer base addresses needed for the operation of the AVC Bit Stream Processing Units (for decoder, it is BSD Unit; for encoder, it is BSE Unit)For both encoder and decoder, currently it is assumed that all codec standards can share the same BSP_BUF_BASE_STATE. The simplicity of this command is the result of moving all the direct MV related processing into the ENC Subsystem. Since all implicit weight calculations and direct MV calculations are done in ENC and all picture buffer management are done in the Host, there is no need to provide POC (POC List - FieldOrderCntList, CurrPic POC - CurrFieldOrderCnt) information to PAK. For decoder, all the direct mode information are sent in a separate slice-level command (AVC_DIRECTMODE_STATE command).In addition, in Encoder, the row stores for CABAC encoding and MB Parameters Construction (MPC) are combined into one single row store. The row stores specified in this command do not combine with those specified in the MFC_PIPE_BUF_ADDR_STATE command for hardware simplification reason.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Pipeline Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MFX_COMMON_STATE Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 4h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 8h Excludes DWord (0,1) Format: =n

MFX_BSP_BUF_BASE_ADDR_STATE

1	31:6	**BSD/MPC Row Store Scratch Buffer Base Address - Read/Write** This field provides the base address of the scratch buffer used by BSD (decoder) and MPC (encoder) unit to store MB information of the previous row for coding each macroblockin the current row. It is a private buffer used by the BSD (decoder) and MPC (encoder)hardware only. Its content is not accessible by software. This Row Store buffer must be64-byte cacheline aligned. Hardware uses the horizontal address of the current macroblockto address this Row Store. For AVC BSD, 2 cacheline (CL) per MB when in MBAFF mode (row of MB pair); 1 CL per MBfor non-MBAFF. So, to support 256 MBs per row (4K screen resolution), 2 * 256 * 64 bytes= 32,768 bytes are required. Cacheline alignment should be followed. For AVC MPC, 1cachline for non-MBAFF, 2 cachelines for MBAFF per MB. For VC1, the BSD row store is512-bit (one cacheline) per MB, times the number of MBs per picture MB row. **Programming Notes** This is one of the four RowStore Scratch Buffers which can programmed to use the internal Media Storage (total size 640 CacheLine). When Deblocking Filter Row Store Scratch Buffer Cache Select is programmed to "1", this will be stored inside MFX Media Internal Storage. Driver then needs to program this Base Address between 0 to 639, indicating starting cacheline address location for this buffer. Driver needs to make sure the whole buffer fits into Media Internal Storage. *(Notes: 1 cachelines per MB for non-mbaff; 2 cachelines per MB pair for mbaff, and the buffer needs to have enough space for 1 MB (pair) row).*
	5:0	**Reserved** Access: RO Format: MBZ
2	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**BSD/MPC Row Store Scratch Buffer Base Address - Read/Write [47:32]** This field is for the upper range of BSD/MPC Row Store Scratch Buffer Base Address. This field is used for 48-bit addressing.
3	31:15	**Reserved** Access: RO Format: MBZ
	14:13	**BSD/MPC Row Store Scratch Buffer - Tiled Resource Mode** Format: U2 **For Media Surfaces:**This field specifies the tiled resource mode. Value / Name / Description 0h / TRMODE_NONE / No tiled resource 1h / TRMODE_TILEYF / 4KB tiled resources

2h	TRMODE_TILEYS	64KB tiled resources
3h	Reserved	

12 — BSD/MPC Row Store Scratch Buffer Cache Select

This field controls if Intra Row Store is going to store inside Media Internal Storage or to LLC.

Value	Name	Description
0		Buffer going to LLC
1		Buffer going to Internal Media Storage

11:9 — Reserved

Access:	RO
Format:	MBZ

8:7 — BSD/MPC Row Store Scratch Buffer - Arbitration Priority Control

Format:	U2

This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.

Value	Name
00b	Highest priority
01b	Second highest priority
10b	Third highest priority
11b	Lowest priority

6:1 — BSD/MPC Row Store Scratch Buffer - Index to Memory Object Control State (MOCS) Tables

Format:	U6

The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.

0 — Reserved

4 / 31:6 — MPR Row Store Scratch Buffer Base Address - Read/Write (Decoder Only)

This field provides the base address of the scratch buffer used by decoder's MPR unit to store MB information of the previous row for decoding each macroblock in the current row. It is a private buffer used by the MPR hardware only. Its content is not accessible by software.

Programming Notes
The MPR Row Store buffer must be 64-byte cacheline aligned. Hardware uses the horizontal address of each macroblock to address the MPR Row Store. Except ILDB Control Data, all other operations does not cross slice boundary. This field is specified in frame-level.2 cacheline (CL) per MB when in MBAFF mode (row of MB pair); 1 CL per MB for non-MBAFF, So, to support 256 MBs per row (4K screen resolution), 2 * 256 * 64 bytes = 32,768 bytes are required. Cacheline alignment should be followed. This field is only valid for AVC decoder mode

MFX_BSP_BUF_BASE_ADDR_STATE

		This is one of the four RowStore Scratch Buffers which can programmed to use the internal Media Storage (total size 640 CacheLine). When Deblocking Filter Row Store Scratch Buffer Cache Select is programmed to "1", this will be cache inside MFX Media Internal Storage. Driver then needs to program this Base Address between 0 to 639, indicating starting cachelines address location for this buffer. Driver needs to make sure the whole buffer fits into Media Internal Storage *(Notes: 1 cachelines per MB for non-mbaff; 2 cachelines per MB pair for mbaff, and the buffer needs to have enough space for 1 MB (pair) row).*
	5:0	**Reserved** Access: RO Format: MBZ
5	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**MPR Row Store Scratch Buffer Base Address - Read/Write [47:32]** This field is for the upper range of MPR Row Store Scratch Buffer Base Address. This field is used for 48-bit addressing.
6	31:15	**Reserved** Access: RO Format: MBZ
	14:13	**MPR Row Store Scratch Buffer - Tiled Resource Mode** Format: U2 **For Media Surfaces:** This field specifies the tiled resource mode.

Value	Name	Description
0h	TRMODE_NONE	No tiled resource
1h	TRMODE_TILEYF	4KB tiled resources
2h	TRMODE_TILEYS	64KB tiled resources
3h	Reserved	

	12	**MPR Row Store Scratch Buffer Cache Select** This field controls if Intra Row Store is going to store inside Media Internal Storage or to LLC.

Value	Name	Description
0		Buffer going to LLC
1		Buffer going to Internal Media Storage

	11:9	**Reserved** Access: RO Format: MBZ

MFX_BSP_BUF_BASE_ADDR_STATE

	8:7	**MPR Row Store Scratch Buffer - Arbitration Priority Control** Format: U2 This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. Value / Name: 00b: Highest priority 01b: Second highest priority 10b: Third highest priority 11b: Lowest priority
	6:1	**MPR Row Store Scratch Buffer - Index to Memory Object Control State (MOCS) Tables** Format: U6 The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**
7	31:6	**Bitplane Read Buffer Base Address** It must be cacheline aligned (i.e. 64 bytes address boundary), so lower bit 0 to 5 are used for controlling information.(In Cantiga, this field must be dword aligned.)Bitplane buffer is a linear buffer. In VC1 Long format, it is written by an application. In VC1 Short Format, it is written and read by H/W only. For VC1 intel Long Format : it is a read-only bufferFor VC1 DXVA2 Short Format : it is a write and a read buffer This field is only valid for VC1 decoder mode.
	5:0	**Reserved** Access: RO Format: MBZ
8	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Bitplane Read Buffer Base Address - Read/Write [47:32]** This field is for the upper range of Bitplane Read Buffer Base Address. This field is used for 48-bit addressing.
9	31:15	**Reserved** Access: RO Format: MBZ
	14:13	**Bitplane Read Buffer - Tiled Resource Mode** Format: U2 **For Media Surfaces:** This field specifies the tiled resource mode. Value / Name / Description: 0h: TRMODE_NONE: No tiled resource

MFX_BSP_BUF_BASE_ADDR_STATE

1h	TRMODE_TILEYF	4KB tiled resources
2h	TRMODE_TILEYS	64KB tiled resources
3h	Reserved	

12:9 Reserved

Access:	RO
Format:	MBZ

8:7 Bitplane Read Buffer - Arbitration Priority Control

Format:	U2

This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.

Value	Name
00b	Highest priority
01b	Second highest priority
10b	Third highest priority
11b	Lowest priority

6:1 Bitplane Read Buffer - Index to Memory Object Control State (MOCS) Tables

Format:	U6

The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers.The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.

0 Reserved

MFX_DBK_OBJECT

MFX_DBK_OBJECT

Source: VideoCS

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_DBK_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h Common Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 9h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 0Bh Excludes DWord (0,1) Format: =n Note: Regardless of the mode, inline data must be present in this command
1	31:6	**Pre Deblocking Source Address** Format: GraphicsAddress[31:6] Specifies the 4K byte aligned frame buffer address for outputting the non-filtered reconstructed YUV picture (i.e. output of final adder in each codec standard, and prior to the deblocking filter unit).
	5:0	**Reserved** Access: RO Format: MBZ

MFX_DBK_OBJECT

2 — 31:16 — Reserved

Access:	RO
Format:	MBZ

2 — 15:0 — Pre Deblocking Source Address High

Format:	GraphicsAddress[47:32]

This field is for the upper range of Pre-Deblocking Source Address. This field is used for 48-bit addressing.

3 — 31:15 — Reserved

Access:	RO
Format:	MBZ

3 — 14:13 — Pre Deblocking Source - Tiled Resource Mode

For Media Surfaces: This field specifies the tiled resource mode.

Value	Name	Description
0h	TRMODE_NONE	No tiled resource
1h	TRMODE_TILEYF	4KB tiled resources
2h	TRMODE_TILEYS	64KB tiled resources
3h	Reserved	

3 — 12:11 — Reserved

Access:	RO
Format:	MBZ

3 — 10 — Pre Deblocking Source - Memory Compression Mode

Distinguishes Vertical from Horizontal compression. Please refer to vol1a**Memory Data Formats** chapter, **Media Memory Compression** section for more details.

Value	Name
1	Vertical Compression Mode

3 — 9 — Pre Deblocking Source - Memory Compression Enable

Format:	Enable

Memory compression will be attempted for this surface.

Value	Name
0	Compression Disable
1	Compression Enable

3 — 8:7 — Pre Deblocking Source - Arbitration Priority Control

This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.

Value	Name
00b	Highest priority
01b	Second highest priority
10b	Third highest priority

MFX_DBK_OBJECT

		11b: Lowest priority
	6:1	**Pre Deblocking Source - Index to Memory Object Control State (MOCS) Tables** The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**
4	31:6	**Deblocking Control Address** Format: GraphicsAddress[31:6] Specifies the 4K byte aligned frame buffer address as input MB-level deblocking parameters to control the way hardware deblock the each micro-block. One 512-bit cacheline is allocated for each Macroblock in raster scan order.
	5:0	**Reserved** Access: RO Format: MBZ
5	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Deblocking Control Address High** Format: GraphicsAddress[47:32] This field is for the upper range of Deblocking Control Address (DeblockCntrlAddr).This field is used for 48-bit addressing.
6	31:15	**Reserved** Access: RO Format: MBZ
	14:13	**Deblocking Control - Tiled Resource Mode** **For Media Surfaces:**This field specifies the tiled resource mode.

Value	Name	Description
0h	TRMODE_NONE	No tiled resource
1h	TRMODE_TILEYF	4KB tiled resources
2h	TRMODE_TILEYS	64KB tiled resources
3h	Reserved	

	12:11	**Reserved** Access: RO Format: MBZ
	10	**Deblocking Control - Memory Compression Mode** Distinguishes Vertical from Horizontal compression. Please refer to **Memory Data Formats**,

MFX_DBK_OBJECT

Media Memory Compression section for more details.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

	9	**Deblocking Control - Memory Compression Enable** Format: Enable Memory compression will be attempted for this surface. Value: 0 — Name: Compression Disable
	8:7	**Deblocking Control - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. 00b: Highest priority 01b: Second highest priority 10b: Third highest priority 11b: Lowest priority
	6:1	**Deblocking Source - Index to Memory Object Control State (MOCS) Tables** The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**
7	31:6	**Deblocking Destination Address** Format: GraphicsAddress[31:6] Specifies the 4K byte aligned frame buffer address for outputting the post-loop filtered reconstructed YUV picture (i.e. output of the deblocking filter unit)
	5:0	**Reserved** Access: RO Format: MBZ
8	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Deblocking Destination Address High** Format: GraphicsAddress[47:32] This field is for the upper range of Deblocking Destination Address. This field is used for 48-bit addressing.
9	31:15	**Reserved**

		Access: RO Format: MBZ
	14:13	**Deblocking Destination - Tiled Resource Mode** **For Media Surfaces:**This field specifies the tiled resource mode. Value / Name / Description: 0h / TRMODE_NONE / No tiled resource 1h / TRMODE_TILEYF / 4KB tiled resources 2h / TRMODE_TILEYS / 64KB tiled resources 3h / Reserved /
	12:11	**Reserved** Access: RO Format: MBZ
	10	**Deblocking Destination - Memory Compression Mode** Distinguishes Vertical from Horizontal compression. Please refer to vol1a**Memory Data Formats** chapter, **Media Memory Compression** section for more details. Value / Name: 0 / Horizontal Compression Mode
	9	**Deblocking Destination - Memory Compression Enable** Format: Enable Memory compression will be attempted for this surface. Value / Name: 0 / Compression Disable
	8:7	**Deblocking Destination - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. Value / Name: 00b / Highest priority 01b / Second highest priority 10b / Third highest priority 11b / Lowest priority
	6:1	**Deblocking Destination - Index to Memory Object Control State (MOCS) Tables** The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**
10	31:6	**Deblock Row Store Address** Format: GraphicsAddress[31:6] This field provides the base address of the scratch buffer (read and write) used by the

deblocking filter unit to store MB information of the previous row for filtering of each macroblock in the current row. The Deblocking Filter Row Store buffer must be 64-byte cacheline aligned. Hardware uses the horizontal address of the current macroblock to address the Deblocking Filter Row Store.

5:0 Reserved

Access:	RO
Format:	MBZ

11 31:16 Reserved

Access:	RO
Format:	MBZ

15:0 Deblock Row Store Address High

Format:	GraphicsAddress[47:32]

This field is for the upper range of Deblock Row Store Address (DeblockRowStoreAddr).This field is used for 48-bit addressing.

12 31:15 Reserved

Access:	RO
Format:	MBZ

14:13 Deblock Row Store - Tiled Resource Mode

For Media Surfaces: This field specifies the tiled resource mode.

Value	Name	Description
0h	TRMODE_NONE	No tiled resource
1h	TRMODE_TILEYF	4KB tiled resources
2h	TRMODE_TILEYS	64KB tiled resources
3h	Reserved	

12:11 Reserved

Access:	RO
Format:	MBZ

10 Deblock Row Store - Memory Compression Mode

Distinguishes Vertical from Horizontal compression. Please refer to vol1a**Memory Data Formats** chapter, **Media Memory Compression** section for more details.

Value	Name
0	Horizontal Compression Mode

9 Deblock Row Store - Memory Compression Enable

Format:	Enable

Memory compression will be attempted for this surface.

MFX_DBK_OBJECT

		Value / **Name** 0 / Compression Disable
	8:7	**Deblock Row Store - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. **Value** / **Name** 00b / Highest priority 01b / Second highest priority 10b / Third highest priority 11b / Lowest priority
	6:1	**CoeffProbability StreamIn Address - Index to Memory Object Control State (MOCS) Tables** The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**

MFX_FQM_STATE

MFX_FQM_STATE

Source: VideoCS

Length Bias: 2

This is a common state command for AVC encoder modes. For encoder, it represents both the forward QM matrices as well as the decoding QM matrices. This is a Frame-level state. Only Scaling Lists specified by an application are being sent to the hardware. The driver is responsible for determining the final set of scaling lists to be used for decoding the current slice, based on the AVC Spec Table 7-2 (Fall-Back Rules A and B).In MFX AVC PAK mode, PAK needs both forward Q scaling lists and IQ scaling lists. The IQ scaling lists are sent as in MFD in raster scan order. But the Forward Q scaling lists are sent in column-wise raster order (column-by-column) to simplify the H/W. Driver will perform all the scan order conversion for both ForwardQ and IQ.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MULTI_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MFX_COMMON_STATE Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 8h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 20h Excludes DWord (0,1) Format: =n
1	31:2	**Reserved** Access: RO Format: MBZ

MFX_FQM_STATE

	1:0	**AVC** Exists If: //AVC- Decoder Only **For AVC QM Type**: This field specifies which Quantizer Matrix is loaded. **Value** / **Name** 0 – AVC_4x4_Intra_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs) 1 – AVC_4x4_Inter_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs) 2 – AVC_8x8_Intra_MATRIX 3 – AVC_8x8_Inter_MATRIX
	1:0	**MPEG2** Exists If: //MPEG2- Decoder Only **For MPEG2 QM Type**: This field specifies which Quantizer Matrix is loaded. **Value** / **Name** 0 – MPEG_INTRA_QUANTIZER_MATRIX 1 – MPEG_NON_INTRA_QUANTIZER_MATRIX 2-3 – Reserved
	1:0	**JPEG** Exists If: //JPEG- Encoder Only **For JPEG QM Type**:This field specifies which Quantizer Matrix is loaded. **Value** / **Name** 0 – JPEG_Luma_Y_QUANTIZER_MATRIX (or R) 1 – JPEG_Chroma_Cb_QUANTIZER_MATRIX (or G) 2 – JPEG_Chroma_Cr_QUANTIZER_MATRIX (or B) **Programming Notes** For JPEG encoder, each quantization element presents 16-bit 1/QM[i][j]. In RGB encoding, because the order input image components can be RGB, GBR, BGR, YUV, the value 0 is used for the first image component, the value 1 is used for the second image component, and the value 2 is used for the third image component.
2..33	1023:0	**Forward Quantizer Matrix** The format of a Quantizer Matrix is an 8x8 matrix in raster order. Each element is an unsigned byte.

MFX_IND_OBJ_BASE_ADDR_STATE

MFX_IND_OBJ_BASE_ADDR_STATE

Source: VideoCS

Length Bias: 2

This state command provides the memory base addresses for all row stores, StreamOut buffer and reconstructed picture output buffers required by the MFD or MFC Engine (that are in addition to the row stores of the Bit Stream Decoding/Encoding Unit (BSD/BSE) and the reference picture buffers).This is a picture level state command and is common among all codec standards and for both encoder and decoder operating modes. However, some fields may only applicable to a specific codec standard. All Pixel Surfaces (original, reference frame and reconstructed frame) in the Encoder are programmed with the same surface state (NV12 and TileY format), except each has its own frame buffer base address. In the tile format, there is no need to provide buffer offset for each slice; since from each MB address, the hardware can calculated the corresponding memory location within the frame buffer directly.

The MFX_IND_OBJ_BASE_ADDR command sets the memory base address pointers for the corresponding indirect Object Data Start Addresses (Offsets) specified in each OBJECT commands. The characteristic of these indirect object data is their variable size (per MB or per Slice).Hence, each OBJECT command must specify the indirect object data offset from the base address to start fetching or writing object data.

While the use of base address is unconditional, the indirection can be effectively disabled by setting the base address to zero.
For decoder, there are:

- 1 read-only per-slice indirect object in the BSD_OBJECT Command, and
- 2 read-only per-MB indirect objects in the IT_OBJECT Command.

For decoder: the Video Command Streamer (VCS) will perform the memory access bound check automatically using the corresponding MFC Indirect Object Access Upper Bound specification. If any access is at or beyond the upper bound, zero value is returned. The request to memory is still being sent, but the corresponding codec's BSD unit will detect this condition and perform the zeroing return. If the Upper Bound is turned off, the beyond bound request will return whatever on the bus (invalid data).
For encoder, there are:

- 1 read-only per-MB indirect object in the PAK_OBJECT Command, and
- 1 write-only per-slice indirect object in the PAK Slice_State Command

For encoder: whenever an out of bound address accessing request is generated, VMX will detect such requests and snap the address to the corresponding [indirect object base address + indirect data start address]. VMX will return all 0s as the data to the requestor. NotationDefinitionPhysicalAddress[n:m] Corresponding bits of a physical graphics memory byte address (not mapped by a GTT) GraphicsAddress[n:m] Corresponding bits of an absolute, virtual graphics memory byte address (mapped by a GTT).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode

MFX_IND_OBJ_BASE_ADDR_STATE

	28:27	**Pipeline** Default Value: 2h MFX_IND_OBJ_BASE_ADDR_STATE Format: OpCode
	26:24	**Common Opcode** Default Value: 0h MFX_IND_OBJ_BASE_ADDR_STATE Format: OpCode
	23:21	**Sub OpcodeA** Default Value: 0h MFX_IND_OBJ_BASE_ADDR_STATE Format: OpCode
	20:16	**SubOpcodeB** Default Value: 3h MFX_IND_OBJ_BASE_ADDR_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 0018h Excludes DWord (0,1) Format: =n
1..2	63:0	**MFX Indirect Bitstream Object - Base Address** Format: **SplitBaseAddress4KByteAligned** Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the MFD_XXX_BSD_OBJECT command for fetching (reading) the compressed Slice Data. This field is only valid in MPEG2, AVC and VC1 decoder VLD mode.
3	31:0	**MFX Indirect Bitstream Object - Attributes** Format: **MemoryAddressAttributes**
4..5	63:0	**MFX Indirect Bitstream Object - Upper Bound** Format: **SplitBaseAddress4KByteAligned** This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the MFD_XXX_BSD_OBJECT command for the compressed Slice Data. Indirect data accessed at this address and beyond will return as 0 by the hardware. Setting this field to 0 will cause this range check to be ignored .If non-zero, this address must be greater than the MFX Indirect Bitstream ObjectBase Address state. Hardware ignores this field if indirect data is not present, i.e. the Indirect Data Length field of the MFD_XXX_BSD_OBJECT command is set to 0.This field is only valid in MPEG2, AVC, VP8, and VC1 decoder VLD mode. **Programming Notes** For **VP8 Encoder**, this field is corresponding to **MFC Indirect PAK-BSE Object - Access Upper Bound in DW24, DW25**. Please program Indirect bitstream upper bound in this field the same as DW24, DW25.

MFX_IND_OBJ_BASE_ADDR_STATE

6..7	63:0	**MFX Indirect MV Object - Base Address** Format: **SplitBaseAddress4KByteAligned** Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the encoder MFC_AVC_PAK_OBJECT command or the decoder MFD_IT_OBJECT command for fetching the per-MB MV data. This field is only valid in AVC encoder mode or in AVC decoder IT mode
8	31:0	**MFX Indirect MV Object - Attributes** Format: **MemoryAddressAttributes**
9..10	63:0	**MFX Indirect MV Object - Upper Bound** Format: **SplitBaseAddress4KByteAligned** This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the MFC_AVC_PAK_OBJECT / MFD_IT_OBJECT command for the per-MB MV data. Indirect data accessed at this address and beyond will return as 0 by the hardware. Setting this field to 0 will cause this range check to be ignored. If non-zero, this address must be greater than the MFX Indirect MV Object Base Address state. Hardware ignores this field if indirect data is not present, i.e. the Indirect Data Length field of the MFC_AVC_PAK_OBJECT / MFD_IT_OBJECT command is set to 0.This field is only valid in AVC encoder mode or in AVC decoder IT mode.
11..12	63:0	**MFD Indirect IT-COEFF Object - Base Address** Format: **SplitBaseAddress4KByteAligned** Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the MFD_IT_OBJECT command for fetching (reading) the per-MB non-scaled coefficient data (all inverse scaling and quantization are done in hardware).This field is only valid in MPEG2, AVC and VC1 decoder IT mode.
13	31:0	**MFD Indirect IT-COEFF Object - Attributes** Format: **MemoryAddressAttributes**
14..15	63:0	**MFD Indirect IT-COEFF Object - Upper Bound** Format: **SplitBaseAddress4KByteAligned** This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the MFD_IT_OBJECT command for the per-MB non-scaled coefficient data. Indirect data accessed at this address and beyond will return as 0 by the hardware. Setting this field to 0 will cause this range check to be ignored. If non-zero, this address must be greater than the MFD Indirect IT-COEFF Object Base Address state. Hardware ignores this field if indirect data is not present, i.e. the Indirect COEFF Data Length field of the MFD_IT_OBJECT command is set to 0.This field is only valid in MPEG2, AVC and VC1 decoder IT mode.
16..17	63:0	**MFD Indirect IT-DBLK Object - Base Address** Format: **SplitBaseAddress4KByteAligned** Specifies the 4K-byte aligned memory base address for the read-only indirect data object pointed in the MFD_IT_OBJECT command for fetching (reading) the per-MB Deblocking filter

MFX_IND_OBJ_BASE_ADDR_STATE

		control data.This field is only valid in AVC decoder IT mode.
18	31:0	**MFD Indirect IT-DBLK Object - Attributes** Format: **MemoryAddressAttributes**
19..20	63:0	**MFD Indirect IT-DBLK Object - Upper Bound** Format: **SplitBaseAddress4KByteAligned** This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the MFD_IT_OBJECT command for the per-MB Deblocking filter control data. Indirect data accessed at this address and beyond will return as 0 by the hardware. Setting this field to 0 will cause this range check to be ignored. If non-zero, this address must be greater than the MFD Indirect IT-DBLK Object Base Address state. Hardware ignores this field if indirect data is not present, i.e. the Indirect Deblocking Control Data Length field of the MFD_IT_OBJECT command is set to 0.This field is only valid in AVC decoder IT mode.
21..22	63:0	**MFC Indirect PAK-BSE Object - Base Address** Format: **SplitBaseAddress4KByteAligned** Specifies the 4K-byte aligned memory base address for the write-only indirect data object pointed in the PAK_SLICE_STATE command for writing out the compressed bitstream. This field is only valid in AVC encoder mode.
23	31:0	**MFC Indirect PAK-BSE Object - Attributes** Format: **MemoryAddressAttributes**
24..25	63:0	**MFC Indirect PAK-BSE Object - Upper Bound** Format: **SplitBaseAddress4KByteAligned** This field specifies the 4K-byte aligned (exclusive) maximum Graphics Memory address access by the indirect data object in the PAK_SLICE_STATE command for the per-slice output bitstream. Indirect data accessed at this address and beyond will be blocked by the hardware and ignored. Setting this field to 0 will cause this range check to be ignored If non-zero, this address must be greater than the MFC Indirect PAK-BSE Object Base Address state. This field is only valid in AVC encoder mode. **Programming Notes** For VP8 Encoder, this field should be programmed the same at both DW4, DW5 **MFX Indirect Bitstream Object - Access Upper Bound** as well as DW24, DW25.

MFX_JPEG_HUFF_TABLE_STATE

MFX_JPEG_HUFF_TABLE_STATE

Source: VideoCS

Length Bias: 2

This Huffman table commands contains both DC and AC tables for either luma or chroma. Once a Huffman table has been defined for a particular destination, it replaces the previous tables stored in that destination and shall be used in the remaining Scans of the current image. A Huffman table will be sent to H/W only when it is loaded from bitstream.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MULTI_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 7h JPEG_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 2h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 033Dh Excludes DWord (0,1) Format: =n
1	31:1	**Reserved** Access: RO Format: MBZ
	0	**HuffTableID (1-bit)** Identifies the huffman table.

MFX_JPEG_HUFF_TABLE_STATE

		Value / **Name** / **Description** 0 / Y / Huffman table for Y
2..4	95:0	**DC_BITS (12 8-bit array)** The number of DC Huffman codes of length i, where i is 1~12
5..7	95:0	**DC_HUFFVAL (12 8-bit array)** The value associated with each DC Huffman code of length i.
8..11	127:0	**AC_BITS (16 8-bit array)** the list of Li, number of Huffman codes of length i, where i is 1~16
12..51	1279:0	**AC_HUFFVAL (160 8-bit array)** the list of Vi,j, the value associated with each Huffman code of length i
52	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**AC_HUFFVAL(2-8 bit array)** In AC table, BITS can have up to 16-bit codeword. Li can be 0 ~ 162. HUFFVAL will have a list of likely random distributed values

MFX_JPEG_PIC_STATE

MFX_JPEG_PIC_STATE

Source: VideoCS

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MULTI_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 7h JPEG Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcode B** Default Value: 0h MEDIA_ Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Format: =n Value: 0001h — Name: **[Default]** — Description: Excludes DWord (0,1)
1	31	**Reserved** Access: RO Format: MBZ
	30:26	**Pixels In Horizontal Last MCU** Exists If: //Encoder Only The number of pixels in the last MCU in a row MCUs. This information is used for completion of partial MCU.

25:21 **Pixels In Vertical Last MCU**

Exists If:	//Encoder Only

The number of pixels in the last MCU in a column MCUs. This information is used for completion of partial MCU.

20 **Vertical Up-Sampling Enable**

Exists If:	//Decoder Only

Only applied to chroma blocks. This flag is used for 2:1 vertical up-sampling for chroma 420 and outputting chroma422 YUY2 or UYVY format. To enable this flag, the input should be interleaved Scan, **InputFormatYUV** should be set to YUV420, and **OutputFormatYUV** should be set to YUY2 or UYVY.

Value	Name	Description
0b		no up-sampling
1b		2:1 vertical up-sampling

19 **Reserved**

Access:	RO
Format:	MBZ

18 **Horizontal Down-Sampling Enable**

Exists If:	//Decoder Only

Only applied to chroma blocks. This flag is used for 2:1 horizontal down-sampling for chroma 422 and outputting chroma420 NV21 format. To enable this flag, the input should be interleaved Scan, **InputFormatYUV** should be set to YUV422V_2Y or YUV422V_4Y, and **OutputFormatYUV** should be set to NV12.

Value	Name	Description
0b		no down-sampling
1b		2:1 horizonatl down-sampling

17 **Vertical Down-Sampling Enable**

Exists If:	//Decoder Only

Only applied to chroma blocks. This flag is used for 2:1 vertical down-sampling for chroma 422 and outputting chroma420 NV21 format. To enable this flag, the input should be interleaved Scan, **InputFormatYUV** should be set to YUV422H_2Y or YUV422H_4Y, and **OutputFormatYUV** should be set to NV12.

Value	Name	Description
0b		no down-sampling
1b		2:1 vertical down-sampling

16 **Average Down Sampling**

Exists If:	//Decoder Only

This flag is used to select a down-sampling method when **VertDownSamplingEnb** or **HoriDownSamplingEnb** is set to 1.

Value	Name	Description
0b		Drop every other line (or column) pixels
1b		Average neighboring two pixels

15:12 **Reserved**

Access:	RO
Format:	MBZ

11:8 **Input Surface Format YUV**

Exists If:	//Encoder Only

This field specifies the surface format to read a YUV image data

Value	Name	Description
0000b		Reserved
0001b	NV12	NV12 for chroma 4:2:0
0010b	UYVY	UYVY for chroma 4:2:2
0011b	YUY2	YUY2 for chroma 4:2:2
0100b	Y8	Y8 for chroma400 Y-only image
0101b	RGB	RGB or YUV for chroma 4:4:4

Programming Notes
This field should be set accordingly for **SurfaceFormat** in MFX_SURFACE_STATE command.
R8G8B8A8_UNORM in this field is used for encoding RGB and YUV chroma 4:4:4. For RGB input, any order of image components R, G, B (e.g., RGB, GBR, BGR, YUV) will be acceptable as far as the order of Quantization tables and Huffman tables match the order of image components.

11:8 **Output Format YUV**

Exists If:	//Decoder Only

This field specifies the surface format to write the decoded JPEG image. Note that any non-interleaved JPEG input should be set to "0000". For the interleaved input Scan data, it can be set either "0000" or the corresponding format.

Value	Name	Description
0000b		3 separate plane for Y, U, and V respectively
0001b		NV12 for chroma 4:2:0
0010b		UYVY for chroma 4:2:2
0011b		YUY2 for chroma 4:2:2

Programming Notes
The **MFX_SURFACE_STATE** command should be set accordingly for each **OutputFormatYUV**. For NV12, **Surface Format** = 4 (PLANAR_420_8) For YUY2, **Surface Format** = 0 (YCRCB_NORMAL) For UYVY, **Surface Format** = 3 (YCRCB_SWAPY) NV12 (0001b) can be set only when Y, U, V are interleaved in a single Scan data with the

following cases

- **InputFormatYUV** is YUV420 and **VertDownSamplingEnb** is disalbed
- **InputFormatYUV**is YUV422H_2Y or YUV422H_4Y, and **VertDownSamplingEnb** is enabled

UYVY (0010b) and YUY2 (0011b) can be set only when Y, U, V are interleaved in a single Scan data with the following cases

- **InputFormatYUV** is YUV420 and **VertUpSamplingEnb** is enabled
- **InputFormatYUV** is YUV422H_2Y or YUV422H_4Y and **VertUpSamplingEnb** is disabled

7:6 **Reserved**

Access:	RO
Format:	MBZ

5:4 **Rotation**

Exists If:	//Decoder Only

Value	Name	Description
00b		no rotation
01b		rotate clockwise 90 degree
10b		rotate counter-clockwise 90 degree (same as rotating 270 degree clockwise)
11b		rotate 180 degree (NOT the same as flipped on the x-axis)

Programming Notes
Rotation can be set to 01b, 10b, or 11b when OutputFormatYUV is set to 0000b. For other OutputFormatYUV, Rotation is not allowed.

3 **Reserved**

Access:	RO
Format:	MBZ

2:0 **Input Format YUV**

Exists If:	//Decoder Only
Format:	U3

Value	Name	Description
0	**[Default]**	YUV400 (grayscale image)
1		YUV420
2		YUV422H_2Y (Horizontally chroma 2:1 subsampled) - horizontal 2 Y-block, 1U and 1V
3		YUV444

		4		YUV411
		5		YUV422V_2Y (Vertically chroma 2:1 subsampled) - vertical 2 Y-blocks, 1U and 1V
		6		YUV422H_4Y - 2x2 Y-blocks, vertical 2U and 2V
		7		YUV422V_4Y - 2x2 Y-blocks, horizontal 2U and 2V

	2:0	**Output MCU Structure**

Exists If:	//Encoder Only

Output MCU Structure(**OutputMcuStructure**) should be set accordingly for each Input Surface Format YUV(**InputSurfaceFormatYUV**):

- If **InputSurfaceFormatYUV** is set to NV12, **OutputMCUStructure** is set to YUV420.
- If **InputSurfaceFormatYUV** is set to UYVY or YUY2, **OutputMCUStructure** is set to YUV422H_2Y.
- If **InputSurfaceFormatYUV** is set to Y8, **OutputMCuStructure** is set to YUV400.
- If **InputSurfaceFormatYUV** is set to RGB (or GBR, BGR, YUV), **OutputMCuStructure** is set to RGB.
- If **InputSurfaceFormatYUV** is set to RGB, the order of encoded blocks in MCU will be same as the order of input image components. If the order of input image components is RGB (or GBR, BGR, YUV), then the order of blocks will be RGB (or GBR, BGR, YUV respectively).

Value	Name	Description
0	YUV400	Grayscale Image
1	YUV420	Both horizontally and vertically chroma 2:1 subsampled
2	YUV422H_2Y	Horizontally chroma 2:1 subsampled - horizontal 2 Y-blocks, 1 U and 1 V block
3	RGB	RGB or YUV444: No subsample
4		
5		
6		
7		

2	31:30	**Reserved**

Access:	RO
Format:	MBZ

	29	**Output Pixel Normalize**

Exists If:	//Decoder Only

JPEG decoded output pixels for Y and U/V in order to adjust display YUV range.

Value	Name	Description	Exists If
0		No Normalization	
1		Normalize output pixels from [0,255] to [16,235]	//Y

MFX_JPEG_PIC_STATE

	1		Normalize output pixels from [0,255] to [16,239]	//U/V

28:16 Frame Height In Blocks Minus 1

Exists If:	//Decoder Only
Format:	U13-1

(The number of blocks in height) - 1.This value is calculated using the number of lines Y and vertical sampling factor of the first component V_1 in Frame header. See the note following this table. For interleaved components, (((Y + (V_1*8 -1)) / (V_1*8)) * V_1) - 1, where "/" is integer division. For non-interleaved components, ((Y + 7) / 8) - 1.

28:16 Frame Height In Blks Minus 1

Exists If:	//Encoder Only
Format:	U13-1

(The number of blocks in height) - 1. This value is calculated using the number of lines Y and vertical sampling factor of the first component V1 in Frame header. See the note following this table.

For interleaved components: (((Y + (V1*8 -1)) / (V1*8)) * V1) - 1For non-interleaved components: ((Y + 7) / 8) - 1

15:13 RoundingQuant

Exists If:	//Encoder Only

Rounding value applied to quantization output

Value	Name	Description
000b	**[Default]**	1/2
001b		(1/2 - 1/128)
010b		(1/2 + 1/128)
011b		(1/2 - 1/64)
100b		(1/2 + 1/64)
101b		(1/2 - 1/32)
110b		(1/2 - 1/16)
111b		(1/2 - 1/8)

12:0 Frame Width In Blocks Minus 1

Exists If:	//Decoder Only
Format:	U13-1

(The number of blocks in width) - 1. This value is calculated using the number of samples per line X and horizontal sampling factor of the first component H_1 in Frame header. See the note following this table. For interleaved components, (((X + (H_1 *8 -1)) / (H_1 *8)) * H_1) - 1.For non-interleaved components, ((X + 7) / 8) - 1.

MFX_JPEG_PIC_STATE		
	12:0	**Frame Width In Blks Minus 1** Exists If: //Encoder Only Format: U13-1 (The number of blocks in width) - 1. This value is calculated using the number of samples per line X and horizontal sampling factor of the first component H1 in Frame header. See the note following this table. For interleaved components: (((X + (H1 *8 -1)) / (H1 *8)) * H1) - 1For non-interleaved components: ((X + 7) / 8) - 1

MFX_MPEG_TS_CONTROL command

MFX_MPEG_TS_CONTROL command

Source: VideoCS

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h Command Type Format: Opcode
	28:27	**Pipeline** Default Value: 2h Format: Opcode
	26:24	**Opcode** Default Value: 0h Format: Opcode
	23:21	**SubOpA** Default Value: 2h Format: Opcode
	20:16	**SubOpB** Default Value: Bh Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Value: 3h — Name: DWORD_COUNT_n **[Default]** — Description: Total length - 2 (excludes DWord0 and DWord1)
1	31:30	**Reserved** Access: RO Format: MBZ
	29	**payload_unit_start_indicator control** Programming Notes: This bit should be programmed zero always.
	28	**Additional Copy info flag in PES header**
	27	**DSM trick mode flag in PES header**
	26	**Original or flag in PES header**

MFX_MPEG_TS_CONTROL command		
	25	**Copy Right flag in PES header**
	24	**Output TS packet grouping select** 0: Return all packets continuously 1: Return 7 packets in 2K aligned buffer (with the remaining bits between the end of the 7th packet and the end of the 2K buffer including the rest being undefined)
	23:20	**StreamID lower Nibble** Stream ID Lower Nibble. Stream ID for Video can be 0xE0 through 0xEF. This 4 bit field indicates the last 4 bits of Stream ID in Mpeg transport stream as indicated in the DCN diagram
	19:13	**Reserved** Access: RO Format: MBZ
	12:0	**Video PacketID Header Parameter** This field specifies the static program fields in MPEG header for each Video packet.
2	31:0	**PCR 90 KHz component least significant bits.**
3	31:23	**27MHz Counter** Full 8-bits of 27Mhz counter
	22:1	**Reserved** Access: RO Format: MBZ
	0	**90 KHz counter MSB** Upper bit (bit 33) of 90khz counter
4	31:0	**PTS Delta** PTS Delta to be applied to 90 KHz count of PCR to generate PTS. This is a Twos complement number and added to 90 KHz PCR counter to generate PTS.
5	31:28	**Continuity Counter** This field specifies the 4b continuity counter given in the MPEGTS packet header. This should be initialized with the value that was read from MMIO at the end of the previous frame. That value will be incremented by HW so there is no need to SW to increment it. For the first frame this should be set to 0.
	27:16	**Reserved** Access: RO Format: MBZ
	15:0	**MPEGTS Packet Count** This field is ignored by HW. Driver can copyMFX_PAK_MPEGTS_STATUS register from the previous frame to DW 5 ofMPEG_TS_CONTROL_command using MI_STORE_REG_MEM

MFX_MPEG2_PIC_STATE

MFX_MPEG2_PIC_STATE

Source: VideoCS

Length Bias: 2

This must be the very first command to issue after the surface state, the pipe select and base address setting commands. For MPEG-2 the encoder is called per slice-group, however the picture state is called per picture. Notice that a slice-group is a group of consecutive slices that no non-trivial slice headers are inserted in between.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MPEG2_PIC_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 3h MPEG2_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 0h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 0h Excludes DWord (0,1)= 00Bh, used for normal decode and encode mode000h, a special case to provide a dummy image state for stitch mode operation. In this case, fields in DW1 which is part of the dummy image state command are ignored by hardware. Format: =n
1	31:28	**f_code[1][1].** Used for backward motion vector prediction. See ISO/IEC 13818-2 7.6.3.1 for details
	27:24	**f_code[1][0].** Used for backward motion vector prediction. See ISO/IEC 13818-2 7.6.3.1 for details

MFX_MPEG2_PIC_STATE

	23:20	**f_code[0][1]** Used for forward motion vector prediction. See ISO/IEC 13818-2 7.6.3.1 for details
	19:16	**f_code[0][0]** Used for forward motion vector prediction. See ISO/IEC 13818-2 7.6.3.1 for details
	15:14	**Intra DC Precision** Format: U2 See ISO/IEC 13818-2 6.3.10 for details.
	13:12	**Picture Structure** This field specifies whether the picture is encoded in the form of a frame picture or one field (top or bottom) picture. See ISO/IEC 13818-2 6.3.10 for details. Format = MPEG_PICTURE_STRUCTURE00 = Reserved01 = MPEG_TOP_FIELD10 = MPEG_BOTTOM_FIELD11 = MPEG_FRAME
	11	**TFF (Top Field First)** When two fields are stored in a picture, this bit indicates if the top field is the first field.For a frame P picture, the value 1 indicates that the top field of the reconstructed frame is the first field output by the decoding process, the same as defined in ISO/IEC 13818-2 6.3.10. Particularly, it is used by the hardware to calculate derivative motion vectors from the dual-prime motion vectors. For a field P picture, hardware uses this bit together with the Picture Structure to determine if the current picture is the Second Field. In this case, the definition of this bit differs from ISO/IEC 13818-2 6.3.10 - software must derive the value for this bit according to the following relation :Picture Structure = top field Picture Structure = bottom field Second Field = 0TFF = 1TFF = 0Second Field = 1TFF = 0TFF = 1
	10	**Frame Prediction Frame DCT** This field provides constraints on the DCT type and prediction type. It affects the syntax of the bitstream.
	9	**Concealment Motion Vector Flag** This field indicates if the concealment motion vectors are coded in intra macroblocks. It affects the syntax of the bitstream.
	8	**Quantizer Scale Type** Format: Boolean This field specifies the quantizer scaling type.

Value	Name	Description
0h		MPEG_QSCALE_LINEAR
1h		D MPEG_QSCALE_NONLINEAR esc

	7	**Intra VLC Format** This field is used by VLD
	6	**Scan Order** Format: Boolean This field specifies the Inverse Scan method for the DCT-domain coefficients in the blocks of the current picture.

MFX_MPEG2_PIC_STATE

		Value / **Name** / **Description** 0h / / MPEG_ZIGZAG_SCAN 1h / / MPEG_ALTERNATE_VERTICAL_SCAN
	5:0	**Reserved** Access: RO Format: MBZ
2	31	**I Slice Concealment Mode** Exists If: //Decoder This field controls how MPEG decoder handles MB concealment in I Slice **Value** / **Name** / **Description** 0h / Intra Concealment / Using Coefficient values to handle MB concealment 1h / Inter Concealment / Using Motion Vectors to handle MB concealment **Programming Notes** If this field is set to "1", driver must provide a valid forward reference picture (both top and bottom Field must be valid)
	30	**Reserved** Access: RO Format: MBZ
	29:28	**P/B Slice Concealment Mode** Exists If: //Decoder This field controls how MPEG decoder handles MB concealment in P/B Slice. **Value** / **Name** / **Description** 00b / INTER / If left MB is NOT Intra MB type (including skipMB), use left MB inter prediction mode [frame/field or forward/backward/bi] and MV final values as concealment. Otherwise (left MB is Intra MB), use forward reference (same polarity for field pic) with MV final values set to 0. 01b / LEFT / If left MB is NOT Intra MB type (including skipMB), use left MB inter prediction mode [frame/field or forward/backward/bi] and MV final values as concealment. Otherwise (left MB is Intra MB), use left MB dct_dc_pred[cc] values for concealment (Macroblock is concealed as INTRA MB and dct_dc_pred[cc] are DC predictor for Luma, Cr, Cb data) 10b / ZERO / Always use forward reference (same polarity for field pic) with MV final values set to 0 (Macroblock is concealed as INTER coded) 11b / INTRA / Use left MB dct_dc_pred[cc] values for concealment (Macroblock is concealed as INTRA MB and dct_dc_pred[cc] are DC predictor for Luma, Cr, Cb data

27 **Reserved**

Access:	RO
Format:	MBZ

26:25 **P/B Slice Predicted BiDir Motion Type Override - Bi-direction MV Type Override**

Exists If:	//Decoder

This field is only applicable if the Concealment Motion Type is predicted to be Bi-directional. (It is only possible if "P/B Slice Concealment Mode" is set to "00" or "01" and left MB is a bi-directional MB).

Value	Name	Description
0h	BID	Keep Bi-direction Prediction
1h	RESERVED	
2h	FWD	Only use Forward Prediction (Backward MV is forced to invalid
3h	BWD	Only use Backward Prediction (Forward MV is forced to invalid)

24 **P/B Slice Predicted Motion Vector Override Final MV value Override**

Exists If:	//Decoder

This field is only applicable if the Concealment Motion Vectors are non-zero. It is only possible if "P/B Slice Concealment Mode" is set to "00" or "01" and left MB has non-zero motion vectors).

Value	Name	Description
0h	Predicted	Motion Vectors use predicted values
1h	ZERO	Motion Vectors force to 0

23:15 **Reserved**

Access:	RO
Format:	MBZ

14 **LoadSlicePointerFlag - LoadBitStreamPointerPerSlice**

Exists If:	//Encoder

To support multiple slice picture and additional header/data insertion before and after an encoded slice. When this field is set to 0, bitstream pointer is only loaded once for the first slice of a frame. For subsequent slices in the frame, bitstream data are stitched together to form a single output data stream. When this field is set to 1, bitstream pointer is loaded for each slice of a frame. Basically bitstream data for different slices of a frame will be written to different memory locations.

Value	Name	Description
0h		Load BitStream Pointer only once for the first slice of a frame
1h		Load/reload BitStream Pointer only once for the each slice, reload the start location of the bitstream buffer from the Indirect PAK-BSE Object Data Start Address field

13:11 **Reserved**

Access:	RO

		Format: MBZ
	10:9	**Picture Coding Type** Format: U2 This field identifies whether the picture is an intra-coded picture (I), predictive-coded picture (P) or bi-directionally predictive-coded picture (B). See ISO/IEC 13818-2 6.3.9 for details. **Value** / **Name** 00b: Reserved 01b: MPEG_I_PICTURE 10b: 10 = MPEG_P_PICTURE 11b: MPEG_B_PICTURE
	8:2	**Reserved** Access: RO Format: MBZ
	1:0	**MismatchControlDisabled** These 2 bits flag disables mismatch control of the inverse transformation for some specific cases during reference reconstruction. To disable MPEG2 IDCT fixed point arithmetic correction. **Value** / **Name** / **Description** 00b: Mismatch control applies to all MBs 01b: Disable mismatch control to all intra MBs whose all AC-coefficients are zero. 10b: Disable mismatch control to all MBs whose all AC-coefficients are zero. 11b: Disable mismatch control to all MBs.
3	31	**Slice Concealment Disable Bit** Exists If: //Decode If VINunit detects the next slice starting position is either out-of-bound or smaller than or equal to the current slice starting position, VIN will set the current slice to be 1 MB and force VMDunit to do slice concealment on the next slice. This bit will disable this feature and the MB data from the next slice will be decoded from bitstream. **Value** / **Name** / **Description** 0h: Enable **[Default]** — VIN will force next slice to be concealment if detects slice boundary error 1h: Disable — VIN will not force next slice to be in concealment

MFX_MPEG2_PIC_STATE

		Programming Notes Driver has an option to detect the scenario given in description (above) and remove the second (out-of-order) slice. In this case, hardware will decode the first slice in completion and do concealment till the third slice. It should yield a picture with better quality this way.
	30:29	**Reserved** Access: RO Format: MBZ
	28:24	**Reserved**
	23:16	**FrameHeightInMBsMinus1[7:0] (Picture Height in Macroblocks)** Format: U8
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**FrameWidthInMBsMinus1[7:0] (Picture Width in Macroblocks)** Format: U8
4	31:16	**MinFrameWSize** Format: U16 Minimum Frame Size [15:0] (16-bit) (Encoder Only) Mininum Frame Size is specified to compensate for intel Rate Control Currently zero fill (no need to perform emulation byte insertion) is done only to the end of the CABAC_ZERO_WORD insertion (if any) at the last slice of a picture. Intel encoder parameter. The caller should always make sure that the value, represented by Minimum Frame Size, is always less than maximum frame size FrameBitRateMax (DWORD 10 bits 29:16). This field is reserved in Decode mode. **Programming Notes** Programmable range is 0..(2^16-1).
	15	**Reserved** Access: RO Format: MBZ
	14:12	**RoundInterAC,** rounding precision for non-Intra AC000: +1/16001: +2/16010: +3/16011: +4/16100: +5/16101: +6/16110: +7/16111: +8/16
	11	**Reserved** Access: RO Format: MBZ
	10:8	**RoundIntraAC** Format: U3 rounding precision for Intra AC000: +1/16001: +2/16010: +3/16011: +4/16100:

		+5/16101: +6/16110: +7/16111: +8/16
	7	**Reserved** Access: RO Format: MBZ
	6:4	**RoundInterDC** rounding Precision for non-Intra-DC000: +1/16001: +2/16010: +3/16011: +4/16100: +5/16101: +6/16110: +7/16111: +8/16
	3	**Reserved** Access: RO Format: MBZ
	2:1	**RoundIntraDC** rounding Precision for Intra-DC00: +1/801: +2/810: +3/811: +4/8
	0	**Reserved** Access: RO Format: MBZ
5	31:17	**Reserved** Access: RO Format: MBZ
	16	**FrameSizeControlMask** Frame size conformance mask This field is used when MacroblockStatEnable is set to 1. 0h: Do not change Slice Quantization Parameter values in MFC_MPEG2_SLICEGROUP_STATE with suggested slice QP value for frame level Rate control 1h: Replace Slice Quantization Parameter values in MFC_MPEG2_SLICEGROUP_STATE with suggested slice QP value for frame level Rate control values in MFC_IMAGE_STATUS control register.
	15:13	**Reserved** Access: RO Format: MBZ
	12	**InterMBForceCBPZeroControlMask** Format: U1 Inter MB Force CBP ZERO mask. 0h: No effect 1h: Zero out all A/C coefficients for the inter MB violating Inter Confirmance

MFX_MPEG2_PIC_STATE

	11:10	**MinFrameWSizeUnits** This field is the Minimum Frame Size Units

Value	Name	Description
00b	compatibility mode	Minimum Frame Size is in old mode (words, 2bytes)
01b	16 byte	Minimum Frame Size is in 16bytes
10b	4Kb	Minimum Frame Size is in 4Kbytes
11b	16Kb	Minimum Frame Size is in 16Kbytes

	9	**MBRateControlMask** MB Rate Control conformance mask This field is ignored when MacroblockStatEnable is disabled or MB level Rate control flag for the current MB is disable in Macroblock Status Buffer.

Value	Name	Description
0h		Do not change QP values of inter macroblock with suggested QP values in Macroblock Status Buffer
1h		Apply RC QP delta for all macroblock

	8:4	**Reserved**

Access:	RO
Format:	MBZ

	3	**FrameBitRateMinReportMask** This is a mask bit controlling if the condition of frame level bit count is less than FrameBitRateMin.

Value	Name	Description
0h	Disable	Do not update bit0 of MFC_IMAGE_STATUS control register.
1h	Enable	set bit0 and bit 1of MFC_IMAGE_STATUS control register if the total frame level bit counter is less than or equal to Frame Bit rate Minimum limit.

	2	**FrameBitRateMaxReportMask** This is a mask bit controlling if the condition of frame level bit count exceeds FrameBitRateMax.

Value	Name	Description
0h	Disable	Do not update bit0 of MFC_IMAGE_STATUS control register.
1h	Enable	set bit0 and bit 1 of MFC_IMAGE_STATUS control register if the total frame level bit counter is greater than or equal to Frame Bit rate Maximum limit.

	1	**InterMBMaxSizeReportMask** This is a mask bit controlling if the condition of any inter MB in the frame exceeds InterMBMaxSize.

Value	Name	Description
0h		Do not update bit0 of MFC_IMAGE_STATUS control register.

		1h | | set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Inter MB Conformance Max size limit.
	0	**IntraMBMaxSizeReportMask** This is a mask bit controlling if the condition of any intra MB in the frame exceeds IntraMBMaxSize. Value | Name | Description 0h | | Do not update bit0 of MFC_IMAGE_STATUS control register. 1h | | set bit0 of MFC_IMAGE_STATUS control register if the total bit counter for the current MB is greater than the Intra MB Conformance Max size limit.
6 [ExistsIf]Encode Only	31:28	**Reserved** Access: RO Format: MBZ
	27:16	**InterMBMaxSize** Default Value: FFFh This field, Inter MB Conformance Max size limit, indicates the allowed max bit count size for Inter MB
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**IntraMBMaxSize** Default Value: FFFh This field, Intra MB Conformance Max size limit, indicates the allowed max bit count size for Intra MB
7	31:0	**Reserved** Access: RO Format: MBZ
8 [ExistsIf]Encode Only	31:24	**SliceDeltaQPMax[3]** Format: S7 This field is the Slice level delta QP for total bit-count above FrameBitRateMax - first 1/8 region This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame exceeds FrameBitRateMax but is within 1/8of FrameBitRateMaxDelta above FrameBitRateMax, i.e., in the range of(FrameBitRateMax, (FrameBitRateMax+FrameBitRateMaxDelta»3). Range: [-30,30]

MFX_MPEG2_PIC_STATE

		Value / **Name** 0h / Disable 1h / Enable
	23:16	**SliceDeltaQPMax[2]** Format: S7 Range: [-30,30] This field is the Slice level delta QP forbit-count above FrameBitRateMax - above 1/8 and below 1/ 4 This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between 1/8 and ofFrameBitRateMaxDelta above FrameBitRateMax, i.e., in the range of((FrameBitRateMax+ FrameBitRateMaxDelta»3), (FrameBitRateMax+FrameBitRateMaxDelta»2).
	15:8	**SliceDeltaQPMax[1]** Format: S7 Range: [-30,30] This field is the Slice level delta QP forbit-count above FrameBitRateMax - above1/ 4 and below 1/2 This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between and of FrameBitRateMaxDeltaabove FrameBitRateMax, i.e., in the range of ((FrameBitRateMax+FrameBitRateMaxDelta»2), (FrameBitRateMax+FrameBitRateMaxDelta»1).
	7:0	**SliceDeltaQPMax[0]** Format: S7 Range: [-30,30] This field is the Slice level delta QP forbit-count above FrameBitRateMax - above 1/ 2This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bitcount for the entire frame is above FrameBitRateMax by more than half the distance of FrameBitRateMaxDelta , i.e., in the range of ((FrameBitRateMax+FrameBitRateMaxDelta»1), infinite).
9 [ExistsIf]Encode Only	31:24	**SliceDeltaQPMin[3]** Format: S7 Range: [-30,30] This field is the Slice level delta QP for total bit-count below FrameBitRateMin - first 1/8 region This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is less than FrameBitRateMin and greater than or equal to 1/8 the distance of FrameBitRateMinDelta from FrameBitRateMin,i.e., in the range of [(FrameBitRateMin-

		FrameBitRateMinDelta»3), FrameBitRateMin).
	23:16	**SliceDeltaQPMin[2]** Format: S7 Range: [-30,30] This field is the Slice level delta QP forbit-count below FrameBitRateMin - below 1/ 8 and above 1/ 4This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between one-eighth and quarter the distance of FrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of[(FrameBitRateMin- FrameBitRateMinDelta»2), (FrameBitRateMin-FrameBitRateMinDelta»3)).
	15:8	**SliceDeltaQPMin[1]** Format: S7 Range: [-30,30] This field is the Slice level delta QP forbit-count below FrameBitRateMin- below 1/4 and above 1/ 2This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bit count for the entire frame is between quarter and half the distance ofFrameBitRateMinDelta from FrameBitRateMin, i.e., in the range of[(FrameBitRateMin- FrameBitRateMinDelta»1), (FrameBitRateMin-FrameBitRateMinDelta»2)).
	7:0	**SliceDeltaQPMin[0]** Format: S7 Range: [-30,30] This field is the Slice Level Delta QP forbit-count below FrameBitRateMin - below 1/ 2This field is used to calculate the suggested slice QP into the MFC_IMAGE_STATUS control register when total bitcount for the entire frame is below FrameBitRateMin by more than half the distance of FrameBitRateMinDelta , i.e., in the range of [0, (FrameBitRateMin-FrameBitRateMinDelta»1).
10 [ExistsIf]Encode Only	31	**FrameBitrateMaxUnit** This field is the Frame Bitrate Maximum Limit Units. Value / Name / Description: 0h / Byte / FrameBitRateMax is in units of 32 Bytes when FrameBitrateMaxUnitMode is 1 and in units of 128 Bytes if FrameBitrateMaxUnitMode is 0 1h / Kilobyte / FrameBitRateMax is in units of 4KBytes Bytes when FrameBitrateMaxUnitMode is 1 and in units of 16KBytes if FrameBitrateMaxUnitMode is 0
	30	**FrameBitrateMaxUnitMode** BitField This field is the Frame Bitrate Maximum Limit Units.dDesc

Value	Name	Description
0h	Compatibility mode	FrameBitRateMaxUnit is in old mode (128b/16Kb)
1h	New mode	FrameBitRateMaxUnit is in new mode (32byte/4Kb)

29:16 **FrameBitRateMax**

This field is the Frame Bitrate Maximum Limit. This field along with FrameBitrateMaxUnit determines maximum allowed bits in a frame before multi-pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count exceeds this value. When FrameBitrateMaxUnitMode is 0(compatibility mode) bits 16:27 should be used, bits 28 and 29 should be 0.

Value	Name	Description
[0-16383]		WhenFrameBitrateMaxUnit =0, this field is measured in unit of 32 bytes.The frame rate in bytes is range from 0-512KBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.
[0-16383]		WhenFrameBitrateMaxUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-64MBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.

15 **FrameBitrateMinUnit**

This field is the Frame Bitrate Minimum Limit Units.

Value	Name	Description
0h	Byte	FrameBitRateMax is in units of 32 Bytes when FrameBitrateMinUnitMode is 1 and in units of 128 Bytes if FrameBitrateMinUnitMode is 0
1h	KiloByte	FrameBitRateMax is in units of 4KBytes Bytes when FrameBitrateMaxUnitMode is 1 and in units of 16KBytes if FrameBitrateMaxUnitMode is 0

14 **FrameBitrateMinUnitMode**

This field is the Frame Bitrate Minimum Limit Units.ValueNameDescriptionProject

Value	Name	Description
0h	compatibility mode	FrameBitRateMaxUnit is in old mode (128b/16Kb)
1h	New Mode	FrameBitRateMaxUnit is in new mode (32byte/4Kb)

13:0 **FrameBitRateMin**

This field is the Frame Bitrate Minimum Limit. This field along with FrameBitrateMinUnit determines minimum allowed bits in a frame before multi-pass gets triggered (when enabled). In other words, multi-pass is triggered when the actual frame byte count happen to be below this value.

It takes on a value in the range of [0-16383].

When FrameBitrateMinUnitMode is 0 (compatibility mode) bits 0:11 should be used, bits 12 and 13 should be 0.

WhenFrameBitrateMinUnit=0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-512KBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.

WhenFrameBitrateMinUnit =1, this field is measured in unit of 4K bytes (1K=1024).The

		frame rate in bytes is range from 0-64MBytes, hence the .this field is programmed from 0 to 16,384 (14-bits) unit.
11 [ExistsIf]Encode Only	31	**Reserved** Access: RO Format: MBZ
	30:16	**FrameBitRateMaxDelta** Default Value: 0h Access: None Format: U15 This field is used to select the slice delta QP when FrameBitRateMax Is exceeded. It shares the same FrameBitrateMaxUnit. When FrameBitrateMaxUnitMode is 0(compatibility mode), only bits 16:27 should be used, bits 28, 29 and 30 should be 0. This field is used to select the slice delta QP when FrameBitRateMax Is exceeded. It shares the same FrameBitrateMaxUnit. When FrameBitrateMaxUnitMode is 0(compatibility mode) bits 16:27 should be used, bits 28, 29 and 30 should be 0. Range : [0-32767] WhenFrameBitrateMaxUnit =0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-1024KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit. WhenFrameBitrateMaxUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-128MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit.
	15	**Reserved** Access: RO Format: MBZ
	14:0	**FrameBitRateMinDelta** This field is used to select the slice delta QP when FrameBitRateMin Is exceeded. It shares the same FrameBitrateMinUnit. When FrameBitrateMinUnitMode is 0(compatibility mode) bits 0:11 should be used, bits12, 13 and 14 should be 0.Note: HW requires the following condition FrameBitRateMinDelta <= 2*FrameBitRateMinMust be true, otherwise it may cause unpredicted behavior.

Value	Name	Description
[0-32767]		WhenFrameBitrateMaxUnit =0, this field is measured in unit of 32 bytes. The frame rate in bytes is range from 0-1024KBytes, hence the .this field is programmed from 0 to 32,767 (15-bits) unit.
[0-32767]		WhenFrameBitrateMaxUnit =1, this field is measured in unit of 4K bytes (1K=1024).The frame rate in bytes is range from 0-128MBytes, hence the .this field is programmed from 0 to 32,767 (14-bits) unit.

MFX_MPEG2_PIC_STATE		
12	31:0	**Reserved** Access: RO Format: MBZ

MFX_PAK_INSERT_OBJECT

MFX_PAK_INSERT_OBJECT

Source: VideoCS

Length Bias: 2

The MFX_PAK_INSERT_OBJECT command is the first primitive command for the AVC, MPEG2, JPEG, and VP8 Encoding Pipeline.

This command is issued to setup the control and parameters of inserting a chunk of compressed/encoded bits into the current bitstream output buffer starting at the specified bit location to perform the actual insertion by transferring the command inline data to the output buffer max, 32 bits at a time.
It is a variable length command as the data to be inserted are presented as inline data of this command. It is a multiple of 32-bit (1 DW), as the data bus to the bitstream buffer is 32-bit wide.
Multiple insertion commands can be issued back to back in a series. It is host software's responsibility to make sure their corresponding data will properly stitch together to form a valid H.264 bitstream.
Internally, MFX hardware will keep track of the very last two bytes' (the very last byte can be a partial byte) values of the previous insertion. It is required that the next Insertion Object Command or the next PAK Object Command to perform the start code emulation sequence check and prevention 0x03 byte insertion with this end condition of the previous insertion.
Hardware will keep track of an output bitstream buffer current byte position and the associated next bit insertion position index. Data to be inserted can be a valid H.264 NAL units or a partial NAL unit. Certain NAL unit has a minimum byte size requirement. As such the hardware will optionally (enabled by STATE Command) determines the number of CABAC_ZERO_WORD to be inserted to the end of the current NAL, based on the minimum byte size of a NAL and the actual bin count of the encoded Slice. Since prior to the CABAC_ZERO_WORD insertion, the RBSP or EBSP is already byte-aligned, so each CABAC_ZERO_WORD insertion is actually a 3-byte sequence 0x00 00 03. The inline data may have already been processed for start code emulation byte insertion, except the possibility of the last 2 bytes plus the very last partial byte (if any). Hence, when hardware performing the concatenation of multiple consecutive insertion commands, or concatenation of an insertion command and a PAK object command, it must check and perform the necessary start code emulation byte insert at the junction. The inline data is required to be byte aligned on the left (first transmitted bit order) and may or may not be byte aligned on the right (last transmitted bits).

The command will specify the bit offset of the last valid DW. Each insertion state command defines a chunk of bits (compressed data) to be inserted at a specific location of the output compressed bitstream in the output buffer. Depend on CABAC or CAVLC encoding mode (from Slice State), PAK Object Command is always ended in byte aligned output bitstream except for CABAC header insertion which is bit aligned. In the aligned cases, PAK will perform 0 filling in CAVLC mode, and 1 filling in CABAC mode.
Insertion data can include any encoded syntax elements bit data before the encoded Slice Data (PAK Object Command) of the current SliceSPS NALPPS NALSEI NALOther Non-Slice NALLeading_Zero_8_bits (as many bytes as there is)Start Code PrefixNAL Header ByteSlice Header Any encoded syntax elements bit data after the encoded Slice Data (PAK Object Command) of the current Slice and prior to the next encoded Slice Data of the next Slice or prior to the end of the bistream, whichever comes firstCabac_Zero_Word or Trailing_Zero_8bits (as many bytes as there is).
Anything listed above before a Slice DataContext switch interrupt is not supported by this command.

DWord	Bit	Description

MFX_PAK_INSERT_OBJECT

0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_PAK_INSERT_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MFX_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 2h Format: OpCode
	20:16	**SubOpcode B** Default Value: 8h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: [1h, FFFh] Excludes DWord (0,1) = Variable Length in DW Format: =n
1	31:18	**Reserved** Access: RO Format: MBZ
	17:16	**DataByteOffset - SrcDataStartingByteOffset[1:0]** Source Data Starting Byte Position within the very first inline DW. **Programming Notes** Must be set to 0 for JPEG encoder
	15	**HeaderLengthExcludeFrmSize** In case this flag is on, bits are NOT accumulated during current access unit coding neither for Cabac Zero Word insertion bits counting or for output in MMIO register MFC_BITSTREAM_BYTECOUNT_FRAME_NO_HEADER. When using HeaderLenghtExcludeFrmSize for header insertion, the software needs to make sure that data comes already with inserted start code emulation bytes. SW shouldn't set EmulationFlag bit (Bit 3 of DWORD1 of MFX_PAK_INSERT_OBJECT).

Value	Name	Description
1	NO_ACCUMULATION	Bits during current call are not accumulated
0	ACCUMULATE	All bits accumulated

Programming Notes
Must be set to 0 for JPEG encoder

14 **Slice Header Indicator**

This bit indicates if the insert object is a slice header. In the VDEnc mode, PAK only gets this command at the beginning of the frame for slice position X=0, Y=0. It internally generates the header that needs to be inserted per slice. For VDEnc mode, this bit should always be set.

Value	Name	Description
1	SLICE_HEADER	Insertion Object is a Slice Header. The command is stored internally by HW and is used for inserting slice headers.
0	LEGACY	Legacy Insertion Object command. The PAK Insertion Object command is not stored in HW.

Programming Notes
In VDENC mode, we support only Slice layer without partitioning RBSP syntax. The payload for PAK_INS_OBJ should contain only start code for Slice header followed by NAL_type and slice header (slice_header() in AVC spec).The payload for PAK_INS_OBJ shouldn't contain CABAC Byte alignment bits. HW adds these alignment bits which are part of slice_data. Example PAK_INS_OBJ payload : 00 00 01 <NAL_type> <slice_header_Byte0> ..<slice_header_Byte LAST>Any zero_bytes that are added before slice header can be inserted by any preceding general PAK_INS_OBJ.

13:8 **DataBitsInLastDW - SrCDataEndingBitInclusion[5:0]**

Source Data to be included in the very last inline DW. Follows the MSBit is the upper bit of each byte within the DW. The lower byte is actually processed first.For example, SrCDataEndingBitInclusion = 9, bit 7:0 and bit 15 are included as valid header data.

Value	Name
[1,32]	

7:4 **SkipEmulByteCnt - Skip Emulation Byte Count**

Skip emulation check for number of starting bytes It can be programmed from 0 to 15 bytes. For example, to skip the start code that has already prefixed in the bitstream.

Programming Notes
Must be set to 0 for JPEG encoder

3 **EmulationFlag - EmulationByteBitsInsertEnable**

Value	Name	Description
0	NONE	No emulation
1	EMULATE	Instruct the hardware to perform Start Code Prefix (0x 00 00 01/02/03/00) Search and Prevention Byte (0x 03) insertion on the insertion data of this command. It is required that hardware will handle a start code prefix crossing the boundary between insertion commands, or an insertion command followed by a PAK Object command.

		Programming Notes Must be set to 0 for JPEG encoder
	2	**LastHeaderFlag - LastSrcHeaderDataInsertCommandFlag** To process a series of consecutive insertion commands, this flag (=1) indicates the current command is the last 'header' insertion in the series. In CABAC, hardware must perform the "1" insert for byte align for Slice Header before Slice Data comes in in the next PAK-OBJECT command.In CAVLC, hardware ignores this bit
	1	**EndOfSliceFlag - LastDstDataInsertCommandFlag** No more insertion command and no more PAK-OBJECT command follows. Flush data out to memory
	0	**BitstreamStartReset - ResetBitStreamStartingPos** Value / Name / Description: 1 / RESET / Reset the bitstream buffer insertion position to the bitstream buffer starting position. 0 / INSERT / Insert the current command inline data starting at the current bitstream buffer insertion position **Programming Notes** Must be set to 1 for JPEG encoder
2..n	31:0	**Insert Data PayLoad** Actual Data to be inserted to the output bitstream buffer.

MFX_PIPE_BUF_ADDR_STATE

MFX_PIPE_BUF_ADDR_STATE

Source: VideoCS

Length Bias: 2

This state command provides the memory base addresses for all row stores, StreamOut buffer and reconstructed picture output buffers required by the MFD or MFC Engine (that are in addition to the row stores of the Bit Stream Decoding/Encoding Unit (BSD/BSE) and the reference picture buffers).
This is a picture level state command and is common among all codec standards and for both encoder and decoder operating modes. However, some fields may only applicable to a specific codec standard. All Pixel Surfaces (original, reference frame and reconstructed frame) in the Encoder are programmed with the same surface state (NV12 and TileY format), except each has its own frame buffer base address. In the tile format, there is no need to provide buffer offset for each slice; since from each MB address, the hardware can calculated the corresponding memory location within the frame buffer directly.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_PIPE_BUF_ADDR_STATE Format: OpCode
	26:24	**Common Opcode** Default Value: 0h MFX_COMMON_STATE Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 2h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Format: =n Fixed Length Value: 3Fh — Name: DWORD_COUNT_n **[Default]** — Description: Excludes DWord (0,1)

MFX_PIPE_BUF_ADDR_STATE

1 — 31:6 — Pre Deblocking Destination Address

Format:	GraphicsAddress[31:6]

Specifies the 4K byte aligned frame buffer address for outputting the non-filtered reconstructed YUV picture (i.e. output of final adder in each codec standard, and prior to the deblocking filter unit).This field is ignored if PreDeblockOutEnable is set to 0 (disable).

1 — 5:0 — Reserved

Access:	RO
Format:	MBZ

2 — 31:16 — Reserved

Access:	RO
Format:	MBZ

2 — 15:0 — Pre Deblocking Destination Address High

Format:	GraphicsAddress[47:32]

This field is for the upper range of Pre-Deblocking Destination Address. This field is ignored if **PreDeblockOutEnable** is set to 0 (disable).

3 — 31:15 — Reserved

Access:	RO
Format:	MBZ

3 — 14:13 — Pre Deblocking - Tiled Resource Mode

For Media Surfaces: This field specifies the tiled resource mode.

Value	Name	Description
0h	TRMODE_NONE	No tiled resource
1h	TRMODE_TILEYF	4KB tiled resources
2h	TRMODE_TILEYS	64KB tiled resources
3h	Reserved	

3 — 12:11 — Reserved

Access:	RO
Format:	MBZ

3 — 10 — Compression Type

This field is valid only is Memory Compression is enabled.

Value	Name
0	Media Compression Enabled **[Default]**
1	Render Compression Enabled

3 — 9 — Pre Deblocking - Memory Compression Enable

Format:	Enable

Memory compression will be attempted for this surface.

Value	Name
0	Compression Disable
1	Compression Enable

Programming Notes

Video Mode	Compression Enable
AVC Frame Only (No MBAFF or Field)	Yes
VP8 (Only Frame is supported	Yes
VC1 (No overlap smoothing, No field)	Yes
MPGE2 (No field)	Yes

JPEG Decode

Chroma Format	Output Format	Compression Enable
422H_2Y,422H_4Y	YUY2	Yes
422H_2Y,422H_4Y	YUY2	Yes
422H_2Y,422H_4Y	UYVY	Yes
422H_2Y, 422H_4Y, 422V_2Y, 422V_4Y	NV12	No
420	YUY2, UYVY	No
420	NV12	Yes

8:7 **Pre Deblocking - Arbitration Priority Control**

This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.

Value	Name
00b	Highest priority
01b	Second highest priority
10b	Third highest priority
11b	Lowest priority

6:0 **Pre Deblocking - Memory Object Control State**

Format:	MEMORY_OBJECT_CONTROL_STATE

Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).

4 31:6 **Post Deblocking Destination Address**

Format:	GraphicsAddress[31:6]

Specifies the 4K byte aligned frame buffer address for outputting the post-loop filtered reconstructed YUV picture (i.e. output of the deblocking filter unit)This field is ignored if PostDeblockOutEnable is set to 0 (disable).

5:0 **Reserved**

Access:	RO
Format:	MBZ

MFX_PIPE_BUF_ADDR_STATE

5	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Post Deblocking Destination Address High** Format: GraphicsAddress[47:32] This field is for the upper range of Post-Deblocking Destination Address. This field is ignored if **PostDeblockOutEnable** is set to 0 (disable).
6	31:15	**Reserved** Access: RO Format: MBZ
	14:13	**Post Deblocking - Tiled Resource Mode** **For Media Surfaces:**This field specifies the tiled resource mode. Value / Name / Description: 0h / TRMODE_NONE / No tiled resource 1h / TRMODE_TILEYF / 4KB tiled resources 2h / TRMODE_TILEYS / 64KB tiled resources 3h / Reserved /
	12:11	**Reserved** Access: RO Format: MBZ
	10	**Compression Type** This field is applicable only when Memory compression is enabled. Value / Name: 0 / Media Compression Enabled **[Default]** 1 / Render Compression Enabled
	9	**Post Deblocking - Memory Compression Enable** Format: Enable Memory compression will be attempted for this surface. Value / Name: 0 / Compression Disable 1 / Compression Enable
	8:7	**Post Deblocking - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. Value / Name: 00b / Highest priority 01b / Second highest priority 10b / Third highest priority

		11b: Lowest priority
	6:0	**Post Deblocking - Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).
7	31:6	**Original Uncompressed Picture Source Address** Format: GraphicsAddress[31:6] Specifies the 64 byte aligned frame buffer address for fetching YUV pixel data from the original uncompressed input picture for encoding.This field is only valid in **encoding** mode.
	5:0	**Reserved** Access: RO Format: MBZ
8	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Original Uncompressed Picture Source Address High** Format: GraphicsAddress[47:32] This field is for the upper range of Original Uncompressed Picture Source Address.This field is valid for **encoding** mode only.
9	31:15	**Reserved** Access: RO Format: MBZ
	14:13	**Original Uncompressed Picture - Tiled Resource Mode** **For Media Surfaces:**This field specifies the tiled resource mode.

Value	Name	Description
0h	TRMODE_NONE	No tiled resource
1h	TRMODE_TILEYF	4KB tiled resources
2h	TRMODE_TILEYS	64KB tiled resources
3h	Reserved	

	12:11	**Reserved** Access: RO Format: MBZ

MFX_PIPE_BUF_ADDR_STATE

	10	**Compression Type** **Description** This field is valid only when memory compression enable is true. **Value** / **Name** 0 – Media Compression Enabled **[Default]** 1 – Render Compression Enabled
	9	**Original Uncompressed Picture - Memory Compression Enable** Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before. **Value** / **Name** 0 – Compression Disable
	8:7	**Original Uncompressed Picture Source - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. **Value** / **Name** 00b – Highest priority 01b – Second highest priority 10b – Third highest priority 11b – Lowest priority
	6:0	**Original Uncompressed Picture Source - Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).
10	31:6	**StreamOut Data Destination Base Address** Format: GraphicsAddress[31:6] Specifies the 64 byte aligned address for outputting the per-MB indirect data to memory when **StreamOutEnable** is set in the MFX_PIPE_MODE_SELECT command. For Decoder : This field is used for transcoding purpose. For Encoder : This field is used for dynamic repeat of frame in PAK for Rate Control. Also used for feeding coding information back to the Host, Video Preprocessing Unit and ENC Unit. All data are written in fixed formats, and therefore all record sizes are known in the hardware. Hardware can calculate the offset into this base address for per-MB data.
	5:0	**Reserved** Access: RO Format: MBZ
11	31:16	**Reserved** Access: RO Format: MBZ

MFX_PIPE_BUF_ADDR_STATE

	15:0	**StreamOut Data Destination Base Address High** Format: GraphicsAddress[47:32] This field is for the upper range of Original Uncompressed Picture Source Address
12	31:15	**Reserved** Access: RO Format: MBZ
	14:13	**StreamOut Data Destination - Tiled Resource Mode** **For Media Surfaces:**This field specifies the tiled resource mode. Value / Name / Description: 0h / TRMODE_NONE / No tiled resource 1h / TRMODE_TILEYF / 4KB tiled resources 2h / TRMODE_TILEYS / 64KB tiled resources 3h / Reserved /
	12:10	**Reserved** Access: RO Format: MBZ
	9	**StreamOut Data Destination - Memory Compression Enable** Note: This is a READ Surface. The setting of this bit should match the settings on how this is written out before. Value / Name: 0 / Compression Disable
	8:7	**StreamOut Data Destination - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. Value / Name: 00b / Highest priority 01b / Second highest priority 10b / Third highest priority 11b / Lowest priority
	6:0	**StreamOut Data Destination - Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).
13	31:6	**Intra Row Store Scratch Buffer Base Address** Format: GraphicsAddress[31:6] This field provides the base address of the scratch buffer (read/write) used by the AVC/VP8 IntraPrediction unit to store MB information of the previous row for processing of each macroblock in the current row. The Intra Row Store buffer must be 64-byte cacheline aligned.

Hardware uses the horizontal address of the current macroblock to address the Intra Row Store. This field is ignored in MPEG2 and VC1 mode. Max 256 cachelines for 4K pixels (1 cacheline for either MBAFF or non-MBAFF)Intra Row Store Scratch Buffer - Arbitration Priority Control

Programming Notes

This is one of the four RowStore Scratch Buffers which can be programmed to use the internal Media Cache (total size 640 CacheLine). When Intra Row Store Scratch Buffer Cache Select is programmed to "1", this data will be stored inside MFX Media Internal Storage. Driver then needs to program this Base Address between 0 to 639, indicating starting cachelines address to Media Cache. Driver needs to make sure the whole buffer fits into MFX Media Internal Storage.

(Notes: 1 cacheline per MB, and the buffer needs to have enough space for 1 MB row).

	5:0	**Reserved**
		Access: RO
		Format: MBZ
14	31:16	**Reserved**
		Access: RO
		Format: MBZ
	15:0	**Intra Row Store Scratch Buffer Base Address High**
		Format: GraphicsAddress[47:32]
		This field is for the upper range of Intra RowStore/Scratch Buffer Base Address This field is ignored in MPEG2 and VC1 mode.
15	31:15	**Reserved**
		Access: RO
		Format: MBZ
	14:13	**Intra Row Store Scratch Buffer - Tiled Resource Mode**
		For Media Surfaces: This field specifies the tiled resource mode.

Value	Name	Description
0h	TRMODE_NONE	No tiled resource
1h	TRMODE_TILEYF	4KB tiled resources
2h	TRMODE_TILEYS	64KB tiled resources
3h	Reserved	

	12	**Intra Row Store Scratch Buffer Cache Select**
		This field controls if Intra Row Store is going to store inside Media Cache or to LLC.

Value	Name	Description
0		Buffer going to LLC.
1		Buffer going to Internal Media Storage

MFX_PIPE_BUF_ADDR_STATE

	11	**Reserved** Access: RO Format: MBZ
	10	**Reserved - Intra Row Store**
	9	**Intra Row Store Scratch Buffer - Memory Compression Enable** Value: 0 — Name: Compression Disable **Programming Notes** This surface is linear surface. This bit must be set to "0" since only TileY/TileYf/TileYs surface is allowed to be compressed
	8:7	**Intra Row Store Scratch Buffer - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. Value: 00b — Name: Highest priority Value: 01b — Name: Second highest priority Value: 10b — Name: Third highest priority Value: 11b — Name: Lowest priority
	6:0	**Intra Row Store Scratch Buffer - Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).
16	31:6	**Deblocking Filter Row Store Scratch Base Address** Format: GraphicsAddress[31:6] Deblocking Filter Row Store is needed for: • AVC and VC1 In-Loop Deblocking Filter • VC1 Overlap-smoothing Filter • AVC, VC1, and MPEG-2 Out-Of-Loop Deblocking Filter (Intel extension) This field provides the 64 byte aligned base address of the scratch buffer (read and write) used by the deblocking filter unit to store MB information of the previous row for filtering of each macroblock in the current row. The Deblocking Filter Row Store buffer must be 64-byte cacheline aligned. Hardware uses the horizontal address of the current macroblock to address the Deblocking Filter Row Store.Max 6 cachelines for VC1 and MPEG2, and max 4 for AVC (for MBAFF, 2 for non-MBAFF) **Programming Notes** This is one of the four RowStore Scratch Buffers which can programmed to use the internal Media Cache (total size 640 CacheLine). When Deblocking Filter Row Store Scratch Buffer Cache Select is programmed to "1", this will be stored inside MFX Media Internal Storage.

		Driver then needs to program this Base Address between 0 to 639, indicating starting cachelines address location for this buffer. Driver needs to make sure the whole buffer fits into Media Internal Storage. *(Notes: 2 cachelines per MB for non-mbaff; 4 cahcelines per MB pair for mbaff, and the buffer needs to have enough space for 1 MB (pair) row).*
	5:0	**Reserved** Access: RO Format: MBZ
17	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Deblocking Filter Row Store Scratch Base Address High** Format: GraphicsAddress[47:32] This field is for the upper range of Deblocking Filter Row Store Scratch Buffer Address.
18	31:15	**Reserved** Access: RO Format: MBZ
	14:13	**Deblocking Filter Row Store - Tiled Resource Mode** **For Media Surfaces:**This field specifies the tiled resource mode. Value / Name / Description: 0h / TRMODE_NONE / No tiled resource 1h / TRMODE_TILEYF / 4KB tiled resources 2h / TRMODE_TILEYS / 64KB tiled resources 3h / Reserved /
	12	**Deblocking Filter Row Store Scratch Buffer Cache Select** This field controls if Intra Row Store is going to store inside Media Internal Storage or to LLC. Value / Name / Description: 0 / / Buffer going to LLC 1 / / Buffer going to Media Internal Storage
	11	**Reserved** Access: RO Format: MBZ
	10	**Deblocking Filter Row Store Scratch - Memory Compression Mode** Value / Name: 0 / Reserved **[Default]**

MFX_PIPE_BUF_ADDR_STATE

	9	**Deblocking Filter Row Store Scratch - Memory Compression Enable** **Value** / **Name** 0 – Compression Disable **Programming Notes** This surface is linear surface. This bit must be set to "0" since only TileY/TileYf/TileYs surface is allowed to be compressed
	8:7	**Deblocking Filter Row Store Scratch - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. **Value** / **Name** 00b – Highest priority 01b – Second highest priority 10b – Third highest priority 11b – Lowest priority
	6:0	**Deblocking Filter Row Store Scratch - Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).
19..50	1023:0	**Reference Picture Base Addr** Format: MFX_REFERENCE_PICTURE_BASE_ADDR[16]
51	31:15	**Reserved** Access: RO Format: MBZ
	14:13	**Reference Picture - Tiled Resource Mode** **For Media Surfaces:**This field specifies the tiled resource mode. **Value** / **Name** / **Description** 0h – TRMODE_NONE – No tiled resource 1h – TRMODE_TILEYF – 4KB tiled resources 2h – TRMODE_TILEYS – 64KB tiled resources 3h – Reserved
	12:9	**Reserved** Access: RO Format: MBZ
	8:7	**Reference Picture - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. **Value** / **Name** 00b – Highest priority

		01b: Second highest priority 10b: Third highest priority 11b: Lowest priority
	6:0	**Refernce Picture - Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).
52	31:6	**Macroblock Buffer Base Address or Decoded Picture Error/Status Buffer Base Address** Format: GraphicsAddress[31:6] **For decoder:** Specifies the 64 byte aligned buffer address for writing a single error/status record into the memory when **Pic Error/Status Report Enable** is set in the MFX_PIPE_MODE_SELECT Command. The error/status record is written by HW at the end of decoding one single picture. The content of this memory location can be later read by the driver only. The record is written in a fixed format, total 96-bits in size always. Please refer to "Media VDBOX -> Video Codec -> Other Codec Functions -> MFX Error Handling -> Decoder" session for the output format. **For encoder:** Specifies the 64 byte aligned buffer address for reading the per-MB indirect data from memory when **MacroblockStatEnable** is set in the MFX_AVC_IMG_STATE Command. This field is used for dynamic repeat of frame in PAK for Rate Control. Also used for feeding coding information back to the Host, Video Preprocessing Unit, and ENC Unit. All data are written in fixed formats, and therefore all record sizes are known in the hardware. Hardware can calculate the offset into this base address for per-MB data.
	5:0	**Reserved** Access: RO Format: MBZ
53	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Macroblock Buffer Base Address or Decoded Picture Error/Status Buffer Base Address High** Format: GraphicsAddress[47:32] This field is for the upper range of Macroblock Status Buffer Base Address
54	31:15	**Reserved** Access: RO Format: MBZ

MFX_PIPE_BUF_ADDR_STATE

	14:13	**Macroblock Status Buffer - Tiled Resource Mode** **For Media Surfaces:**This field specifies the tiled resource mode. **Value** / **Name** / **Description** 0h / TRMODE_NONE / No tiled resource 1h / TRMODE_TILEYF / 4KB tiled resources 2h / TRMODE_TILEYS / 64KB tiled resources 3h / Reserved /
	12:11	**Reserved** Access: RO Format: MBZ
	10	**Macroblock Status Buffer - Memory Compression Mode** **Value** / **Name** 0 / Reserved **[Default]**
	9	**Macroblock Status Buffer - Memory Compression Enable** **Value** / **Name** 0 / Compression Disable **Programming Notes** This surface is linear surface. This bit must be set to "0" since only TileY/TileYf/TileYs surface is allowed to be compressed
	8:7	**Macroblock Status Buffer - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. **Value** / **Name** 00b / Highest priority 01b / Second highest priority 10b / Third highest priority 11b / Lowest priority
	6:0	**Macroblock Status Buffer - Memory Object Control State** Format: **MEMORY_OBJECT_CONTROL_STATE** Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).
55	31:6	**Macroblock ILDB StreamOut Buffer Base Address** Format: GraphicsAddress[31:6] Specifies the 64 byte aligned buffer address for writing MB ILDB parameter per MB to memory when **Debocker streamout enable** is set in the MFX_PIPE_MODE_SELECT Command. The ildb MB control parameters are written by HW at the end of each decoding MB. Only AVC edge information is being streamed out. It is used in AVC decode mode only.

MFX_PIPE_BUF_ADDR_STATE

	5:0	**Reserved** Access: RO Format: MBZ
56	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Macroblock ILDB StreamOut Buffer Base Address High** Format: GraphicsAddress[47:32] This field is for the upper range of Deblocking Filter Row Store Scratch Address
57	31:15	**Reserved** Access: RO Format: MBZ
	14:13	**Macroblock ILDB StreamOut - Tiled Resource Mode** **For Media Surfaces:**This field specifies the tiled resource mode. Value / Name / Description: 0h / TRMODE_NONE / No tiled resource 1h / TRMODE_TILEYF / 4KB tiled resources 2h / TRMODE_TILEYS / 64KB tiled resources 3h / Reserved /
	12:11	**Reserved** Access: RO Format: MBZ
	10	**Macroblock ILDB StreamOut Buffer - Memory Compression Mode** Value / Name: 0 / Reserved **[Default]**
	9	**Macroblock ILDB StreamOut Buffer - Memory Compression Enable** Value / Name: 0 / Compression Disable **Programming Notes** This surface is linear surface. This bit must be set to "0" since only TileY/TileYf/TileYs surface is allowed to be compressed
	8:7	**Macroblock ILDB StreamOut Buffer - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. Value / Name: 00b / Highest priority

		01b: Second highest priority 10b: Third highest priority 11b: Lowest priority
	6:0	**Macroblock ILDB StreamOut Buffer - Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).
58	31:6	**Second Macroblock ILDB StreamOut Buffer Base Address** Format: GraphicsAddress[31:6] 64 byte aligned buffer. Specifies the 64 byte aligned buffer address for writing MB ILDB parameter per MB to memory when Debocker streamout enable is set in the MFX_PIPE_MODE_SELECT Command. The ildb MB control parameters are written by HW at the end of each decoding MB. Only AVC edge information is being streamed out. It is used in AVC decode mode only.
	5:0	**Reserved** Access: RO Format: MBZ
59	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Second Macroblock ILDB StreamOut Buffer Base Address High** Format: GraphicsAddress[47:32] This field is for the upper range of Second Macroblock ILDB StreamOutBuffer Base Address.
60	31:15	**Reserved** Access: RO Format: MBZ
	14:13	**Second Macroblock ILDB StreamOut Buffer - Tiled Resource Mode** **For Media Surfaces:**This field specifies the tiled resource mode. Value / Name / Description: 0h / TRMODE_NONE / No tiled resource 1h / TRMODE_TILEYF / 4KB tiled resources 2h / TRMODE_TILEYS / 64KB tiled resources 3h / Reserved /
	12:11	**Reserved** Access: RO

		Format: MBZ
	10	**Second Macroblock ILDB StreamOut Buffer - Memory Compression Mode** Value: 0 — Name: Reserved **[Default]**
	9	**Second Macroblock ILDB StreamOut Buffer - Memory Compression Enable** Value: 0 — Name: Compression Disable **Programming Notes**: This surface is linear surface. This bit must be set to "0" since only TileY/TileYf/TileYs surface is allowed to be compressed
	8:7	**Second Macroblock ILDB StreamOut Buffer - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.
	6:0	**Second Macroblock ILDB StreamOut Buffer - Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).
61	31	**Reference Picture 15 - Compression Type** This field is valid only when memory compression is enabled. Value: 0 — Name: Media Compression Enabled **[Default]** Value: 1 — Name: Render Compression Enabled
	30	**Reference Picture 15 - Memory Compression Enable** Value: 0 — Name: Compression Disable Value: 1 — Name: Compression Enable
	29	**Reference Picture 14 - Compression Type** Value: 0 — Name: Media Compression Enabled **[Default]** Value: 1 — Name: Render Compression Enabled
	28	**Reference Picture 14 - Memory Compression Enable** Value: 0 — Name: Compression Disable Value: 1 — Name: Compression Enable

27 **Reference Picture 13 - Compression Type**

Value	Name
0	Media Compression Enabled **[Default]**
1	Render Compression Enabled

26 **Reference Picture 13 - Memory Compression Enable**

Value	Name
0	Compression Disable
1	Compression Enable

25 **Reference Picture 12 - Compression Type**

Value	Name
0	Media Compression Enabled **[Default]**
1	Render Compression Enabled

24 **Reference Picture 12 - Memory Compression Enable**

Value	Name
0	Compression Disable
1	Compression Enable

23 **Reference Picture 11 - Compression Type**

Value	Name
0	Media Compression Enabled **[Default]**
1	Render Compression Enabled

22 **Reference Picture 11 - Memory Compression Enable**

Value	Name
0	Compression Disable
1	Compression Enable

21 **Reference Picture 10-Compression Type**

Value	Name
0	Media Compression Enabled **[Default]**
1	Render compression enabled

20 **Reference Picture 10 - Memory Compression Enable**

Value	Name
0	Compression Disable
1	Compression Enable

	19	**Reference Picture 9 - Compression Type** **Value** / **Name** 0 – Media Compression Enabled **[Default]** 1 – Render Compression Enabled
	18	**Reference Picture 9 - Memory Compression Enable** **Value** / **Name** 0 – Compression Disable 1 – Compression Enable
	17	**Reference 8 - Compression Type** **Value** / **Name** 0 – Media Compression Enabled **[Default]** 1 – Render Compression Enabled
	16	**Reference Picture 8 - Memory Compression Enable** **Value** / **Name** 0 – Compression Disable 1 – Compression Enable
	15	**Reference Picture 7 - Compression Type** **Value** / **Name** 0 – Media Compression Enabled **[Default]** 1 – Render Compression Enabled
	14	**Reference Picture 7 - Memory Compression Enable** **Value** / **Name** 0 – Compression Disable 1 – Compression Enable
	13	**Reference Picture 6 - Compression Type** **Value** / **Name** 0 – Media Compression Enabled **[Default]** 1 – Render Compression Enabled
	12	**Reference Picture 6 - Memory Compression Enable** **Value** / **Name** 0 – Compression Disable 1 – Compression Enable

	11	**Reference Picture 5 -Compression Type** **Value** / **Name** 0 – Media Compression Enabled **[Default]** 1 – Render Compression Enabled
	10	**Reference Picture 5 - Memory Compression Enable** **Value** / **Name** 0 – Compression Disable 1 – Compression Enable
	9	**Reference PIcture 4 - Compression Type** **Value** / **Name** 0 – Media Compression Enabled **[Default]** 1 – Render Compression Enabled
	8	**Reference Picture 4 - Memory Compression Enable** **Value** / **Name** 0 – Compression Disable 1 – Compression Enable
	7	**Reference PIcture 3 - Compression Type** **Value** / **Name** 0 – Media Compression Enabled **[Default]** 1 – Render Compression Enabled
	6	**Reference Picture 3 - Memory Compression Enable** **Value** / **Name** 0 – Compression Disable 1 – Compression Enable
	5	**Reference Picture 2 - Compression Type** **Value** / **Name** 0 – Media Compression Enabled **[Default]** 1 – Render Compression Enabled
	4	**Reference Picture 2 - Memory Compression Enable** **Value** / **Name** 0 – Compression Disable 1 – Compression Enable

MFX_PIPE_BUF_ADDR_STATE

	3	**Reference Picture 1 - Compression Type** **Value** / **Name** 0 – Media Compression Enabled **[Default]** 1 – Render Compression Enabled
	2	**Reference Picture 1 - Memory Compression Enable** **Value** / **Name** 0 – Compression Disable 1 – Compression Enable
	1	**Reference Picture 0 - Compression Type** **Value** / **Name** 0 – Media Compression Enabled **[Default]** 1 – Render Compression Enabled
	0	**Reference Picture 0 - Memory Compression Enable** **Value** / **Name** 0 – Compression Disable 1 – Compression Enable
62	31:6	**Scaled Reference Surface Base Address** Format: GraphicsAddress[31:6] Specifies the 64 byte aligned down scaled reference frame buffer addresses that needs to be used by the PAK down-scaler to write the down scaled pixels. Only the luma pixels will be downscaled and written to the surface
	5:0	**Reserved** Access: RO Format: MBZ
63	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Scaled Reference Surface Base Address High** Format: GraphicsAddress[47:32] This field is for the upper range of Scaled Reference Surface Base Address.
64	31:15	**Reserved** Access: RO Format: MBZ

MFX_PIPE_BUF_ADDR_STATE

	14:13	**Scaled Reference Surface - Tiled Resource Mode** For Media Surfaces:This field specifies the tiled resource mode **Value** / **Name** / **Description** 0h / TRMODE_NONE / No tiled resource 1h / TRMODE_TILEYF / No tiled resource 2h / TRMODE_TILEYS / No tiled resource 3h / Reserved /
	12:11	**Reserved** Access: RO Format: MBZ
	10	**Scaled Reference Surface - Render Compression Enable** **Value** / **Name** 0 / Disable **[Default]** 1 / Enable
	9	**Scaled Reference Surface - Memory Compression Enable** Format: Enable Memory compression shouldn't be enabled for this surface.
	8:7	**Scale Reference Surface - Arbitration Priority Control** This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. **Value** / **Name** 00b / Highest Priority 01b / Second Highest Priority 10b / Third Highest Priority 11b / Lowest Priority
	6:0	**Scaled Reference Surface - Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).
65	31:6	**SliceSize StreamOut Data Destination Base Address** Format: GraphicsAddress[31:6] Specifies the 64 byte aligned Slice Size streamout surface address. Here slice sizes are written out. This surface can be used to determine the slice start location.
	5:0	**Reserved** Access: RO Format: MBZ

MFX_PIPE_BUF_ADDR_STATE

66 — 31:16 — Reserved

Access:	RO
Format:	MBZ

66 — 15:0 — SliceSize StreamOut Data Destination Base Address High

Format:	GraphicsAddress[47:32]

This field is for the upper range of Slice Size Streamout Surface Base Address.

67 — 31:15 — Reserved

Access:	RO
Format:	MBZ

67 — 14:13 — SliceSize StreamOut Data Destination - Tiled Resource Mode

For Media Surfaces: This Surface is never tiled.

Value	Name	Description
0h	TRMODE_NONE	No tiled resource
1h	TRMODE_TILEYF	No tiled resource
2h	TRMODE_TILEYS	No tiled resource
3h	Reserved	

67 — 12:11 — Reserved

Access:	RO
Format:	MBZ

67 — 10 — SliceSize StreamOut Data Destination - Memory Compression Mode

Value	Name
0	Reserved **[Default]**

67 — 9 — SliceSize StreamOut Data Destination - Memory Compression Enable

Format:	Enable

Memory compression is never enabled for this surface

67 — 8:7 — SliceSize StreamOut Data Destination - Arbitration Priority Control

This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.

Value	Name
00b	Highest Priority
01b	Second Highest Priority
10b	Third Highest Priority
11b	Lowest Priority

67 — 6:0 — SliceSize StreamOut Data Destination - Memory Object Control State

Format:	MEMORY_OBJECT_CONTROL_STATE

Specifies the memory object control state for this surface and the associated Auxiliary surface (if any).

MFX_PIPE_MODE_SELECT

MFX_PIPE_MODE_SELECT

Source: VideoCS

Length Bias: 2

Specifies which codec and hardware module is being used to encode/decode the video data, on a per-frame basis.
The MFX_PIPE_MODE_SELECT command specifies which codec and hardware module is being used to encode/decode the video data, on a per-frame basis. It also configures the hardware pipeline according to the active encoder/decoder operating mode for encoding/decoding the current picture. Commands issued specifically for AVC and MPEG2 are ignored when VC1 is the active codec.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_COMMON Format: OpCode
	26:24	**Opcode** Default Value: 0h MFX_COMMON_STATE Format: OpCode
	23:21	**SubOpA** Default Value: 0h Format: OpCode
	20:16	**SubOpB** Default Value: 0h MFX_PIPE_MODE_SELECT Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Format: =n Value: 3h — Name: DWORD_COUNT_n **[Default]** — Description: Excludes DWord (0,1)
1	31:24	**AES Control** Format: AES_CONTROL

23:19 **Reserved**

Access:	RO
Format:	MBZ

18 **Extended stream out enable**

Format:	U1

This bit can be set only when VDEnc_Mode is set.
When this bit is set and MB stream out is enabled, per MB 1CL of data is streamed out. The actual contents of the stream out are listed in Media VDBOX > Encoder VDEnc mode StreamOut Data Structure Definition.
When this bit is not set, per MB CL data is streamed out. The actual contents of the stream out are listed in Media VDBOX > Encoder StreamOut Mode Data Structure Definition.

17 **Decoder Short Format Mode**

For IT mode, this bit must be 0.

Value	Name	Description
0	Short Format Driver Interface **[Default]**	AVC/VC1/MVC/VP8 Short Format Mode is in use **Note: There is no Short Format for VP8 yet, so this field must be set to 1 for VP8.**
1	Long Format Driver Interface	AVC/VC1/MVC/VP8 Long Format Mode is in use.

16:15 **Decoder Mode select**

Each coding standard supports two entry points: VLD entry point and IT (IDCT) entry point. This field selects which one is in use. This field is only valid if Codec Select is 0 (decoder).

Value	Name	Description
0h	VLD Mode	All codec minimum must support this mode Configure the MFD Engine for VLD ModeNote: All codec minimum must support this mode
1h	IT Mode	Configure the MFD Engine for IT ModeNote: Only VC1 and MPEG2 support this mode
2h	Deblocker Mode	Configure the MFD Engine for Standalone Deblocker Mode. Require streamout AVC edge control information from preceeding decoding pass.
3h	Interlayer Mode	Configure the MFX Engine for standalone interlayer upsampling for motion info, residual and reconstructed pixel. Require information being streamout from the preceding encoding and decoding pass of a reference layer.>

14 **Standalone VDEnc_Mode Enable**

Format:	Enable

This field indicates to PAK if this is standalone VDEnc mode. This is primarily a validation mode.

Value	Name
0h	VDEnc+PAK
1h	PAK Only

13 **VDEnc_Mode**

This field indicates if PAK is working in legacy MBEnc mode or the VDEnc mode.

Value	Name	Description
0h	MBEnc mode	PAK is working in legacy mode
1h	VDEnc mode	PAK is working in VDEnc mode

12 **Deblocker Stream-Out Enable**

This field indicates if Deblocker information is going to be streamout during VLD decoding. For AVC, it is needed to enable the deblocker streamout as the AVC Disable_DLKFilterIdc is a slice level parameters. Driver needs to determine ahead of time if at least one slice of the current frame/ has deblocker ON.

Value	Name	Description
0h	Disable	Disable streamout of deblocking control information for standalone deblocker operation. It needs other fields to determine one or two deblocking surface streamout (Post Deblocking Output Enable, Pre Deblocking Output Enable, interlayer idc and regular deblock idc).
1h	Enable	

11 **Pic Error/Status Report Enable.**

This field control whether the error/status reporting is enable or not.0: Disable1: Enable In decoder modes: Error reporting is written out once per frame. The Error Report frame ID listed in DW3 along with the VLD/IT error status bits are packed into one cache and written to the "Decoded Picture Error/Status Buffer address" listed in the MFX_PIPE_BUF_ADDR_STATE Command. Note: driver shall program different error buffer addresses between pictures; otherwise, hardware might overwrite previous written data if driver does not read it fast enough. In encoder modes: Not used

Please refer to "Media VDBOX -> Video Codec -> Other Codec Functions -> MFX Error Handling -> Decoder" session for the output format.

Value	Name
0h	Disable
1h	Enable

10 **Stream-Out Enable**

This field controls whether the macroblock parameter stream-out is enabled during VLD decoding for transcoding purpose.

Value	Name
0h	Disable
1h	Enable

Programming Notes
In decoder modes: The Stream-Out feature is added to support transcoding. While decoding the input compressed stream, selected decoded information may be used by the encoder for re-compression. In encoder modes: This feature used to perform dynamic Multipass of PAK for conformance purpose. Also it provides feedback to host (ENC) for future needs. Software can use this bit to disable writing PAK steam data to the streamout buffer for last pass of frame in

PAK. Thus, save memory bandwidth.

9 Post Deblocking Output Enable (PostDeblockOutEnable)

This field controls the output write for the reconstructed pixels AFTER the deblocking filter. In MPEG2 decoding mode, if this is enabled, VC1 deblocking filter is used.

Value	Name
0h	Disable
1h	Enable

8 Pre Deblocking Output Enable (PreDeblockOutEnable)

This field controls the output write for the reconstructed pixels BEFORE the deblocking filter.

Value	Name
0h	Disable
1h	Enable

7 Scaled Surface Enable

This field indicates if the scaled surface is enabled. This field enables the 4x HME down scalar of the reconstructed image. Only supported for AVC and VP8 formats.

Value	Name
0h	Disable
1h	Enable

6 Frame Statistics StreamOut Enable

This field controls the frame level statistics streamout from the PAK.

Note: This field needs to be always "Enabled" in VD_Enc mode. In case of non-VDEnc mode, this can be used to control the frame statistics output from the PAK.

Value	Name
0h	Disable
1h	Enable

5 Stitch Mode

Exists If:	//CodecSel=Encode and StandardSel=AVC

Value	Name	Description
0h	Not in stitch mode	
1h	In the special stitch mode	This mode can be used for any Codec as long as bitfield conditions are met.

4 Codec Select

Value	Name	Description
0h	Decode	
1h	Encode	Valid only if StandardSel is AVC, MPEG2)

MFX_PIPE_MODE_SELECT

Standard Select

Value	Name	Description
0000b	MPEG2	
0001b	VC1	
0010b	AVC	Covers both AVC and MVC
0011b	JPEG	
0101b	VP8	Decoder, Encoder
0110b	Reserved	
0111b	Reserved	
1111b	UVLD	SW decoder w/ embedded micro-controller and co-processor

Reserved

Access:	RO
Format:	MBZ

Pic Status/Error Report ID

Exists If:	//Decoder Mode Only
Format:	U32

In decoder modes: Error reporting is written out once per frame. This field along with the VLD error status bits are packed into one cache and written to the memory location specified by "Decoded Picture Error/Status Buffer address" listed in the MFX_PIPE_BUF_ADDR_STATE Command.

Value	Name	Description
0h	32-bit unsigned	Unique ID Number
1h	Reserved	

Reserved

Access:	RO
Format:	MBZ

MFX_QM_STATE

MFX_QM_STATE

Source: VideoCS

Length Bias: 2

This is a common state command for AVC encoder modes. For encoder, it represents both the forward QM matrices as well as the decoding QM matrices. This is a Frame-level state. Only Scaling Lists specified by an application are being sent to the hardware. The driver is responsible for determining the final set of scaling lists to be used for decoding the current slice, based on the AVC Spec Table 7-2 (Fall-Back Rules A and B).In MFX AVC PAK mode, PAK needs both forward Q scaling lists and IQ scaling lists. The IQ scaling lists are sent as in MFD in raster scan order. But the Forward Q scaling lists are sent in column-wise raster order (column-by-column) to simplify the H/W. Driver will perform all the scan order conversion for both ForwardQ and IQ.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_MULTI_DW Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MFX_COMMON_STATE Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 7h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 20h Excludes DWord (0,1) Format: =n
1	31:2	**Reserved** Access: RO Format: MBZ

MFX_QM_STATE

1:0 AVC

Exists If:	//AVC- Decoder Only

For AVC QM Type: This field specifies which Quantizer Matrix is loaded.

Value	Name
0	AVC_4x4_Intra_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)
1	AVC_4x4_Inter_MATRIX, (Y-4DWs, Cb-4DWs, Cr-4DWs, reserved-4DWs)
2	AVC_8x8_Intra_MATRIX
3	AVC_8x8_Inter_MATRIX

1:0 MPEG2

Exists If:	//MPEG2- Decoder Only

For MPEG2 QM Type: This field specifies which Quantizer Matrix is loaded.

Value	Name
0	MPEG_INTRA_QUANTIZER_MATRIX
1	MPEG_NON_INTRA_QUANTIZER_MATRIX
2-3	Reserved

1:0 JPEG

Exists If:	//JPEG- Encoder Only

For JPEG QM Type:This field specifies which Quantizer Matrix is loaded.

Value	Name
0	JPEG_Luma_Y_QUANTIZER_MATRIX (or R)
1	JPEG_Chroma_Cb_QUANTIZER_MATRIX (or G)
2	JPEG_Chroma_Cr_QUANTIZER_MATRIX (or B)

Programming Notes
For JPEG encoder, each quantization element presents 16-bit 1/QM[i][j].In RGB encoding, because the order input image components can be RGB, GBR, BGR, YUV, the value 0 is used for the first image component, the value 1 is used for the second image component, and the value 2 is used for the third image component.

2..33 1023:0 Forward Quantizer Matrix

The format of a Quantizer Matrix is an 8x8 matrix in raster order. Each element is an unsigned byte.

MFX_STATE_POINTER

MFX_STATE_POINTER

Source: VideoCS

Length Bias: 2

The MFX_STATE_POINTER command, issued at picture level, is used to set up the indirect pointers for VCS to fetch all the MFX states (Image state, Slice state, etc.) needed for the encoding/decoding process in PAK/IT mode. The encoding/decoding states are presented by state commands, which are grouped into separate sets (picture level, slice level, etc.), and each is stored in its own memory buffer referred by an indirect state pointer. The content of each indirect state buffer is a list of MFX state commands with no special format requirements. The sequence of commands in each indirect state buffer is terminated by a MI_BATCH_BUFFER_END command(acts as the last command marker). Therefore, indirect state buffers can have different and variable length of command sequences.

The indirection is designed to facilitate context switching in the middle of a codec operation. The smallest granularity of interruption is designed to be at a completed MB row in AVC/VC1/MPEG2 IT and AVC PAK operating modes as well as in VC1/MPEG2 VLD mode. There is no support for context switch in AVC VLD mode. Hardware supports up to 4 separate indirect state pointers, allowing software to manage the grouping of state commands. During context switch, hardware restores (re-issues) the latest version of each indirect state pointer, if present.

MFX_STATE_POINTER command can only program one indirect state pointer at a time. MI_FLUSH will invalidate all indirect state buffer pointers inside VCS.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFX_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MFX_COMMON_STATE Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 6h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ

MFX_STATE_POINTER

	11:0	**DWord Length** Default Value: 0h DWORD_COUNT_n Format: =n
1	31:5	**State Pointer** Format: GeneralStateOffset[31:5] Specifies the 32-byte aligned address of an Indirect State Buffer. This pointer is relative to the General State Base Address.
	4:2	**Reserved** Access: RO Format: MBZ
	1:0	**State Pointer Index** Specifies one of the four indirect state pointers to program.

Value	Name	Description
00b		indirect state pointer 0 (image state)
01b		indirect state pointer 1 (slice state)sc
10b		indirect state pointer 2
11b		indirect state pointer 3

MFX_STITCH_OBJECT

MFX_STITCH_OBJECT

Source: VideoCS

Length Bias: 2

The MFC_STITCH_OBJECT command is used when stitch-enabled is set to 1, while CodecSel and Standard Sel are set to ENCODE and AVC, respectively. This command is used, for example, to stitch multiple bitstreams to form a transport stream.
It is a variable length command as the data to be inserted are presented as either inline data and/or indirect data of this command. Multiple insertion commands can be issued back to back in a series. It is host software's responsibility to make sure their corresponding data will properly stitch together to forma valid output.
Hardware keeps track of an output bitstream buffer current byte position and the associated next bit insertion position index. Context switch interrupt is not supported by this command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFC_STITCH_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 0h MFX_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 2h Format: OpCode
	20:16	**SubOpcode B** Default Value: Ah Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: [0h, FFFh] Excludes DWord (0,1) = Variable Length in DW (>= 3) Format: =n If it is 3, it indicates the absent of inline data.

MFX_STITCH_OBJECT

1	31:18	**Reserved** Access: RO Format: MBZ
	17:16	**Source Data Starting Byte Offset** Source Data Starting Byte Position within the very first inline DW.
	15:14	**Reserved** Access: RO Format: MBZ
	13:8	**Source Data Ending Bit Inclusion** Source Data to be included in the very last inline DW. Follows the MSBit is the upper bit of each byte within the DW. The lower byte is actually processed first. For example, SrCDataEndingBitInclusion =9, bit 7:0 and bit 15 are included as valid header data. Value: [1,32] — Name:
	7:4	**Reserved**
	3	**Reserved**
	2	**Last Source Header Data Insert Command Flag** To process a series of consecutive insertion commands, this flag (=1) indicates the current command is the last 'header' insertion in the series. In CABAC, hardware must perform the "1" insert for byte align for Slice Header before Slice Data comes in in the next PAK-OBJECT command. In CAVLC, hardware ignores this bit.
	1	**Last Destination Data Insert Command Flag** THIS FIELD MUST BE THE SAME AS Last Source Header Data Insert Command Flag No more insertion command and no more PAK-OBJECT command follows. Flush data out to memory
	0	**Reserved**
2	31:19	**Reserved** Access: RO Format: MBZ
	18:0	**Indirect Data Length** Format: U19 This field provides the length in bytes of the indirect data. A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect Data Start Address field is ignored. This field must have the same alignment as the Indirect Object Data Start Address.
3	31:0	**Indirect Data Start Address** Format: GraphicsAddress[31:0] This field specifies the Graphics Memory starting address of the data to be loaded into the kernel for processing. This pointer is relative to the MFX Indirect Bitstream Object Base Address. Hardware ignores this field if indirect data is not present.

MFX_STITCH_OBJECT		
4..n	31:0	**Insert Data PayLoad** Inline data to be inserted to the output bitstream buffer

MFX_SURFACE_STATE

MFX_SURFACE_STATE

Source:	VideoCS
Length Bias:	2

Description

This command is common for all encoding/decoding modes, to specify the uncompressed YUV picture (i.e. destination surface) or intermediate streamout in/out surface (e.g. coefficient/residual) (field, frame or interleaved frame) format for reading and writing:

- Uncompressed, original input picture to be encoded
- Reconstructed non-filtered/filtered display picturec(becoming reference pictures as well for subsequent temporal inter-prediction)

Since there is only one media surface state being active during the entire encoding/decodingprocess, all the uncompressed/reconstructed pictures are defined to have the same surface state. For each media object call (decoding or encoding) to distinguish among them, a surface ID is added to specify for each type of surface. The primary difference among picture surface states is their individual programmed base addresses, which are provided by other state commands and not included in this command. MFX engine is making the association of surface states and corresponding buffer base addresses.
MFX engine currently supports only one media surface type for video and that is the NV12 (Planar YUV420 with interleaved U (Cb) and V (Cr). For optimizing memory efficiency based on access patterns, only TileY is supported. For JPEG decoder, only IMC1 and IMC3 are supported. Pitch can be wider than the Picture Width in pixels and garbage will be there at the end of each line. The following describes all the different formats that are supported and not supported in MFX :

- NV12 - 4:2:0 only;UV interleaved; Full Pitch, U and V offset is set to 0 (the only format supported forvideo codec); vertical UV offset is MB aligned; UV xoffsets = 0. JPEG does not supportNV12 format because non-interleave JPEG has performance issue with partial write (in interleaved UV format)
- IMC 1 & 3 - Full Pitch, U and V are separate plane; (JPEG only; U plane + garbage first in full pitch followed by V plane + garbage in fullpitch). U and V vertical offsets are block aligned; U and V xoffset = 0; there is no gap between Y, U and V planes. IMC1 and IMC3 are different by a swap of U and V. This is the only format supported in JPEG for all video subsampling types (4:4:4, 4:2:2 and 4:2:0)
- We are not supporting IMC 2 & 4 - Full Pitch, U and V are separate plane (JPEG only; Uplane first in full pitch followed by V plane in full pitch - U and V plane are side-by-side). U and V vertical offsets are 16-pixel aligned; V xoffset is half-pitch aligned; U xoffset is 0; there is no gap between Y, U and V planes. IMC2 and IMC4 are different by a swap of U and V.
- We are not supporting YV12 - half pitch for each U and Vplane, and separate planes for Y, U and V (U plane first in half pitch followed by Vplane in half pitch). For YV12, U and V vertical offsets are block aligned; U and Vxoffset = 0; there is no gap between Y, U and V planes

Note that the following data structures are not specified through the media surface state

- 1D buffers for row-store and other miscellaneous information.
- 2D buffers for per-MB data-structures (e.g. DMV biffer, MB info record, ILDB Control and Tcoeff/Stocoeff).

MFX_SURFACE_STATE

This surface state here is identical to the Surface State for deinterlace and sample_8x8messages described in the Shared Function Volume.
For non pixel data, such as row stores, indirect data (Compressed Slice Data, AVC MV record, Coeff record and AVC ILDB record) and streamin/out and output compressed bitstream, a linear buffer is employed. For row stores, the H/W is designed to guarantee legal memory accesses (read and write). For the remaining cases, indirect object base address, indirect object address upper bound, object data start address (offset) and object data length are used to fully specified their corresponding buffer. This mechanism is chosen over the pixel surface type because of their variable record sizes.
All row store surfaces are linear surface. Their addresses are programmed in Pipe_Buf_Base_State orBsp_Buf_Base_Addr_State

This surface state here is identical to the Surface State for deinterlace and sample_8x8messages described in the Shared Function Volume and Sampler Chapter.

Programming Notes

VC1 I picture scaling: Even though VC1 allows I reconstructed picture scaling (via RESPIC), as such scaling is only allowed at I picture. All subsequent P (and B) pictures must have the same picture dimensions with the preceding I picture. Therefore, all reference pictures for P or B picture can share the same surface state with the current P and B picture. Note : H/W is not processing RESPIC. Application is no longer expecting intel decoder pipeline and kernel to perform this function, it is going to be done in the video post-processing scaler or display controller scale as a separate step and controller.

All video codec surfaces must be NV12 Compliant, except JPEG. U/V vertical must be MB aligned for all video codec (further contrained for field picture), but JPEG can be block aligned. All video codec and JPEG uses Tiled - Y format only, for uncompressed pixel surfaces.

Even for JPEG planar 420 surface, application may provide only 1 buffers, but there is still only one single surface state for all of them. If IMC equal to 1, 2, 3 or 4, U and V have the pitch same as Y. And U and V will have different offset, each offset is block aligned.

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	3h PARALLEL_VIDEO_PIPE
		Format:	OpCode
	28:27	**Pipeline**	
		Default Value:	2h MFX_COMMON
		Format:	OpCode
	26:24	**Opcode**	
		Default Value:	0h MFX_COMMON_STATE
		Format:	OpCode
	23:21	**SubOpA**	
		Default Value:	0h
		Format:	OpCode

MFX_SURFACE_STATE

	20:16	**SubOpB** Default Value: 1h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Format: =n **Value** / **Name** / **Description** 4h / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)
1	31:4	**Reserved** Access: RO Format: MBZ
	3:0	**Surface Id** Format: U4 **Value** / **Name** / **Description** 0100b / Source Input Picture (encoder) / 8-bit uncompressed data 0101b / Reconstructed Scaled Reference Picture / 8-bit data
2	31:18	**Height** Format: U14-1 This field specifies the height of the Picture in units of pixels/residuals. For PLANAR surface formats, this field indicates the height of the Y (luma) plane. Note : Video Codecs must program less than and equal to 4K.(In future, it will be ideal to have this field define in a WORD boundary.)AVC - multiple of 2 MB rows for field pictureVC1 - mulitple of 4 pixels for field pictureMPEG2 - multiple of 2 MB rows for field picJPEG - mulitple of integral MCU (8 or 16 pixels) per picture **Value** / **Name** / **Description** [0,16383] / / representing heights [1,16384] **Programming Notes** • For AVC : For frame picture is a multiple of 16; for field picture is a multiple of 32 • For VC1 : For progressive frames, the frame height and frame width is a multiple of 2 pixels. For interlaced frames, the frame height shall be a multiple of 4 pixels, and its width is a multiple of 2 pixels, based on a PLANAR_420 surface.
	17:4	**Width** Format: U14-1

This field specifies the width of the Picture in units of pixels/residuals. For PLANAR surface formats, this field indicates the width of the Y (luma) plane.

Value	Name	Description
[0,16383]		representing widths [1,16384]

Programming Notes

- The Width specified by this field multiplied by the pixel size in bytes must be less than or equal to the surface pitch (specified in bytes via the Surface Pitch field).
- Width (field value + 1) must be a multiple of 2 for PLANAR_420,
- MFX HW does not use this field, the picture width is read from IMG State instead, because this field may not equal to the actual picture width. This field is used by the KMD to allocate surface in GTT.

3:2 **Reserved**

Access:	RO
Format:	MBZ

1:0 **Cr(V)/Cb(U) Pixel Offset V Direction**

Format:	U0.2

Specifies the distance to the U/V values with respect to the even numbered Y channels in the V direction

Programming Notes

This field is ignored for all formats except PLANAR_420_8

3 31:28 **Surface Format**

Format:	U4

Specifies the format of the surface. All of the Y and G channels will use table 0 and all of the Cr/Cb/R/B channels will use table 1.Usage: For 420 planar YUV surface, use 4; for monochrome surfaces, use 12. For monochrome surfaces, hardware ignores control fields for Chroma planes. This field must be set to 4 - PLANAR_420_8, or 12 - Y8_UNORMNot used for MFX, and is ignored. But for JPEG decoding, this field should be programmed to the same format as JPEG_PIC_STATE. For video codec, it should set to 4 always.

Value	Name	Description
0	YCRCB_NORMAL	
1	YCRCB_SWAPUVY	
2	YCRCB_SWAPUV	
3	YCRCB_SWAPY	
4	PLANAR_420_8	(NV12, IMC1,2,3,4, YV12)
5	PLANAR_411_8	Deinterlace Only
6	PLANAR_422_8	Deinterlace Only
7	STMM_DN_STATISTICS	Deinterlace Only
8	R10G10B10A2_UNORM	Sample_8x8 Only

9	R8G8B8A8_UNORM	Sample_8x8 Only
10	R8B8_UNORM (CrCb	Sample_8x8 Only
11	R8_UNORM (Cr/Cb)	Sample_8x8 Only
12	Y8_UNORM	Sample_8x8 Only

27 **Interleave Chroma**

Format:	Enable

This field indicates that the chroma fields are interleaved in a single plane rather than stored as two separate planes. This field is only used for PLANAR surface formats. For AVC/VC1/MPEG VLD and IT modes : set to Enable to support interleave U/V only. For JPEG : set to Disable for all formats (including 4:2:0) - because JPEG does not support NV12. (This field is needed only if JPEG will support NV12; otherwise is ignored.)

Value	Name
1	Enable
0	Disable

26:20 **Reserved**

Access:	RO
Format:	MBZ

19:3 **Surface Pitch**

Format:	U17-1

This field specifies the surface pitch in (#Bytes).

Value	Name
[0,131071]	

Programming Notes

For tiled surfaces, the pitch must be a multiple of the tile width (i.e.128 bytes aligned). If Half Pitch for Chroma is set, this field must be a multiple of two tile widths for tiled surfaces, or a multiple of 2 bytes for linear surfaces. For Y-tiled surfaces: Range = [127, 131071] to [128B,128KB] = [1 tile, 1024 tiles]

For TileYF and TileYS surfaces, the range is dependent on the Cu parameter (refer to Memory Data Formats section for the definition of the Cu parameter depending on the case).
The range in bytes is [2^{Cu}-1, 131071] -> [(2^{Cu})B, 128KB] = [1 tile, 128KB/($2^{Cu)\ tiles}$]
The field specifies the surface pitch in (#Bytes - 1)

If Media Memory Compression is enabled, the following max pitch size restriction must be honored. For larger resolution, Media Memory compression Must be disabled.

Tiling Mode	Pixel Format	Max Frame Width (bytes)	Max Frame Width (pixels)	Max Pitch (bytes)
Legacy 4K	8bpp	16k	16k	16k + 127
	16bpp	16k	8k	16k + 127
	32bpp	16k	4k	16k + 127

	64bpp	16k	2k	16k + 127
	128bpp	16k	1k	16k + 127
TileYF	8bpp	8k	8k	8k + 63
	16bpp	16k	8k	16k + 127
	32bpp	16k	4k	16k + 127
	64bpp	16k	2k	16k + 255
	128bpp	16k	1k	16k + 255
TileYS	8bpp	16k	16k	16k + 255
	16bpp	16k	8k	16k + 511
	32bpp	16k	4k	16k + 511
	64bpp	16k	2k	16k + 1023
	128bpp	16k	1k	16k + 1023

2 **Half Pitch for Chroma**

Format:	Enable

(This field must be set to Disable)This field indicates that the chroma plane(s) will use a pitch equal to half the value specified in the Surface Pitch field. This field is only used for PLANAR surface formats. This field is ignored by MFX (unless we support YV12)

1 **Tiled Surface**

Format:	Boolean

(This field must be set to TRUE: Tiled)This field specifies whether the surface is tiled. This field is ignored by MFX

Value	Name	Description
0	False	Linear
1	True	Tiled

Programming Notes
Linear surfaces can be mapped to Main Memory (uncached) or System Memory (cacheable, snooped). Tiled surfaces can only be mapped to Main Memory. The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit.

0 **Tile Walk**

Format:	Boolean

(This field must be set to 1: TILEWALK_YMAJOR)This field specifies the type of memory tiling (XMajor or YMajor) employed to tile this surface. See Memory Interface Functions for details on memory tiling and restrictions. This field is ignored when the surface is linear. This field is ignored by MFX. Internally H/W is always treated this set to 1 for all video codec and for JPEG.

Value	Name	Description
0h	XMAJOR	TILEWALK_XMAJOR

1h	YMAJOR	TILEWALK_YMAJOR

Programming Notes

The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit

4 — 31 — Reserved

Access:	RO
Format:	MBZ

4 — 30:16 — X Offset for U(Cb)

Format:	U15

This field specifies the horizontal offset in pixels from the Surface Base Address to the start (origin) of the U(Cb) plane or the interleaved UV plane if Interleave Chroma is enabled. This field is only used for PLANAR surface formats. This field must be set to zero.X Offset for U(Cb) in pixel (This field must be zero for NV12 and IMC 1 and 3)

Programming Notes

For PLANAR_420 and PLANAR_422 surface formats, this field must be zero.

4 — 15 — Reserved

Access:	RO
Format:	MBZ

4 — 14:0 — Y Offset for U(Cb)

Format:	U15

This field specifies the vertical offset in rows from the Surface Base Address to the start (origin) of the U(Cb) plane or the interleaved UV plane if Interleave Chroma is enabled. This field is only used for PLANAR surface formats.

Programming Notes

For PLANAR_420 and PLANAR_422 surface formats, this field must be multiple of 16 pixels - i.e. multiple MBs. For JPEG, this field must be a multiple of 16 pixels.

5 — 31:29 — Reserved

Access:	RO
Format:	MBZ

5 — 28:16 — X Offset for V(Cr)

Format:	U13

This field must be zero for NV12 and IMC 1 and 3

This field specifies the horizontal offset in pixels from the Surface Base Address to the start (origin) of the V(Cr) plane. This field is only used for PLANAR surface formats with Interleave Chroma disabled.

Programming Notes

		For PLANAR_420 and PLANAR_422 surface formats, this field must indicate an even number of pixels.
	15:0	**Y Offset for V(Cr)** Format: U16 This field specifies the vertical offset in rows from the Surface Base Address to the start (origin) of the V(Cr) plane. This field is only used for PLANAR surface formats with Interleave Chroma disabled. This field is ignored by all video codec, only used by JPEG. **Programming Notes** For PLANAR_420 surface formats, this field must be multiple of 16 pixels - i.e. multiple MBs. For JPEG, this field must be a multiple of 16 pixels.

MFX_VC1_DIRECTMODE_STATE

MFX_VC1_DIRECTMODE_STATE

Source: VideoCS

Length Bias: 2

Exists If: //VC1 decoding in VLD modes

This is a picture level command and should be issued only once, even for a multi-slices picture. There is only one DMV buffer for read (when processing a B-picture) and one for write (when processing a P-Picture). Each DMV record is 64 bits per MB, to store the top and bottom field MVs (32-bit MVx,y each).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_VC1_DIRECTMODE_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 2h VC1_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 2h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 0005h Excludes DWord (0,1) Format: =n
1..2	63:0	**Direct MV Write Buffer - Base Address** Format: SplitBaseAddress64ByteAligned This field provides the base address of the DMV write buffer to store the motion vectors decoded in the current picture. It is a private buffer used by the MPR hardware only. Its content is not accessed by software. This buffer must be 64-byte cacheline aligned. The write buffer size is 557,056 bytes for 1 frame. Scalable with frame height, but do not scale with frame width as the hardware assumes frame width (in MBs) fixed at 128 (smallest power of 2 value larger than 120 -

MFX_VC1_DIRECTMODE_STATE		
		1920x1088 screen resolution).This field is only valid for a P picture
3	31:0	**Direct MV Write Buffer - Attributes** Format: **MemoryAddressAttributes**
4..5	63:0	**Direct MV Reference Buffer - Base Address** Format: **SplitBaseAddress64ByteAligned** This field provides the base address of the DMV buffer for reference picture. It is a private buffer used by the MPR hardware only. Its content is not accessed by software. All these buffers must be 64-byte cacheline aligned. This field is only valid for a B picture.
6	31:0	**Direct MV Reference Buffer - Attributes** Format: **MemoryAddressAttributes**

MFX_VC1_PRED_PIPE_STATE

MFX_VC1_PRED_PIPE_STATE

Source: VideoCS

Length Bias: 2

This command is used to set the operating states of the MFD Engine beyond the BSD unit. It is used with both VC1 Long and Short format. Driver is responsible to take the intensity compensation enable signal, the LumScale and the LumShift provided from the VC1 interface, and maintain a history of these values for reference pictures. Together with these three parameters specified for the current picture being decoded, driver will derive and supply the above sets of LumScaleX, LumShiftX and intensity compensation enable (single or double, forward or backward) signals. H/W is responsible to take these state values, and use them to build the lookup table (including the derivation of iScale and iShift) for remapping the reference frame pixels, as well as performing the actual pixel remapping calculations/process.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_VC1_PRED_PIPE_STATE Format: OpCode
	26:24	**Media Command Opcode** Default Value: 2h VC1_COMMON Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 1h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 0004h Excludes DWord (0,1) Format: =n
1	31:16	**Reserved** Access: RO Format: MBZ

MFX_VC1_PRED_PIPE_STATE

	15:14	**vin_intensitycomp_Double_FWDen** Format: U2 for forward reference picture only, to enable top or/and bottom of the reference field enable for single compensation. For frame, may only need one bit. This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	13:12	**vin_intensitycomp_Double_BWDen** Format: U2 for backward reference picture only, no double for backward reference. This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	11:10	**vin_intensitycomp_Single_FWDen** Format: U2 for forward reference picture only, to enable top or/and bottom of the reference field enable for single compensation. For frame, may only need one bit. This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	9:8	**vin_intensitycomp_Single_BWDen** Format: U2 for backward reference picture only, no double for backward reference. This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	7:4	**Reference Frame Boundary Replication Mode** Format: U4 This is a bit field with each bit indicating the corresponding picture's boundary replication mode. Bit 11: reference 3Bit 10: reference 2Bit 9: reference 1Bit 8: reference 00 = progressive frame replication1 = interlace frame replication This field is maintained and provided by driver for both long and short VC1 interface format.
	3:0	**Reserved** Access: RO Format: MBZ

MFX_VC1_PRED_PIPE_STATE

2	31:30	**Reserved** Access: RO Format: MBZ
	29:24	**LumShift2- single - FWD** Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	23:22	**Reserved** Access: RO Format: MBZ
	21:16	**LumShift1 - single - FWD** Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	15:14	**Reserved** Access: RO Format: MBZ
	13:8	**LumScale2 - single - FWD** Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**LumScale1 - Single - FWD** Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.

MFX_VC1_PRED_PIPE_STATE

3	31:30	**Reserved** Access: RO Format: MBZ
	29:24	**LumShift2- double - FWD** Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	23:22	**Reserved** Access: RO Format: MBZ
	21:16	**LumShift1 - double -FWD** Format: U6 This field is maintained andprovided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	15:14	**Reserved** Access: RO Format: MBZ
	13:8	**LumScale2 - double - FWD** Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**LumScale1 - double - FWD** Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.

MFX_VC1_PRED_PIPE_STATE

4	31:30	**Reserved** Access: RO Format: MBZ
	29:24	**LumShift2- single - BWD** Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	23:22	**Reserved** Access: RO Format: MBZ
	21:16	**LumShift1 - single - BWD** Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	15:14	**Reserved** Access: RO Format: MBZ
	13:8	**LumScale2 - single - BWD** Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**LumScale1 - Single - BWD** Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.

MFX_VC1_PRED_PIPE_STATE

5	31:30	**Reserved** Access: RO Format: MBZ
	29:24	**LumShift2- double - BWD** Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	23:22	**Reserved** Access: RO Format: MBZ
	21:16	**LumShift1 - double -BWD** Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	15:14	**Reserved** Access: RO Format: MBZ
	13:8	**LumScale2 - double - BWD** Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**LumScale1 - double - BWD** Format: U6 This field is maintained and provided by driver for both long and short VC1 interface format. And is derived from the intensity compensation enable flag, wBitstreamPCEelement and wBitstreamFcodes parameters provided by the DXVA2 VC1 interface to the driver for each current picture.

MFX_VP8_BSP_BUF_BASE_ADDR_STATE

MFX_VP8_BSP_BUF_BASE_ADDR_STATE

Source: VideoCS

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Video Codec Format: OpCode
	26:24	**Media Command OpCode** Default Value: 4h VP8 Format: OpCode
	23:21	**Sub Opcode A** Default Value: 2h VP8 Common Format: OpCode
	20:16	**Sub Opcode B** Default Value: 3h MFX_VP8_BSP_BUF_BASE_ADDR_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Format: =n Value / Name / Description 000h / Excludes DWord (0,1) **[Default]** / A special case to provide a dummy image state for stitch mode operation. In this case, fields in DW1 which is part of the dummy image state command are ignored by hardware." 008h / / Used for normal encode mode
1..2	63:0	**Frame Header - Base Address** Format: **SplitBaseAddress64ByteAligned** 64 byte aligned, 48-bit Abs. Address StreamIn Surface **Note:**The format is linear vs. tile for better performance.

MFX_VP8_BSP_BUF_BASE_ADDR_STATE

3	31:0	**Frame Header - Attributes** Format: **MemoryAddressAttributes**
4..5	63:0	**Intermediate Buffer - Base Address** Format: **SplitBaseAddress64ByteAligned** 64 byte aligned, 48-bit AbsAddr StreamIn Surface **Note:**The format is linear vs. tile for better performance.
6	31:0	**Intermediate Buffer - Attributes** Format: **MemoryAddressAttributes**
7..14	255:0	**Intermediate Buffer Partition Offset** **Programming Notes** All **Intermediate Buffer Partition-[i] Offset** (i = 1 to 8) and **Intermediate Buffer Max Size** need to be cacheline aligned (64Byte aligned).
15	31:0	**Intermediate Buffer Max Size** Format: U32
16..17	63:0	**Final Frame - Base Address** Format: **SplitBaseAddress64ByteAligned** 64 byte aligned, 48-bit AbsAddr StreamIn Surface **Note:**The format is linear vs. tile for better performance.
18	31:0	**Final Frame - Attributes** Format: **MemoryAddressAttributes**
19	31:6	**Reserved** Access: RO Format: MBZ
	5:0	**Final Frame Byte Offset** Format: U6 Specify byte offset within a 64-byte cacheline where the bitstream should be inserted at.
20..21	63:0	**Streamout - Base Address** Format: **SplitBaseAddress64ByteAligned** 64 byte aligned,48-bit AbsAddr StreamIn Surface **Note:**The format is linear vs. tile for better performance.
22	31:0	**Streamout - Attributes** Format: **MemoryAddressAttributes**

MFX_VP8_BSP_BUF_BASE_ADDR_STATE

23..24	63:0	**Coeff Probs StreamIn Surface - Base Address** Format: **SplitBaseAddress64ByteAligned** 64 byte aligned, 48-bit AbsAddr StreamIn Surface **Note:**The format is linear vs. tile for better performance.
25	31:0	**Coeff Probs StreamIn Surface - Attributes** Format: **MemoryAddressAttributes**
26..27	63:0	**Token Statistics Surface - Base Address** Format: **SplitBaseAddress64ByteAligned** 64 byte aligned, 48-bit Abs. Address StreamIn Surface **Note:**The format is linear vs. tile for better performance.
28	31:0	**Token Statistics Surface - Attributes** Format: **MemoryAddressAttributes**
29..30	63:0	**MPC RowStore Surface - Base Address** Format: **SplitBaseAddress64ByteAligned** Abs. Address StreamIn/StreamOut Surface. **Note:** The format is linear vs. tile for better performance. GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and MBZ.
31	31:0	**MPC RowStore Surface - Attributes** Format: **MemoryAddressAttributes**

MFX_VP8_Encoder_CFG

MFX_VP8_Encoder_CFG

Source: VideoCS

Length Bias: 2

This must be the very first command to issue after the surface state, the pipe select and base address setting commands and must be issued before MFX_VP8_PIC_STATE.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Video Codec Format: OpCode
	26:24	**Media Command OpCode** Default Value: 4h VP8 Format: OpCode
	23:21	**Sub Opcode A** Default Value: 2h VP8 Common Format: OpCode
	20:16	**Sub Opcode B** Default Value: 1h MFX_VP8_ENCODER_CFG Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Format: =n Value / Name / Description 000h / Excludes DWord (0,1) **[Default]** / A special case to provide a dummy image state for stitch mode operation. In this case, fields in DW1 which is part of the dummy image state command are ignored by hardware." 01Dh / / Used for normal encode mode
1	31:11	**Reserved** Access: RO Format: MBZ

MFX_VP8_Encoder_CFG

	10	**VBSPunitPowerClock Gating Disable** Format: U1 VBSPunit Power Clock Gating Disable.
	9	**Compressed Bitstream Output Disable** Format: U1 Disable Compressed Bitstream Output.**(Both Final Bitstream and Intermediate bit buffer)**
	8	**Finer BRC Enable** Format: U1 Enable Finer BRC Feature.
	7	**Per Segment Delta Qindex / LoopFilter Disable** Format: U1 Disable Per Segment Delta Qindex / Loop Filter in Rate Control.
	6	**Rate Control Initial Pass** Format: U1 Value / Name: 1 – Initial pass 0 – Subsequence Pass(es)
	5	**Skip Final Bitstream when Over / Under flow** Format: U1 Skip Final Bitstream conditionally on Over/Under flow in rate control and intermediate Bit Buffer Overrun.
	4	**Update Segment Feature Data Flag** Exists If: //VP8 Encoder Format: U1 Enable for Frame Header per Segment Quantizer / LoopFilter Update
	3	**Bitstream Statistics Output Enable** Enable Bitstream Statistics Output at Memory Surface in MFX_VBSP_BUF_ADDR_STATE DW[26:28]
	2	**Token Statistics Output Enable** Enable Token Statistics Output at Memory Surface in MFX_VBSP_BUF_ADDR_STATE DW[26:28]

MFX_VP8_Encoder_CFG

	1	**Final Bitstream Output Disable** Format: U1 Disable Final Bitstream Output.
	0	**Performance Counter Enable** Format: U1 Enable Performance Counter in Streamout.
2	31:8	**Reserved** Access: RO Format: MBZ
	7	**Qindex_Clamp_High_mask for overflow** Format: U1 If current frame is overflow and this mask is set, it would mask out MFX_VP8_Img_Status register. DW1.bit1. In another word, subsequent passes would be skipped.
	6	**Qindex_Clamp_High_mask for underflow** Format: U1 If current frame is underflow and this mask is set, it would mask out MFX_VP8_Img_Status register. DW1.bit0. In another word, subsequent passes would be skipped
	5	**Final Bistream Buffer Overrun Enable Mask** Format: U1 Enable Final Bitstream Buffer Overrun detection feature.
	4	**Intermediate Bit Buffer Overrun Enable Mask** Format: U1 Enable Intermediate Bit Buffer Overrun detection feature.
	3	**Max Intra MB Bit Count Check Enable Mask** Format: U1 Enable Max. Intra MB bit count check in Streamout.
	2	**Max Inter MB Bit Count Check Enable Mask** Format: U1 Enable Max. Inter MB bit count check in Streamout.

MFX_VP8_Encoder_CFG

	1	**Min Frame Bit Count Rate Control Enable Mask** Format: U1 Enable Min. Frame Rate Control. This is a mask bit controlling if the condition of frame level bit count is less than or equal to FrameBitRateMin. **Value** / **Name** / **Description** 1 / / If (Total Frame Level Bit Counter) =< (Frame Bit Rate Minimum limit)Set bit[0] and bit[1] of MFX_VP8_IMAGE_STATUS Control Register. 0 / / Do not update bit[0] of MFX_VP8_IMAGE_STATUS Control Register.
	0	**Max Frame bit count Rate Control Enable Mask** Format: U1 Enable Max. Frame Rate Control. This is a mask bit controlling if the condition of frame level bit count is greater than or equal to FrameBitRateMax. **Value** / **Name** / **Description** 1 / / If (Total Frame Level Bit Counter) >= (Frame Bit Rate Maximum Limit)Set bit[0] and bit[1] of MFX_VP8_IMAGE_STATUS control register. 0 / / Do not update bit[0] of MFX_VP8_IMAGE_STATUS control register.
3	31:28	**Reserved** Access: RO Format: MBZ
	27:16	**Max Intra MB Bit Count Limit** Format: U12 12-bit bit count for Max Intra MB Limit.
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Max Inter MB bit count** Format: U12 12-bit bit count for Max Inter MB Limit.
4	31	**Frame Bitrate Min Unit Mode** Format: U1 This field is the Frame Bitrate Minimum Limit Units. **Value** / **Name** / **Description** 0h / Compatibility Mode / Frame BitRate Min Unit is in old mode **(128b/16Kb)**

MFX_VP8_Encoder_CFG

		1h | New Mode | Frame BitRate Min Unit is in new mode **(32byte/4Kb)**
	30	**Frame Bit Rate Min Unit** Format: U1 *This field is Frame Bitrate Minimum Mode.* 0 = 32-B 1 = 4-KB
	29:16	**Frame Bit Rate Min** Format: U14 If either BRC Underflow or overflow is enabled. Frame Bit Rate Min and Frame Bit Rate Max need to be programmed with unambiguous values
	15	**Frame Bitrate Max Unit Mode** Format: U1 This field is the Frame Bitrate Maximum Limit Units. 0h | Compatibility Mode | Frame BitRate Max Unit is in old mode **(128b/16Kb)** 1h | New Mode | Frame BitRate Max Unit is in new mode **(32byte/4Kb)**
	14	**Frame Bit Rate Max Unit** Format: U1 *This field is Frame Bitrate Maximum Mode* 0 = 32-B 1 = 4-KB
	13:0	**Frame Bit Rate Max** Format: U14 If either BRC Underflow or overflow is enabled. Frame Bit Rate Min and Frame Bit Rate Max need to be programmed with unambiguous values
5	31:24	**Frame Delta QIndex Max[3]** This field is the Frame level delta Qindex for total bit-count above FrameBitRateMax - First 1/8 Region. This field is used to calculate the suggested Frame Qindex into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame exceeds FrameBitRateMax but is within 1/8 of FrameBitRateMaxDelta above **FrameBitRateMax**;i.e., In the range of (FrameBitRateMax, (FrameBitRateMax + FrameBitRateMaxDelta » 3)].

MFX_VP8_Encoder_CFG		
	23:16	**Frame DeltaQ Index Max[2]** This field is the Frame level delta Qindex for bit-count above FrameBitRateMax - Above 1/8 and Below 1/4. This field is used to calculate the suggested Frame Qindex into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between 1/8 and 1/4 of FrameBitRateMaxDelta above **FrameBitRateMax**;i.e., In the range of ((FrameBitRateMax + FrameBitRateMaxDelta » 3), (FrameBitRateMax+ FrameBitRateMaxDelta » 2)].
	15:8	**Frame Delta QIndex Max[1]** This field is the Frame level delta QINDEX for bit-count above FrameBitRateMax - Above 1/4 and Below 1/2. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between 1/4 and 1/2 of FrameBitRateMaxDelta above **FrameBitRateMax**;i.e., In the range of [(FrameBitRateMax+ FrameBitRateMaxDelta » 2), (FrameBitRateMax+ FrameBitRateMaxDelta » 1)].
	7:0	**Frame Delta QIndex Max [0]** This field is the Frame level delta QINDEX for bit-count above FrameBitRateMax - Above 1/2. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is above FrameBitRateMax by more than half the distance of **FrameBitRateMax**;i.e., In the range of [(FrameBitRateMax + FrameBitRateMaxDelta » 1), (Infinite)].
6	31:24	**Frame Delta QIndex Min[3]** This field is the Frame level delta QINDEX for total bit-count below FrameBitRateMin - First 1/8 Region. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is less than FrameBitRateMin and greater than or equal to 1/8 the distance of FrameBitRateMinDelta from **FrameBitRateMin**; i.e., In the range of [(FrameBitRateMin - FrameBitRateMinDelta » 3), FrameBitRateMin].
	23:16	**Frame Delta QIndex Min[2]** This field is the Frame level delta QINDEX for bit-count below FrameBitRateMin - Below 1/ 8 and Above 1/4. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between one-eighth and quarter the distance of FrameBitRateMinDelta from **FrameBitRateMin**;i.e., In the range of [(FrameBitRateMin - FrameBitRateMinDelta » 2), (FrameBitRateMin- FrameBitRateMinDelta » 3)].
	15:8	**Frame Delta QIndex Min[1]** This field is the Frame level delta QINDEX for bit-count below FrameBitRateMin- Below 1/4 and Above 1/2. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire

MFX_VP8_Encoder_CFG		
		frame is between quarter and half the distance of FrameBitRateMinDelta from **FrameBitRateMin**;i.e., In the range of [(FrameBitRateMin - FrameBitRateMinDelta » 1), (FrameBitRateMin - FrameBitRateMinDelta » 2)].
	7:0	**Frame Delta QIndex Min[0]** This field is the Frame Level Delta QINDEX for bit-count below FrameBitRateMin - Below 1/2. This field is used to calculate the suggested Frame QINDEX into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is below FrameBitRateMin by more than half the distance of **FrameBitRateMin**;i.e., In the range of [0, (FrameBitRateMin - FrameBitRateMinDelta » 1).
7	31:0	**Per Segment Frame Delta QIndex Max[1]**
8	31:0	**Per Segment Frame Delta QIndex Min[1]**
9	31:0	**Per Segment Frame Delta QIndex Max[2]**
10	31:0	**Per Segment Frame Delta QIndex Min[2]**
11	31:0	**Per Segment Frame Delta QIndex Max[3]**
12	31:0	**Per Segment Frame Delta QIndex Min[3]**
13	31:24	**Frame Delta Loop Filter Max[3]** Format: U8 This field is the Frame level delta LoopFilter for total bit-count above FrameBitRateMax - First 1/8 region. This field is used to calculate the suggested Frame LoopFilter into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame exceeds FrameBitRateMax but is within 1/8 of FrameBitRateMaxDelta above FrameBitRateMax.i.e., in the range of (FrameBitRateMax, (FrameBitRateMax+ FrameBitRateMaxDelta » 3)].
	23:16	**Frame Delta Loop Filter Max[2]** Format: U8 This field is the Frame level delta LoopFilter for bit-count above FrameBitRateMax - Above 1/8 and Below 1/4. This field is used to calculate the suggested Frame LoopFilter into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between 1/8 and 1/4 of FrameBitRateMaxDelta above FrameBitRateMax.i.e., in the range of ((FrameBitRateMax + FrameBitRateMaxDelta » 3) and (FrameBitRateMax + FrameBitRateMaxDelta » 2)].
	15:8	**Fram eDelta Loop Filter Max[1]** Format: U8 This field is the Frame level delta LOOPFILTER for bit-count above FrameBitRateMax - Above1/ 4 and Below 1/2. This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire

MFX_VP8_Encoder_CFG

		frame is between 1/4 and1/2 of FrameBitRateMaxDelta above FrameBitRateMax.i.e., in the range of ((FrameBitRateMax+ FrameBitRateMaxDelta » 2) and (FrameBitRateMax+ FrameBitRateMaxDelta » 1)].
	7:0	**Frame Delta Loop Filter Max[0]** Format: U8 This field is the Frame level delta LOOPFILTER for bit-count above FrameBitRateMax - Above 1/2. This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is above FrameBitRateMax by more than half the distance of FrameBitRateMaxDelta.i.e., in the range of ((FrameBitRateMax+ FrameBitRateMaxDelta » 1), infinite).
14	31:24	**Frame Delta Loop Filter Min[3]** Format: U8 This field is the Frame level delta LOOPFILTER for total bit-count below FrameBitRateMin - First 1/8 region. This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is less than FrameBitRateMin and greater than or equal to 1/8 the distance of FrameBitRateMinDelta from FrameBitRateMin.i.e., in the range of [(FrameBitRateMin - FrameBitRateMinDelta » 3), FrameBitRateMin).
	23:16	**Frame Delta Loop Filter Min[2]** Format: U8 This field is the Frame level delta LOOPFILTER for bit-count below FrameBitRateMin - Below 1/ 8 and Above 1/4. This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between one-eighth and quarter the distance of FrameBitRateMinDelta from FrameBitRateMin.i.e., in the range of [(FrameBitRateMin - FrameBitRateMinDelta » 2), (FrameBitRateMin - FrameBitRateMinDelta » 3)).
	15:8	**Frame Delta Loop Filter Min[1]** Format: U8 This field is the Frame level delta LOOPFILTER for bit-count below FrameBitRateMin- Below 1/4 and Above 1/2. This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is between quarter and half the distance of FrameBitRateMinDelta from FrameBitRateMin.i.e., in the range of [(FrameBitRateMin - FrameBitRateMinDelta » 1) and (FrameBitRateMin - FrameBitRateMinDelta » 2)).

MFX_VP8_Encoder_CFG

	7:0	**Frame Delta Loop Filter Min[0]** Format: U8 This field is the Frame Level Delta LOOPFILTER for bit-count below FrameBitRateMin - Below 1/ 2. This field is used to calculate the suggested Frame LOOPFILTER into the MFX_VP8_IMAGE_STATUS control register when total bit count for the entire frame is below FrameBitRateMin by more than half the distance of FrameBitRateMinDelta.i.e., in the range of [0, (FrameBitRateMin - FrameBitRateMinDelta » 1).
15	31:0	**Per Segment Frame Delta LoopFilter Max[1]**
16	31:0	**Per Segment Frame Delta LoopFilter Min[1]**
17	31:0	**Per Segment Frame Delta LoopFilter Max[2]**
18	31:0	**Per Segment Frame Delta LoopFilter Min[2]**
19	31:0	**Per Segment Frame Delta LoopFilter Max[3]**
20	31:0	**Per Segment Frame Delta LoopFilter Min[3]**
21	31	**Reserved** Access: RO Format: MBZ
	30:16	**FrameBitRateMinDelta** Format: U15 This field is used to select the frame delta QINDEX when FrameBitRateMin Is exceeded. It shares the same FrameBitrateMinUnit. Value / Name / Description [0-4095] / / When FrameBitrateMinUnit is in Bytes, this range is in Bytes.When FrameBitrateMinUnit is in KB, this range is in KB units.
	15	**Reserved** Access: RO Format: MBZ
	14:0	**Frame Bit Rate Max Delta** Format: U15 This field is used to select the frame delta QINDEX when FrameBitRateMax Is exceeded. It shares the same FrameBitrateMaxUnit. Value / Name / Description [0-4095] / / When FrameBitrateMinUnit is in Bytes, this range is in Bytes.When FrameBitrateMinUnit is in KB, this range is in KB units.
22	31:24	**Reserved** Access: RO

MFX_VP8_Encoder_CFG

		Format: MBZ
	23	**Show Frame** Format: U1 VP8 Frame Tag, Show Frame Field
	22:20	**Bitstream Format Version** Format: U3 VP8 Frame Tag, Verison Field
	19:18	**Reserved** Access: RO Format: MBZ
	17:16	**Min Frame WSize Unit** Format: U2 Value / Name / Description: 0h – Compatibility Mode – MinFrameWSizeUnit is in old mode (128b/16Kb) 1h – New Mode – MinFrameWSizeUnit is in new mode (32byte/4Kb)
	15:0	**Min Frame WSize** Exists If: //Encoder Only This field (in Word, 16-bit) is specified to compensate for Intel Rate Control. Zero padding would be performed.
23	31:16	**Vertical_Size_Code** Format: U16 Frame Tag Vertical Size Code, composed of{VerticalScale[15:14], FrameHeight[13:0]}
	15:0	**Horizontal_Size_Code** Format: U16 Frame Tag Horizontal Size Code, composed of{HorizontalScale[15:14], FrameWidth[13:0]}
24	31:0	**Frame Header Bit Count** Format: U32 Binarized Header Bit Count.
25	31:0	**Frame Header Bin Buffer Qindex Update Pointer** Format: U32

MFX_VP8_Encoder_CFG

		Binarized Header Qindex Update PointerIf Segment Enabled and UpdateSegmentFeature enabled, 4 per segment Qindices would be updated in Binarized header (Only ABS mode supported).Else Base Qindex would be updated
26	31:0	**Frame Header Bin Buffer LoopFilter Update Pointer** Format: U32 Binarized Header LoopFilter Update PointerIf Segment Enabled and UpdateSegmentFeature enabled, 4 per segment LoopFilters would be updated in Binarized header (Only ABS mode supported).ElseBase LoopFilter would be updated.
27	31:0	**Frame Header Bin Buffer Token Update Pointer** Format: U32 Binarized Header TokenUpdate Pointer
28	31:0	**Frame Header Bin Buffer MVUpdate Pointer** Format: U32 Binarized Header MVUpdate Pointer.
29 **Programming Notes:** The only value permitted for CV7 through CV0 is 0xf	31:28	**ClampValues - CV7**
	27:24	**CV6**
	23:20	**CV5**
	19:16	**CV4**
	15:12	**CV3**
	11:8	**CV2**
	7:4	**CV1**
	3:0	**CV0 - Clamp Value 0** Format: U4 If the magnitude of coefficients at locations assigned with CV0 (mapping shown below) exceeds 2CV0-1, they are replaced with 2CV0-1. For coefficients at locations marked as 'none', no clamping is performed. The following mappings are only applied to luma and chroma blocks\subblocks containing AC coefficiencts (blocks\sublocks with only DC coeffs will not be clamped). **For 4x4 frame block, each coefficient is mapped to one of the eight CV values as following:** none CV7 CV5 CV4 CV7 CV6 CV4 CV3 CV5 CV4 CV2 CV1 CV4 CV3 CV1 CV0

For 4x4 field block, each coefficient is mapped to one of the eight CV values as following:

none	CV6	CV3	CV1
CV7	CV6	CV3	CV1
CV5	CV4	CV2	CV0
CV5	CV4	CV2	CV0

Value	Name
0-15	

MFX_VP8_PAK_OBJECT

MFX_VP8_PAK_OBJECT

Source: VideoCS

Length Bias: 2

The MFX_VP8_PAK_OBJECT command is the second primitive command for the VP8 Encoding Pipeline. The MV Data portion of the bitstream is loaded as indirect data object. Before issuing a MFX_VP8_PAK_OBJECT command, all VP8 MFX states need to be valid; therefore the commands used to set these states need to have been issued prior to the issue of this command. MB record must be consecutive with no gaps, hence we do not need MB(x,y) in each MB command .Internal counter will keep track of the current MB address, starting from the first MB.MFX_VP8_PAK_OBJECT command follows the MbType definition like MFD. Encoding statistical data such as the total size of the output bitstream are provided through MMIO registers. Software may access these registers through MI_STORE_REGISTER_MEM command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_VP8_PAK_OBJECT Format: OpCode
	26:24	**Media Command Opcode** Default Value: 4h VP8_ENC Format: OpCode
	23:21	**SubOpcode A** Default Value: 2h Format: OpCode
	20:16	**SubOpcode B** Default Value: 9h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 5h DWORD_COUNT_n Format: =n
1	31:30	**Reserved** Access: RO Format: MBZ

MFX_VP8_PAK_OBJECT

	29	**Enable Inline MV data** Format: Enable This field denotes if the MV data will be sent inline following the other inline data instead of being indirect.
	28:10	**Reserved** Access: RO Format: MBZ
	9:0	**Indirect PAK-MV Data Length** Format: U10 This field provides the length in bytes of the indirect data, which contains all the MVs for the current MB (in any partitioning and subpartitioning form). A value zero indicates that indirect data fetching is disabled - subsequently, the Indirect PAK-MV Data Start Address field is ignored. This field must have the same alignment as the Indirect PAK-MV Data Start Address. This field must be DW aligned (since each MV is 4 bytes in size).Driver has to derived this field from MVsize (MVquantity in DXVA, exact size) *4 bytes per MV.
2	31:29	**Reserved** Access: RO Format: MBZ
	28:0	**Indirect PAK-MV Data Start Address Offset** This field specifies the memory starting address (offset) of the MV data to be fetched into PAK Subsystem for processing. This pointer is relative to the MFC Indirect PAK-MV Object Base Address.Hardware ignores this field if indirect data is not present, i.e. the Indirect PAK-MV Data Length is set to 0. It is a Dword aligned address in all AVC encoding configuration, since each MV is 4 bytes in size. Value: [0,512MB) — Name:
3..6	127:0	**Inline Data** All the required MB level controls and parameters for encoding are captured as Inline Data Description - VP8 PAK OBJECT. It has a fixed size of 4 DWs. Its definition is described in the next section.

MFX_VP8_PIC_STATE

MFX_VP8_PIC_STATE

Source: VideoCS

Length Bias: 2

This must be the very first command to issue after the surface state, the pipe select and base address setting commands and must be issued before MFX_VP8_IMG_STATE.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Video Codec Format: OpCode
	26:24	**Media Command OpCode** Default Value: 4h VP8 Format: OpCode
	23:21	**Sub Opcode A** Default Value: 0h VP8 Common Format: OpCode
	20:16	**Sub Opcode B** Default Value: 0h MFX_VP8_PIC_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Format: =n Value / Name / Description 000h / Excludes DWord (0,1) **[Default]** / A special case to provide a dummy image state for stitch mode operation. In this case, fields in DW1 which is part of the dummy image state command are ignored by hardware." 024h / / Used for normal decode and encode mode
1	31:24	**Reserved** Access: RO Format: MBZ

MFX_VP8_PIC_STATE

	23:16	**Frame Height Minus 1** Exists If: //Decoder / Encoder Format: U8 Picture Height in integer number of MBs minus 1, so the min pic height can be program is 16 rows of pixels.
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**Frame Width Minus 1** Exists If: //Decoder / Encoder Format: U8 Picture Width in integer number of MBs minus 1, so the min pic width can be program is 16 pixels.
2	31:26	**Reserved** Access: RO Format: MBZ
	25:24	**Log2 Num of Partition** Exists If: //Decoder / Encoder Format: U2 Value / Name: 0 – 1 Token partition 1 – 2 Token partition 2 – 4 Token partition 3 – 8 Token partition
	23:19	**Reserved** Access: RO Format: MBZ
	18:16	**Deblock Sharpness Level** Exists If: //Decoder / Encoder Format: U3 Specifiy the sharpness level, as one of the regular deblocking strength control parameters. **Programming Notes** Set to 0 to disable the use of sharpness control.
	15:14	**Reserved** Access: RO

MFX_VP8_PIC_STATE

	Format:	MBZ

13 Alternate Ref Pic MV SignBias Flag

Exists If:	//Decoder / Encoder

Alternate Reference Picture MV sign bias flag, specified for non-key frame only.

12 Golden Ref Picture MV SignBias Flag

Exists If:	//Decoder / Encoder

Golden Reference Picture MV sign bias flag, specified for non-key frame only.

11 Mode Reference Loop Filter Delta Enabled

Exists If:	//Decoder / Encoder

Value	Name	Description
0		Mode or Reference Loop Filter Delta Adjustment for current frame is disabled.
1		Mode or Reference Loop Filter Delta Adjustment for current frame is enabled.

10 MB NoCoeff SkipFlag

Exists If:	//Decoder / Encoder

Frame level control if Skip MB (with no non-zero coefficient) is allowed or not.

Value	Name	Description
0		All MBs will have its MB level signaling mb_skip_coeff forced to 0. That is, no skip of coefficient record in the bitstream (even their values are all 0s)
1		Skip MB is enabled in the per MB record.

9 Update MBSegment Map Flag

Exists If:	//Decoder / Encoder

Value	Name	Description
0		Disable segmentation update
1		Enable segmentation update, and to enable reading segment_id for each MB.

8 Segment Enable Flag

Exists If:	//Decoder / Encoder

Value	Name	Description
0		Disable Segmentation processing in the current frame
1		Enable Segmentation processing in the current frame

7 Segmentation ID StreamIn Enable

Exists If:	//Decoder Only

Value	Name

0	StreamIn Disabled
1	StreamIn Enabled

Programming Notes
When 0, no input needed.

6 **Segmentation ID StreamOut Enable**

Exists If:	//Decoder Only

Value	Name
0	StreamOut Disabled
1	StreamOut Enabled

Programming Notes
When 0, no output needed.

5 **sKeyFrameFlag**

Exists If:	//Decoder / Encoder

Value	Name
0	Non-Key Frame (P-Frame)
1	Key Frame (I-Frame)

4 **DBLKFilterType**

Exists If:	//Decoder / Encoder

To specify VP8 Profile of operation.

Value	Name	Description
0		Use a full feature normal deblocking filter
1		Use a simple filter for deblocking

3:2 **Reserved**

Access:	RO
Format:	MBZ

1 **Chroma Full Pixel MC Filter Mode**

Exists If:	//Decoder / Encoder

To specify VP8 Profile of operation.

Value	Name	Description
0		Chroma MC filter operates in sub-pixel mode
1		Chroma MC filter only operates in full pixel position, i.e. no sub-pixel interpolation.

0 **MC Filter Select**

Exists If:	//Decoder / Encoder

To specify VP8 Profile of operation.

Value	Name	Description
0		6-tap filter (regular filter mode)
1		2-tap bilinear filter (simple profile/version mode)

3 | 31:30 | Reserved

Access:	RO
Format:	MBZ

29:24 | DBLKFilterLevel for Segment3

Exists If:	//Decoder / Encoder
Format:	U6

Value	Name	Description
0	Signifies disable in loop deblocking operation	This is used to set a VP8 profile without in loop deblocker.

Programming Notes
There are max 4 segments per frame, each segment can have its own deblocking filter level. When segmentation is disabled, only segment 0 parameter is used for the entire frame.

23:22 | Reserved

Access:	RO
Format:	MBZ

21:16 | DBLKFilterLevel for Segment2

Exists If:	//Decoder / Encoder
Format:	U6

Value	Name	Description
0	Signifies disable in loop deblocking operation	This is used to set a VP8 profile without in loop deblocker.

Programming Notes
There are max 4 segments per frame, each segment can have its own deblocking filter level. When segmentation is disabled, only segment 0 parameter is used for the entire frame.

15:14 | Reserved

Access:	RO
Format:	MBZ

13:8 | DBLKFilterLevel for Segment1

Exists If:	//Decoder / Encoder
Format:	U6

MFX_VP8_PIC_STATE

		Value / **Name** / **Description** 0 / Signifies disable in loop deblocking operation / This is used to set a VP8 profile without in loop deblocker. **Programming Notes** There are max 4 segments per frame, each segment can have its own deblocking filter level. When segmentation is disabled, only segment 0 parameter is used for the entire frame.
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**DBLKFilterLevel for Segment0** Exists If: //Decoder / Encoder Format: U6 **Value** / **Name** / **Description** 0 / Signifies disable in loop deblocking operation / This is used to set a VP8 profile without in loop deblocker. **Programming Notes** There are max 4 segments per frame, each segment can have its own deblocking filter level. When segmentation is disabled, only segment 0 parameter is used for the entire frame.
4	31	**Reserved** Access: RO Format: MBZ
	30:24	**Seg 3 Qindex** Exists If: //Encoder Only Format: U7 Quantizer Value for Segment ID 3
	23	**Reserved** Access: RO Format: MBZ
	22:16	**Seg 2 Qindex** Exists If: //Encoder Only Format: U7 Quantizer Value for Segment ID 2

MFX_VP8_PIC_STATE

	15	**Reserved** Access: RO Format: MBZ
	14:8	**Seg 1 Qindex** Exists If: //Encoder Only Format: U7 Quantizer Value for Segment ID 1
	7	**Reserved** Access: RO Format: MBZ
	6:0	**Seg 0 Qindex** Exists If: //Encoder Only Format: U7 Quantizer Value for Segment ID 0. **Programming Notes** This is the **[Default]** Qindex
5	31:29	**Reserved** Access: RO Format: MBZ
	28	**UVac Qindex Delta Sign** Exists If: //Encoder Only Format: U1 Sign of Quantization index delta for UVac
	27:24	**UVac QindexDelta** Exists If: //Encoder Only Format: U4 Absolute Quantization index delta for UVac
	23:21	**Reserved** Access: RO Format: MBZ
	20	**UVdc Qindex Delta Sign** Exists If: //Encoder Only Format: U1 Sign of Quantization index delta for UVdc

MFX_VP8_PIC_STATE

	19:16	**UVdc Qindex Delta** Exists If: //Encoder Only Format: U4 Absolute Quantization index delta for UVdc
	15:13	**Reserved** Access: RO Format: MBZ
	12	**Y2ac Qindex Sign** Exists If: //Encoder Only Format: U1 Sign of Quantization index delta for Y2ac
	11:8	**Y2ac Qindex Delta** Exists If: //Encoder Only Format: U4 Absolute Quantization index delta for Y2ac
	7:5	**Reserved** Access: RO Format: MBZ
	4	**Y2ac Qindex Delta Sign** Exists If: //Encoder Only Format: U1 Sign of Quantization index delta for Y2dc This is the **[Default]** Qindex Delta Sign
	3:0	**Y2dc Qindex Delta** Exists If: //Encoder Only Format: U4 Absolute Quantization index delta for Y2dc This is the **[Default]** Qindex Delta
6	31:5	**Reserved** Access: RO Format: MBZ
	4	**Y1dc Qindex Delta Sign** Exists If: //Encoder Only

		Format: U1 Sign of Quantization index delta for Y1dc This is the **[Default]**Qindex Delta Sign
	3:0	**Y1dc Qindex Delta** Exists If: //Encoder Only Absolute Quantization index delta for Y1dc This is the **[Default]**Qindex Delta
7	31:15	**Reserved** Access: RO Format: MBZ
	14:8	**Clamp Qindex high** Exists If: //Encoder Only Format: U7 Maximum Clamp Value for Qindex used in quantization.
	7	**Reserved** Access: RO Format: MBZ
	6:0	**Clamp Qindex Low** Exists If: //Encoder Only Format: U7 Minimum Clamp Value for Qindex used in quantization.
8	31:25	**Reserved** Access: RO Format: MBZ
	24:16	**Quantizer Value [1][BlockType3=UVAC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Access: RO Format: MBZ
	8:0	**Quantizer Value [1][BlockType2=UVDC]** Exists If: //Decoder Only

MFX_VP8_PIC_STATE

		Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]
9	31:25	**Reserved** Access: RO Format: MBZ
	24:16	**Quantizer Value [1][BlockType5=Y2AC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Access: RO Format: MBZ
	8:0	**Quantizer Value [1][BlockType4=Y2DC]** Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
10	31:25	**Reserved** Access: RO Format: MBZ
	24:16	**Quantizer Value [2][BlockType1=Y1AC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Access: RO Format: MBZ
	8:0	**Quantizer Value [2][BlockType0=Y1DC]** Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
11	31:25	**Reserved** Access: RO Format: MBZ
	24:16	**Quantizer Value [2][BlockType3=UVAC]** Exists If: //Decoder Only Format: U9

MFX_VP8_PIC_STATE

		Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]
	15:9	**Reserved** Access: RO Format: MBZ
	8:0	**Quantizer Value [2][BlockType2=UVDC]** Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
12	31:25	**Reserved** Access: RO Format: MBZ
	24:16	**Quantizer Value [2][BlockType5=Y2AC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Access: RO Format: MBZ
	8:0	**Quantizer Value [2][BlockType4=Y2DC]** Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
13	31:25	**Reserved** Access: RO Format: MBZ
	24:16	**Quantizer Value [3][BlockType1=Y1AC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Access: RO Format: MBZ
	8:0	**Quantizer Value [3][BlockType0=Y1DC]** Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**

MFX_VP8_PIC_STATE

14	31:25	**Reserved** Access: RO Format: MBZ
	24:16	**Quantizer Value [3][BlockType3=UVAC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Access: RO Format: MBZ
	8:0	**Quantizer Value [3][BlockType2=UVDC]** Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
15	31:25	**Reserved** Access: RO Format: MBZ
	24:16	**Quantizer Value [3][BlockType5=Y2AC]** Exists If: //Decoder Only Format: U9 **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
	15:9	**Reserved** Access: RO Format: MBZ
	8:0	**Quantizer Value [3][BlockType4=Y2DC]** Exists If: //Decoder Only **Quantizer Value [n = Segment_Id = 0..3][BlockType = 0..5]**
16..17	63:0	**CoeffProbability StreamIn Base Address** Exists If: //Decoder Only Format: **SplitBaseAddress4KByteAligned** It is specified for non-key frame only.It is the final computed probability table for parsing Coeff in the bitstream.The buffer is unsigned 8-bit * 1056 entries (CoeffProbs[4][8][3][11].
18	31:15	**Reserved** Access: RO Format: MBZ

14:13 **CoeffProbability StreamIn - Tiled Resource Mode**

Exists If:	//Decoder Only
Format:	U2

For Media Surfaces: This field specifies the tiled resource mode.

Value	Name	Description
0h	TRMODE_NONE	No tiled resource
1h	TRMODE_TILEYF	4KB tiled resources
2h	TRMODE_TILEYS	64KB tiled resources
3h	Reserved	

12:11 **Reserved**

Access:	RO
Format:	MBZ

10 **CoeffProbability StreamIn - Memory Compression Mode**

Exists If:	//Decoder Only
Format:	U1

Distinguishes Vertical from Horizontal compression. Please refer to vol1a**Memory Data Formats** chapter, **Media Memory Compression** section for more details.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

9 **CoeffProbability StreamIn - Memory Compression Enable**

Exists If:	//Decoder Only
Format:	Enable

Memory compression will be attempted for this surface.

8:7 **CoeffProbability StreamIn - Arbitration Priority Control**

Exists If:	//Decoder Only
Format:	U2

This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.

Value	Name
00b	Highest priority
01b	Second highest priority
10b	Third highest priority
11b	Lowest priority

6:1 **CoeffProbability StreamIn Address - Index to Memory Object Control State (MOCS) Tables**

Exists If:	//Encoder Only
Format:	U6

The index to define the L3 and system cache memory properties. The details of the controls are

		further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.
	0	**Reserved**
19	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**MBSegmentIDTreeProbs[2]** Exists If: //Decoder / Encoder Format: U8 MBSegmentIDTreeProbs[2:0] probability tree table for CPBAC parsing Segment_ID of each MB.
	15:8	**MBSegmentIDTreeProbs[1]** Exists If: //Decoder / Encoder Format: U8 MBSegmentIDTreeProbs[2:0] probability tree table for CPBAC parsing Segment_ID of each MB.
	7:0	**MBSegmentIDTreeProbs[0]** Exists If: //Decoder / Encoder Format: U8 MBSegmentIDTreeProbs[2:0] probability tree table for CPBAC parsing Segment_ID of each MB.
20	31:24	**MBNoCoeffSkipFalseProb** Exists If: //Decoder / Encoder Format: U8 8-bit probability value for CPBAC parsing of the MBNoCoeffSkip Flag in the bistream.
	23:16	**IntraMBProb** Exists If: //Decoder / Encoder Format: U8 8-bit probability value for CPBAC parsing of the intra or inter MB type flag in the bitstream.
	15:8	**InterPredFromLastRefProb** Exists If: //Decoder / Encoder Format: U8 8-bit probability value for CPBAC parsing of the flag in the bitstream that determines which reference frame to be used for the current MB motion compensation.

MFX_VP8_PIC_STATE

	7:0	**InterPredFromGRefRefProb** Exists If: //Decoder / Encoder Format: U8 8-bit probability value for CPBAC parsing of the flag in the bitstream that determines which reference frame to be used for the current MB motion compensation.
21	31:24	**YModeProb[3]** Exists If: //Decoder / Encoder Format: U8 YModeProb[3:0] probability tree table for CPBAC parsing Luma MBType of each MB.
	23:16	**YModeProb[2]** Exists If: //Decoder / Encoder Format: U8 YModeProb[3:0] probability tree table for CPBAC parsing Luma MBType of each MB.
	15:8	**YModeProb[1]** Exists If: //Decoder / Encoder Format: U8 YModeProb[3:0] probability tree table for CPBAC parsing Luma MBType of each MB.
	7:0	**YModeProb[0]** Exists If: //Decoder / Encoder Format: U8 YModeProb[3:0] probability tree table for CPBAC parsing Luma MBType of each MB.
22	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**UVModeProb[2]** Exists If: //Decoder / Encoder Format: U8 UVModeProb[2:0] probability tree table for CPBAC parsing Chroma MBType of each MB.
	15:8	**UVModeProb[1]** Exists If: //Decoder / Encoder Format: U8 UVModeProb[2:0] probability tree table for CPBAC parsing Chroma MBType of each MB.

MFX_VP8_PIC_STATE

	7:0	**UVModeProb[0]** Exists If: //Decoder / Encoder Format: U8 UVModeProb[2:0] probability tree table for CPBAC parsing Chroma MBType of each MB.
23	31:24	**MVUpdateProbs[0][3]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	23:16	**MVUpdateProbs[0][2]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[0][1]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[0][0]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
24	31:24	**MVUpdateProbs[0][7]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	23:16	**MVUpdateProbs[0][6]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].

MFX_VP8_PIC_STATE

	15:8	**MVUpdateProbs[0][5]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[0][4]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
25	31:24	**MVUpdateProbs[0][11]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB.
	23:16	**MVUpdateProbs[0][10]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB.
	15:8	**MVUpdateProbs[0][9]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB.
	7:0	**MVUpdateProbs[0][8]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB.
26	31:24	**MVUpdateProbs[0][15]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].

MFX_VP8_PIC_STATE

	23:16	**MVUpdateProbs[0][14]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[0][13]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[0][12]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
27	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**MVUpdateProbs[0][18]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[0][17]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[0][16]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].

MFX_VP8_PIC_STATE

28	31:24	**MVUpdateProbs[1][3]** Exists If: //Decoder Only Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	23:16	**MVUpdateProbs[1][2]** Exists If: //Decoder Only Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[1][1]** Exists If: //Decoder Only Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[1][0]** Exists If: //Decoder Only Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
29	31:24	**MVUpdateProbs[1][7]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	23:16	**MVUpdateProbs[1][6]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[1][5]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].

MFX_VP8_PIC_STATE

	7:0	**MVUpdateProbs[1][4]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
30	31:24	**MVUpdateProbs[1][11]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	23:16	**MVUpdateProbs[1][10]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[1][9]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[1][8]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
31	31:24	**MVUpdateProbs[1][15]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	23:16	**MVUpdateProbs[1][14]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].

MFX_VP8_PIC_STATE

	15:8	**MVUpdateProbs[1][13]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[1][12]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
32	31:24	**Reserved** Access: RO Format: MBZ
	23:16	**MVUpdateProbs[1][18]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	15:8	**MVUpdateProbs[1][17]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
	7:0	**MVUpdateProbs[1][16]** Exists If: //Decoder / Encoder Format: U8 MVUpdateProbs[1:0][18:0] probability table for CPBAC parsing of MV update value of each MB. To map into DWord, it becomes MVUpdate[1:0][19:0].
33	31	**Reserved** Access: RO Format: MBZ
	30:24	**RefLFDelta3 (for ALTREF FRAME)** Exists If: //Decoder / Encoder Format: S6

		Delta value for reference frame based adjustment of the MB-level's filter level value. RefLFDeltas [ref_frametype = 0 to 3]
		Programming Notes Please note that although RefDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.
	23	**Reserved** Access: RO Format: MBZ
	22:16	**RefLFDelta2 (for GOLDEN FRAME)** Exists If: //Decoder / Encoder Format: S6
		Delta value for reference frame based adjustment of the MB-level's filter level value. RefLFDeltas [ref_frametype = 0 to 3]
		Programming Notes Please note that although RefDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.
	15	**Reserved** Access: RO Format: MBZ
	14:8	**RefLFDelta1 (for LAST FRAME)** Exists If: //Decoder / Encoder Format: S6
		Delta value for reference frame based adjustment of the MB-level's filter level value. RefLFDeltas [ref_frametype = 0 to 3]
		Programming Notes Please note that although RefDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.
	7	**Reserved** Access: RO Format: MBZ
	6:0	**RefLFDelta0 (for INTRA FRAME)** Exists If: //Decoder / Encoder Format: S6

		Delta value for reference frame based adjustment of the MB-level's filter level value. RefLFDeltas [ref_frametype = 0 to 3] **Programming Notes** Please note that although RefDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.
34	31	**Reserved** Access: RO Format: MBZ
	30:24	**ModeLFDelta3 (for SPLITMV mode)** Exists If: //Decoder / Encoder Format: S6 Delta value for mode based adjustment of the MB-level's filter level value. ModeLFDeltas[MB_Type = 0 to 3] **Programming Notes** Please note that although ModeLFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.
	23	**Reserved** Access: RO Format: MBZ
	22:16	**ModeLFDelta2 (for Nearest, Near and New mode)** Exists If: //Decoder / Encoder Format: S6 Delta value for mode based adjustment of the MB-level's filter level value. ModeLFDeltas[MB_Type = 0 to 3] **Programming Notes** Please note that although ModeLFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.
	15	**Reserved** Access: RO Format: MBZ
	14:8	**ModeLFDelta1(for ZEROMV mode)** Exists If: //Decoder / Encoder

		Format: S6 Delta value for mode based adjustment of the MB-level's filter level value. ModeLFDeltas[MB_Type = 0 to 3] **Programming Notes** Please note that although ModeLFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.
	7	**Reserved** Access: RO Format: MBZ
	6:0	**ModeLFDelta0 (for B_PRED mode)** Exists If: //Decoder / Encoder Format: S6 Delta value for mode based adjustment of the MB-level's filter level value. ModeLFDeltas[MB_Type = 0 to 3] **Programming Notes** Please note that although ModeLFDelta is signed 2's complement, bitstream is sign bit + 6 bit magnitude.
35..36	63:0	**Segmentation ID Stream Base Address** Exists If: //Decoder Only Format: **SplitBaseAddress4KByteAligned** It is specified when **SegmentationIDStreamInEnable** or **SegmentationIDStreamOutEnable** is specified. **Programming Notes** Each cache has only 8 bits for 4 segmentation ID from 4 continuous MBs.
37	31:15	**Reserved** Access: RO Format: MBZ
	14:13	**Segmentation ID Stream - Tiled Resource Mode** Exists If: //Decoder Only Format: U2 **For Media Surfaces:**This field specifies the tiled resource mode. **Value** / **Name** / **Description** 0h / TRMODE_NONE / No tiled resource 1h / TRMODE_TILEYF / 4KB tiled resources

2h	TRMODE_TILEYS	64KB tiled resources
3h	Reserved	

12:11 Reserved

Access:	RO
Format:	MBZ

10 Segmentation ID Stream - Memory Compression Mode

Format:	U1

Distinguishes Vertical from Horizontal compression. Please refer to vol1a**Memory Data Formats** chapter, **Media Memory Compression** section for more details.

Value	Name
0	Horizontal Compression Mode
1	Vertical Compression Mode

9 Segmentation ID Stream - Memory Compression Enable

Format:	Enable

Memory compression will be attempted for this surface.

8:7 Segmentation ID Stream - Arbitration Priority Control

Exists If:	//Decoder Only
Format:	U2

This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface.

Value	Name
00b	Highest priority
01b	Second highest priority
10b	Third highest priority
11b	Lowest priority

6:1 CoeffProbability StreamIn Address - Index to Memory Object Control State (MOCS) Tables

Format:	U6

The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime.

0 Reserved

MFX_WAIT

MFX_WAIT

Source: VideoCS

Length Bias: 1

This command can be considered the same as an MI_NOOP except that the command parser will not parse the next command until the following happens

- **AVC or VC1 BSD mode:** The command will stall the parser until completion of the BSD object
- **IT, encoder, and MPEG2 BSD mode:** The command will stall the parser until the object package is sent down the pipeline This command should be used to ensure the preemption enable window occurs during the time the object command is being executed down the pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 03h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Command Subtype** Default Value: 01h MFX_SINGLE_DW Format: OpCode
	26:16	**Sub-Opcode** Default Value: 0h MFX_WAIT Format: OpCode
	15:10	**Reserved** Access: RO Format: MBZ
	9	**Reserved**
	8	**MFX Sync Control Flag**
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2

MI_ARB_CHECK

MI_ARB_CHECK

Source: CommandStreamer

Length Bias: 1

Description
This command allows software to enable or disable pre-fetch mechanism for command buffers in hardware.
The command allows software to add preemption points in the ring buffer. The command streamer will preempt in the case arbitration is enabled, there is a pending execution list and this command is currently being parsed.

Programming Notes
MI_ARB_CHK command can be programmed in a ring buffer or batch buffer. MI_ARB_CHK command must not be programmed in INDIRECT_CTX and BB_PER_CTX_PTR buffers.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 05h MI_ARB_CHECK Format: OpCode
	22:16	**Reserved** Access: RO Format: MBZ
	15:8	**Mask Bits** Programming Notes: Must be set to modify corresponding bits in Bits 7:0. (For implemented bits)
	7:1	**Reserved** Access: RO Format: MBZ
	0	**Pre-Parser Disable** This command allows software to enable or disable pre-parser of command buffer functionality from within a command sequence on per context basis. This ability allows the command stream to prefetch batch buffers that are yet to be parsed by looking ahead in the command FIFO. Even with this disabled, driver will have to ensure it does not self-modify commands already prefetched into the command buffer within the same batch buffer. Value 1 — Name: — Description: When set early fetch and parsing of future command buffers is disabled in hardware.

MI_ARB_CHECK

		0		When reset early fetch and parsing of future command buffers is enabled in hardware when "Pre-Fetch Disable" in GFX_MODE register is not set.

Programming Notes

- Mask bit [8] must be set to modify this bit. By default pre-fetch in hardware is enabled. The status of this bit is engine context save/restored.
- This is not a preemptible command if the corresponding mask bit is set for this field.

MI_ARB_ON_OFF

MI_ARB_ON_OFF

Source: CommandStreamer

Length Bias: 1

The MI_ARB_ON_OFF instruction is used to disable/enable context switching. This instruction can be used to prevent submission of a new execlist from interrupting a command sequence, however lite restore preemption is allowed with in the arbitration disabled command execution zone. Note that context switching will remain disabled until re-enabled through use of this command. This command will also prevent a switch in the case of waiting on events, running out of commands. These will effectively hang the device if allowed to occur while arbitration is off (context switching is disabled.) This command should always be used as an off-on pair with the sequence of instructions to be protected from context switch between MI_ARB_OFF and MI_ARB_ON. Software must use this arbitration control with caution since it has the potential to increase the response time of the Render Engine to pre-emption requests. This is a privileged command; it will not be effective (will be converted to a no-op) if executed from within a non-privileged batch buffer.

The MI_ARB_ON_OFF instruction is used to disable/enable context switching. Context switching could be either due to preemption or un-successful wait for events or semaphore waits. This instruction can be used to prevent submission of a new execlist from interrupting a command sequence, however lite restore preemption is allowed with in the arbitration disabled command execution zone. Note that context switching will remain disabled until re-enabled through use of this command. This command will also prevent a switch in the case of waiting on events, running out of commands. These will effectively hang the device if allowed to occur while arbitration is off (context switching is disabled.)

Programming Notes

This command must be always be programmed in pairs of off/on in the same command dispatch. Sequence of instructions to be protected from cntext switch or preemption must be programmed between the MI_ARB_OFF and MI_ARB_ON.Software must use this arbitration control with caution since it has the potential to increase the response time of the Render Engine to pre-emption requests. This is a privileged command; it will not be effective (will be converted to a no-op) if executed from within a non-privileged batch buffer.

MI_ARB_ON_OFF command must not be programmed as part of the POSH command execution.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 08h MI_ARB_ON_OFF Format: OpCode
	22:2	**Reserved** Access: RO Format: MBZ

MI_ARB_ON_OFF

	1	**Arbitration Mode** Source: RenderCS Format: Enable This bit controls whether or not lite restore is allowed when arbitration is disabled thru clearing the Arbitration Enable bit. If arbitration is enabled then the value of this bit does not change the behavior of the hardware. **Value** / **Name** 0h: Allow Lite Restore **[Default]** 1h: Lite Restore Disabled
	0	**Arbitration Enable** Format: Enable This field enables or disables context switches due to pre-emption (a new execlist). **Value** / **Name** 1: Arbitration Enabled **[Default]** 0: Arbitration Disabled

MI_ATOMIC

MI_ATOMIC

Source: BSpec

Length Bias: 2

Description

MI_ATOMIC is used to carry atomic operation on data in graphics memory. Atomic operations are supported on data granularity of 4B, 8B and 16B. The atomic operation leads to a read-modify-write operation on the data in graphics memory with the option of returning value. The data in graphics memory is modified by doing arithmetic and logical operation with the inline/indirect data provided with the MI_ATOMIC command. Inline/Indirect provided in the command can be one or two operands based on the atomic operation. Ex: Atomic-Compare operation needs two operands while Atomic-Add operation needs single operand and Atomic-increment requires no operand. Refer "Atomics" sub-section of "L3 Cache and URB" section of the B-spec for detailed atomic operations supported. Atomic operations can be enabled to return value by setting "Return Data Control" field in the command, return data is stored to CS_GPR registers.
CS_GPR4/5 registers are updated with memory Return Data based on the "Data Size". Each GPR register is qword in size and occupies two MMIO registers.
Note: Any references to CS_GPR registers in the command should be understood as the CS_GPR registers belonging to the corresponding engines *CS_GPR registers.

Engine Name	Corresponding GPR Registers
RCS, POCS	CS_GPR, POCS_GPR
BCS	BCS_GPR
VCS	VCS_GPR
VECS	VECS_GPR

Indirect Source Operands:
Operand1 is sourced from [CS_GPR1, CS_GPR0]
Operand2 is sourced from [CS_GPR3, CS_GPR2]
Read return Data is stored in [CS_GPR_5, CS_GPR4]

When "Data Size" is DWORD lower dword of CS_GPR4 (Qword) is updated with the dword data returned from memory. When "Data Size" is QWORD only CS_GPR4 (Qword) is updated with the qword data returned from memory. When the data size is OCTWORD CS_GPR4/5 are updated with the OCTWORD data returned from memory. CS_GPR4 is loaded with lower qword returned from memory and CS_GPR5 is loaded with upper qword returned from memory.

Programming Notes

- When Inline Data mode is not set, Dwords 3..10 must not be included as part of the command. Dword Length field in the header must be programmed accordingly.
- When Inline Data Mode is set, Dwords3..10 must be included based on the Data Size field of the header. Both Operand-1 and Operand-2 dwords must be programmed based on the Data Size field. Operand-2 must be programmed to 0x0 if the atomic operation doesn't require it. Dword Length field in the header must be programmed accordingly.

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td>Command Type
<table>
<tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>28:23</td><td>MI Command Opcode
<table>
<tr><td>Default Value:</td><td>2Fh MI_ATOMIC</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>22</td><td>Memory Type
This bit will be ignored and treated as if clear when executing from a non-privileged batch buffer. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear.
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>Per Process Graphics Address</td><td></td></tr>
<tr><td>1h</td><td>Global Graphics Address</td><td>This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.</td></tr>
</table></td></tr>
<tr><td></td><td>21</td><td>Post-Sync Operation
<table>
<tr><td>Source:</td><td>RenderCS, PositionCS</td></tr>
</table>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0h</td><td>No Post Sync Operation</td><td>Command is executed as usual.</td></tr>
<tr><td>1h</td><td>Post Sync Operation</td><td>MI_ATOMIC command is executed as a pipelined PIPE_CONTROL flush command with Atomics operation as post sync operation. Flush completion only guarantees the workload prior to this command is pushed till Windower unit and completion of any outstanding flushes issued prior to this command.
When this bit set following restriction apply to atomic operation:
Non-Compare atomic operations are supported on data granularity of 4B and 8B. DW3 is the lower dword of the operand and DW4 is the upper dword of the operand for the atomic operation.
Compare atomic operations are supported on data granularity of 4B. DW3 is Operand-0 and DW4 is Operand-1 for the atomic operation.
Atomic operations to GGTT/PPGTT memory surface are supported.
Only Inline data mode for atomic operand is supported, no support for indirect data mode.
No support for Return Data Control functionality.
No support for atomic operations on data granularity of 16B.</td></tr>
</table></td></tr>
</table>

		• No support for compare atomic operations on data granularity of 8B.

Programming Notes
Any desired pipeline flush operation can be achieved by programming PIPE_CONTROL command prior to this command.
When this bit is set Command Streamer sends a flush down the pipe and the atomic operation is saved as post sync operation. Command streamer goes on executing the following commands. Atomic operation saved as post sync operation is executed at some point later on completion of corresponding flush issued.
When this bit is set atomic semaphore signal operation will be out of order with rest of the MI commands programmed in the ring buffer or batch buffer, it will be in order with respect to the post sync operations resulting due to PIPE_CONTROL command.

20:19 **Data Size**

This field indicates the size of the operand in dword/qword/octword on which atomic operation will be performed. Data size must match with the Atomic Opcode. Operation Data size could be 4B, 8B or 16B

Value	Name	Description
0h	DWORD	Operand size used by Atomic Operation is DWORD.
1h	QWORD	Operand Size used by Atomic Operation is QWORD.
2h	OCTWORD	Operand Size used by Atomic Operation is OCTWORD.
3h	RESERVED	

18 **Inline Data**

This bit when set indicates the source operands are provided in line within the command. When reset the source operands are in CS_GPR registers.

Programming Notes
CS_GPR registers must be programmed with appropriate values before issuing MI_ATOMIC command with this field reset.

17 **CS STALL**

This bit when set command stream waits for completion of this command before executing the next command.

Programming Notes
Render Command Streamer Only: CS will not guarantee atomic operation to be complete upon setting this bit along with Post Sync Operation set. When Post Sync Operation is set, this bit has no significance.

16 **Return Data Control**

Source:	RenderCS, BlitterCS, VideoCS, VideoEnhancementCS

When Return Data Control is set the read return feature will be enabled during the atomic operation. Data is stored in CS_GPR5/4 registers unconditionally on completion of the atomic operation. On data return CS_GPR5/4 Registers are updated based on the "Data Size" field. When

		"Data Size" is DWORD lower dword of CS_GPR4 (Qword) is updated with the dword data returned from memory. When "Data Size" is QWORD only CS_GPR4 (Qword) is updated with the qword data returned from memory. When the data size is OCTWORD CS_GPR4/5 are updated with the OCTWORD data returned from memory. CS_GPR4 is loaded with lower qword returned from memory and CS_GPR5 is loaded with upper qword returned from memory.
	15:8	**ATOMIC OPCODE** This field selects the kind of atomic operation to be performed. Refer "Atomics" sub-section of "L3 Cache and URB" section for atomic opcode encoding and operation.

Value	Name
1h	Atomic_AND
2h	Atomic_OR
3h	Atomic_XOR
4h	Atomic_MOVE
5h	Atomic_INC
6h	Atomic_DEC
7h	Atomic_ADD
8h	Atomic_SUB
87h	Atomic_FADD
88h	Atomic_FSUB
9h	Atomic_RSUB
Ah	Atomic_IMAX
Bh	Atomic_IMIN
Ch	Atomic_UMAX
Dh	Atomic_UMIN
Eh	Atomic_CMP/WR
9Bh	Atomic_Min_Float16
9Ah	Atomic_Max_Float16
9Eh	Atomic_FLOATCMP/WR16
21h	Atomic_AND8B
22h	Atomic_OR8B
23h	Atomic_XOR8B
24h	Atomic_MOVE8B
25h	Atomic_INC8B
26h	Atomic_DEC8B
27h	Atomic_ADD8B
28h	Atomic_SUB8B
29h	Atomic_RSUB8B

		2Ah: Atomic_IMAX8B 2Bh: Atomic_IMIN8B 2Ch: Atomic_UMAX8B 2Dh: Atomic_UMIN8B 2Eh: Atomic_CMP/WR8B 8Ah: Atomic_MAX_Float32 8Bh: Atomic_MIN_Float32 8Eh: Atomic_CMP/WR_Float32
	7:0	**DWord Length** Format: =n Total Length - 2. Excludes DWord (0,1). 1h: Inline Data 0 **[Default]** ([Inline Data]==0) 9h: Inline Data 1 ([Inline Data]==1)
1	31:2	**Memory Address** Format: GraphicsAddress[31:2] This field contains the graphics memory address of the data on which atomic operation has to be performed. Atomic operation can be performed on data granularity of 4B, 8B or 16B and hence the Address has to be correspondingly aligned to 4B,8B or 16B respectively.
	1	**Reserved** Access: RO Format: MBZ
	0	**Reserved** Access: RO Format: MBZ
2	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Memory Address High** This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space.
3	31:0	**Operand1 Data Dword 0** Format: U32 Dword0 of Operand1 when Inline Data mode is set.
4	31:0	**Operand2 Data Dword 0** Format: U32 Dword0 of Operand2 when Inline Data mode is set.

MI_ATOMIC		
5	31:0	**Operand1 Data Dword 1** Format: U32 Dword1 of Operand1 when Inline Data mode is set.
6	31:0	**Operand2 Data Dword 1** Format: U32 Dword1 of Operand2 when Inline Data mode is set.
7	31:0	**Operand1 Data Dword 2** Format: U32 Dword2 of Operand1 when Inline Data mode is set.
8	31:0	**Operand2 Data Dword 2** Format: U32 Dword2 of Operand2 when Inline Data mode is set.
9	31:0	**Operand1 Data Dword 3** Format: U32 Dword3 of Operand1 when Inline Data mode is set.
10	31:0	**Operand2 Data Dword 3** Format: U32 Dword3 of Operand2 when Inline Data mode is set.

MI_BATCH_BUFFER_END

MI_BATCH_BUFFER_END

Source: CommandStreamer

Length Bias: 1

The MI_BATCH_BUFFER_END command is used to terminate the execution of commands stored in a batch buffer initiated using a MI_BATCH_BUFFER_START command.

Programming Notes

SW must ensure it buffers the size of the command buffer beyond this command. The command buffer for the command streamer is 0.5KB.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 0Ah MI_ BATCH_BUFFER_END Format: OpCode
	22:16	**Reserved** Access: RO Format: MBZ
	15	**Reserved** Access: RO Format: MBZ
	14:1	**Reserved** Access: RO Format: MBZ
	0	**End Context** Format: Enable This bit must only be set within a context image. If this command is parsed with this bit set then the engine will consider the context image restore complete. This bit is ignored if parsed during a batch buffer.

MI_BATCH_BUFFER_START

MI_BATCH_BUFFER_START

Source: CommandStreamer

Length Bias: 2

The MI_BATCH_BUFFER_START command is used to initiate the execution of commands stored in a batch buffer. For restrictions on the location of batch buffers, see Batch Buffers in the Device Programming Interface chapter of *MI Functions*. The batch buffer can be specified as privileged or non-privileged, determining the operations considered valid when initiated from within the buffer and any attached (chained) batch buffers. See Batch Buffer Protection in the Device Programming Interface chapter of *MI Functions*.

Programming Notes

- A batch buffer initiated with this command must end either with a MI_BATCH_BUFFER_END command or by chaining to another batch buffer with an MI_BATCH_BUFFER_START command.
- It is essential that the address location beyond the current page be populated inside the GTT. HW performs over-fetch of the command addresses and any over-fetch requires a valid TLB entry. A single extra page beyond the batch buffer is sufficient. Otherwise the driver can check if the current page has enough space for the size of the overfetch.

SW must ensure it buffers the size of the batch buffer includes additional buffer equal to the command buffer beyond the end. The command buffer for the command streamer is 0.5KB.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 31h MI_BATCH_BUFFER_START Format: OpCode
	22	**Second Level Batch Buffer** Exists If: //MI_MODE:NestedBatchBufferEnable=='0' The command streamer contains three storage elements; one for the ring head address, one for the batch head address, and one for the second level batch head address. When performing batch buffer chaining, hardware simply updates the head pointer of the first level batch address storage. There is no stack in hardware. When this bit is set, hardware uses the 2nd level batch head address storage element. Chaining of second level batch buffers is supported. A chained second level batch buffer is inferred in hardware when aMI_BATCH_BUFFER_START command with "Second Level Batch Buffer" bit field set is executed from a second level batch buffer, hardware simply updates the head pointer of the second level batch address storage. Upon MI_BATCH_BUFFER_END, it will automatically return to the first level batch buffer address. This allows hardware to mimic a simple 3-level stack. Value / Name / Description

0h	First level batch	Place the batch buffer address in the 1st (traditional) level batch address storage element.
1h	Second level batch	Place the batch buffer address in the second-level batch address storage element.

Programming Notes

This field is ignored when MI_BATCH_BUFFER_START is executed from a ring. Whether this is a zero or one, the command streamer will move to the first level batch buffer.

22 **Nested Level Batch Buffer**

Exists If:	//MI_MODE:NestedBatchBufferEnable=='1'

Description

If this bit is set, the command streamer will move to the next level of batch buffer. Once it executes a MI_BATCH_BUFFER_END in the next level, it will return to the batch buffer executing this command and execute the next command.
If clear then it will remain in the same batch buffer level and on executingMI_BATCH_BUFFER_END, it will return to the previous level. Otherwise known as batch buffer chaining.
Hardware supports three levels of nesting, namely First Level, Second Level and Third Level.
This bit must not be set in any of the MI_BATCH_BUFFER_START commands programmed as part of the 3rd level batch buffer's command sequence.

Value	Name	Description
0h	Chain	Stay in the same batch buffer level.
1h	Nested	Move to the next level of batch buffer.

Programming Notes

This field is ignored when MI_BATCH_BUFFER_START is executed from a ring. Whether this is a zero or one, the command streamer will move to the first level batch buffer.

Following programming guidelines must be follow for programming commands in third level batch buffer:

- Preemptable commands must not be programmed inside third level batch buffer. Refer section Preemption/Execlist Scheduling (**Preemptable Commands**) for the list of preemptable commands supported on per engine basis. Preemptable commands for RenderCS are mentioned below.
  - 3DPRIMTIVE and MI_ARB_CHK command in 3D mode of PIPELINE_SELECT operation.
  - GPGPU_WALKER, MEDIA_WALKER, MEDIA_OBJECT, PIPE_CONTROL, MI_ATOMIC (bit 21 set)and MEDIA_STATE_FLUSH in GPGPU or MEDIA

mode of PIPELINE_SELECT operation.

- Any Non-Pipeline state command in GPGPU mode of PIPELINE_SELECT operation.

- MI_SEMAPHORE_WAIT and MI_WAIT_FOR_EVENT commands must not be programmed inside third level batch buffers. These commands are preemptable and also can result in context switch and hence must not be programmed inside third level batch buffer.

21 **POSH Start**

Source:	RenderCS, PositionCS

Batch buffers dedicated to be executed by POSH pipe are indicated by setting the field POSH Start in the MI_BATCH_BUFFER_START command header. Once POSH Start is set in a batch buffer all the following chained batch buffers and next level batch buffers will implicitly inherit the POSH Start field value. Once POSH Start is set in a batch buffer all the following command sequences are to be **executed** by POCS until the corresponding batch buffer sequencing is terminated through MI_BATCH_BUFFER_END/MI_CONDITIONAL_BATCH_BUFFER_END command.
Example:

- Once POSH Start is encountered in a first level batch buffer by POCS, it will get reset only when the first level batch buffer execution is terminated through batch buffer end and the command execution sequence goes back to the ring buffer,
- Similarly, once POSH Start is encountered in a second level batch buffer by POCS, it will get reset only when the second level batch buffer execution is terminated through batch buffer end and the command execution sequence goes back to the first level buffer,
- Similarly, once when POSH Start is encountered in a third level batch buffer by POCS, it will get reset only when the third level batch buffer execution is terminated through batch buffer end and the command execution sequence goes back to the second level batch buffer.

Command sequences executed from the POSH Start batch buffer may lead to chained batch buffers or next level batch buffers. Batch buffers executed by POCS may have MI Commands, 3DSATE commands and 3DPRIMTIVE commands for POSH pipe. RCS on parsing MI_BATCH_BUFFER_START command with POSH Start enabled NOOPS the command and moves on the following command.

Value	Name	Description
0	**[Default]**	Batch buffer is not "POSH Start" enabled.
1		Batch buffer is "POSH Start" enabled.

Programming Notes
Resource Streamer must not be enabled for "POSH Enable" or "POSH Start" batch buffers.

20 **POSH Enable**

Source:	RenderCS, PositionCS

POSH Enable field in the MI_BATCH_BUFFER_START command is a hint to POCS to traverse (parse, don't execute) the batch buffer to look for POSH Start batch buffers. POSH Enable field is only inherited to the chained batch buffer and doesn't get inherit to the next level batch buffers unlike POSH Start field. POSH Enable field must be explicitly set in the MI_BATCH_BUFFER_START command which calls the next level batch buffers in order for the POCS to parse them to look for POSH Start batch buffers. POSH Start field takes precedence over the POSH Enable field in POCS.
Example:

- Once POSH Enable is encountered in a first level batch buffer, POCS will traverse the whole of the first level batch buffers (including chained first level) to check for POSH Start field in MI_BATCH_BUFFER_START command. POCS by default will not traverse the second level batch buffers. SW must explicitly set the POSH Enable field for the second level batch buffer called from first level batch buffer if the second level batch buffer have to be traversed by POCS.
- Similarly, Once POSH Enable is encountered in a second level batch buffer, POCS will traverse the whole of the second level batch buffers (including chained second level) to check for POSH Start field in MI_BATCH_BUFFER_START command. POCS by default will not traverse the third level batch buffers. SW must explicitly set the POSH Enable field for the third level batch buffer called from second level batch buffer if the third level batch buffer have to be traversed by POCS.
- Similarly, Once POSH Enable is encountered in a third level batch buffer, POCS will traverse the whole of the third level batch buffers (including chained second level) to check for POSH Start field in MI_BATCH_BUFFER_START command.

RCS ignores POSH Enable field and has no implications due to the POSH Enable field set in the MI_BATCH_BUFFER_START command.

Value	Name	Description
0	**[Default]**	Batch buffer is not "POSH Enable".
1		Batch buffer is "POSH Enable"/

Programming Notes
POCS executes only the MI_BATCH_BUFER_START commands programmed in the ring buffer with POSH Enable set and NOOPS (predicates) all the other commands in the ring buffer. POCS only parses/traverses the batch buffer with POSH Enable to check for any batch buffer programmed with POSH Start set. SW must set POSH Enable field in the MI_BATCH_BUFFER_START command programmed in ring buffer if the commands in the corresponding batch buffer or the chained batch buffers (includes Second Level and third level) has at least one batch buffer start command with POSH Start set (also implies 3DPRIMITIVE command for which POSH is enabled).
Resource Streamer must not be enabled for "POSH Enable" or "POSH Start" batch

buffers.

19 **Enable Command Cache**

Source:	RenderCS

Description

Command Buffer DMA engine on processing the MI_BATCH_BUFFER_START command with Enable Command Cache set will make the corresponding command buffer read requests cacheable in L3, it also applies the command caching to all subsequent chained and next level batch buffers. Command buffer DMA engine uses the Command Buffer Cache Size programmed in the CMD_BUF_CCTL register to limit the read requests of a cacheable batch buffer to be cached in L3. DMA engine does this by tracking the amount of read requests made cacheable and stops caching when the read requested data size equals to the size of the command cache allocated. This avoids thrashing of cache for Batch Buffers larger in size compared to the command buffer cache allocated in L3. DMA engine resets the command caching tracker on events listed below.

- On an End Of Tile in PTRBR/POSH mode of operation.
- On a context save of a context.
- On command cache invalidation through PIPE_CONTROL.

Command buffer caching must not be enabled for batch buffers in GGTT address space.
For best performance in terms of utilizing the command buffer section of L3 cache for PTBR mode, the recommendation is to ensure command buffers not to cross 128KB boundary. Avoiding a large overlap around 128KB boundary will minimize the cross distribution of 1KB chunks randomly between the L3 banks creating hotspots otherwise.

Command Buffer DMA engine on processing the MI_BATCH_BUFFER_START command with Enable Command Cache set will make the corresponding command buffer read requests cacheable in L3, it also applies the command caching to all subsequent chained and next level batch buffers. Command buffer DMA engine uses the Command Buffer Cache Size programmed in the CMD_BUF_CCTL register to limit the read requests of a cacheable batch buffer to be cached in L3. DMA engine does this by tracking the amount of read requests made cacheable and stops caching when the read requested data size equals to the size of the command cache allocated. This avoids thrashing of cache for Batch Buffers larger in size compared to the command buffer cache allocated in L3. DMA engine resets the command caching tracker on events listed below.

- On an End Of Tile in PTRBR mode of operation.
- On a context save of a context.
- On command cache invalidation through PIPE_CONTROL.

Command buffer caching must not be enabled for batch buffers in GGTT address space.

For best performance in terms of utilizing the command buffer section of L3 cache for PTBR mode, the recommendation is to ensure command buffers not to cross 128KB boundary. Avoiding a large overlap around 128KB boundary will minimize the cross distribution of 1KB chunks randomly between the L3 banks creating hotspots otherwise.

Value	Name	Description
1		Command Cache enabled
0	**[Default]**	Command Cache disable

18 **Reserved**

Access:	RO
Format:	MBZ

17:16 **Reserved**

Access:	RO
Format:	MBZ

15 **Predication Enable**

Source:	RenderCS, PositionCS, BlitterCS, VideoCS, VideoEnhancementCS, ComputeCS

This bit is used to enable predication of this command. If this bit is set and Bit 0 of the MI_PREDICATE_RESULT_1 register is clear, this command is ignored. Otherwise the command is performed normally.

14:10 **Reserved**

Access:	RO
Format:	MBZ

9 **Reserved**

8 **Address Space Indicator**

Description
Batch buffers accessed via PPGTT are considered as non-privileged. Certain operations (e.g., MI_STORE_DATA_IMM commands to GGTT memory) are prohibited within non-privileged buffers. More details mentioned in User Mode Privileged command section. When MI_BATCH_BUFFER_START command is executed from within a batch buffer (i.e., is a "chained" or "second level" batch buffer command), the current active batch buffer's "Address Space Indicator" and this field determine the "Address Space Indicator" of the next buffer in the chain. • Chained or Second level batch buffer can be in GGTT or PPGTT if the parent batch buffer is in GGTT. • Chained or Second level batch buffer can only be in PPGTT if the parent batch buffer is in PPGTT. This is enforced by Hardware.
Batch buffers accessed via PPGTT are considered as non-privileged. Certain

operations (e.g., MI_STORE_DATA_IMM commands to GGTT memory) are prohibited within non-privileged buffers. More details mentioned in User Mode Privileged command section. When MI_BATCH_BUFFER_START command is executed from within a batch buffer (i.e., is a "chained" or "second level" batch buffer command), the current active batch buffer's "Address Space Indicator" and this field determine the "Address Space Indicator" of the next buffer in the chain.

- Chained or Nested level batch buffer can be in GGTT or PPGTT if the parent batch buffer is in GGTT.
- Chained or Nested level batch buffer can only be in PPGTT if the parent batch buffer is in PPGTT. This is enforced by Hardware.

Value	Name	Description
0h	GGTT	This batch buffer is located in GGTT memory and is privileged.
1h	PPGTT	This batch buffer is located in PPGTT memory and is Non-Privileged.

Programming Notes
This field must be '0' unless the Per-Process GTT Enable is '1'

7:0 **DWord Length**

Default Value:	1h Excludes DWord (0,1)
Format:	=n

Total - Bias. Excludes DWord (0,1).

1..2 The address is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper 63 down to 48 bits are reserved bits which are ignored.

63:2 **Batch Buffer Start Address**

Format:	VIRTUAL_ADDR[63:2]

1:0 **Reserved**

Access:	RO
Format:	MBZ

MI_CONDITIONAL_BATCH_BUFFER_END

MI_CONDITIONAL_BATCH_BUFFER_END

Source: CommandStreamer

Length Bias: 2

Description

The MI_CONDITIONAL_BATCH_BUFFER_END command is used to conditionally terminate the execution of commands stored in a batch buffer initiated using a MI_BATCH_BUFFER_START command.
Termination of the current level of batch buffer from which MI_CONDITIONAL_BATCH_BUFFER_END is executed or termination of all levels of batch buffer behavior is controlled by the **End Current Batch Buffer Level** bit in the command header.

Programming Notes

Any updates to the memory location exercised by this command must be ensured to be coherent in memory prior to programming of this command. If the memory location is being updated by a prior executed MI command (ex: MI_STORE_REGISTER_MEM ..etc) on the same engine, SW must follow one of the below programming note prior to programming of this command to ensure data is coherent in memory.
Option1: Programming of "4" dummy MI_STORE_DATA_IMM (write to scratch space) commands prior to programming of this command. Example: MI_STORE_REGISTE_MEM (0x2288, 0x2CF0_0000)
MI_STORE_DATA_IMM (4 times) (Dummy data, Scratch Address)
MI_CONDITIONAL_BATCH_BUFFER_END(0x2CF0_0000)
Option2: Programming of a PIPE_CONTROL with Post-Sync Operation selected to Write Immediate Data to scratch space address with Command Streamer Stall Enable set prior to programming of this command. Example: MI_STORE_REGISTE_MEM (0x2288, 0x2CF0_0000) PIPE_CONTROL (Stall, Write Immediate Data), MI_CONDITIONAL_BATCH_BUFFER_END(0x2CF0_0000) .
Option3: MI_ATOMIC (write to scratch space) with "CS STALL" set prior to programming of this command. Example: MI_STORE_REGISTE_MEM (0x2288, 0x2CF0_0000) MI_ATOMIC (MOV, Dummy data, Scratch Address), MI_CONDITIONAL_BATCH_BUFFER_END(0x2CF0_0000) .

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 36h MI_CONDITIONAL_BATCH_BUFFER_END Format: OpCode
	22	**Use Global GTT** Default Value: 0h Format: Boolean If set, this command will use the global GTT to translate the **Compare Address** and this command must be executing from a privileged (secure) batch buffer. If clear, the PPGTT will be used to translate the **Compare Address**.

MI_CONDITIONAL_BATCH_BUFFER_END

	21	**Compare Semaphore** Format: Boolean This bit provides a means to enable or disable compare operation. **Value / Name / Description** 1h — The data from the Compare Data Dword (inline) is compared to the data in memory pointed by the Compare Address as per the Compare Operation. Based on the outcome of the compare operation it may result in either continue execution of the batch buffer or it may result in termination of the batch buffer. 0h — This command will be treated as NOOP and usual batch buffer execution flow continues.
	20	**Reserved**
	19	**Compare Mask Mode** **Value / Name / Description** 0h — Compare Mask Mode Disabled — Compare address points to Dword in memory consisting of Data Dword(DW0). HW will compare Data Dword(DW0) against Semaphore Data Dword. 1h — Compare Mask Mode Enabled — Compare address points to Qword in memory consisting of compare Mask (DW0) and Data Dword(DW1). HW will do AND operation on Mask(DW0) with Data Dword(DW1) and then compare the result against Semaphore Data Dword. **Programming Notes** When "Compare Mask Mode" is enabled, "Compare Address" must be qword aligned.
	18	**End Current Batch Buffer Level** This field specifies if the current level of the batch buffer execution must or the complete batch (including parent) buffer execution must be terminated on compare operation evaluating false. **Value / Name / Description** 1 — Execution of the command results in terminating the batch buffer level from which this command has been executed and command execution returns to the previous/parent batch buffer. Ex: • when executed from a first level batch buffer, execution of batch buffer is terminated and the command execution resumes to the ring buffer. This is similar to as if MI_BATCH_BUFFER_END command was executed from first level batch buffer. • when executed from a second level batch buffer, execution of second level batch buffer is terminated and

			the command execution resumes to the first level batch buffer. This is similar to as if MI_BATCH_BUFFER_END command was executed from second level batch buffer. • when executed from a third level batch buffer (if supported), execution of third level batch buffer is terminated and the command execution resumes to the second level batch buffer. This is similar to as if MI_BATCH_BUFFER_END command was executed from third level batch buffer.
		0		Execution of the command result in termination of all levels of batch buffer and command execution returns to the ring buffer.

17:16	**Reserved**
	Access: RO
	Format: MBZ

15	**Reserved**
	Access: RO
	Format: MBZ

14:12 **Compare Operation**

This field specifies the operation that will be executed to create the result that will either allow to continue or terminate the batch buffer.

MAD = Memory Address Data Dword

IDD =Inline Data Dword

Value	Name	Description
0h	MAD_GREATER_THAN_IDD	If Indirect fetched data is greater than inline data then continue.
1h	MAD_GREATER_THAN_OR_EQUAL_IDD	If Indirect fetched data is greater than or equal to inline data then continue.
2h	MAD_LESS_THAN_IDD	If Indirect fetched data is less than inline data then continue.
3h	MAD_LESS_THAN_OR_EQUAL_IDD	If Indirect fetched data is less than or equal to inline data then continue.
4h	MAD_EQUAL_IDD	If Indirect fetched data is equal to inline data then continue.
5h	MAD_NOT_EQUAL_IDD	If Indirect fetched data is not equal to inline data then continue.
6h	Reserved	
7h	Reserved	

MI_CONDITIONAL_BATCH_BUFFER_END

	11:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) Format: =n
1	31:0	**Compare Data Dword** Data dword to compare memory. The Data dword is supplied by software to control execution of the command buffer. If the compare is enabled and the data at Compare Address is greater than this dword, the execution of the command buffer should continue.
2..3 Qword address to fetch Data Qword from memory. This field specifies the 4GB aligned base address of Gfx 4GB virtual address space within the host's 64-bit virtual address space. Virtual Address is a 64-bit value [63:0], but only a portion of it is used by hardware. Upper reserved bits are ignored and MBZ.	63:3	**Compare Address** Format: VIRTUAL_ADDR[63:3]
	2:0	**Reserved** Access: RO Format: MBZ

MI_COPY_MEM_MEM

MI_COPY_MEM_MEM

Source: BlitterCS

Length Bias: 2

The MI_COPY_MEM_MEM command reads a DWord from memory and stores the value of that DWord to back to memory. The source and destination addresses are specified in the command. The command temporarily halts command execution.

Programming Notes

This command should not be used within a "non_privilege" batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation. This command can be used within ring buffers and/or privilege batch buffers to access global virtual space.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 2Eh MI_COPY_MEM_MEM Format: OpCode
	22	**Use Global GTT Source** It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear. 0h – Per Process Graphics Address 1h – Global Graphics Address: This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
	21	**Use Global GTT Destination** It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear. 0h – Per Process Graphics

MI_COPY_MEM_MEM

			Address	
			1h	Global Graphics Address	This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
	20:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 3 Excludes DWord (0,1) Format: =n
1..2 Surface Type: MMIO Register Virtual Address is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and MBZ.	63:0	**Destination Memory Address** Format: VIRTUAL_ADDR[63:0]
3..4 Surface Type: MMIO Register Virtual Address is a 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and MBZ.	63:0	**Source Memory Address** Format: VIRTUAL_ADDR[63:0]

MI_COPY_MEM_MEM

MI_COPY_MEM_MEM

Source: RenderCS

Length Bias: 2

The MI_COPY_MEM_MEM command reads a DWord from memory and stores the value of that DWord to back to memory. The source and destination addresses are specified in the command. The command temporarily halts command execution.

Programming Notes

This command should not be used within a "non_privilege" batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation. This command can be used within ring buffers and/or privilege batch buffers to access global virtual space.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 2Eh MI_COPY_MEM_MEM Format: OpCode
	22	**Use Global GTT Source** It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear. 0h – Per Process Graphics Address 1h – Global Graphics Address: It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear.
	21	**Use Global GTT Destination** This bit will be ignored and treated as if clear when executing from a non-privileged batch buffer. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer. This bit *must* be '1' if the **Per Process GTT Enable** bit is clear. This bit will determine write to memory uses Global or Per Process GTT. 0h – Per Process Graphics Address

		1h: Global Graphics Address — This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
	20:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 3 Excludes DWord (0,1) Format: =n
1..2	63:2	**Destination Memory Address** Format: GraphicsAddress[63:2] Surface Type: MMIO Register This field specifies the address of the memory location where the value fetched specified in the DWord address above will be written. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[63:2] for a DWord register GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	1:0	**Reserved** Access: RO Format: MBZ
3..4	63:2	**Source Memory Address** Format: GraphicsAddress[63:2] Surface Type: MMIO Register This field specifies the address of the memory location where the value is located that will be written to the address below. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[63:2] for a DWord register GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	1:0	**Reserved** Access: RO Format: MBZ

MI_COPY_MEM_MEM

MI_COPY_MEM_MEM

Source: VideoCS

Length Bias: 2

The MI_COPY_MEM_MEM command reads a DWord from memory and stores the value of that DWord to back to memory. The source and destination addresses are specified in the command. The command temporarily halts command execution.

Programming Notes

This command should not be used within a "non-privileged" batch buffer to access global virtual space; doing so will be treated as privilege access violation. Refer to the "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to learn more about HW behavior on encountering a privilege access violation.
This command can be used within ring buffers and/or privilege batch buffers to access global virtual space.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 2Eh MI_MEM_TO_MEM Format: OpCode
	22	**Use Global GTT Source** It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear. 0h – Per Process Graphics Address 1h – Global Graphics Address: This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
	21	**Use Global GTT Destination** It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear. 0h – Per Process Graphics Address 1h – Global Graphics Address: This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch

MI_COPY_MEM_MEM

		buffer.
	20:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 3 Excludes DWord (0,1) Format: =n
1..2	63:2	**Destination Memory Address** Format: VIRTUAL_ADDR[63:2] Surface Type: MMIO Register
	1:0	**Reserved** Access: RO Format: MBZ
3..4	63:2	**Source Memory Address** Format: VIRTUAL_ADDR[63:2] Surface Type: MMIO Register
	1:0	**Reserved** Access: RO Format: MBZ

MI_COPY_MEM_MEM

MI_COPY_MEM_MEM

Source: VideoEnhancementCS

Length Bias: 2

The MI_COPY_MEM_MEM command reads a DWord from memory and stores the value of that DWord to back to memory. The source and destination addresses are specified in the command. The command temporarily halts command execution.

Programming Notes

This command should not be used within a "non-privileged" batch buffer to access global virtual space; doing so will be treated as privilege access violation. Refer to the "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to learn more about HW behavior on encountering a privilege access violation.
This command can be used within ring buffers and/or privilege batch buffers to access global virtual space.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 2Eh MI_MEM_TO_MEM Format: OpCode
	22	**Use Global GTT Source** It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear. 0h – Per Process Graphics Address 1h – Global Graphics Address: This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
	21	**Use Global GTT Destination** It is allowed for this bit to be set when executing this command from a privileged (secure) batch buffer or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear. 0h – Per Process Graphics Address 1h – Global Graphics Address: This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch

MI_COPY_MEM_MEM

		buffer.
	20:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 3 Excludes DWord (0,1) Format: =n
1..2	63:2	**Destination Memory Address** Format: VIRTUAL_ADDR[63:2] Surface Type: MMIO Register
	1:0	**Reserved** Access: RO Format: MBZ
3..4	63:2	**Source Memory Address** Format: VIRTUAL_ADDR[63:2] Surface Type: MMIO Register
	1:0	**Reserved** Access: RO Format: MBZ

MI_DISPLAY_FLIP

MI_DISPLAY_FLIP

Source:	RenderCS, BlitterCS
Length Bias:	2

The MI_DISPLAY_FLIP command is used to request a specific display plane to switch (flip) to display a new buffer. The buffer is specified with a starting address and pitch. The tiled attribute of the buffer start address is programmed as part of the packet.
The operation this command performs is also known as a "display flip request" operation - in that the flip operation itself will occur at some point in the future. This command specifies when the flip operation is to occur: either synchronously with vertical retrace to avoid tearing artifacts (possibly on a future frame), or asynchronously (as soon as possible) to minimize rendering stalls at the cost of tearing artifacts.

Programming Notes

1. This command simply requests a display flip operation. Command execution then continues normally. There is no guarantee that the flip (even if asynchronous) will occur prior to subsequent commands being executed. (Note that completion of the MI_FLUSH command does not guarantee that outstanding flip operations have completed). The MI_WAIT_FOR_EVENT command can be used to provide this synchronization - by pausing command execution until a pending flip has actually completed. This synchronization can also be performed by use of the Display Flip Pending hardware status. See Display Flip Synchronization in the Device Programming Interface chapter of MI Functions.
2. After a display flip operation is requested, software is responsible for initiating any required synchronization with subsequent buffer clear or rendering operations. For multi-buffering (e.g., double buffering) operations, this will typically require updating SURFACE_STATE or the binding table to change the rendering (back) buffer. In addition, prior to any subsequent clear or rendering operations, software must typically ensure that the new rendering buffer is not actively being displayed. Again, the MI_WAIT_FOR_EVENT command or Display Flip Pending hardware status can be used to provide this synchronization. See Display Flip Synchronization in the Device Programming Interface chapter of MI Functions.
3. The display buffer command uses the X and Y offset for the tiled buffers from the Display Interface registers. Software is allowed to change the offset via the MMIO interface irrespective of the flip commands enqueued in the command stream. For tiled buffers, the display subsystem uses the X and Y offset in generation of the final request to memory. The offset is always updated on the next vblank for both Synchronous and Asynch Flips. It is not necessary to have a flip enqueued to update the X and Y offset.
4. The display buffer command uses the linear dword offset for the linear buffers from the Display Interface registers. Software is allowed to change the offset via the MMIO interface irrespective of the flip commands enqueued in the command stream. For linear buffers, the display subsystem uses the dword offset in generation of the final request to memory.
    - For synchronous flips the offset is updated on the next vblank. It is not necessary to have a sync flip enqueued to update the dword offset.
5. DWord 3 (Left Eye Display Buffer Base Address) must not be set with synchronous flips or asynchronous flips. It is only allowed to be sent with stereo 3D flips.

Asynchronous flip completion time depends greatly on how much data has been prefetched for power savings,

MI_DISPLAY_FLIP

and can take up to 1 full frame to complete. For faster flip completion, disable FBC and render compression and allocate a small amount of data buffer for the plane.

"Command Streamer Plane Number" mapping to "Display Plane Name" are listed in display B-spec -"Plane capability and Interoperability".

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 14h MI_DISPLAY_FLIP Format: OpCode
	22	**Async Flip Indicator** Format: Enable This bit should always be set if DW2 [1:0] == '01' (async flip). This field is required due to HW limitations. This bit is used by the render pipe while DW2 is used by the display hardware.
	21:19	**Reserved** Access: RO Format: MBZ
	18:17	**Reserved**
	16:14	**Reserved** Access: RO Format: MBZ
	13:8	**Display Plane Select** Value / Name: 0h Display Plane 1 1h Display Plane 2 2h Display Plane 3 3h Reserved 4h Display Plane 4 5h Display Plane 5 6h Display Plane 6 7h Display Plane 7 8h Display Plane 8 9h Display Plane 9 Ah Display Plane 10 Bh Display Plane 11

Ch	Display Plane 12
Dh	Display Plane 13
Eh	Display Plane 14
Fh	Display Plane 15
10h	Display Plane 16
11h	Display Plane 17
12h	Display Plane 18
13h	Display Plane 19
14h	Display Plane 20
15h	Display Plane 21
16h	Display Plane 22
17h	Display Plane 23
18h	Display Plane 24
19h	Display Plane 25
1Ah	Display Plane 26
1Bh	Display Plane 27
1Ch	Display Plane 28
1Dh	Display Plane 29
1Eh	Display Plane 30
1Fh	Display Plane 31
20h	Display Plane 32
[21h, 3Fh]	Reserved

7:0 **DWord Length**

Format:	=n

Total Length - 2. Excludes DWord (0,1).
For Synchronous Flips and Asynchronous Flips, this field must be programmed to 1h for a total length of 3.
For Stereo 3D Flips, this field must be programmed to 2h for a total length of 4.

Value	Name	Exists If
1h	**[Default]**	([Flip Type]!='Stereo 3D Flip')
2h	**[Default]**	([Flip Type]=='Stereo 3D Flip')

1 31 **Stereoscopic 3D Mode**

Default Value:	0h
Format:	Enable

This bit must be set if the Extra Display Buffer Address is part of this command. This bit is used to notify the display there is an extra DW before processing the Display Flip.

	30:16	**Reserved** Access: RO Format: MBZ
	15:6	**Display Buffer Pitch** Default Value: 0h Format: U10 *For Synchronous Flips and Stereo 3D Flips only*, this field specifies the pitch of the new display buffer. For Asynchronous Flips, this parameter is programmed so that all the flips in a flip chain should maintain the same pitch as programmed with the last synchronous flip or stereo 3D flip or direct through MMIO. See the Display Plane Stride register for details.
	5:3	**Reserved** Access: RO Format: MBZ
	2:0	**Tile Parameter** Format: Enable For Asynchronous Flips, this parameter cannot be changed. All the flips in a flip chain should maintain the same tile parameter as programmed with the last synchronous flip or direct through MMIO.

Value	Name	Description
0h	Linear **[Default]**	For Synchronous Flips Only
1h	Tiled X	
2h-3h	Reserved	
4h	Tiled Y Legacy (Y B)	

2	31:12	**Display Buffer Base Address** Format: GraphicsAddress[31:12] This field specifies Bits 31:12 of the Graphics Address of the new display buffer. In stereo 3D mode this is the right eye base address. In non-stereo 3D mode this is the only base address. (Refer to the Display Address Start Address Register description in the Display Registers chapter).

Programming Notes

- The Display buffer must reside completely in Main Memory.
- This address is always translated via the global (rather than per-process) GTT.

	11	**Reserved** Access: RO Format: MBZ
	10:7	**Reserved** Access: RO Format: MBZ

	6:4	**Reserved**
	3	**Reserved** Access: RO Format: MBZ
	2	**Reserved**
	1:0	**Flip Type** This field specifies whether the flip operation should be performed asynchronously to vertical retrace. **Value** / **Name** / **Description** 00b / Sync Flip **[Default]** / The flip will occur during the vertical blanking interval - thus avoiding any tearing artifacts. 01b / Async Flip / The flip will occur "as soon as possible" - and may exhibit tearing artifacts 10b / Stereo 3D Flip / The flip will occur during the vertical blanking interval (left or right eye blank selectable through display MMIO register) - thus avoiding any tearing artifacts. 11b / Reserved / **Programming Notes** • The Display Buffer Pitch and Tile parameter cannot be changed for asynchronous flips (i.e., the new buffer must have the same pitch/tile format as the previous buffer). Only display surface base address can be changed. • For Asynch Flips the Buffers used must be 32KB aligned. • Asynch flips are supported on primary or universal planes only. • For Stereo 3D flips, both left and right eye buffers must have the same pitch and tile format.
3	31:12	**Left Eye Display Buffer Base Address** Format: GraphicsAddress[31:12] This field specifies Bits 31:12 of the Graphics Address of the new display buffer for the stereo 3D left eye. In non-stereo 3D mode this address is not used. (Refer to the Display Address Start Address Register description in the Display Registers chapter). **Programming Notes** • The Display buffer must reside completely in Main Memory. • This address is always translated via the global (rather than per-process) GTT.
	11:0	**Reserved** Access: RO Format: MBZ

MI_FLUSH_DW

MI_FLUSH_DW

Source: BlitterCS

Length Bias: 2

The MI_FLUSH_DW command is used to perform an internal "flush" operation. The parser pauses on an internal flush until all drawing engines have completed any pending operations. In addition, this command can also be used to: Flush any dirty data to memory. Invalidate the TLB cache inside the hardware

Usage note: After this command is completed with a Store DWord enabled, CPU access to graphics memory will be coherent (assuming the Render Cache flush is not inhibited).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 26h MI_FLUSH_DW Format: OpCode
	22	**Reserved** Access: RO Format: MBZ
	21	**Store Data Index** Format: U1 This field is valid only if the post-sync operation is not 0. If this bit is set, the store data address is index into the global hardware status page when destination address type in the command is set to 1 (GGTT). The store data address is index into the per-process hardware status page when destination address type in the command is set to 0 (PPGTT).
	20:19	**Reserved** Access: RO Format: MBZ
	18	**TLB Invalidate** Format: U1 If ENABLED, all TLBs belonging to Blitter Engine will be invalidated once the flush operation is complete. This bit is only valid when the Post-Sync Operation field is a value of 1h or 3h. If GFX_MODE (0x229c) bit 13, this command will cause a

config write to MMIO register space with the address 0x4f100.

17 **Reserved**

Access:	RO
Format:	MBZ

16 **Reserved**

Access:	RO
Format:	MBZ

15:14 **Post-Sync Operation**

BitFieldDesc

Value	Name	Description
0h	No Write	No write occurs as a result of this instruction. This can be used to implement a "trap" operation, etc.
1h	Write Immediate Data QWord	Write the QWord containing Immediate Data Low, High DWs to the Destination Address
2h	Reserved	Reserved
3h	Write TIMESTAMP Register	Write the TIMESTAMP register to the Destination Address. The upper 28 bits of the TIMESTAMP register are tied to '0'.

Programming Notes

If executed in a non-secure batch buffer, the address given is in a PPGTT address space. If in a secure ring or batch, the address given is in GGTT space.

13:10 **Reserved**

Access:	RO
Format:	MBZ

9 **Flush LLC**

Format:	Enable

If enabled, at the end of the current MI_FLUSH_DW the last level cache is cleared of all the cachelines which have been determined as being part of the Frame Buffer.

8 **Notify Enable**

Format:	U1

If ENABLED, a Sync Completion Interrupt will be generated (if enabled by the MI Interrupt Control registers) once the

MI_FLUSH_DW

		sync operation is complete. See Interrupt Control Registers in Memory Interface Registers for details.
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**DWord Length** Default Value: 3h DWORD_COUNT_n Format: =n Total Length - 2. Excludes DWord (0,1).
1..2 This field specifies the destination address where the DWord or QWord will be stored. Note that the address can only be QWord aligned, irrespective of data size. GraphicsAddress is 64-bit value [63:0], but only a portion of it is used by hardware. The upper reserved bits are ignored and MBZ. Some GraphicsAddress fields only specify the upper address bits. For example GraphicsAddress[47:12] would be a 4KB page address.	63:48	**Reserved** Access: RO Format: MBZ
	47:3	**Destination Address** Format: GraphicsAddress[47:3]
	2	**Destination Address Type** Defines the address space of the Destination Address. Value / Name / Description 0h / PPGTT / Use PPGTT address space for DW write 1h / GGTT / Use GGTT address space for DW write **Programming Notes** Ignored if "No write" is the selected in Operation.
	1:0	**Reserved** Access: RO Format: MBZ
3..4	63:0	**Immediate Data** This field specifies the DWord value to be written to the targeted location. Only valid when 15:14 in header is set to 1h. **Programming Notes** To avoid hitting a known hardware bug, drivers cannot send a QW write when bit 5 of the address is '1'

MI_FLUSH_DW

MI_FLUSH_DW

Source: VideoCS

Length Bias: 2

The MI_FLUSH_DW command is used to perform an internal "flush" operation. The parser pauses on an internal flush until all drawing engines have completed any pending operations. In addition, this command can also be used to :Flush any dirty data to memory. Invalidate the TLB cache inside the hardware Usage note: After this command is completed with a Store DWord enabled, CPU access to graphics memory will be coherent (assuming the Render Cache flush is not inhibited).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 26h MI_FLUSH_DW Format: OpCode
	22	**Reserved**
	21	**Store Data Index** Format: U1 This field is valid only if the post-sync operation is not 0. If this bit is set, the store data address is index into the global hardware status page when destination address type in the command is set to 1 (GGTT). The store data address is index into the per-process hardware status page when destination address type in the command is set to 0 (PPGTT).
	20:19	**Reserved** Access: RO Format: MBZ
	18	**TLB Invalidate** Format: U1 If ENABLED, all TLBs belonging to Video Engine will be invalidated once the flush operation is complete. This bit is only valid when the Post-Sync Operation field is a value of 1h or 3h.
	17:16	**Reserved** Access: RO Format: MBZ
	15:14	**Post-Sync Operation** BitFieldDesc

MI_FLUSH_DW

Value	Name	Description
0h	No Write	No write occurs as a result of this instruction. This can be used to implement a "trap" operation, etc.
1h	Write Immediate Data	HW implicitly detects the Data size to be Qword or Dword to be written to memory based on the command dword length programmed . When Dword Length indicates Qword, Writes the QWord containing Immediate Data Low, High DWs to the Destination Address . When Dword Length indicates Dword, Writes the DWord containing Immediate Data Low to the Destination Address
2h	Reserved	Reserved
3h		Write the TIMESTAMP register to the Destination Address. The upper 28 bits of the TIMESTAMP register are tied to '0'.

Programming Notes
If executed in a PPGTT (non-secure) batch buffer, the post-sync op is always inhibited. This command does not write anything to memory.

13:10 **Reserved**

Access:	RO
Format:	MBZ

9 **Flush LLC**

Format:	Enable

If enabled, at the end of the current MI_FLUSH_DW the last level cache is cleared of all the cachelines which have been determined as being part of the Frame Buffer.

8 **Notify Enable**

Format:	U1

If ENABLED, a Sync Completion Interrupt will be generated (if enabled by the MI Interrupt Control registers) once the sync operation is complete. See Interrupt Control Registers in Memory Interface Registers for details.

7 **Video Pipeline Cache invalidate**

Format:	U1

Enable the invalidation of the video cache at the end of this flush

6 **Reserved**

Access:	RO
Format:	MBZ

5:0 **DWord Length**

Format:	=n

		Value / **Name** 2h — DWord 3h — QWord **[Default]**
1..2	63:57	**Reserved** Access: RO Format: MBZ
	56:48	**Reserved** Access: RO Format: MBZ
	47:3	**Address** Format: GraphicsAddress[47:3]U64 This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space. GraphicsAddress is a 64-bit value [63:0], but only a portion of it is used by the hardware. Upper bits [63:48] are ignored and MBZ. Some GraphicsAddress fields only specify the upper address bits. For example GraphicsAddress[47:12] is a 4KB page address.
	2	**Destination Address Type** Defines address space of Destination Address 0h — PPGTT — Use PPGTT address space for DW write 1h — GGTT — Use GGTT address space for DW write **Programming Notes** Ignored if "No write" is the selected in Operation.
	1:0	**Reserved** Access: RO Format: MBZ
3..4	63:0	**Immediate Data** This field specifies the DWord value to be written to the targeted location. DW2 is the lower DW if QW is desired. Only valid when 15:14 in header is set to 1h To avoid hitting a known hardware bug, drivers cannot send a QW write when bit 5 of the address is '1'

MI_FLUSH_DW

MI_FLUSH_DW

Source: VideoEnhancementCS

Length Bias: 2

The MI_FLUSH_DW command is used to perform an internal "flush" operation. The parser pauses on an internal flush until all drawing engines have completed any pending operations. In addition, this command can also be used to:

- Flush any dirty data to memory.
- Invalidate the TLB cache inside the hardware

Usage note: After this command is completed with a Store DWord enabled, CPU access to graphics memory will be coherent (assuming the Render Cache flush is not inhibited).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 26h MI_FLUSH_DW Format: OpCode
	22	**Reserved**
	21	**Store Data Index** Format: U1 This field is valid only if the post-sync operation is not 0. If this bit is set, the store data address is index into the global hardware status page when destination address type in the command is set to 1 (GGTT). The store data address is index into the per-process hardware status page when destination address type in the command is set to 0 (PPGTT).
	20:19	**Reserved** Access: RO Format: MBZ
	18	**TLB Invalidate** Format: U1 If ENABLED, all TLBs belonging to Video Enhancement Engine will be invalidated once the flush operation is complete. This bit is only valid when the Post-Sync Operation field is a value of 1h or 3h. If GFX_MODE (0x229c) bit 13, this command will cause a config write to MMIO register space with the address 0x4f100.

MI_FLUSH_DW

17:16 Reserved

Access:	RO
Format:	MBZ

15:14 Post-Sync Operation

Value	Name	Description
0h	No Write	No write occurs as a result of this instruction. This can be used to implement a "trap" operation, etc.
1h	Write Immediate Data	Write the QWord containing Immediate Data Low, High DWs to the Destination Address
2h	Reserved	Reserved
3h	Write TIMESTAMP register	Write the TIMESTAMP register to the Destination Address. The upper 28 bits of the TIMESTAMP register are tied to '0'.

Programming Notes

If executed in non-secure batch buffer, the address given will be in a PPGTT address space. If in a secure ring or batch, address given will be in GGTT space

13:10 Reserved

Access:	RO
Format:	MBZ

9 Flush LLC

Format:	Enable

If enabled, at the end of the current MI_FLUSH_DW the last level cache is cleared of all the cachelines which have been determined as being part of the Frame Buffer.

8 Notify Enable

Format:	U1

If ENABLED, a Sync Completion Interrupt will be generated (if enabled by the MI Interrupt Control registers) once the sync operation is complete. See Interrupt Control Registers in Memory Interface Registers for details.

7:6 Reserved

Access:	RO
Format:	MBZ

5:0 DWord Length

Default Value:	3h Excludes DWord (0,1) = 2 for DWord, 3 for QWord
Format:	=n

MI_FLUSH_DW

1	31:3	**Address** Format: GraphicsAddress[31:3]U28 This field specifies Bits 31:3 of the Address where the DWord or QWord will be stored. Note that the address can only be QWord aligned, irrespective of data size.
	2	**Destination Address Type** Defines address space of Destination Address Value / Name / Description: 0h / PPGTT / Use PPGTT address space for DW write 1h / GGTT / Use GGTT address space for DW write **Programming Notes** Ignored if "No write" is the selected in Operation.
	1:0	**Reserved** Access: RO Format: MBZ
2	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Address High** Format: GraphicsAddress[47:32]U64 This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space
3..4	63:0	**Immediate Data** This field specifies the DWord value to be written to the targeted location. DW2 is the lower DW if QW is desired. Only valid when 15:14 in header is set to 1h To avoid hitting a known hardware bug, drivers cannot send a QW write when bit 5 of the address is '1'

MI_FORCE_WAKEUP

MI_FORCE_WAKEUP

Source: CommandStreamer

Length Bias: 2

This command is used to communicate Force Wakeup request to PM unit. No functionality is performed by this command when none of the mask bits are set and is equivalent to NOOP. Example for usage model: VCS Ring Buffer: MI_FORCE_WAKEUP (Force Render Awake set to '1') MI_SEMPAHORE_SIGNAL (Signal context id 0xABC to Render Command Streamer) MI_FORCE_WAKEUP (Force Render Awake set to '0') MI_BATCH_BUFFER_START STATE Commands .. .. MI_FORCE_WAKEUP (Force Render Awake set to '1') MI_LOAD_REGISTER_IMMEDIATE (Load register 0x23XX in render command streamer with data 0xFFF) MI_FORCE_WAKEUP (Force Render Awake set to '0') .. MI_BATCH_BUFFER_END

Programming Notes

This command must not be programmed in the command stream for Render Engine Command Streamer or Position Commmand Streamer or Compute Engine Command Streamer.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 1Dh MI_FORCE_WAKEUP Format: OpCode
	22:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h Format: =n Total Length - 2. Excludes DWord (0,1).
1	31:16	**Mask Bits** Format: Mask[15:0] **Programming Notes** Must be set to modify corresponding Bits 15:0. (Mask bits must not be set for reserved bits).
	15:10	**Reserved** Access: RO Format: MBZ
	9	**Reserved**

MI_FORCE_WAKEUP

	8	**Reserved**
	7:5	**Reserved** Access: RO Format: MBZ
	4	**Force Media-Slice3 Awake** Format: U1 When set, Command Streamer sends message to PM to force awake the media engines in media slice 3, i.e., VCS0, VCS1, and VECS0 (next instructions require media engine awake). Command streamer waits for acknowledge from PM before parsing the next command. When reset, command streamer sends message to PM to disable force awake of media engine (next instructions do not require the media engine to be awake). Command streamer waits for acknowledge from PM before parsing the next command. **Programming Notes** Mask bit [20] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed. Use of this Cross-engine Force Wake is deprecated and not allowed.
	3	**Force Media-Slice2 Awake** Format: U1 When set, Command Streamer sends message to PM to force awake the media engines in media slice 2, i.e., VCS0, VCS1, and VECS0 (next instructions require media engine awake). Command streamer waits for acknowledge from PM before parsing the next command. When reset, command streamer sends message to PM to disable force awake of media engine (next instructions do not require the media engine to be awake). Command streamer waits for acknowledge from PM before parsing the next command. **Programming Notes** Mask bit [19] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed. Use of this Cross-engine Force Wake is deprecated and not allowed.
	2	**Force Media-Slice1 Awake** Format: U1 When set, Command Streamer sends message to PM to force awake the media engines in media slice 1, i.e., VCS0, VCS1, and VECS0 (next instructions require media engine awake). Command streamer waits for acknowledge from PM before parsing the next command. When reset, command streamer sends message to PM to disable force awake of media engine (next instructions do not require the media engine to be awake). Command streamer waits for acknowledge from PM before parsing the next command. **Programming Notes** Mask bit [18] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed. Use of this Cross-engine Force Wake is deprecated and not allowed.

MI_FORCE_WAKEUP

	1	**Force Render Awake** Source: BlitterCS, VideoCS, VideoEnhancementCS Format: U1 **Description** When set, Command Streamer sends message to PM to force awake render engine (next instructions require render engine or compute engine awake). Command streamer waits for acknowledge from PM before parsing the next command. When reset, command streamer sends message to PM to disable force awake of render engine (next instructions do not require the render engine or compute engine to be awake. Command streamer waits for acknowledge from PM before parsing the next command. **Programming Notes** Mask bit [17] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed. MI_FORCE_WAKEUP command programmed in RenderCS command buffer must not set "Force Render Awake" bit. Use of this Cross-engine Force Wake is deprecated and not allowed.
	0	**Force Media-Slice0 Awake** Format: U1 When set, Command Streamer sends message to PM to force awake the media engines in media slice 0, i.e., VCS0, VCS1, and VECS0 (next instructions require media engine awake). Command streamer waits for acknowledge from PM before parsing the next command. When reset, command streamer sends message to PM to disable force awake of media engine (next instructions do not require the media engine to be awake). Command streamer waits for acknowledge from PM before parsing the next command. **Programming Notes** Mask bit [16] has to be set for HW to look at this field when MI_FORCE_WAKEUP command is parsed. Use of this Cross-engine Force Wake is deprecated and not allowed.

MI_LOAD_REGISTER_IMM

MI_LOAD_REGISTER_IMM

Source:	CommandStreamer
Length Bias:	2

The MI_LOAD_REGISTER_IMM command requests a write of up to a DWord constant supplied in the command to the specified Register Offset (i.e., offset into Memory-Mapped Register Range).
Any offset that is to a destination outside of the GT core will allow the parser to continue once the cycle is at the GT boundry and not destination. Any other address will ensure the destination is updated prior to parsing the next command

Programming Notes

Many MMIO bits require the engine to be IDLE prior to updating the value. Command streamer does not implicitly put in a pipeline flush in the cases a MMIO bit requires the engine to be IDLE. In the case there was a 3DPRIMITIVE command or GPGPU_WALKER command without any stalling PIPE_CONTROL, one must be inserted prior to a MI_LOAD_REGISTER_IMMEDIATE that is updating a bit that requires the engine to be IDLE.

When executed from a non-privileged batch buffer, MMIO writes are only allowed to the registers listed in User Mode Non-Privileged Registers for the corresponding engine, any writes targeting the register not listed in the User Mode Non-Privileged Register will convert this command to a NOOP.

The following addresses should NOT be used for LRIs:

1. 0x8800 - 0x88FF
2. >= 0xC0000

Limited LRI cycles to the Display Engine (0x40000-0xBFFFF) are allowed, but must be spaced to allow only one pending at a time. This can be done by issuing an SRM to the same address immediately after each LRI.

Programming an MMIO register is equivalent to programming a non-pipeline state to the hardware and hence an explicit stalling flush needs to be programmed prior to programming this command. However for certain MMIO registers based on their functionality doing an explicit stalling flush is exempted. Listed below are the exempted registers.

- 3DPRIM_END_OFFSET - Auto Draw End Offset
- 3DPRIM_START_VERTEX - Load Indirect Start Vertex
- 3DPRIM_VERTEX_COUNT - Load Indirect Vertex Count
- 3DPRIM_INSTANCE_COUNT - Load Indirect Instance Count
- 3DPRIM_START_INSTANCE - Load Indirect Start Instance
- 3DPRIM_BASE_VERTEX - Load Indirect Base Vertex
- 3DPRIM_XP0 - Load Indirect Extended Parameter 0
- 3DPRIM_XP1 - Load Indirect Extended Parameter 1
- 3DPRIM_XP2 - Load Indirect Extended Parameter 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND

MI_LOAD_REGISTER_IMM

Format:	OpCode

28:23 MI Command Opcode

Default Value:	22h MI_LOAD_REGISTER_IMM
Format:	OpCode

22:20 Reserved

Access:	RO
Format:	MBZ

19 Add CS MMIO Start Offset

This bit controls the functionality of the "Register Address" field in the command.

Value	Name	Description
1		"Register Address" field in the command is treated as an offset from the executing Command Streamers MMIO start offset. Bits [22:2] of the "Register Address" are considered as dword offset to be added to the MMIO start offset of the corresponding command streamer. However, during context restore bits [11:2] of the "Register Address" are considered as dword offset to be added to the MMIO start offset of the corresponding command streamer. Refer "Register Access and User Mode Privilege Access" section to get the list of all the instances of the Command Streamers and their associated MMIO Start Offset's. /// Command executed from Ring Buffer or Batch Buffer Example: MI_LOAD_REGISTER_IMMEDIATE, ADD_CS_MMIO_START_OFFSET: true, Data:0xABCD, Register Address: 0x00_2000 The above command when executed on RenderCS will result in a write to MMIO offset 0x00_4000 (0x00_2000 + 0x00_2000) instead to 0x00_2000. Note that RenderCS MMIO start offset is 0x2000. For illustration table below shows the result of this command executed from few instances of the command streamers from different engines.

S.No	Command Streamer	Command Streamer MMIO Base	LRI Register Offset	LRI Written Address
1	RenderCS	0x2000	0x2000	0x00_4000
2	VideoCS0	0x1C_0000	0x2000	0x1C_2000
3	VideoCS1	0x1C_4000	0x2000	0x1C_6000
4	VideoEnhancement0	0x1C_8000	0x2000	0x1C_A000
5	VideoCS2	0x1D_0000	0x2000	0x1D_2000
6	VideoCS3	0x1D_4000	0x2000	0x1D_6000
7	VideoEnhancement1	0x1D_8000	0x2000	0x1D_A000

// Command executed from context image during context restore
Example: MI_LOAD_REGISTER_IMMEDIATE, ADD_CS_MMIO_START_OFFSET: true, Data:0xABCD, Register Address: 0x1C_2030
The above command when executed on RenderCS will result in a write to

MMIO offset 0x2030 (0x00_2000 + 0x030) instead to 0x1C_2030. Note that RenderCS MMIO start offset is 0x2000. For illustration table below shows the result of this command executed from few instances of the command streamers from different engines during context restore.

S.No	Command Streamer	Command Streamer MMIO Base	LRI Register Offset	LRI Written Address
1	RenderCS	0x2000	0x1C_2030	0x00_2030
2	VideoCS0	0x1C_0000	0x1C_2030	0x1C_0030
3	VideoCS1	0x1C_4000	0x1C_2030	0x1C_4030
4	VideoEnhancement0	0x1C_8000	0x1C_2030	0x1C_8030
5	VideoCS2	0x1D_0000	0x1C_2030	0x1D_0030
6	VideoCS3	0x1D_4000	0x1C_2030	0x1D_4030
7	VideoEnhancement1	0x1D_8000	0x1C_2030	0x1D_8030

0	**[Default]**	"Register Address" field in the command is absolute and not an offset from the executing command streamer MMIO start offset..

18 **Reserved**

Access:	RO
Format:	MBZ

17 **MMIO Remap Enable**

This bit provides a mechanism in HW to remap the MMIO address in the MI command to the engine instance on which the command is getting executed, remapping in HW is done using engine specific remap table. Render and Compute engine share a common remapping table to facilitate remapping across engines, where as a dedicated remap table for each of Video Decode and Video Enhancement engine class.

A MMIO remapping table per engine class is created with MMIO address belonging to multiple instances of an engine within an engine class. However Render and Compute engine share a common remapping table to facilitate remapping across engines, where as a dedicated remap table for each of Video Decode and Video Enhancement engine class.

This mode provides mechanism for SW to always use MMIO address belonging to fixed instance (instance zero) with in an engine class during command buffer creation agnostic to the instance on which it will get scheduled. This will also allow context interoperability across instances with in an engine class and extends to across engines in case of Render and Compute.

Value	Name	Description
1		MMIO remapping will be applied to the MMIO address prior to using for any other functionality of the command.
0		MMIO remapping will not be applied to the MMIO address.

Programming Notes

- SW must always use MMIO address belonging to Instance-0 of an engine while enabling "MMIO Remap" in MI commands.

- MMIO Remapping will be done by HW prior to doing any other functionality associated with the MI command or the privilege checks.
- "Add CS MMIO Start Offset" must not be enabled when "MMIO Remap" is Enabled and Vice-versa.
- When remapping is not found in the remap table, HW will use the MMIO address directly without any modification.

	16:13	**Reserved** Access: RO Format: MBZ
	12	**Reserved**
	11:8	**Byte Write Disables** Range: Must specify a valid register write operation If [11:8] is '1111b', then this command will behave as a NOOP. Otherwise, the value is forwarded to the destination register.
	7:0	**DWord Length** Default Value: 1h Excludes DWord (0,1) Format: =n
1..2	63:32	**Data DWord** Mask: Bytes Write Disables Format: U32 This field specifies the DWord value to be written to the targeted location.
	31:23	**Reserved** Access: RO Format: MBZ
	22:2	**Register Offset** Format: MmioAddress[22:2] This field specifies bits [22:2] of the offset into the Memory Mapped Register Range (i.e., this field specifies a DWord offset).
	1:0	**Reserved** Access: RO Format: MBZ

MI_LOAD_REGISTER_MEM

MI_LOAD_REGISTER_MEM

Source: RenderCS, BlitterCS, VideoCS, VideoEnhancementCS

Length Bias: 2

The MI_LOAD_REGISTER_MEM command requests from a memory location and stores that DWord to a register.

Programming Notes

The command temporarily halts commands that will cause cycles down the 3D pipeline.

The following addresses should NOT be used for MMIO writes:

- 0x8800 - 0x88FF
- >= 0xC0000

Limited MMIO writes cycles to the Display Engine 0x40000-0xBFFFF) are allowed, but must be spaced to allow only one pending at a time. This can be done by issuing an SRM to the same address immediately after each MMIO write.

This command should not be used within a non-privilege batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation.

This command is not allowed to update the privilege register range when executed from a non-privilege batch buffer.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 29h MI_LOAD_REGISTER_MEM Format: OpCode
	22	**Use Global GTT** Format: Boolean This bit if set when executing from a non-privileged batch buffer will be treated as privilege access violation. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer or ring buffer. This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
	21	**Async Mode Enable** Format: Enable If this bit is set then the command stream will not wait for completion of this command before executing the next command. Please refer to the LOAD_INDIRECT and Predicate registers for usage of this bit.

20 **Reserved**

Access:	RO
Format:	MBZ

19 **Add CS MMIO Start Offset**

This bit controls the functionality of the Register Address field in the command.

Value	Name	Description
1		Register Address field in the command is treated as an offset from the executing Command Streamers MMIO start offset.Bits [22:2] of the Register Address are considered as dword offset to be added to the MMIO start offset of the corresponding command streamer. Example: MI_LOAD_REGISTER_MEM, ADD_CS_MMIO_START_OFFSET: true, Memory Address:0xABCD, Register Address: 0x1C_0030 The above command when executed on RenderCS will result in a write to MMIO offset 0x1C_2030 (0x00_2000 + 0x1C_0030) instead to 0x1C_0030. Note that RenderCS MMIO start offset is 0x2000.
0	**[Default]**	Register Address field in the command is absolute and not an offset from the executing command streamer MMIO start offset.

18 **Reserved**

Access:	RO
Format:	MBZ

17 **MMIO Remap Enable**

This bit provides a mechanism in HW to remap the MMIO address in the MI command to the engine instance on which the command is getting executed, remapping in HW is done using engine specific remap table. Render and Compute engine share a common remapping table to facilitate remapping across engines, where as a dedicated remap table for each of Video Decode and Video Enhancement engine class.

A MMIO remapping table per engine class is created with MMIO address belonging to multiple instances of an engine within an engine class. However Render and Compute engine share a common remapping table to facilitate remapping across engines, where as a dedicated remap table for each of Video Decode and Video Enhancement engine class.

This mode provides mechanism for SW to always use MMIO address belonging to fixed instance (instance zero) with in an engine class during command buffer creation agnostic to the instance on which it will get scheduled. This will also allow context interoperability across instances with in an engine class and extends to across engines in case of Render and Compute.

Value	Name	Description
1		MMIO remapping will be applied to the MMIO address prior to using for any other functionality of the command.
0		MMIO remapping will not be applied to the MMIO address.

Programming Notes
• SW must always use MMIO address belonging to Instance-0 of an engine while enabling "MMIO Remap" in MI commands.

		• MMIO Remapping will be done by HW prior to doing any other functionality associated with the MI command or the privilege checks. • "Add CS MMIO Start Offset" must not be enabled when "MMIO Remap" is Enabled and Vice-versa. • When remapping is not found in the remap table, HW will use the MMIO address directly without any modification.
	16:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) Format: =n
1	31:23	**Reserved** Access: RO Format: MBZ
	22:2	**Register Address** Format: MMIOAddress[22:2] This field specifies Bits 22:2 of the Register offset the DWord will be written to. As the register address must be DWord-aligned, Bits 1:0 of that address MBZ.
	1:0	**Reserved** Access: RO Format: MBZ
2..3	63:2	**Memory Address** Format: GraphicsAddress[63:2] This field specifies the address of the memory location where the register value specified in the DWord above will read from. The address specifies the DWord location of the data. Range = GraphicsVirtualAddress[63:2] for a DWord register GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	1:0	**Reserved** Access: RO Format: MBZ

MI_LOAD_REGISTER_REG

MI_LOAD_REGISTER_REG

Source: CommandStreamer

Length Bias: 2

The MI_LOAD_REGISTER_REG command reads from a source register location and writes that value to a destination register location.

Any offset that is to a destination outside of the GT core will allow the parser to continue once the cycle is at the GT boundry and not destination. Any other address will ensure the destination is updated prior to parsing the next command

Programming Notes

The command temporarily halts commands that will cause cycles down the 3D pipeline.

Destination register with mask implemented (Ex: Some registers have bits [31:16] as mask bits and bits[15:0] as data) will not get updated unless the value read from source register has the bits corresponding to the mask bits set. Note that any mask implemented register when read returns "0" for the bits corresponding to mask location. When the source and destination are mask implemented registers, destination register will not get updated with the source register contents.

This command is not allowed to update the privilege register range when executed from a non-privilege batch buffer.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 2Ah MI_LOAD_REGISTER_REG Format: OpCode
	22:20	**Reserved** Access: RO Format: MBZ
	19	**Add CS MMIO Start Offset Destination** This bit controls the functionality of the Register Address Destination field in the command. Value 1 (Name): Destination Register Address field in the command is treated as an offset from the executing Command Streamers MMIO start offset. Bits [22:2] of the Destination Register Address are considered as dword offset to be added to the MMIO start offset of the corresponding command streamer. Example: MI_LOAD_REGISTER_REGISTER_REG, DEST_ADD_CS_MMIO_START_OFFSET: true, SRC_ADD_CS_MMIO_START_OFFSET:true, Source Register Address:0x1C_0130, Destination Register Address: 0x1C_0030

		The above command when executed on RenderCS will result in a MMIO read from 0x1C_2130 (0x00_2000 + 0x1C_0130) and write to MMIO offset 0x1C_2030 (0x00_2000 + 0x1C_0030) instead of read from 0x1C_0130 and write to 0x1C_0030. Note that RenderCS MMIO start offset is 0x2000.
0	**[Default]**	Destination Register Address field in the command is absolute and not an offset from the executing command streamer MMIO start offset.

18 **Add CS MMIO Start Offset Source**

This bit controls the functionality of the Register Address Source field in the command.

Value	Name	Description
1		Source Register Address field in the command is treated as an offset from the executing Command Streamers MMIO start offset. Bits [22:2] of the Source Register Address are considered as dword offset to be added to the MMIO start offset of the corresponding command streamer. Example: MI_LOAD_REGISTER_REGISTER_REG, DEST_ADD_CS_MMIO_START_OFFSET: false, SRC_ADD_CS_MMIO_START_OFFSET:true, Source Register Address:0x1C_0130, Destination Register Address: 0x1C_0030 The above command when executed on RenderCS will result in a MMIO read from 0x1C _2130 instead of read from 0x1C_0130 and write to MMIO offset 0x1C_0030. Note that RenderCS MMIO start offset is 0x2000.
0	**[Default]**	Register Address field in the command is absolute and not an offset from the executing command streamer MMIO start offset.

17 **MMIO Remap Enable Destination**

This bit provides a mechanism in HW to remap the " Destination Register" MMIO address in the MI command to the engine instance on which the command is getting executed, remapping in HW is done using engine specific remap table. Render and Compute engine share a common remapping table to facilitate remapping across engines, where as a dedicated remap table for each of Video Decode and Video Enhancement engine class.

A MMIO remapping table per engine class is created with MMIO address belonging to multiple instances of an engine within an engine class. However Render and Compute engine share a common remapping table to facilitate remapping across engines, where as a dedicated remap table for each of Video Decode and Video Enhancement engine class.

This mode provides mechanism for SW to always use MMIO address belonging to fixed instance (instance zero) with in an engine class during command buffer creation agnostic to the instance on which it will get scheduled. This will also allow context interoperability across instances with in an engine class and extends to across engines in case of Render and Compute.

Value	Name	Description
1		MMIO remapping will be applied to the MMIO address prior to using for any other functionality of the command.
0		MMIO remapping will not be applied to the MMIO address.

Programming Notes

- SW must always use MMIO address belonging to Instance-0 of an engine while enabling "MMIO Remap" in MI commands.
- MMIO Remapping will be done by HW prior to doing any other functionality associated with the MI command or the privilege checks.
- "Add CS MMIO Start Offset" must not be enabled when "MMIO Remap" is Enabled and Vice-versa.
- When remapping is not found in the remap table, HW will use the MMIO address directly without any modification.

16 **MMIO Remap Enable Source**

This bit provides a mechanism in HW to remap the "Source Register"MMIO address in the MI command to the engine instance on which the command is getting executed, remapping in HW is done using engine specific remap table. Render and Compute engine share a common remapping table to facilitate remapping across engines, where as a dedicated remap table for each of Video Decode and Video Enhancement engine class.

A MMIO remapping table per engine class is created with MMIO address belonging to multiple instances of an engine within an engine class. However Render and Compute engine share a common remapping table to facilitate remapping across engines, where as a dedicated remap table for each of Video Decode and Video Enhancement engine class.

This mode provides mechanism for SW to always use MMIO address belonging to fixed instance (instance zero) with in an engine class during command buffer creation agnostic to the instance on which it will get scheduled. This will also allow context interoperability across instances with in an engine class and extends to across engines in case of Render and Compute.

Value	Name	Description
1		MMIO remapping will be applied to the MMIO address prior to using for any other functionality of the command.
0		MMIO remapping will not be applied to the MMIO address.

Programming Notes

- SW must always use MMIO address belonging to Instance-0 of an engine while enabling "MMIO Remap" in MI commands.
- MMIO Remapping will be done by HW prior to doing any other functionality associated with the MI command or the privilege checks.
- "Add CS MMIO Start Offset" must not be enabled when "MMIO Remap" is Enabled and Vice-versa.
- When remapping is not found in the remap table, HW will use the MMIO address directly without any modification.

15:8 **Reserved**

Access:	RO
Format:	MBZ

7:0 **DWord Length**

MI_LOAD_REGISTER_REG

		Default Value: 1h Excludes DWord (0,1) Format: =n
1	31:23	**Reserved** Access: RO Format: MBZ
	22:2	**Source Register Address** Format: MMIOAddress[22:2] This field specifies Bits 22:2 of the Register offset the DWord will be written to. As the register address must be DWord-aligned, Bits 1:0 of that address MBZ.
	1:0	**Reserved** Access: RO Format: MBZ
2	31:23	**Reserved** Access: RO Format: MBZ
	22:2	**Destination Register Address** Format: MMIOAddress[22:2] This field specifies Bits 22:2 of the Register offset the DWord will be written to. As the register address must be DWord-aligned, Bits 1:0 of that address MBZ.
	1:0	**Reserved** Access: RO Format: MBZ

MI_LOAD_SCAN_LINES_EXCL

MI_LOAD_SCAN_LINES_EXCL

Source: BlitterCS

Length Bias: 2

The MI_LOAD_SCAN_LINES_EXCL command is used to initialize the Scan Line Window registers for a specific Display Pipe. If the display refresh is inside this window the Display Engine signals the command parser to release the WAIT_FOR_EVENT command (i.e., the parser will wait while outside of the window). This command overrides the Scan Line Window defined by any previous MI_LOAD_SCAN_LINES_INCL or MI_LOAD_SCAN_LINES_EXCL commands targeting the specific display pipe.
Note: The two scan-line numbers are inclusive. If programmed to the same values, that single line defines the region in question.
Always place an even number of MI_LOAD_SCAN_LINES_EXCL/INCL at a time into the ring buffer. If only a single MI_LOAD_SCAN_LINES_EXCL/INCL is desired, just add a second identical MI_LOAD_SCAN_LINES_EXCL/INCL command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 13h MI_LOAD_SCAN_LINES_EXCL Format: OpCode
	22	**Reserved** Access: RO Format: MBZ
	21:19	**Display Pipe Select** Format: U3 This field selects which Display Engine (pipe) this command is targeting. Value / Name: 0h: Display Pipe A 1h: Display Pipe B 2h, 3h: Reserved 4h: Display Pipe C 5h: Display Pipe D 6h, 7h: Reserved
	18:17	**Reserved**
	16:6	**Reserved** Access: RO Format: MBZ

<table>
<tr><th colspan="3">MI_LOAD_SCAN_LINES_EXCL</th></tr>
<tr><td></td><td>5:0</td><td>DWord Length<table><tr><td>Default Value:</td><td>0h Excludes DWord (0,1)</td></tr><tr><td>Format:</td><td>=n</td></tr></table></td></tr>
<tr><td rowspan="2">1</td><td>31:16</td><td>Start Scan Line Number<table><tr><td>Format:</td><td>U16</td></tr></table>This field specifies the starting scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]</td></tr>
<tr><td>15:0</td><td>End Scan Line Number<table><tr><td>Format:</td><td>U16</td></tr></table>This field specifies the ending scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]</td></tr>
</table>

MI_LOAD_SCAN_LINES_EXCL

MI_LOAD_SCAN_LINES_EXCL

Source: RenderCS

Length Bias: 2

The MI_LOAD_SCAN_LINES_EXCL command is used to initialize the Scan Line Window registers for a specific Display Pipe. If the display refresh is **inside** this window the Display Engine signals the command parser to release the WAIT_FOR_EVENT command (i.e., the parser will wait while outside). This command overrides the Scan Line Window defined by any previous MI_LOAD_SCAN_LINES_INCL or MI_LOAD_SCAN_LINES_EXCL commands targeting the specific display pipe. **Note:** The two scan-line numbers are inclusive. If programmed to the same values, that single line defines the region in question. Always place an even number of MI_LOAD_SCAN_LINES_EXCL/INCL at a time into the ring buffer. If only a single MI_LOAD_SCAN_LINES_EXCL/INCL is desired, just add a second identical MI_LOAD_SCAN_LINES_EXCL/INCL command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 13h MI_LOAD_SCAN_LINES_EXCL Format: OpCode
	22	**Reserved** Access: RO Format: MBZ
	21:19	**Display Pipe Select** Format: U3 This field selects which Display Engine (pipe) this command is targeting. Value / Name: 0h Display Pipe A 1h Display Pipe B 2h Reserved 3h Reserved 4h Display Pipe C 5h Display Pipe D
	18:17	**Reserved**
	16:6	**Reserved** Access: RO Format: MBZ
	5:0	**DWord Length**

MI_LOAD_SCAN_LINES_EXCL

		Default Value: 0h Format: =n
1	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**Start Scan Line Number** Format: U13 Range: [0, Display Buffer height in lines-1] This field specifies the starting scan line number of the Scan Line Window.
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**End Scan Line Number** Format: U13 This field specifies the ending scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]

MI_LOAD_SCAN_LINES_INCL

MI_LOAD_SCAN_LINES_INCL

Source: BlitterCS

Length Bias: 2

The MI_LOAD_SCAN_LINES_INCL command is used to initialize the Scan Line Window registers for a specific Display Engine. If the display refresh is outside this window the Display Engine signals the command parser to release the WAIT_FOR_EVENT command (i.e., the parser will wait while inside of the window). This command overrides the Scan Line Window defined by any previous MI_LOAD_SCAN_LINES_INCL or MI_LOAD_SCAN_LINES_EXCL commands targeting the specific display.
Always place an even number of MI_LOAD_SCAN_LINES_EXCL/INCL at a time into the ring buffer. If only a single MI_LOAD_SCAN_LINES_EXCL/INCL is desired, just add a second identical

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 12h MI_LOAD_SCAN_LINES_INCL Format: OpCode
	22	**Reserved** Access: RO Format: MBZ
	21:19	**Display Pipe Select** Format: U3 This field selects which Display Engine (pipe) this command is targeting. Value / Name: 0h: Display Pipe A 1h: Display Pipe B 2h, 3h: Reserved 4h: Display Pipe C 5h: Display Pipe D 6h, 7h: Reserved
	18:17	**Reserved**
	16:6	**Reserved** Access: RO Format: MBZ
	5:0	**DWord Length** Default Value: 0h Excludes DWord (0,1)

MI_LOAD_SCAN_LINES_INCL		
		Format: =n
1	31:16	**Start Scan Line Number** Format: U16 This field specifies the starting scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]
	15:0	**End Scan Line Number** Format: U16 This field specifies the ending scan line number of the Scan Line Window. Range: [0, Display Buffer height in lines-1]

MI_LOAD_SCAN_LINES_INCL

MI_LOAD_SCAN_LINES_INCL

Source: RenderCS

Length Bias: 2

The MI_LOAD_SCAN_LINES_INCL command is used to initialize the Scan Line Window registers for a specific Display Engine. If the display refresh is **outside** this window the Display Engine signals the command parser to release the WAIT_FOR_EVENT command (i.e., the parser will wait while inside the window). This command overrides the Scan Line Window defined by any previous MI_LOAD_SCAN_LINES_INCL or MI_LOAD_SCAN_LINES_EXCL commands targeting the specific display.
Always place an even number of MI_LOAD_SCAN_LINES_EXCL/INCL at a time into the ring buffer. If only a single MI_LOAD_SCAN_LINES_EXCL/INCL is desired, just add a second identical MI_LOAD_SCAN_LINES_EXCL/INCL command.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 12h MI_LOAD_SCAN_LINES_INCL Format: OpCode
	22	**Reserved** Access: RO Format: MBZ
	21:19	**Display Pipe Select** Format: U3 This field selects which Display Engine (pipe) this command is targeting. Value / Name: 0h Display Pipe A 1h Display Pipe B 2h Reserved 3h Reserved 4h Display Pipe C 5h Display Pipe D
	18:17	**Reserved**
	16:6	**Reserved** Access: RO Format: MBZ
	5:0	**DWord Length**

MI_LOAD_SCAN_LINES_INCL

		Default Value: 0h Format: =n
1	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**Start Scan Line Number** Format: U13 Range: [0, Display Buffer height in lines-1] This field specifies the starting scan line number of the Scan Line window.
	15:13	**Reserved** Access: RO Format: MBZ
	12:0	**End Scan Line Number** Format: U13 Range: [0, Display Buffer height in lines-1] This field specifies the ending scan line number of the Scan Line Window.

MI_MATH

MI_MATH

Source: VideoCS

Length Bias: 2

The MI_MATH command allows software to send instructions to the ALU in the Command Streamer.
This command is the means by which the ALU is accessed. ALU instructions form the data payload of the MI_MATH command. An ALU instruction takes one DWord in size. The MI_MATH DWord Length is programmed based on the number of ALU instructions included, limited only by the max DWord Length supported.
When the command streamer parses an MI_MATH command, it sends the included ALU instructions to the ALU. The ALU processes any instruction in a single clock. See the ALU section for more details.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 1Ah MI_MATH Format: OpCode
	22:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Format: =n Value / Name: 0h **[Default]**; [1,255]
1..n	31:0	**ALU INSTRUCTION** Format: U32

MI_MATH

MI_MATH

Source: BlitterCS

Length Bias: 2

The MI_MATH command allows software to send instructions to the ALU in the Command Streamer.
This command is the means by which the ALU is accessed. ALU instructions form the data payload of the MI_MATH command. An ALU instruction takes one DWord in size. The MI_MATH DWord Length is programmed based on the number of ALU instructions included, limited only by the max DWord Length supported.
When the command streamer parses an MI_MATH command, it sends the included ALU instructions to the ALU. The ALU processes any instruction in a single clock. See the ALU section for more details.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 1Ah MI_MATH Format: OpCode
	22:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Format: =n Value / Name 0h **[Default]** [1,255]
1..n	31:0	**ALU INSTRUCTION** Format: U32

MI_MATH

MI_MATH

Source: RenderCS

Length Bias: 2

The MI_MATH command allows SW to send instruction to ALU in Render Command Streamer.
MI_MATH command is the means by which ALU can be accessed. ALU instructions form the data payload of MI_MATH command, ALU instruction is dword in size. MI_MATH Dword Length should be programmed based on the number of ALU instruction packed, max number is limited by the max Dword Length supported. When MI_MATH command is parsed by command streamer it outputs the payload dwords (ALU instructions) to the ALU. ALU takes single clock to process any given instruction. Refer to B-spec "Command Streamer (CS) ALU Programming" section in Command Streamer Programming.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 1Ah MI_MATH Format: OpCode
	22:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Format: =n Value: [0-255] Name:
1..n	31:0	**ALU INSTRUCTION** Format: U32

MI_MATH

MI_MATH

Source: VideoEnhancementCS

Length Bias: 2

The MI_MATH command allows software to send instructions to the ALU in the Command Streamer.
This command is the means by which the ALU is accessed. ALU instructions form the data payload of the MI_MATH command. An ALU instruction takes one DWord in size. The MI_MATH DWord Length is programmed based on the number of ALU instructions included, limited only by the max DWord Length supported.
When the command streamer parses an MI_MATH command, it sends the included ALU instructions to the ALU. The ALU processes any instruction in a single clock. See the ALU section for more details.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 1Ah MI_MATH Format: OpCode
	22:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Format: =n Value / Name 0h **[Default]** [1,255]
1..n	31:0	**ALU INSTRUCTION** Format: U32

MI_NOOP

MI_NOOP

Source: BlitterCS

Length Bias: 1

The MI_NOOP command basically performs a "no operation" in the command stream and is typically used to pad the command stream (e.g., in order to pad out a batch buffer to a QWord boundary). However, there is one minor (optional) function this command can perform - a 22-bit value can be loaded into the MI NOPID register. This provides a general-purpose command stream tagging ("breadcrumb") mechanism (e.g., to provide sequencing information for a subsequent breakpoint interrupt).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 0h MI_NOOP Format: OpCode
	22	**Identification Number Register Write Enable** Format: Enable This field enables the value in the Identification Number field to be written into the MI NOPID register. If disabled, that register is unmodified - making this command an effective "no operation" function. Value / Name / Description 0h / Disable / Do not write the NOP_ID register. 1h / Enable / Write the NOP_ID register.
	21:0	**Identification Number** Format: U22 This field contains a 22-bit number which can be written to the MI NOPID register.

MI_NOOP

MI_NOOP

Source: RenderCS

Length Bias: 1

The MI_NOOP command basically performs a "no operation" in the command stream and is typically used to pad the command stream (e.g., in order to pad out a batch buffer to a QWord boundary). However, there is one minor (optional) function this command can perform - a 22-bit value can be loaded into the MI NOPID register. This provides a general-purpose command stream tagging ("breadcrumb") mechanism (e.g., to provide sequencing information for a subsequent breakpoint interrupt).

Performance

The MI_NOOP process time is reduced to 1 clock. An example use of the improved NOOP throughput is for some multi-pass media applications where some unwanted media object commands are replaced by MI_NOOP commands without repacking the commands in a batch buffer.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 0h MI_NOOP Format: OpCode
	22	**Identification Number Register Write Enable** Format: Enable This field enables the value in the Identification Number field to be written into the MI NOPID register. If disabled, that register is unmodified, making this command an effective "no operation" function. Value / Name / Description 0h / Disable / Do not write the NOP_ID register. 1h / Enable / Write the NOP_ID register.
	21:0	**Identification Number** Format: U22 This field contains a 22-bit number which can be written to the MI NOPID register.

MI_NOOP

MI_NOOP

Source: VideoCS

Length Bias: 1

The MI_NOOP command basically performs a "no operation" in the command stream and is typically used to pad the command stream (e.g., in order to pad out a batch buffer to a QWord boundary). However, there is one minor (optional) function this command can perform - a 22-bit value can be loaded into the MI NOPID register. This provides a general-purpose command stream tagging ("breadcrumb") mechanism (e.g., to provide sequencing information for a subsequent breakpoint interrupt).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 00h MI_NOOP Format: OpCode
	22	**Identification Number Register Write Enable** Format: Enable This field enables the value in the Identification Number field to be written into the MI NOPID register. If disabled, that register is unmodified - making this command an effective "no operation" function. Value: 1 — Name: Write the NOP_ID register.
	21:0	**Identification Number** Format: U22 This field contains a 22-bit number which can be written to the MI NOPID register.

MI_NOOP

MI_NOOP

Source: VideoEnhancementCS

Length Bias: 1

The MI_NOOP command basically performs a "no operation" in the command stream and is typically used to pad the command stream (e.g., in order to pad out a batch buffer to a QWord boundary). However, there is one minor (optional) function this command can perform - a 22-bit value can be loaded into the MI NOPID register. This provides a general-purpose command stream tagging ("breadcrumb") mechanism (e.g., to provide sequencing information for a subsequent breakpoint interrupt).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 00h MI_NOOP Format: OpCode
	22	**Identification Number Register Write Enable** Format: Enable This field enables the value in the Identification Number field to be written into the MI NOPID register. If disabled, that register is unmodified - making this command an effective "no operation" function. Value 1 — Write th NOP_ID Register Value 0 — Do not write the NOP_ID register
	21:0	**Identification Number** Format: U22 This field contains a 22-bit number which can be written to the MI NOPID register.

MI_PREDICATE

MI_PREDICATE

Source: RenderCS

Length Bias: 1

Programming Notes
This command is supported by PositionCS.
This command is supported by ComputeCS

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 0Ch MI_PREDICATE Format: OpCode
	22:8	**Reserved** Access: RO Format: MBZ
	7:6	**Load Operation** This field controls if/how the Predicate state bit is modified. Value / Name / Description: 0h / KEEP / The Predicate state bit is unmodified. 1h / Reserved / 2h / LOAD / The Predicate state bit is loaded with the combine operation result. 3h / LOADINV / The Predicate state bit is loaded with the inverted combine operation result.
	5	**Reserved** Access: RO Format: MBZ
	4:3	**Combine Operation** This field controls if/how the result of the compare operation is combined with the current Predicate state bit. Value / Name / Description: 0h / SET / The combine operation output the compare result unmodified. 1h / AND / The combine operation outputs the AND of the compare result and the current Predicate state bit. 2h / OR / The combine operation outputs the OR of the compare result and the current

		Predicate state bit.
3h	XOR	The combine operation outputs the XOR of the compare result and the current Predicate state bit.

2 Reserved

Access:	RO
Format:	MBZ

1:0 Compare Operation

This field controls how Data DWord 0 and Data DWord 1 fields are used to generate a compare operation result and possibly modify the PredicateData register.

Value	Name	Description
0h	TRUE	The compare operation outputs TRUE. The PredicateData register is unmodified.
1h	FALSE	The compare operation outputs FALSE. The PredicateData register is unmodified.
2h	SRCS_EQUAL	(MItemp0 - MItemp1) is computed and loaded into the PredicateData register. The compare operation outputs (MItemp0 == MItemp1).
3h	DELTAS_EQUAL	(MItemp0 - MItemp1) is computed and compared to the PredicateData register. If the values are equal, the compare operation outputs TRUE, otherwise it outputs FALSE. The PredicateData register is unmodified.

MI_REPORT_HEAD

MI_REPORT_HEAD

Source: BlitterCS

Length Bias: 1

The MI_REPORT_HEAD command causes the Head Pointer value of the active ring buffer to be written to a cacheable (snooped) system memory location.
When the **Execlist Enable** bit is reset:
The location written is relative to the address programmed in the Hardware Status Page Address Register.

Programming Notes

This command must not be executed from a Batch Buffer (Refer to the description of the HWS_PGA register).
When the **Execlist Disable** is clear, the head pointer will be reported to the PP HW Status Page.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 07h MI_REPORT_HEAD Format: OpCode
	22:0	**Reserved** Access: RO Format: MBZ

MI_REPORT_HEAD

MI_REPORT_HEAD

Source: RenderCS

Length Bias: 1

The MI_REPORT_HEAD command causes the Head Pointer value of the active ring buffer to be written to a cacheable (snooped) system memory location. When Execlist Enable is set, the head pointer will be reported to the PP HW Status Page. The location written is relative to the address programmed in the Hardware Status Page Address Register.

Programming Notes

This command must not be executed from a Batch Buffer. (Refer to the description of the HWS_PGA register.)

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 07h MI_REPORT_HEAD Format: OpCode
	22:0	**Reserved** Access: RO Format: MBZ

MI_REPORT_HEAD

MI_REPORT_HEAD

Source: VideoCS

Length Bias: 1

The MI_REPORT_HEAD command causes the Head Pointer value of the ring buffer to be written to a cacheable (snooped) system memory location. When the Per-Process Virtual Address Space and Execlist Enable bitis reset: The location written is relative to the address programmed in the Hardware Status Page Address Register. When the Execlist Enable is set, the head pointer will be reported to the PP HW Status Page.

Programming Notes

This command must not be executed from a Batch Buffer (Refer to the description of the HWS_PGA register).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 07h MI_REPORT_HEAD Format: OpCode
	22:0	**Reserved** Access: RO Format: MBZ

MI_REPORT_HEAD

MI_REPORT_HEAD

Source: VideoEnhancementCS

Length Bias: 1

The MI_REPORT_HEAD command causes the Head Pointer value of the ring buffer to be written to a cacheable (snooped) system memory location.
When the **Per-Process Virtual Address Space and Execlist Enable bit** is reset: The location written is relative to the address programmed in the Hardware Status Page Address Register. When the **Execlist Enable** is set, the head pointer will be reported to the PP HW Status Page.

Programming Notes

This command must not be executed from a Batch Buffer (Refer to the description of the HWS_PGA register).

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 07h MI_REPORT_HEAD Format: OpCode
	22:0	**Reserved** Access: RO Format: MBZ

MI_REPORT_PERF_COUNT

MI_REPORT_PERF_COUNT

Source: RenderCS, ComputeCS

Length Bias: 2

The MI_REPORT_PERF_COUNT command causes the GFX hardware to write out a snap-shot of performance counters to the address specified in this command along with constant ID field supplied and the time-stamp counter. This write is required to be treated as a cacheable write irrespective of GTT entry memory type. This command is specific to the render engine.

Programming Notes

This command can be inserted after events of interest (frequently before and after a 3DPRIMITIVE command). SW is entirely responsible for managing the ID field and addresses used by such a series of commands.

GTT_SELECT must not be set to 1 (i.e. GGTT) when MI_REPORT_PERF_COUNT command is programmed in a non-privileged batch buffer. Refer to the "User Mode Privileged commands" Table in MI_BATCH_BUFFER_START command section for more details. All batch buffers in PPGTT are considered as Non-privileged.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 28h MI_REPORT_PERF_COUNT Format: OpCode
	22:6	**Reserved** Access: RO Format: MBZ
	5:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) Format: =n Total Length - 2
1..2	63:6	**Memory Address** Format: GraphicsAddress[63:6] This field specifies 64B aligned GFX MEM address where the chap counter values are reported. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47] **Programming Notes** This field is ignored if "Report to OABUFFER" bit is set.
	5	**Reserved**

MI_REPORT_PERF_COUNT

		Access: RO Format: MBZ
	4	**Core Mode Enable** Format: U1 This bit is set then the address will be offset by the Core ID:If Core ID 0, then there is no offset If Core ID 1, then the Memory is offset by the size of the data(64b).
	3:1	**Reserved** Access: RO Format: MBZ
	0	**Use Global GTT** Format: Boolean This field when set (i.e. bit = 1) selects the GGTT for address translation. When this bit is 0 (default value), HW should use PGTT for address translation.
3	31:0	**Report ID** Format: U32 This field specifies the ID provided by SW for a given report command. It can be tracked to use different flavors of these reports based on where in command-stream they are inserted. This field is reported only when Counter Select Field is 0. **Programming Notes** If a privilege access violation occurs, the REPORT ID field in the report generated by the next legitimate MI_REPORT_PERF_COUNT will be corrupted.

MI_RS_STORE_DATA_IMM

MI_RS_STORE_DATA_IMM

Source: RenderCS

Length Bias: 2

The MI_RS_STORE_DATA_IMM command requests a write of the DWord constant supplied in the packet to the specified Memory Address.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 2Bh MI_RS_STORE_DATA_IMM Format: OpCode MI_RS_STORE_DATA_IMM
	22	**Reserved** Access: RO Format: MBZ
	21	**Reserved**
	20:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) Format: =n
1..2	63:2	**Destination Address** Format: GraphicsAddress[63:2] This field specifies Bits 47:2 of the Address where the DWord will be stored. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47]. When render engine is PPGTT enabled this Address is translated using PPGTT, else GGTT is used for translation.
	1	**Reserved** Access: RO Format: MBZ
	0	**Core Mode Enable** If this bit is set then the address will be offset by the Core ID:

MI_RS_STORE_DATA_IMM		
		If Core ID 0, then there is no offset If Core ID 1, then the Memory is offset by the size of the data.
3	31:0	**Data DWord 0** Format: U32 This field specifies the DWord value to be written to the targeted location.

MI_SEMAPHORE_SIGNAL

MI_SEMAPHORE_SIGNAL

Source: CommandStreamer

Length Bias: 2

Description

An engine on executing this command generates a signal (interrupt) to the GUC (scheduler or FW) by reporting the Producer Token Number programmed in SEMAPHORE_TOKEN register. Each engine implements its own SEMAPHORE_TOKEN register. SEMAPHORE_TOKEN register is privileged and context save/restored. Scheduler can take appropriate action on decoding the reported Producer Token Number. Typically MI_ATOMIC (non-posted) command will be used to update the memory semaphore before signaling the consumer context.
Each engine implements SEMAPHORE_SIGNAL_PORT register for receiving semaphore signal from the scheduler (SW or FW). A write to the SEMAPHORE_SIGNAL_PORT with data as 0xFFFF_FFFF is decoded as semaphore signal received by the corresponding engine. An engine waiting on un-successful MI_SEMAPHORE_WAIT (signal mode) command will reacquire the semaphore data from memory and re-evaluate the semaphore comparison on receiving the semaphore signal. SEMAPHORE_SIGNAL_PORT register is privileged. Writing to the SEMAPHORE_SIGNAL_PORT of an idle engine (no context) does not trigger any action in HW and is of no use.
SEMAPHORE_TOKEN, MI_SEMAPHORE_SIGNAL, SEMAPHORE_SIGNAL_PORT and MI_SEMAPHORE_WAIT together can be used to create semaphores between producer context and consumer context.
MI_SEMPAHORE_SIGNAL command from a producer context can be used to signal a consumer context waiting on MI_SEMAPHORE_WAIT (signal mode) command through scheduler (SW or FW).

- Typically MI_ATOMIC (non-posted) command will be used to update the memory semaphore by the producer context before signaling the consumer context.
- Scheduler on receiving the signal will process the Producer Token Number and if required will signal the consumer context running on an engine by writing 0xFFFF_FFFF to the corresponding engines SEMAPHORE_SIGNAL_PORT.
- A consumer context will wait on MI_SEMAPHORE_WAIT (signal mode) command until the semaphore comparison is successful. An engine waiting on un-successful MI_SEMAPHORE_WAIT (signal mode) command will reacquire the semaphore data from memory and re-evaluate the semaphore comparison on receiving the semaphore signal. MI_SEMAPHORE_WAIT command has Wait Token Number as inline data programmed by the SW. Context switched out an un-successful MI_SEMAPHORE_WAIT command will report Wait Token Number as Wait Detail field in the CSB structure.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 1Bh MI_SEMAPHORE_SIGNAL Format: OpCode

MI_SEMAPHORE_SIGNAL

	22	**Reserved** Access: RO Format: MBZ
	21	**Post-Sync Operation** Source: RenderCS **Value / Name / Description** 0h — No Post Sync Operation — Command is executed as usual. 1h — Post Sync Operation — MI_SEMAPHORE_SIGNAL command is executed as a pipelined PIPE_CONTROL flush command with Semaphore Signal as post sync operation. Flush completion only guarantees the workload prior to this command is pushed till Windower unit and completion of any outstanding flushes issued prior to this command. **Programming Notes** Any desired pipeline flush operation can be achieved by programming PIPE_CONTROL command prior to this command. When this bit is set Command Streamer sends a flush down the pipe and the atomic operation is saved as post sync operation. Command streamer goes on executing the following commands. Atomic operation saved as post sync operation is executed at some point later on completion of corresponding flush issued. When this bit is set atomic semaphore signal operation will be out of order with rest of the MI commands programmed in the ring buffer or batch buffer, it will be in order with respect to the post sync operations resulting due to PIPE_CONTROL command. This bit must not be set when executed by ComputeCS.
	20:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:0	**Reserved** Access: RO Format: MBZ

MI_SEMAPHORE_WAIT

MI_SEMAPHORE_WAIT

Source:	CommandStreamer
Length Bias:	2

Description

This command supports memory based Semaphore WAIT. Memory based semaphores will be used for synchronization between the Producer and the Consumer contexts. Producer and Consumer Contexts could be running on different engines or on the same engine inside GT. Producer Context implements a Signal and Consumer context implements a Wait.
Command Streamer on parsing this command fetches data from the Semaphore Address mentioned in this command and compares it with the inline Semaphore Data Dword.

- If comparison passes, the command streamer moves to the next command.
- If comparison fails Command streamer switches out the context. Context switch can be inhibited by setting "Inhibit Synchronous Context Switch" in CTXT_SR_CTL register.
- If "Inhibit Synchronous context Switch" is enabled and comparison fails, Command Streamer evaluates the Compare Operation based on the Wait Mode until the compare operation is true or Wait is canceled by SW.
- CS generates semaphore wait interrupt to the scheduler when MI_SEMAPHORE_WAIT command is un-successful and when "Inhibit Synchronous Context Switch" is set. Scheduler can use this interrupt to preempt the context waiting on semaphore wait.

MI_SEMAPHORE_WAIT command also supports register based Semaphore WAIT. Command Streamer on parsing this command fetches data from the MMIO offset mentioned in this command and compares it with the inline Semaphore Data Dword. This functionality is supported when Register Poll bit is set in the command header. In register poll mode of operation Wait Mode supported is always Poll mode and no Signal mode is supported.

- If comparison passes, the command streamer moves to the next command.
- Unlike in Memory based semaphore, there is no context switch on an un-successful semaphore wait in Register Poll mode, however preemption is supported on unsuccessful semaphore wait in Register Poll mode. Semaphore wait interrupt is not generated by default on wait un-successful in Register Poll mode.
- Also unlike in Memory based semaphore, generation of an interrupt for a semaphore wait in "Register Poll" mode is not dependent on the value of bit "Inhibit Synchronous Context Switch" in register "CTXT_SR_CTL"
- Register Poll mode of Semaphore Wait command operation is non-privileged and will be supported from PPGTT batch buffers.
- HW will trigger Render DOP CG on semaphore wait unsuccessful by default and can be disabled if not desired by programming Register Poll Mode Semaphore Wait Event IDLE message Disable bit in INSTPM register. Note that Render DOP CG will not be triggered on register semaphore wait un-successful from INDIRECT_CTX pointer or BB_PER_CTX_PTR buffers.

Programming Notes

MI_SEMAPHORE_WAIT

MI_SEMAPHORE_WAIT command must not be used in the middle of a tile pass on the posh pipe.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 1Ch MI_SEMAPHORE_WAIT Format: OpCode
	22	**Memory Type** This bit will be ignored and treated as if clear when executing from a non-privileged batch buffer. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer. This bit *must* be 1 if the **Per Process GTT Enable** bit is clear. 0h – Per Process Graphics Address 1h – Global Graphics Address – This command will use the global GTT to translate the Address and this command must beexecuting from a privileged (secure) batch buffer.
	21:19	**Reserved** Access: RO Format: MBZ
	18	**Reserved** Access: RO Format: MBZ
	17	**Reserved**
	16	**Register Poll Mode** This field control the semaphore wait behavior of polling from memory vs MMIO register. 1h – Register Poll **[Default]** – In this mode HW periodically reads the semaphore data from MMIO register instead of memory for comparison until the condition is satisfied. Periodicity will be mentioned in a SEMA_WAIT_POLL register. When operating in register poll mode, DW2 Semaphore Address (bits 22:2) carries the register MMIO offset to be polled. In register poll mode Memory Type field of this command are ignored by HW. 0h – Memory Poll – In this mode HW will functional as in regular mode and checks for semaphore data in memory.

Programming Notes
In register poll mode of operation of MI_SEMAPHORE_WAIT command, context switch is not supported on un-successful wait. Wait Mode must be always set to Polling Mode when Register Poll Mode is enabled. Preemption is supported on unsuccessful semaphore wait in Register Poll mode if operation.

15 **Wait Mode**

This bit specifies the WAIT behavior when the semaphore comparison fails and before the context is switched out.

Value	Name	Description
1h	Polling Mode	In this mode HW periodically reads the semaphore data from memory for comparison until it is context switched out. Periodicity will be mentioned in a SEMA_WAIT_POLL register.
0h	Signal Mode	[] In this mode HW will reacquire the semaphore data from memory for evaluating semaphore wait condition on receiving SIGNAL.Scheduler or SW can generate a SIGNAL to an engine by writing a value 0xFFFF_FFFF to the engines corresponding SEMAPHORE_SIGNAL_PORT register.

Programming Notes
Wait Mode must be always set to Polling Mode when Register Poll Mode is enabled.

14:12 **Compare Operation**

This field specifies the operation that will be executed to create the result that will either allow the context to continue or wait.

SAD = Semaphore Address Data
SDD = Semaphore Data Dword

Value	Name	Description
0h	SAD_GREATER_THAN_SDD	If Indirect fetched data is greater than inline data then continue.
1h	SAD_GREATER_THAN_OR_EQUAL_SDD	If Indirect fetched data is greater than or equal to inline data then continue.
2h	SAD_LESS_THAN_SDD	If Indirect fetched data is less than inline data then continue.
3h	SAD_LESS_THAN_OR_EQUAL_SDD	If Indirect fetched data is less than or equal to inline data then continue.
4h	SAD_EQUAL_SDD	If Indirect fetched data is equal to inline data then continue.
5h	SAD_NOT_EQUAL_SDD	If Indirect fetched data is not equal to inline data then continue.
6h	Reserved	
7h	Reserved	

MI_SEMAPHORE_WAIT

	11:10	**Reserved** Access: RO Format: MBZ
	9:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Format: =n Value: 3h — Name: Excludes DWord (0,1) **[Default]**
1	31:0	**Semaphore Data Dword** Format: U32 This Data dword is supplied by software to control execution of the command buffer. This value is used as part of the comparison to result in waiting or continuing in the command parser if enabled.
2..3	63:2	**Semaphore Address** Format: VIRTUAL_ADDR[63:2] **Register Poll Mode:** In Register Poll mode of operation, Bits 22:2 (Bits 63:23 are reserved MBZ, HW enforced) specify the MMIO offset of the register for the semaphore. **Non Register Poll Mode:** This field is the Graphics Memory Address of the 32-bit value for the semaphore. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form.
	1:0	**Reserved** Access: RO Format: MBZ
4	31:22	**Reserved** Access: RO Format: MBZ
	21:10	**Reserved** Access: RO Format: MBZ
	9:5	**Wait Token Number** When context is switched out due to Semaphore wait, WaitTokenNumber is reported as Wait Detail in the CSB structure.

MI_SEMAPHORE_WAIT		
	4:2	**Reserved** Access: RO Format: MBZ
	1:0	**Reserved** Access: RO Format: MBZ

MI_SET_CONTEXT

MI_SET_CONTEXT

Source: RenderCS

Length Bias: 2

The MI_SET_CONTEXT command is used to specify the logical context associated with the hardware context. A logical context is an area in memory used to store hardware context information, and the context is referenced via a 2KB-aligned pointer. If the (new) logical context is different (i.e., at a different memory address), the device saves the current HW context values to the current logical context address, and then restores (loads) the new logical context by reading the context from the new address and loading it into the hardware context state. If the logical context address specified in this command matches the current logical context address, this command is effectively treated as a NOOP. Specific to the Render command stream only. This command also includes some controls over the context save/restore process. The Force Restore bit can be used to refresh the on-chip device state from the same memory address if the indirect state buffers have been modified. The Restore Inhibit bit can be used to prevent the new context from being loaded at all. This must be used to prevent an uninitialized context from being loaded. Once software has initialized a context (by setting all state variables to initial values via commands), the context can then be stored and restored normally. When switching from a generic media context to a 3D context, the generic media state must be cleared via the Generic Media State Clear bit 16 in PIPE_CONTROL (or bit 4 in MI_FLUSH) before saving 3D context. MI_SET_CONTEXT commands are permitted only within a ring buffer (not within a batch buffer).
All context is saved and restored from a GGTT space.
This command does not initiate any interrupt due to context switch of any kind and does not support any workaround batch buffer or indirect context offset feature.

Programming Notes
For ring buffer mode, the first 128B(2 cache lines) of the context image are saved as zeros.
In execution list mode, this command must be preceded with a MI_ARB_ON_OFF command to disable arbitration and followed by a MI_ARB_ON_OFF command to enable arbitration. The first 320 bytes(5 cache lines) of the context image will be saved as zeros.
Arbitration Mode must be set to not allow lite restore prior to this command being executed. This bit is a field in the MI_ARB_ON_OFF command when in Execution list Mode.
This command needs to be always followed by a single MI_NOOP instruction to workaround a silicon issue.
MI_ARB_ON_OFF with 'Arbitration Enable Reset' set should be programmed before an MI_SET_CONTEXT command. MI_ARB_ON_OFF with 'Arbitration Enable' set should be programmed after an MI_SET_CONTEXT command.
MI_SET_CONTEXT command must not be programmed for a POSH enabled context.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 18h MI_SET_CONTEXT

		Format: OpCode
	22:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h Format: =n
1	31:12	**Logical Context Address** Format: GraphicsAddress[31:12] This field contains the 4KB-aligned graphics memory address of the Logical Context that is to be loaded into the hardware context. If this address is equal to the CCID register associated with the current ring, no load will occur. Prior to loading this new context, the device will save the existing context as required. After the context switch operation completes, this address will be loaded into the associated CCID register. This field needs to be 4KB aligned virtual address.
	11:9	**Reserved** Access: RO Format: MBZ
	8	**Reserved, Must be 1** Format: MBO
	7:5	**Reserved** Access: RO Format: MBZ
	4	**Core Mode Enable** Format: Enable If set the Context Image will be offset based off the Core ID: If Core ID 0, no offset If Core ID 1, 36KB Offset
	3	**Resource Streamer State Save Enable** Format: Enable If set, the resource streamer state identified in the Logical Context Data section of the Memory Data Formats chapter is saved as part of switching away from this logical context. This bit will be stored in the associated CCID register to control the context save operation when switching away from this context (as part of a subsequent MI_SET_CONTEXT command).

MI_SET_CONTEXT		
	2	**Resource Streamer State Restore Enable** Format: Enable If set, the resource streamer state identified in the Logical Context Data section of the Memory Data Formats chapter is loaded (or restored) as part of switching to this logical context. This bit affects the switch (if required) to the context specified in Logical Context Address. This bit will also be stored in the associated CCID register to control a subsequent context save operation when switching to this context (as part of a subsequent ring buffer switch).
	1	**Force Restore** When switching to this logical context a comparison between Logical Context Address and the contests of the CCID register is performed. Normally, matching addresses prevent a context restore from occurring; however, when this bit is set a context restore is forced to occur. This bit cannot be set with Restore Inhibit. Note: This bit is not saved in the associated CCID register. It only affects the processing of this command.
	0	**Restore Inhibit** If set, the restore of the HW context from the logical context specified by Logical Context Address is inhibited (i.e., the existing HW context values are maintained). This bit must be used to prevent the loading of an uninitialized logical context. If clear, the context switch proceeds normally. This bit cannot be set with Force Restore. Note: This bit is not saved in the associated CCID register. It only affects the processing of this command.

MI_SET_PREDICATE

MI_SET_PREDICATE

Source: CommandStreamer

Length Bias: 1

Description

This command provides a mechanism to NOOP a section of commands programmed in the command buffer. This command on execution evaluates the condition based on the Predicate Enable field and sets the predicate status accordingly in HW. On predicate status set, HW NOOPS the commands subsequently parsed until the predicate status is re-evaluated and reset on executing MI_SET_PREDICATE. MI_SET_PREDICATE is the only command executed by HW when parsed during predicate status set. Resource Streamer doesn't take any action on parsing this command.

Programming Notes

- MI_SET_PREDICATE predication scope must be confined to commands programmed within a Batch Buffer (May include Nested and Chained Batch Buffers)..
- MI_SET_PREDICATE with Predicate Enable Must always have a corresponding MI_SET_PREDICATE with Predicate Disable within the same Batch Buffer.

Only the following command(s) can be programmed between the MI_SET_PREDICATE command enabled for predication: 3DSTATE_URB_VS 3DSTATE_URB_HS 3DSTATE_URB_DS 3DSTATE_URB_GS 3DSTATE_PUSH_CONSTANT_ALLOC_VS 3DSTATE_PUSH_CONSTANT_ALLOC_HS 3DSTATE_PUSH_CONSTANT_ALLOC_DS 3DSTATE_PUSH_CONSTANT_ALLOC_GS 3DSTATE_PUSH_CONSTANT_ALLOC_PS MI_LOAD_REGISTER_IMM MEDIA_VFE_STATE MEDIA_OBJECT MEDIA_OBJJECT_WALKER MEDIA_INTERFACE_DESCRIPTOR_LOAD 3DSTATE_WM_HZ_OP MI_STORE_DATA_IMM

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 01h MI_SET_PREDICATE Format: OpCode
	22:6	**Reserved** Access: RO Format: MBZ
	5:4	**Reserved** Access: RO Format: MBZ

MI_SET_PREDICATE

3:0 **Predicate Enable**

Source:	RenderCS, PositionCS, ComputeCS

Value	Name	Description
0h	NOOP Never	Predication is Disabled and CS will process commands as usual.
1h	NOOP on Result2 clear	Following Commands will be NOOPED by CS only if the MI_PREDICATE_RESULT_2 is clear.
2h	NOOP on Result2 set	Following Commands will be NOOPED by CS only if the MI_PREDICATE_RESULT_2 is set.
3h	NOOP on Result clear	Following Commands will be NOOPED by CS only if the MI_PREDICATE_RESULT is clear.
4h	NOOP on Result set	Following Commands will be NOOPED by CS only if the MI_PREDICATE_RESULT is set.
5h, 6h, 7h, 8h, 9h, Ah	Reserved	
Bh, Ch, Dh, Eh	Reserved	
Fh	NOOP Always	Following Commands will be NOOPED by RCS unconditionally.

3:0 **Predicate Enable**

Source:	BlitterCS, VideoCS, VideoEnhancementCS

Value	Name	Description
0h	Predicate Disable	Predication is Disabled and CS will process commands as usual.
1h	NOOP on Result2 Clear	Following Commands will be NOOPED by CS only if the MI_PREDICATE_RESULT_2 is clear.
2h	NOOP on Result2 Set	Following Commands will be NOOPED by CS only if the MI_PREDICATE_RESULT_2 is set.
3h,4h,5h, 6h, 7h, 8h, 9h, Ah, Bh, Ch, Dh, Eh	Reserved	
Fh	NOOP Always	Following Commands will be NOOPED by CS unconditionally.

MI_STORE_DATA_IMM

MI_STORE_DATA_IMM

Source: CommandStreamer

Length Bias: 2

The MI_STORE_DATA_IMM command requests a write of the QWord constant supplied in the packet to the specified Memory Address. As the write targets a System Memory Address, the write operation is coherent with the CPU cache (i.e., the processor cache is snooped).

This command supports writing to multiple consecutive dwords or qwords memory locations from the starting address.

Programming Notes

- This command should not be used within a "non-privilege" batch buffer to access global virtual space, doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation. This command can be used within ring buffers and/or privilege batch buffers to access global virtual space.
- This command can be used for general software synchronization through variables in cacheable memory (i.e., where software does not need to poll un-cached memory or device registers).
- This command simply initiates the write operation with command execution proceeding normally. Although the write operation is guaranteed to complete eventually, there is no mechanism to synchronize command execution with the completion (or even initiation) of these operations.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 20h MI_STORE_DATA_IMM Format: OpCode
	22	**Use Global GTT** Format: Boolean If set, this command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer. If clear, the PPGTT will be used. It is allowed for this bit to be clear when executing this command from a privileged (secure) batch buffer. This bit must be '1' if the Per Process GTT Enable bit is clear.
	21	**Store Qword** Format: Boolean If set, this command generates Qword writes to memory, two "Data Dword" are paired to form a Qword. Number of qwords generated depends upon the number of "Data Dword" programmed

in the command. If 'x' number of "Data Dwords" are programmed in this command it results in "x/2" qword writes to memory. If reset this command generates Dwords writes to memory. Number of dwords generated depends upon the number of "Data Dword" programmed in the command. If 'x' number of "Data Dwords" are programmed in this command it results in "x" dword writes to memory.

20:13 Reserved

Access:	RO
Format:	MBZ

12 Reserved

Access:	RO
Format:	MBZ

11 Reserved

Access:	RO
Format:	MBZ

10 Reserved

9:0 DWord Length

Format:	=n

Value	Name
2h	Store Dword **[Default]**
3h	Store Qword

Programming Notes
DWord Length programmed must not exceed 0x3FE.
If RS is enabled in the batch buffer, then the value of this field must not exceed 0x3F.

1..2 **63:2 Address**

Format:	VIRTUAL_ADDR[63:2]

GraphicsAddress is 64-bit value [63:0], but only a portion of it is used by hardware. The uppermost bits are ignored and MBZ. This field specifies Bits 47:2 of the Address where the DWord will be stored. As the store address must be DWord-aligned, Bits 1:0 of that address MBZ. This address must be 8B aligned for a store "QW" command.

1 Reserved

Access:	RO
Format:	MBZ

0 Core Mode Enable

Format:	U1

This bit is set then the address will be offset by the Core ID:If Core ID 0, then there is no offset If

<table>
<tr><th colspan="3">MI_STORE_DATA_IMM</th></tr>
<tr><td></td><td></td><td>Core ID 1, then the Memory is offset by the size of the data(32b or 64b based off number of DW length).</td></tr>
<tr><td>3</td><td>31:0</td><td>Data DWord 0<table><tr><td>Format:</td><td>U32</td></tr></table>This field specifies the DWord value to be written to the targeted location. For a QWord write this DWord is the lower DWord of the QWord to be reported (DW 0).</td></tr>
<tr><td>4</td><td>31:0</td><td>Data DWord 1<table><tr><td>Format:</td><td>U32</td></tr></table>This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).</td></tr>
</table>

MI_STORE_DATA_INDEX

MI_STORE_DATA_INDEX

Source: CommandStreamer

Length Bias: 2

The MI_STORE_DATA_INDEX command requests a write of the data constant supplied in the packet to the specified offset from the System Address defined by the Hardware Status Page Address Register. As the write targets a System Address, the write operation is coherent with the CPU cache (i.e., the processor cache is snooped).

Programming Notes

- Use of this command with an invalid or uninitialized value in the Hardware Status Page Address Register is UNDEFINED.
- This command can be used for general software synchronization through variables in cacheable memory(i.e., where software does not need to poll uncached memory or device registers).
- This command simply initiates the write operation with command execution proceeding normally. Although the write operation is guaranteed to complete eventually, there is no mechanism to synchronize command execution with the completion (or even initiation) of these operations.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 21h MI_STORE_DATA_INDEX Format: OpCode
	22	**Reserved** Access: RO Format: MBZ
	21	**Use Per-Process Hardware Status Page** If this bit is set, this command will index into the per-process hardware status page at offset 0K from the LRCA. If clear, the Global Hardware Status Page will be indexed.
	20:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 1h Format: =n

MI_STORE_DATA_INDEX

1	31:12	**Reserved** Access: RO Format: MBZ
	11:2	**Offset** Format: U10 zero-based DWord offset into the HW status page. This field specifies the offset (into the hardware status page) to which the data will be written. Note that the first few DWords of this status page are reserved for special-purpose data storage - targeting these reserved locations via this command is UNDEFINED. This address must be 8B aligned for a store QW command. Value: [16, 1023] — Name:
	1:0	**Reserved** Access: RO Format: MBZ
2	31:0	**Data DWord 0** Format: U32 This field specifies the DWord value to be written to the targeted location. For a QWord write this DWord is the lower DWord of the QWord to be reported (DW 0).
3	31:0	**Data DWord 1** Format: U32 This field specifies the upper DWord value to be written to the targeted QWord location (DW 1).

MI_STORE_REGISTER_MEM

MI_STORE_REGISTER_MEM

Source: CommandStreamer

Length Bias: 2

The MI_STORE_REGISTER_MEM command requests a register read from a specified memory mapped register location in the device and store of that DWord to memory. The register address is specified along with the command to perform the read.

Programming Notes

- The command temporarily halts command execution.
- The memory address for the write is snooped on the host bus.
- This command should not be used from within a "non-privilege" batch buffer to access global virtual space. doing so will be treated as privilege access violation. Refer "User Mode Privilege Command" in MI_BATCH_BUFFER_START command section to know HW behavior on encountering privilege access violation. This command can be used within ring buffers and/or "privilege" batch buffers to access global virtual space.
- This command will cause undefined data to be written to memory if given register addresses for the PGTBL_CTL_0 or FENCE registers.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 24h MI_STORE_REGISTER_MEM Format: OpCode
	22	**Use Global GTT** Format: Boolean It is allowed for this bit to be set when executing this command from a privileged (secure) batch or ring buffer. This bit must be clear when programmed from within a non-privileged batch buffer. This bit must be 1 if the Per Process GTT Enable bit is clear. This command will use the global GTT to translate the Address and this command must be executing from a privileged (secure) batch buffer.
	21	**Predicate Enable** Source: RenderCS, PositionCS, ComputeCS If set, this command is executed (or not) depending on the current value of the MI_PREDICATE internal state bit in MMIO register MI_PREDICATE_RESULT[0]. This command is ignored if PredicateEnable is set and value in the MMIO register MI_PREDICATE_RESULT[0] is 0. This command may also be dropped when MI_SET_PREDICATE condition to drop is true.

20 **Reserved**

Access:	RO
Format:	MBZ

19 **Add CS MMIO Start Offset**

This bit controls the functionality of the Register Address field in the command.

Value	Name	Description
1		Register Address field in the command is treated as an offset from the executing Command Streamers MMIO start offset. Bits [22:2] of the Register Address are considered as dword offset to be added to the MMIO start offset of the corresponding command streamer. Example: MI_STORE_REGISTER_MEM, ADD_CS_MMIO_START_OFFSET: true, Memory Address:0xABCD, Register Address: 0x1C_0030 The above command when executed on RenderCS will result in updating the memory address with the content of the MMIO offset 0x1C_2030 (0x00_2000 + 0x1C_0030) instead to 0x1C_0030. Note that RenderCS MMIO start offset is 0x2000.
0	**[Default]**	Register Address field in the command is absolute and not an offset from the executing command streamer MMIO start offset.

18 **Reserved**

Access:	RO
Format:	MBZ

17 **MMIO Remap Enable**

This bit provides a mechanism in HW to remap the MMIO address in the MI command to the engine instance on which the command is getting executed, remapping in HW is done using engine specific remap table. Render and Compute engine share a common remapping table to facilitate remapping across engines, where as a dedicated remap table for each of Video Decode and Video Enhancement engine class.

A MMIO remapping table per engine class is created with MMIO address belonging to multiple instances of an engine within an engine class. However Render and Compute engine share a common remapping table to facilitate remapping across engines, where as a dedicated remap table for each of Video Decode and Video Enhancement engine class.

This mode provides mechanism for SW to always use MMIO address belonging to fixed instance (instance zero) with in an engine class during command buffer creation agnostic to the instance on which it will get scheduled. This will also allow context interoperability across instances with in an engine class and extends to across engines in case of Render and Compute.

Value	Name	Description
1		MMIO remapping will be applied to the MMIO address prior to using for any other functionality of the command.
0		MMIO remapping will not be applied to the MMIO address.

Programming Notes
• SW must always use MMIO address belonging to Instance-0 of an engine while enabling

		"MMIO Remap" in MI commands. • MMIO Remapping will be done by HW prior to doing any other functionality associated with the MI command or the privilege checks. • "Add CS MMIO Start Offset" must not be enabled when "MMIO Remap" is Enabled and Vice-versa. • When remapping is not found in the remap table, HW will use the MMIO address directly without any modification.
	16:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) Format: =n
1	31:23	**Reserved** Access: RO Format: MBZ
	22:2	**Register Address** Format: MMIOAddress[22:2] This field specifies Bits 22:2 of the Register offset the DWord will be read from. As the register address must be DWord-aligned, Bits 1:0 of that address MBZ. **Programming Notes** • Storing a VGA register is not permitted and will store an UNDEFINED value. • The values of PGTBL_CTL0 or any of the FENCE registers cannot be stored to memory; UNDEFINED values will be written to memory if the addresses of these registers are specified.
	1:0	**Reserved** Access: RO Format: MBZ
2..3	63:2	**Memory Address** Format: GraphicsAddress[63:2] This field specifies the address of the memory location where the register value specified in the DWord above will be written. The address specifies the DWord location of the data.Range = GraphicsVirtualAddress[63:2] for a DWord register GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].
	1:0	**Reserved** Access: RO Format: MBZ

MI_SUSPEND_FLUSH

MI_SUSPEND_FLUSH		
Source: BlitterCS Length Bias: 1		
Blocks PM Flush Requests.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 0Bh MI_SUSPEND_FLUSH Format: OpCode
	22:1	**Reserved** Access: RO Format: MBZ
	0	**Suspend Flush** Format: Enable This field suspends flush due to a PM flush request.

MI_SUSPEND_FLUSH

MI_SUSPEND_FLUSH		
Source: RenderCS Length Bias: 1		
Blocks PM Flush Requests.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 0Bh MI_SUSPEND_FLUSH Format: OpCode
	22:1	**Reserved** Access: RO Format: MBZ
	0	**Suspend Flush** Format: Enable This field suspends flush due to a PM flush request.

MI_SUSPEND_FLUSH

MI_SUSPEND_FLUSH		
Source: VideoCS Length Bias: 1		
Blocks PM Flush Requests.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 0Bh MI_SUSPEND_FLUSH Format: OpCode
	22:1	**Reserved** Access: RO Format: MBZ
	0	**Suspend Flush** Format: Enable This field suspends flush due to a PM flush request.

MI_SUSPEND_FLUSH

MI_SUSPEND_FLUSH		
Source: VideoEnhancementCS Length Bias: 1		
Blocks PM Flush Requests.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 0Bh MI_SUSPEND_FLUSH Format: OpCode
	22:1	**Reserved** Access: RO Format: MBZ
	0	**Suspend Flush** Format: Enable This field suspends flush due to a PM flush request.

MI_TOPOLOGY_FILTER

MI_TOPOLOGY_FILTER		
Source:	RenderCS	
Length Bias:	1	
This command is used to specify a specific 3DPrimType value, where the CS will ignore all 3DPRIMITIVE commands that do no have a matching 3DPrimType. This primitive culling is optional (turned off by using this command with a Topology Filter Value of 0). **This command is specific to the Render command stream only.**		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 0Dh MI_TOPOLOGY_FILTER Format: OpCode
	22:6	**Reserved** Access: RO Format: MBZ
	5:0	**Topology Filter Value** Format: 3D_Prim_Topo_Type When non-zero, the CS will discard all 3DPRIMITIVE commands which do not match the specified 3DPrimTopologyType. When zero, no filtering is performed (normal operation).

MI_UPDATE_GTT

MI_UPDATE_GTT

Source: BSpec

Length Bias: 2

The MI_UPDATE_GTT command is used to update GTT page table entries in a coherent manner and at a predictable place in the command flow.
A PIPE_CONTROL flush command with "CS Stall" bit set must be programmed prior to MI_UPDATE_GTT command, since work associated with preceding commands that are still in the pipeline may be referencing GTT entries that will be changed by its execution. The flush must also invalidate TLBs and read caches that may become invalid as a result of the changed GTT entries. A PIPE_CONTROL flush command with "CS Stall" bit set must be programmed post MI_UPDATE_GTT command to ensure the GGTT is updated with modified page table entries before the following workload references the modified entries.
PIPE_CONTROL flush is not required if it can be guaranteed that the pipeline is free of any work that relies on changing GTT entries (such as MI_UPDATE_GTT contained in a paging DMA buffer that is doing only update/mapping activities and no rendering).
MI_UPDTE_GTT command is privilege operation and will be converted to a no-op and an error flagged if it is executed from within a non-secure batch buffer.
PPGTT updates cannot be done via **MI_UPDATE_GTT**, gfx driver will have to use MI_STORE_DATA_IMM for PPGTT inline updates.

The MI_UPDATE_GTT command is used to update GGTT page table entries in a coherent manner and at a predictable place in the command flow. A MI_FLUSH_DWORD flush command with "CS Stall" bit set must be programmed prior to MI_UPDATE_GTT command, since work associated with preceding commands that are still in the pipeline may be referencing GTT entries that will be changed by its execution. The flush must also invalidate TLBs and read caches that may become invalid as a result of the changed GTT entries. A MI_FLUSH_DWORD flush command with "CS Stall" bit set must be programmed post MI_UPDATE_GTT command to ensure the GGTT is updated with modified page table entries before the following workload references the modified entries. MI_FLUSH_DWORD flush is not required if it can be guaranteed that the pipeline is free of any work that relies on changing GTT entries (such as MI_UPDATE_GTT contained in a paging DMA buffer that is doing only update/mapping activities and no rendering). MI_UPDATE_GTT command is privilege operation and will be converted to a no-op and an error flagged if it is executed from within a non-secure batch buffer. PPGTT updates cannot be done via **MI_UPDATE_GTT** , gfx driver will have to use MI_STORE_DATA_IMM for PPGTT inline updates.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 23h MI_UPDATE_GTT Format: OpCode
	22:10	**Reserved** Access: RO

MI_UPDATE_GTT

		Format: MBZ
	9:0	**DWord Length** Format: =n n = 2b (where b = # of Entry Data included) **Value** / **Name** [2,1022] / 2 / **[Default]**
1	31:12	**Entry Address** Format: GraphicsAddress[31:12] This field holds the QW offset of the first table entry to be modified in GGTT.
	11:0	**Reserved** Access: RO Format: MBZ
2..n	63:0	**Entry Data** Format: U64 This Dword becomes the new page table entry. See PPGTT/Global GTT Table Entries (PTEs) in Memory Interface Registers.

MI_USER_INTERRUPT

MI_USER_INTERRUPT

Source: BlitterCS

Length Bias: 1

The MI_USER_INTERRUPT command is used to generate a User Interrupt condition. The parser will continue parsing after processing this command. See User Interrupt.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 02h MI_USER_INTERRUPT Format: OpCode
	22:0	**Reserved** Access: RO Format: MBZ

MI_USER_INTERRUPT

<table>
<tr><th colspan="3">MI_USER_INTERRUPT</th></tr>
<tr><td colspan="3">Source: RenderCS
Length Bias: 1</td></tr>
<tr><td colspan="3">The MI_USER_INTERRUPT command is used to generate a User Interrupt condition. The parser will continue parsing after processing this command. See User Interrupt.</td></tr>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="3">0</td><td>31:29</td><td>Command Type<table><tr><td>Default Value:</td><td>0h MI_COMMAND</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td>28:23</td><td>MI Command Opcode<table><tr><td>Default Value:</td><td>02h MI_USER_INTERRUPT</td></tr><tr><td>Format:</td><td>OpCode</td></tr></table></td></tr>
<tr><td>22:0</td><td>Reserved<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
</table>

MI_USER_INTERRUPT

MI_USER_INTERRUPT

Source: VideoCS

Length Bias: 1

The MI_USER_INTERRUPT command is used to generate a User Interrupt condition. The parser will continue parsing after processing this command. See User Interrupt.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 02h MI_USER_INTERRUPT Format: OpCode
	22:0	**Reserved** Access: RO Format: MBZ

MI_USER_INTERRUPT

MI_USER_INTERRUPT		
Source: VideoEnhancementCS Length Bias: 1		
The MI_USER_INTERRUPT command is used to generate a User Interrupt condition. The parser will continue parsing after processing this command. See User Interrupt.		
DWord	**Bit**	**Description**
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 02h MI_USER_INTERRUPT Format: OpCode
	22:0	**Reserved** Access: RO Format: MBZ

MI_WAIT_FOR_EVENT

MI_WAIT_FOR_EVENT

Source: RenderCS, BlitterCS

Length Bias: 1

The MI_WAIT_FOR_EVENT command is used to pause command stream processing of an engine (render or blitter) until a specific display event occurs (V-Blank) or a specific condition (Flip Pending, Scanline Pending) exists.

- Display engine can be configured to generate periodic V-Blank event to an engine.
- An engine on executing MI_LOAD_SCANLINE_INCL/EXCL command for a display pipe, expects a corresponding scanline event response from display engine. Engine tracks the pending scanline response for each of the display pipe separately.
- An engine on executing MI_DISPLAY_FLIP command for a display plane, expects a corresponding flip done event response from display engine. Engine tracks the pending flip done response (flip pending) for each of the display plane separately. Display flip could be of type Sync Flip or Async Flip, hence engine also tracks the type of flip (Sync or Async) along with the pending flip done response for a given display plane.

Only one event or condition can be specified in the command -- specifying multiple events or conditions is UNDEFINED. The command parser will halt until the event occurs or condition exists on parsing this command. Note that if a specified condition (Pending Flip Done response or Pending Scanline response) does not exist at the time the parser executes this command, the parser proceeds, treating this command as a no-operation (Ex: Command is no-operation when parsed with Display Plane Flip Pending Wait Enable set to Display Plane-1 when there is no outstanding flip done response for Display Plane-1 in the engine).

Execution List Mode of Scheduling:

An engine on evaluating unsuccessful MI_WAIT_FOR_EVENT(results in pausing command stream) triggers synchronous context switch stating the switch reason in Context Status Buffer. With exception of not triggering synchronous context switch on unsuccessful MI_WAIT_FOR_EVENT due to Flip Pending on an Async flip. Note that synchronous context switch can be inhibited through programming Inhibit Synchronous Context Switch bit in CTXT_SR_CTL register or by disabling arbitration through MI_ARB_ON_OFF command around MI_WAIT_FOR_EVENT. When synchronous context switch is inhibited and the engine is waiting on an unsuccessful MI_WAIT_FOR_EVENT, a submission of new execlist will trigger preemption process switching out the context. With exception of not triggering preemption on an unsuccessful MI_WAIT_FOR_EVENT due to Flip Pending on an Async flip.

Engine will always re-evaluate the wait condition for context switched out due to unsuccessful MI_WAIT_FOR_EVENT on resubmission of the context.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 03h MI_WAIT_FOR_EVENT Format: OpCode

MI_WAIT_FOR_EVENT

	22	**Reserved** Access: RO Format: MBZ
	21	**Display Plane 1 C Vertical Blank Wait Enable** Format: Enable This field enables a wait until the next Display Plane 1 C "Vertical Blank" event occurs. This event is described as the start of the next Display C vertical blank period. Note that this can cause a wait for up to an entire refresh period. See Vertical Blank Event in the Device Programming Interface chapter of MI Functions.
	20	**Display Plane 6 Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 2 C Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	19	**Display Plane 12 Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 4 C Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	18	**Display Plane 11 Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 4 B Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	17	**Display Plane 10 Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 4 A Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	16	**Display Plane 9 Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 3 C Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).

MI_WAIT_FOR_EVENT

	15	**Display Plane 3 Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 1 C Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	14	**Display Plane 1 C Scan Line Wait Enable** Format: Enable This field enables a wait while a Display Plane 1 C "Scan Line" condition exists. This condition is defined as the the start of the scan line specified in the Pipe C Display Scan Line Count Range Compare Register.
	13:12	**Reserved** Access: RO Format: MBZ
	11	**Display Plane 1 B Vertical Blank Wait Enable** Format: Enable This field enables a wait until the next Display Plane 1 B "Vertical Blank" event occurs. This event is described as the start of the next Display B vertical blank period. Note that this can cause a wait for up to an entire refresh period.
	10	**Display Plane 5 Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 2 B Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	9	**Display Plane 2 Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane B Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	8	**Display Plane 1 B Scan Line Wait Enable** Format: Enable This field enables a wait while a Display Plane 1 B "Scan Line" condition exists. This condition is defined as the the start of the scan line specified in the Pipe B Display Scan Line Count Range Compare Register.
	7	**Display Plane 8 Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 3 B Flip Pending condition. If a flip

		request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	6	**Display Plane 7 Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 3 A Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	5:4	**Reserved** Access: RO Format: MBZ
	3	**Display Plane 1 A Vertical Blank Wait Enable** Format: Enable This field enables a wait until the next Display Plane 1 A "Vertical Blank" event occurs. This event is described as the start of the next Display A vertical blank period. Note that this can cause a wait for up to an entire refresh period.
	2	**Display Plane 4 Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 2 A Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	1	**Display Plane 1 Flip Pending Wait Enable** Format: Enable This field enables a wait for the duration of a Display Plane 1 A Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).
	0	**Display Plane 1 A Scan Line Wait Enable** Format: Enable This field enables a wait while a Display Plane 1 A "Scan Line" condition exists. This condition is defined as the the start of the scan line specified in the Pipe A Display Scan Line Count Range Compare Register.

MI_WAIT_FOR_EVENT_2

MI_WAIT_FOR_EVENT_2

Source: RenderCS, BlitterCS

Length Bias: 1

The MI_WAIT_FOR_EVENT_2 command is used to pause command stream processing of an engine (render or blitter) until a specific display event occurs (V-Blank) or a specific condition (Flip Pending, Scanline Pending) exists.

- Display engine can be configured to generate periodic V-Blank event to an engine.
- An engine on executing MI_LOAD_SCANLINE_INCL/EXCL command for a display pipe, expects a corresponding scanline event response from display engine. Engine tracks the pending scanline response for each of the display pipe separately.
- An engine on executing MI_DISPLAY_FLIP command for a display plane, expects a corresponding flip done event response from display engine. Engine tracks the pending flip done response (flip pending) for each of the display plane separately. Display flip could be of type Sync Flip or Async Flip, hence engine also tracks the type of flip (Sync or Async) along with the pending flip done response for a given display plane.

Only one event or condition can be specified in the command -- specifying multiple events or conditions is UNDEFINED. The command parser will halt until the event occurs or condition exists on parsing this command. Note that if a specified condition (Pending Flip Done response or Pending Scanline response) does not exist at the time the parser executes this command, the parser proceeds, treating this command as a no-operation (Ex: Command is no-operation when parsed with Display Plane Flip Pending Wait Enable set to Display Plane-1 when there is no outstanding flip done response for Display Plane-1 in the engine).

Execution List Mode of Scheduling:

An engine on evaluating unsuccessful MI_WAIT_FOR_EVENT_2 (results in pausing command stream) triggers synchronous context switch stating the switch reason in Context Status Buffer. With exception of not triggering synchronous context switch on unsuccessful MI_WAIT_FOR_EVENT_2 due to Flip Pending on an Async flip. Note that synchronous context switch can be inhibited through programming Inhibit Synchronous Context Switch bit in CTXT_SR_CTL register or by disabling arbitration through MI_ARB_ON_OFF command around MI_WAIT_FOR_EVENT_2. When synchronous context switch is inhibited and the engine is waiting on an unsuccessful MI_WAIT_FOR_EVENT_2, a submission of new execlist will trigger preemption process switching out the context. With exception of not triggering preemption on an unsuccessful MI_WAIT_FOR_EVENT_2 due to Flip Pending on an Async flip.

Engine will always re-evaluate the wait condition for context switched out due to unsuccessful MI_WAIT_FOR_EVENT2 on resubmission of the context.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 0h MI_COMMAND Format: OpCode
	28:23	**MI Command Opcode** Default Value: 04h MI_WAIT_FOR_EVENT_2 Format: OpCode

MI_WAIT_FOR_EVENT_2

22:15	**Reserved**	
	Access:	RO
	Format:	MBZ
14:12	**Display Pipe Scan Line Wait Enable**	
	Format:	Enable
	This field enables a wait while a Display Pipe "Scan Line" condition exists. This condition is defined as the the start of the scan line specified in the Pipe B Display Scan Line Count Range Compare Register.	
	Value	**Name**
	0h	No Wait
	1h	Display Pipe A
	2h	Display Pipe B
	3h	Display Pipe C
	4h	Display Pipe D
	[5h,7h]	Reserved
11	**Reserved**	
	Access:	RO
	Format:	MBZ
10:8	**Display Pipe Vertical Blank Wait Enable**	
	Format:	Enable
	This field enables a wait until the next Display Pipe "Vertical Blank" event occurs. This event is described as the start of the next Display A vertical blank period. Note that this can cause a wait for up to an entire refresh period.	
	Value	**Name**
	0h	No Wait
	1h	Display Pipe A
	2h	Display Pipe B
	3h	Display Pipe C
	4h	Display Pipe D
	[5h,7h]	Reserved
7:6	**Reserved**	
	Access:	RO
	Format:	MBZ
5:0	**Display Plane Flip Pending Wait Enable**	
	Format:	Enable
	This field enables a wait for the duration of a Display Plane Flip Pending condition. If a flip request is pending, the parser will wait until the flip operation has completed (i.e., the new front buffer address has now been loaded into the active front buffer registers).	

MI_WAIT_FOR_EVENT_2

Value	Name
0h	No Wait
1h	Display Plane-1
2h	Display Plane-2
3h	Display Plane-3
4h	Display Plane-4
5h	Display Plane-5
6h	Display Plane-6
7h	Display Plane-7
8h	Display Plane-8
9h	Display Plane-9
Ah	Display Plane-10
Bh	Display Plane-11
Ch	Display Plane-12
Dh	Display Plane-13
Eh	Display Plane-14
Fh	Display Plane-15
10h	Display Plane-16
11h	Display Plane-17
12h	Display Plane-18
13h	Display Plane-19
14h	Display Plane-20
15h	Display Plane-21
16h	Display Plane-22
17h	Display Plane-23
18h	Display Plane-24
19h	Display Plane-25
1Ah	Display Plane-26
1Bh	Display Plane-27
1Ch	Display Plane-28
1Dh	Display Plane-29
1Eh	Display Plane-30
1Fh	Display Plane-31
20h	Display Plane-32
[21h, 3Fh]	Reserved

Monitor Event

MSD_MONITOR_EVENT - Monitor Event

Source: EuSubFunctionGateway

Length Bias: 1

Gateway will record for this thread if this Event ID is signaled.

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of GRF registers sent as the message payload. Value: 1 — Name: One **[Default]** — Description: See MDP_EVENT Event Data Payload definition.
	24:20	**Response Length** Default Value: 0 None Format: U5 Specifies the number of GRF registers expected as the message response payload.
	19:3	**Reserved** Access: RO Format: MBZ
	2:0	**Monitor Event Subfunction** Default Value: 0x2 Format: OpCode

Monitor No Event

MSD_MONITOR_NO_EVENT - Monitor No Event

Source: EuSubFunctionGateway

Length Bias: 1

Gateway will stop recording any Events for this thread.

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of GRF registers sent as the message payload. Value 1: One **[Default]** — Data payload is ignored.
	24:20	**Response Length** Default Value: 0 None Format: U5 Specifies the number of GRF registers expected as the message response payload.
	19:3	**Reserved** Access: RO Format: MBZ
	2:0	**Monitor No Event Subfunction** Default Value: 0x3 Format: OpCode

Move

mov - Move

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	true
Source Modifier:	true

The mov instruction moves the components in src0 into the channels of dst. If src0 and dst are of different types, format conversion is performed. If src0 is a scalar immediate, the immediate value is loaded into enabled channels of dst.
A mov with the same source and destination type, no source modifier, and no saturation is a raw move. i.e. the destination is written with an unmodified copy of the source. A packed byte destination region (B or UB type with HorzStride == 1 and ExecSize > 1) can only be written using raw move.

When denorm mode is flush to zero, a raw mov instruction with saturation modifier will not flush the denorm input or output to zero (Denorm is preserved).

Format:

```
[(pred)] mov[.cmod] (exec_size) dst src0
```

Programming Notes

A *mov* instruction with a source modifier always copies a denorm source value to a denorm destination value(in the manner of a raw move).

There is no direct conversion from B/UB to DF or DF to B/UB. Use two instructions and a word or DWord intermediate type.

There is no direct conversion from B/UB to Q/UQ or Q/UQ to B/UB. Use two instructions and a word or DWord intermediate integer type.

There is no direct conversion from HF to DF or DF to HF. Use two instructions and F (Float) as an intermediate type.

There is no direct conversion from HF to Q/UQ or Q/UQ to HF. Use two instructions and F (Float) or a word integer type or a DWord integer type as an intermediate type.

Restriction

ALT mode is not honored by raw move.

An accumulator can be a source or destination operand but not both.

IP register must not be used as destination operand when EU Fusion is enabled.

Syntax

```
[(pred)] mov[.cmod] (exec_size) reg reg
 [(pred)] mov[.cmod] (exec_size) reg imm32
 [(pred)] mov[.cmod] (exec_size) reg imm64
```

mov - Move

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         dst.chan[n] = src0.chan[n];
     }
 }
```

Src Types	Dst Types
*B,*W,*D	*B,*W,*D
*B,*W,*D	F
F	*B,*W,*D
F	F
*B,*W,*D	HF
F	HF
HF	*B,*W,*D
HF	F
HF	HF

DWord	Bit	Description
0..3	127:96	**Src0.ImmValue[31:0]** Exists If: ([Src0.IsImm]==true)
	95:92	**CondCtrl** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **FlagModifier**
	95:64	**Src0.ImmValue[63:32]** Exists If: ([Src0.IsImm]==true) AND (([Src0.DataType]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df))
	87:84	**Src0.VertStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **VertStride**
	83:81	**Src0.Width** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **Width**
	80	**Src0.AddrMode** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))

		Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Reserved** Access: RO Format: MBZ
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — Name: false **[Default]** Value: 1 — Name: true
	45:44	**Src0.Mod** Format: **SrcMod**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: **RegDataType**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: **ImmDataType**

	39:36	**Dst.DataType** Format: RegDataType
	35	**Dst.AddrMode** Format: AddrMode
	34	**Saturate** Format: Saturate
	33	**AccWrCtrl** Format: AccWrCtrl
	32	**AtomicCtrl** Format: AtomicCtrl
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". **Value** / **Name** / **Description** 0 / Normal **[Default]** / Normal. Per channel write enable used for final write enable generation. 1 / NoMask / NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	30	**Reserved**
	29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. **Value** / **Name** / **Description** 0 / NoCompaction **[Default]** / No compaction. 128-bit native instruction supporting all instruction options. 1 / Compacted / Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
	28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields **Value** / **Name** / **Description** 0 / Positive / Positive polarity of predication. Use the predication mask produced

	[Default]	by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 PredCtrl

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 FlagRegNum[0]

This field specifies bit[0] of the register number for a flag register operand.

22 FlagSubRegNum

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 ChanOff

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16 ExecSize

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

15:0 Header

Format:	Header

Move Indexed

movi - Move Indexed

Source: EuIsa

Length Bias: 4

The movi instruction performs a fast component-wise indexed move for subfields from src0 to dst. The source operand must be an indirectly-addressed register. All channels of the source operand share the same register number, which is provided by the register field of the first address subregister, with a possible immediate register offset. The register fields of the subsequent address subregisters are ignored by hardware. The subregister number of a source channel is provided by the subregister field of the corresponding address subregister, with a possible immediate subregister offset.
The destination register may be either a directly-addressed or an indirectly-addressed register.
This instruction effectively performs a subfield shuffling from one register to another.

Format:

```
[(pred)] movi (exec_size) dst src0 src1
```

Restriction
Source operand cannot be accumulators. The source operand must be a general register.
The source and destination must have the same type.
The address register for the source must be a0.0 or a0.8.
The destination register (directly or indirectly addressed) must be 16-byte aligned.
The destination region (directly or indirectly addressed) must point to the same GRF register.
The destination stride in bytes must equal the source element size in bytes.
All the index registers (address subregisters) used must point to the same GRF register.
The instruction must use 1x1 indirect regioning.
The destination offset is only used to create channel enables. Each element of the destination is directly mapped to the index registers for the movi instruction. i.e. a0.0 -> dst.0, a0.1 -> dst.1, a0.2 -> dst.2, etc.
Only 8 address subregisters are used (a0.0-a0.7 or a0.8-a0.15). Destination element will be sourced from address register (a0.0 or a0.8), for example: movi (8) r31.0:uw r[a0.0,0]<1;1,0>:uw // r31.0:uw<-a0.0:uw, r31.1:uw<-a0.1:uw, etc. movi (8) r31.0:uw r[a0.8,0]<1;1,0>:uw // r31.0:uw<-a0.8:uw, r31.1:uw<-a0.9:uw, etc. movi (8) r31.8:uw r[a0.0,0]<1;1,0>:uw // r31.8:uw<-a0.0:uw, r31.9:uw<-a0.1:uw, etc. movi (8) r31.8:uw r[a0.8,0]<1;1,0>:uw // r31.8:uw<-a0.8:uw, r31.9:uw<-a0.9:uw, etc. movi (8) r31.0:ud r[a0.0,0]<1;1,0>:ud // r31.0:ud<-a0.0:ud, r31.1:ud<-a0.1:ud, etc. movi (8) r31.0:ud r[a0.8,0]<1;1,0>:ud // r31.0:ud<-a0.8:ud, r31.1:ud<-a0.9:ud, etc.
Conditional Modifier is not allowed for this instruction.

DWord	Bit	Description
0..3	127:96	**Src0.ImmValue[31:0]** Exists If: ([Src0.IsImm]==true)
	95:92	**CondCtrl** Exists ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND

movi - Move Indexed

		If: ([Src0.DataType]!=:df) Format: **FlagModifier**
	95:64	**Src0.ImmValue[63:32]** Exists If: ([Src0.IsImm]==true) AND (([Src0.DataType]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df))
	87:84	**Src0.VertStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **VertStride**
	83:81	**Src0.Width** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **Width**
	80	**Src0.AddrMode** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**

movi - Move Indexed

	47	**Reserved** Access: RO Format: MBZ
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. 0 – false **[Default]** 1 – true
	45:44	**Src0.Mod** Format: **SrcMod**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: **RegDataType**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: **ImmDataType**
	39:36	**Dst.DataType** Format: **RegDataType**
	35	**Dst.AddrMode** Format: **AddrMode**
	34	**Saturate** Format: **Saturate**
	33	**AccWrCtrl** Format: **AccWrCtrl**
	32	**AtomicCtrl** Format: **AtomicCtrl**
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". 0 – Normal **[Default]**: Normal. Per channel write enable used for final write enable generation. 1 – NoMask: NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	30	**Reserved**
	29	**CmptCtrl** Format: MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

movi - Move Indexed

	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Multiply

mul - Multiply

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	true
Source Modifier:	true

The mul instruction performs component-wise multiplication of src0 and src1 and stores the results in dst. When multiplying integer datatypes, if src0 is DW and src1 is W, irrespective of the destination datatype, the accumulator maintains full 48-bit precision. This is required to handle the macro for 32x32 multiplication. The macro described in the mach instruction should be used to obtain the full precision 64-bit multiplication results.

Note: A 32x32 multiply operation is handled natively, without a macro. When operating in this mode, the resulting 64-bit data is packed, unlike the macro, where the lower and upper 32 bits of the result are written to different general registers by two separate instructions. Refer to the macro description for details.

When multiplying integer data types, if one of the sources is a DW, the resulting full precision data is stored in the accumulator. However, if the destination data type is either W or DW, the low bits of the result are written to the destination register and the remaining high bits are discarded. This results in undefined Overflow and Sign flags. Therefore, conditional modifiers and saturation (.sat) cannot be used in this case.

Format:

```
[(pred)] mul[.cmod] (exec_size) dst src0 src1
```

Restriction

Integer source operands cannot be accumulators.

When multiplying a DW and any lower precision integer, the DW operand must on src0.

When multiplying a DW and any lower precision integer, source modifier is not supported.

Syntax

```
[(pred)] mul[.cmod] (exec_size) reg reg reg
 [(pred)] mul[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         dst.chan[n] = src0.chan[n] * src1.chan[n];
     }
 }
```

Src Types	Dst Types
*B	*B

mul - Multiply

*B	*W
*B	*D
*W	*W
*W	*D
F	F
HF	HF

DWord	Bit	Description
0..3	127:126	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	127:96	**Src1.ImmValue[31:0]** Exists If: ([Src1.IsImm]==true)
	125:122	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	121:120	**Src1.Mod** Exists If: ([Src1.IsImm]==false) Format: **SrcMod**
	119:116	**Src1.VertStride** Exists If: ([Src1.IsImm]==false) Format: **VertStride**
	115:113	**Src1.Width** Exists If: ([Src1.IsImm]==false) Format: **Width**
	112	**Src1.AddrMode** Exists If: ([Src1.IsImm]==false) Format: **AddrMode**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect) Format: **IndirectOperand**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct) Format: **DirectOperand**
	97:96	**Src1.HorzStride** Exists If: ([Src1.IsImm]==false) Format: **HorzStride**

mul - Multiply

	95:92	**CondCtrl** Format: **FlagModifier**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==true) Format: **ImmDataType**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==false) Format: **RegDataType**
	87:84	**Src0.VertStride** Format: **VertStride**
	83:81	**Src0.Width** Format: **Width**
	80	**Src0.AddrMode** Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value. 0 – false **[Default]** 1 – true

mul - Multiply

	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. **Value** / **Name** 0 – false **[Default]** 1 – true
	45:44	**Src0.Mod** Format: **SrcMod**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: **RegDataType**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: **ImmDataType**
	39:36	**Dst.DataType** Format: **RegDataType**
	35	**Dst.AddrMode** Format: **AddrMode**
	34	**Saturate** Format: **Saturate**
	33	**AccWrCtrl** Format: **AccWrCtrl**
	32	**AtomicCtrl** Format: **AtomicCtrl**
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". **Value** / **Name** / **Description** 0 – Normal **[Default]** – Normal. Per channel write enable used for final write enable generation. 1 – NoMask – NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	30	**Reserved**
	29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations

supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Multiply Accumulate

mac - Multiply Accumulate

Source:	Eulsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	true
Source Modifier:	true

The mac instruction takes component-wise multiplication of src0 and src1, adds the results with the corresponding accumulator values, and then stores the final results in dst.

Format:

```
[(pred)] mac[.cmod] (exec_size) dst src0 src1
```

Programming Notes

When source and destination datatypes are different, the implied datatype for the accumulator operand is always the destination datatype.

Integer source operands cannot be explicit accumulators.

Restriction

The conditional modifier and saturation (.sat) must not be used when src0 or src1 are dwords.

Syntax

```
[(pred)] mac[.cmod] (exec_size) reg reg reg
 [(pred)] mac[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         dst.chan[n] = src0.chan[n] * src1.chan[n] + acc0.chan[n];
     }
 }
```

Src Types	Dst Types
*B,*W	*B,*W,*D
F	F
HF	HF

DWord	Bit	Description
0..3	127:126	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ

mac - Multiply Accumulate

	127:96	**Src1.ImmValue[31:0]** Exists If: ([Src1.IsImm]==true)
	125:122	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	121:120	**Src1.Mod** Exists If: ([Src1.IsImm]==false) Format: **SrcMod**
	119:116	**Src1.VertStride** Exists If: ([Src1.IsImm]==false) Format: **VertStride**
	115:113	**Src1.Width** Exists If: ([Src1.IsImm]==false) Format: **Width**
	112	**Src1.AddrMode** Exists If: ([Src1.IsImm]==false) Format: **AddrMode**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect) Format: **IndirectOperand**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct) Format: **DirectOperand**
	97:96	**Src1.HorzStride** Exists If: ([Src1.IsImm]==false) Format: **HorzStride**
	95:92	**CondCtrl** Format: **FlagModifier**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==true) Format: **ImmDataType**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==false) Format: **RegDataType**
	87:84	**Src0.VertStride** Format: **VertStride**

mac - Multiply Accumulate

	83:81	**Src0.Width** Format: **Width**
	80	**Src0.AddrMode** Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
	45:44	**Src0.Mod** Format: **SrcMod**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: **RegDataType**

mac - Multiply Accumulate

	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: **ImmDataType**
	39:36	**Dst.DataType** Format: **RegDataType**
	35	**Dst.AddrMode** Format: **AddrMode**
	34	**Saturate** Format: **Saturate**
	33	**AccWrCtrl** Format: **AccWrCtrl**
	32	**AtomicCtrl** Format: **AtomicCtrl**
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". **Value** / **Name** / **Description** 0 / Normal **[Default]** / Normal. Per channel write enable used for final write enable generation. 1 / NoMask / NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	30	**Reserved**
	29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. **Value** / **Name** / **Description** 0 / NoCompaction **[Default]** / No compaction. 128-bit native instruction supporting all instruction options. 1 / Compacted / Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
	28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no

predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16 **ExecSize**

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

15:0 **Header**

Format:	Header

Multiply Accumulate High

mach - Multiply Accumulate High

Source:	Eulsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	true

The mach instruction performs DWord integer multiply-accumulate operation and outputs the high DWord (bits 63:32). For each enabled channel, this instruction multiplies the DWord in src0 with the high word of the DWord in src1, left shifts the result by 16 bits, adds it with the corresponding accumulator values, and keeps the whole 64-bit result in the accumulator. It then stores the high DWord (bits 63:32) of the results in dst.
This instruction is intended to be used to emulate 32-bit DWord integer multiplication by using the large number of bits available in the accumulator. Usage of accumulator content is restricted to the emulation sequence.
For example, the following instructions perform vector multiplication of two 32-bit signed integer sources from r2 and r3 and store the resulting vectors with the high 32 bits in r5 and the low 32 bits in r6.

```
mul (8) acc0:d r2.0<8;8,1>:d r3.0<16;8,2>:uw
 mach (8) r5.0<1>:d r2.0<8;8,1>:d r3.0<8;8,1>:d
 mov (8) r6.0<1>:d acc0:d // Low 32 bits.
```

Here is a different example including negation. An added preliminary mov is required for source modification on src1.

```
mov (8) r3.0<1>:d -r3<8;8,1>:d
 mul (8) acc0:d r2.0<8;8,1>:d r3.0<16;8,2>:uw
 mach (8) r5.0<1>:d r2.0<8;8,1>:d r3.0<8;8,1>:d // High 32 bits
 mov (8) r6.0<1>:d acc0:d // Low 32 bits.
```

The mach should have channel enable from the destHI of IMUL, the mov should have the channel enable from the destLO of IMUL. As mach is used to generate part of the 64-bit DWord integer results, saturation modifier should not be used. In fact, saturation modifier should not be used for any of these four instructions. Source and destination operands must be DWord integers. Source and destination must be of the same type, signed integer or unsigned integer. If dst is UD, src0 and src1 may be UD and/or D. However, if any of src0 and src1 is D, source modifier (abs) must be present to convert it to match with dst. If dst is D, src0 and src1 must also be D. They cannot be UD as it may cause unexpected overflow because the computed results are limited to 64 bits.

Format:

```
[(pred)] mach[.cmod] (exec_size) dst src0 src1
```

Restriction
Accumulator is an implicit source and thus cannot be an explicit source operand.
The accumulator is an implicit destination and thus cannot be an explicit destination operand.

Syntax
`[(pred)] mach[.cmod] (exec size) reg reg reg`

mach - Multiply Accumulate High

```
[(pred)] mach[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
    for ( n = 0; n < exec_size; n++ ) {
       if ( WrEn.chan[n] ) {
          temp.chan[n][63:0] = (src1.chan[n][31:16] *
              src0.chan[n][31:0]) « 16 + acc.chan[n][63:0];
          if (AccWrEn) {
             acc.chan[n][63:0] = temp.chan[n][63:0];
             dst.chan[n][31:0] = temp.chan[n][63:32];
          }
          else {
             dst.chan[n][31:0] = temp.chan[n][31:0];
          }
       }
     }
```

Description

A source modifier must not be used on src1 for the macro operation. This applies to both mul and mach of the macro. If source modifier is required, an additional mov instruction may be used before the macro.

Src Types	Dst Types
D	D
UD	UD

DWord	Bit	Description	
0..3	127:126	**Reserved**	
		Exists If:	([Src1.IsImm]==false)
		Format:	MBZ
	127:96	**Src1.ImmValue[31:0]**	
		Exists If:	([Src1.IsImm]==true)
	125:122	**Reserved**	
		Exists If:	([Src1.IsImm]==false)
		Format:	MBZ
	121:120	**Src1.Mod**	
		Exists If:	([Src1.IsImm]==false)
		Format:	**SrcMod**
	119:116	**Src1.VertStride**	
		Exists If:	([Src1.IsImm]==false)
		Format:	**VertStride**
	115:113	**Src1.Width**	
		Exists If:	([Src1.IsImm]==false)
		Format:	**Width**

mach - Multiply Accumulate High

	112	**Src1.AddrMode** Exists If: ([Src1.IsImm]==false) Format: **AddrMode**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect) Format: **IndirectOperand**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct) Format: **DirectOperand**
	97:96	**Src1.HorzStride** Exists If: ([Src1.IsImm]==false) Format: **HorzStride**
	95:92	**CondCtrl** Format: **FlagModifier**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==true) Format: **ImmDataType**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==false) Format: **RegDataType**
	87:84	**Src0.VertStride** Format: **VertStride**
	83:81	**Src0.Width** Format: **Width**
	80	**Src0.AddrMode** Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct)

		Format: DirectOperand
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: IndirectOperand
	49:48	**Dst.HorzStride** Format: HorzStride
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
	45:44	**Src0.Mod** Format: SrcMod
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: RegDataType
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: ImmDataType
	39:36	**Dst.DataType** Format: RegDataType
	35	**Dst.AddrMode** Format: AddrMode
	34	**Saturate** Format: Saturate
	33	**AccWrCtrl** Format: AccWrCtrl
	32	**AtomicCtrl** Format: AtomicCtrl
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

		<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>Normal **[Default]**</td><td>Normal. Per channel write enable used for final write enable generation.</td></tr><tr><td>1</td><td>NoMask</td><td>NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.</td></tr></table>
	30	**Reserved**
	29	**CmptCtrl** <table><tr><td>Format:</td><td>MBZ</td></tr></table> Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. <table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>NoCompaction **[Default]**</td><td>No compaction. 128-bit native instruction supporting all instruction options.</td></tr><tr><td>1</td><td>Compacted</td><td>Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.</td></tr></table>
	28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields <table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>Positive **[Default]**</td><td>Positive polarity of predication. Use the predication mask produced by PredCtrl.</td></tr><tr><td>1</td><td>Negative</td><td>Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.</td></tr></table>
	27:24	**PredCtrl** <table><tr><td>Format:</td><td>**PredCtrl**</td></tr></table> This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.
	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-

		registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Multiply Add

mad - Multiply Add

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	true
Source Modifier:	true

Description

The mad instruction takes component-wise multiplication of src1 and src2, adds the results with the corresponding src0 values, and then stores the final results in dst.
The conditional modifier and saturation (.sat) must not be used when src1 or src2 are dwords.

Plane and Linear Interpolation instructions are removed. The following macros must be used to emulate Plane and Linear Interpolation operations. Plane Instruction Emulation The below plane instructionpln (16) r20.0<1>:f r10.4<0;1,0>:f r4.0<8;8,1>:fis emulated as belowmad (8) acc0<1>:f r10.7<0;1,0>:f r4.0<8;8,1>:f r10.4<0;1,0>:fmad (8) r20.0<1>:f acc0<8;8,1>:f r5.0<8;8,1>:f r10.5<0;1,0>:fmad (8) acc0<1>:f r10.7<0;1,0>:f r6.0<8;8,1>:f r10.4<0;1,0>:fmad (8) r21.0<1>:f acc0<8;8,1>:f r7.0<8;8,1>:f r10.5<0;1,0>:fln case of SIMD8 pln instruction only the first pair of mad instructions are used. Linear Interpolation Instruction Emulation The below lrp instructionlrp (16) r40.0<1>:f r10.0<8;8,1>:f r20.0<8;8,1>:f r30.0<8;8,1>:fis emulated as belowmad (8) acc0<1>:f r30.0<8;8,1>:f r10.0<8;8,1>:f r20.0<8;8,1>:fmad (8) r40.0<1>:f acc0<8;8,1>:f -r10.0<8;8,1>:f r30.0<8;8,1>:fmad (8) acc0<1>:f r31.0<8;8,1>:f r11.0<8;8,1>:f r21.0<8;8,1>:fmad (8) r41.0<1>:f acc0<8;8,1>:f -r11.0<8;8,1>:f r31.0<8;8,1>:fln case of SIMD8 lrp instruction only the first pair of mad instructions are used.

Format:

```
[(pred)] mad[.cmod] (exec_size) dst src0 src1 src2
```

Restriction

Src1/Src2 for Integer source operands cannot be accumulators. Src0 is allowed to use accumulator.

When multiplying a DW and any lower precision integer, source modifier is not supported.

All three-source instructions have certain restrictions, described in Instruction Formats.

Syntax

```
[(pred)] mad[.cmod] (exec_size) reg  reg   reg reg
 [(pred)] mad[.cmod] (exec_size) reg  reg   reg imm16
 [(pred)] mad[.cmod] (exec_size) reg  imm16 reg reg
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         dst.chan[n] = src1.chan[n] * src2.chan[n] + src0.chan[n];
     }
 }
```

mad - Multiply Add

Src Types	Dst Types
F	F
HF	HF
B	W
W, D	W, D

DWord	Bit	Description
0..3	127:114	**Src2.Operand** Exists If: ([Src2.IsImm]==false) AND ([Header][Opcode]!=madm) Format: **DirectOperand**
	127:114	**Src2.Operand** Exists If: ([Src2.IsImm]==false) AND ([Header][Opcode]==madm) Format: **MacroOperand**
	127:112	**Src2.ImmValue[15:0]** Exists If: ([Src2.IsImm]==true)
	113:112	**Src2.HorzStride** Exists If: ([Src2.IsImm]==false) Format: **HorzStride**
	111:98	**Src1.Operand** Exists If: ([Header][Opcode]!=madm) Format: **DirectOperand**
	111:98	**Src1.Operand** Exists If: ([Header][Opcode]==madm) Format: **MacroOperand**
	97:96	**Src1.HorzStride** Format: **HorzStride**
	95:92	**CondCtrl** Format: **FlagModifier**
	91	**Src1.VertStride[1]** Format: **TernaryVertStride[1:1]**
	90:88	**Src1.DataType** Format: **TernaryDataType**
	87:86	**Src1.Mod** Format: **SrcMod**
	85:84	**Src2.Mod** Format: **SrcMod**

mad - Multiply Add

	83	**Src1.VertStride[0]** Format: **TernaryVertStride[0:0]**
	82:80	**Src2.DataType** Format: **TernaryDataType**
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) AND ([Header][Opcode]!=madm) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) AND ([Header][Opcode]==madm) Format: **MacroOperand**
	79:64	**Src0.ImmValue[15:0]** Exists If: ([Src0.IsImm]==true)
	65:64	**Src0.HorzStride** Exists If: ([Src0.IsImm]==false) Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Header][Opcode]!=madm) Format: **DirectOperand** **Programming Notes** The Dst.Operand must be 64 bit aligned. i.e. Dst.Operand.SubRegNum[2:0] must be zero,
	63:50	**Dst.Operand** Exists If: ([Header][Opcode]==madm) Format: **MacroOperand**
	49	**Reserved** Format: MBZ
	48	**Dst.HorzStride** This field provides the distance in unit of data elements between two adjacent data elements within a row (horizontal) in the register region for the operand. **Value** / **Name** 0 / 1 element 1 / 2 element
	47	**Src2.IsImm** This field indicate that Source 2 operand is carrying an immediate value. **Value** / **Name** 0 / false 1 / true

mad - Multiply Add

46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. **Value** / **Name** 0 / false 1 / true
45:44	**Src0.Mod** Format: SrcMod
43	**Src0.VertStride[1]** Format: TernaryVertStride[1:1]
42:40	**Src0.DataType** Format: TernaryDataType
39	**ExecDataType** This field indicate the datatype mode of ternary instruction. Integer or Float. **Value** / **Name** 0 / Integer 1 / Float
38:36	**Dst.DataType** Format: TernaryDataType
35	**Src0.VertStride[0]** Format: TernaryVertStride[0:0]
34	**Saturate** Format: Saturate
33	**AccWrCtrl** Format: AccWrCtrl
32	**AtomicCtrl** Format: AtomicCtrl
31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". **Value** / **Name** / **Description** 0 / Normal **[Default]** / Normal. Per channel write enable used for final write enable generation. 1 / NoMask / NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	**Reserved**
29	**CmptCtrl** Format: MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

mad - Multiply Add

	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

No Operation

nop - No Operation

Source:	Eulsa
Length Bias:	4
Predication:	false
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

Do nothing. The nop instruction takes an instruction dispatch but performs no operation. It can be used for assembly patching in memory, or to insert a delay in the program sequence.

Format:

```
nop
```

Restriction

The nop instruction takes no instruction options other than Breakpoint.

Syntax

```
nop
```

Pseudocode

```
{
     ;  // The null statement, which does nothing.
 }
```

DWord	Bit	Description
0..3	127:31	**Reserved** Access: RO Format: MBZ
	30	**Reserved**
	29:28	**Reserved** Access: RO Format: MBZ
	27:26	**Reserved** Format: MBZ
	25:18	**Reserved** Access: RO Format: MBZ
	17:16	**Reserved** Format: MBZ

nop - No Operation		
	15:0	**Header** Format: **Header**

Oword Aligned Block Read MSD

MSD0R_OWAB - Oword Aligned Block Read MSD

Source: EuSubFunctionDataPort0

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18	**Legacy Message** Default Value: 0h Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 01h Format: Opcode Aligned Block Read message
	13	**Reserved** Access: RO Format: MBZ
	12:11	**Reserved** Access: RO Format: MBZ

MSD0R_OWAB - Oword Aligned Block Read MSD		
	10:8	**Data Elements** Format: **MDC_DB_OW** Specifies the number of contiguous Owords to be read
	7:0	**Binding Table Index** Format: **MDC_BTS_A32** Specifies the Binding Table Index for the message

Oword Block Read MSD

MSD0R_OWB - Oword Block Read MSD

Source: EuSubFunctionDataPort0

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18	**Legacy Message** Default Value: 0h Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 00h Format: Opcode Block Read message
	13	**Block Message Subtype** Default Value: 0 Format: Opcode Oword Block Read/Write subtype
	12:11	**Reserved** Access: RO

MSD0R_OWB - Oword Block Read MSD

		Format: MBZ
	10:8	**Data Elements** Format: **MDC_DB_OW** Specifies the number of contiguous Owords to be read or written
	7:0	**Binding Table Index** Format: **MDC_BTS_SLM_A32** Specifies the Binding Table Index for the message

Oword Block Write MSD

MSD0W_OWB - Oword Block Write MSD

Source: EuSubFunctionDataPort0

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18	**Legacy Message** Default Value: 0h Format: Opcode Legacy Message
	17:14	**Message Type** Default Value: 08h Format: Opcode Block Write message
	13	**Block Message Subtype** Default Value: 0 Format: Opcode Oword Block subtype

MSD0W_OWB - Oword Block Write MSD

	12:11	**Reserved** Access: RO Format: MBZ
	10:8	**Data Elements** Format: MDC_DB_OW Specifies the number of contiguous Owords to be read or written
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

PIPE_CONTROL

PIPE_CONTROL

Source: RenderCS, ComputeCS

Length Bias: 2

The PIPE_CONTROL command is used to effect the synchronization described above.

Programming Notes	Source
SW must follow below programming restrictions when programming PIPECONTROL command for POCS: • Write cache flush bits must not be set (Render Target Cache Flush Enable, DC Flush Enable, Depth Cache Flush Enable) • Post Sync Operations must not be set to Write PS Depth Count • Stall at Pixel Scoreboard must not be set • Notify Enable must not be set. • Depth Stall Enable must not be set. • Generic Media State Clear must not be set. • PSD Sync Enable must not be set.	

SW must follow below programming restrictions when programming PIPE_CONTROL command:
• "Command Streamer Stall Enable" must be always set.
• Post Sync Operations must not be set to Write PS Depth Count
• Following bits must not be set when programmed for ComputeCS
  • "Render Target Cache Flush Enable", "Depth Cache Flush Enable" and "Tile Cache Flush Enable"
  • "Depth Stall Enable", Stall at Pixel Scoreboard and "PSD Sync Enable".
  • "OVR Tile 0 Flush", "TBIMR Force Batch Closure", "AMFS Flush Enable" "VF Cache Invalidation Enable" and "Global Snapshot Count Reset".
	ComputeCS

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 3h GFXPIPE_3D Format: OpCode
	26:24	**3D Command Opcode** Default Value: 2h PIPE_CONTROL Format: OpCode

	23:16	**3D Command Sub Opcode** Default Value: 0h PIPE_CONTROL Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11	**Reserved** Format: MBZ
	10	**Reserved** Access: RO Format: MBZ
	9	**HDC Pipeline Flush** If set, HDC and LSC ensures it's pipeline is flushed and the memory transactions are coherent in L3$ as part of the flush operation. The HDC read-only cache is also flushed as well. This will not result in flushing L3$ potion that caches DC writes.
	8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 4h DWORD_COUNT_n Format: =n Total Length - 2. Excludes DWord (0,1).
1	31	**Reserved** Access: RO Format: MBZ
	30	**L3 Fabric Flush** **Description** L3 Fabric Flush will ensure all the pending transactions in the L3 Fabric are flushed to global observation point. HW does implicit L3 Fabric Flush on all stalling flushes (both explicit and implicit) and on PIPECONTROL having Post Sync Operation enabled. **Pipelined L3 Fabric Flush:** When Depth Stall Enable is set L3 Fabric Flush is pipelined along with the workload and issued from raster stage in the pipeline. Flush marker for this flush type on reaching raster stage, will ensure all the prior workload is complete and then L3 Fabric Flush is performed before allowing the workload following the flush marker to execute. By default L3 Fabric Flush happens top of the pipe as part of post-synchronization operation

on a flush completion.

Usages:

This mechanism is used to flush the updated compressed control state of an surface during fast clear to global observable point before the rendering operations are started using these surfaces. Most common usage case is to flush the L1 (Color, Depth) caches with L3 Fabric Flush and Depth Stall Enable post Fast Clears prior to starting the rendering operations. Refer Render Target Fast Clear and Depth Buffer Clear sections for more details.

- For a sequence of color fast clears
  - A single PIPE_CONTROL command with Render Target Cache Flush, L3 Fabric Flush and Depth Stall set at the end of the sequence suffices. This assumes the output of the fast clears are used only post this PIPE_CONTROL command.

- For a sequence of depth fast clears
  - A single PIPE_CONTROL command with Depth Cache Flush, L3 Fabric Flush and Depth Stall set at the end of the sequence suffices. This assumes the output of the depth fast clears are used only post this PIPE_CONTROL command.

- For a sequence of mixed color/depth fast clears.
  - A single PIPE_CONTROL command with Depth Cache Flush, Render Target Cache Flush, L3 Fabric Flush and Depth Stall set at the end of the sequence suffices. This assumes the output of the color/depth fast clears are used only post this PIPE_CONTROL command.

Value	Name	Description
1		HW will do a L3 Fabric Flush on completion of flush for the corresponding PIPECONTROL. Setting this bit will force HW to send a marker downstream for a flush completion.
0		HW will not force a "L3 Fabric Flush" on completion of flush and will only do on need basis.

29 **Command Cache Invalidate Enable**

Format:	Enable

When set the command cache for commands parsed at the top of the pipe will be invalidated. This bit is independent from the other bits in this command and will be executed prior to the pipeline being flushed.

28 **Tile Cache Flush Enable**

Setting this bit will force Tile Cache (contains both color and depth data) to be flushed to memory prior to this synchronization point completing. This bit must be set for all write fence sync operations to assure that results from operations initiated prior to this command are visible in memory once software observes this synchronization.

Value	Name	Description
0		Tile Cache is not flushed.
1		Tile cache is flushed.

Programming Notes

- SW must always set CS Stall bit when Tile Cache Flush Enable bit is set in the PIPECONTROL command.
- SW must ensure level1 depth and color caches are flushed prior to flushing the tile cache. This can be achieved by following means:
  - Single PIPECONTROL command to flush level1 caches and the tile cache. Attributes listed below must be set. OR
    - Tile Cache Flush Enable
    - Render Target Cache Flush Enable
    - Depth Cache Flush Enable
  - Flushing of L1 caches followed by flushing of tile cache through two different PIEPCONTROL commands. SW must ensure not to issue any rendering commands between the two PIPECONTROL commands.

Tile Cache Enabled Mode:

- SW must always set CS Stall bit when Tile Cache Flush Enable bit is set in the PIPECONTROL command.
- SW must ensure level1 depth and color caches are flushed prior to flushing the tile cache. This can be achieved by following means:
  - Single PIPECONTROL command to flush level1 caches and the tile cache. Hardware will sequence the flushing of L1 caches followed by the Tile cache. Attributes listed below must be set. OR
    - Tile Cache Flush Enable
    - Render Target Cache Flush Enable
    - Depth Cache Flush Enable
  - Flushing of L1 caches followed by flushing of tile cache through two different PIEPCONTROL commands. SW must ensure not to issue any rendering commands between the two PIPECONTROL commands.

Unified Cache (Tile Cache Disabled):
In unified cache mode of operation Color and Depth (Z) streams are cached in DC space of L2 along with Data Port stream. On a Tile Cache Flush only Color and Depth (Z) streams from DC space of L2 are flushed to globally observable and where as DC Flush Enable will only flush Data Port stream from the DC space of L2 to globally observable. Refer L3 configuration section for Unified cache usage model. In this mode of operation there is no dedicated memory allocated for Tile Cache in L2. When the Color and Depth (Z) streams are enabled to be cached in the DC space of L2, Software must use Render Target Cache Flush Enable and Depth Cache Flush Enable along with Tile Cache Flush for getting the color and depth (Z) write data to be globally observable. In this mode of operation it is not required to set CS Stall upon setting Tile Cache Flush bit.

27	**Reserved**
26	**Flush LLC**

Format:	Enable

If enabled, at the end of the current pipe-control the last level cache is cleared of all the cachelines which have been determined as being part of the Frame Buffer.

Programming Notes
SW must always program Post-Sync Operation to "Write Immediate Data" when Flush LLC is set.

25 **AMFS Flush Enable**

Format:	Enable

If enabled, at the end of the current pipe-control the AMFS unit stalls until all spawned texel shaders are completed, and then the AMFS unit flushes internal cache(s) to memory.
This bit should be enabled when a procedural texture transitions from the write state to the read state.

24 **Destination Address Type**

Defines address space of Destination Address

Value	Name	Description
0h	PPGTT	Use PPGTT address space for DW write
1h	GGTT	Use GGTT address space for DW write

Programming Notes
Ignored if ""No Write" is selected in Operation.

23 **LRI Post Sync Operation**

Value	Name	Description
0h	No LRI Operation	No LRI operation occurs as a result of this instruction. The Post-Sync Operation field is valid and may be used to specify an operation.
1h	MMIO Write Immediate Data	Write the DWord contained in Immediate Data Low (DW3) to the MMIO offset specified in the Address field.

Programming Notes
This bit causes a post sync operation with an LRI (Load Register Immediate) operation. If this bit is set then the Post-Sync Operation field must be cleared.

22 **Reserved**

21 **Store Data Index**

Format:	U1

This field is valid only if the post-sync operation is not 0. If this bit is set, the store data address is index into the global hardware status page when destination address type in the command is set to 1 (GGTT). The store data address is index into the per-process hardware status page when destination address type in the command is set to 0 (PPGTT).

PIPE_CONTROL

20 **Command Streamer Stall Enable**

Format:	U1

If ENABLED, the sync operation will not occur until all previous flush operations pending a completion of those previous flushes will complete, including the flush produced from this command. This enables the command to act similar to the legacy MI_FLUSH command.

19 **Global Snapshot Count Reset**

Format:	U1

Value	Name	Description
0h	Don't Reset	Do not reset the snapshot counts or Statistics Counters.
1h	Reset	Reset the snapshot count in Gen4 for all the units and reset the Statistics Counters except as noted above.

Programming Notes
This bit must not be exercised on any product.
TIMESTAMP is not reset by PIPE_CONTROL with this bit set. When Post Sync Operation is set to "Write PS Depth Count" along with Global Snapshot Count Reset, PS Depth Count is Reported first before resetting the value.

18 **TLB Invalidate**

Format:	U1

If ENABLED, all TLBs belonging to Render Engine will be invalidated once the flush operation is complete. Note that if the flush TLB invalidation mode is clear, a TLB invalidate will occur irrespective of this bit setting
If ENABLED, PIPE_CONTROL command will flush the in flight data written out by render engine to Global Observation point on flush done. Also Requires stall bit ([20] of DW1) set.

Programming Notes
If ENABLED, all TLBs belonging to Render Engine will be invalidated once the flush operation is complete. Note that if the flush TLB invalidation mode is clear, a TLB invalidate will occur irrespective of this bit setting.
Post Sync Operation or CS stall must be set to ensure a TLB invalidate occurs. Otherwise no cycle will occur to the TLB cache to invalidate.

17 **PSD Sync Enable**

Format:	Enable

If set, Pixel Shader Dispatch Units will stall successive PS threads from being dispatched until all the prior PS threads complete. Once all PSDs are synced up, post-sync LRI can be optionally used to change EU enables.

Programming Notes
If this bit is set, these bits must not be set: • Depth Stall Enable • Command Streamer Stall Enable • Render Target Cache Flush Enable • Stall At Pixel Scoreboard

16 **Generic Media State Clear**

Format:	Disable

If set, all generic media state context information will be invalidated. Any state invalidated will not be saved as part of the render engine context image. The state only becomes valid once it is parsed by the command streamer.

15:14 **Post Sync Operation**

Description
This field specifies an optional action to be taken upon completion of the synchronization operation.
This field must be cleared if the LRI Post-Sync Operation bit is set.

Value	Name	Description	Programming Notes
0h	No Write	No write occurs as a result of this instruction. This can be used to implement a "trap" operation, etc.	
1h	Write Immediate Data	Write the QWord containing Immediate Data Low, High DWs to the Destination Address	
2h	Write PS Depth Count	Write the 64-bit PS_DEPTH_COUNT register to the Destination Address	Workaround : Driver must program PIPE_CONTROL with only Depth Stall Enable bit set prior to programming a PIPE_CONTROL with Write PS Depth Count Post sync operation.
3h	Write Timestamp	Write the 64-bit TIMESTAMP register(i.e. "Reported Timestamp Count" 0x2358 for render pipe) to the Destination Address.	

Programming Notes
If executed in non-secure batch buffer, the address given will be in a PPGTT address space. If in a secure ring or batch, address given will be in GGTT space

PIPE_CONTROL

13 **Depth Stall Enable**

Format:	Enable

This bit must be set when obtaining a "visible pixel" count to preclude the possible inclusion in the PS_DEPTH_COUNT value written to memory of some fraction of pixels from objects initiated after the PIPE_CONTROL command.

Value	Name	Description
0h	Disable	3D pipeline will not stall subsequent primitives at the Depth Test stage.
1h	Enable	3D pipeline will stall any subsequent primitives at the Depth Test stage until the Sync and Post-Sync operations complete.

Programming Notes
This bit must be DISABLED for operations other than writing PS_DEPTH_COUNT.
This bit will have no effect (besides preventing write cache flush) if set in a PIPE_CONTROL command issued to the Media pipe.

12 **Render Target Cache Flush Enable**

Format:	Enable

Setting this bit will force Render Cache to be flushed to memory prior to this synchronization point completing. This bit must be set for all write fence sync operations to assure that results from operations initiated prior to this command are visible in memory once software observes this synchronization.

Value	Name	Description
0h	Disable Flush	Render Target Cache is NOT flushed.
1h	Enable Flush	Render Target Cache is flushed.

Programming Notes
Whenever a Binding Table Index (BTI) used by a Render Target Message points to a different RENDER_SURFACE_STATE, SW must issue a Render Target Cache Flush by enabling this bit.
When render target flush is set due to new association of BTI, PS Scoreboard Stall bit must be set in this packet.

11 **Instruction Cache Invalidate Enable**

Format:	Enable

Setting this bit is independent of any other bit in this packet. This bit controls the invalidation of the L1 and L2 at the top of the pipe i.e. at the parsing time.

10 **Texture Cache Invalidation Enable**

Format:	Enable

Setting this bit is independent of any other bit in this packet. This bit controls the invalidation of the texture caches at the top of the pipe i.e. at the parsing time.

9 **Indirect State Pointers Disable**

Format:	Enable

Description
At the completion of the post-sync operation associated with this pipe control packet, the indirect state pointers in the hardware are considered invalid; the indirect pointers are not saved in the context. If any new indirect state commands are executed in the command stream while the pipe control is pending, the new indirect state commands are preserved.
Using Invalidate State Pointer (ISP) only inhibits context restoring of Push Constant (3DSTATE_CONSTANT_*) commands. Push Constant commands are only considered as Indirect State Pointers. Once ISP is issued in a context, SW must initialize by programming push constant commands for all the shaders (at least to zero length)before attempting any rendering operation for the same context.

Programming Notes
When executed in POCS results in inhibiting the context restore of 3DSTATE_CONSTANT_VS from POSH engine context.

8 **Notify Enable**

Format:	Enable

If ENABLED, a Sync Completion Interrupt will be generated (if enabled by the MI Interrupt Control registers) once the sync operation is complete. See Interrupt Control Registers in Memory Interface Registers for details.

7 **Pipe Control Flush Enable**

Format:	Enable

Hardware on parsing PIPECONTROL command with Pipe Control Flush Enable set will wait for all the outstanding post sync operations corresponding to previously executed PIPECONTROL commands are complete before making forward progress.

6 **Reserved**

5 **DC Flush Enable**

Format:	Enable

Setting this bit enables flushing of the L3$ portions that caches DC writes.

Programming Notes
DC Flush (L3 Flush)by default doesn't result in flushing/invalidating the IA Coherent lines from L3$, however this can be achieved by setting control bit **Pipe line flush Coherent lines** in L3SQCREG4 register.

4 **VF Cache Invalidation Enable**

Format:	Enable

Setting this bit is independent of any other bit in this packet. This bit controls the invalidation of VF address based cache at the top of the pipe i.e. at the parsing time.

Programming Notes

When executed from POCS it results in invalidating the VFR cache.

3 **Constant Cache Invalidation Enable**

Format:	Enable

Setting this bit is independent of any other bit in this packet. This bit controls the invalidation of the constant cache at the top of the pipe i.e. at the parsing time.

2 **State Cache Invalidation Enable**

Format:	Enable

Setting this bit is independent of any other bit in this packet. This bit controls the invalidation of the L1 and L2 state caches at the top of the pipe i.e. at the parsing time.

Programming Notes

When State Cache redirect to CS Section enable bit is set in MMIO register SLICE_COMMON_ECO_CHCICKEN1 (0731Ch), Command Cache Invalidate Enable must be also set upon setting State Cache Invalidate bit in PIPE_CONTROL command.

1 **Stall At Pixel Scoreboard**

Format:	Enable

Defines the behavior of PIPE_CONTROL command at the pixel scoreboard.

Value	Name	Description
0h	Disable	Stall at the pixel scoreboard is disabled.
1h	Enable	Stall at the pixel scoreboard is enabled.

Programming Notes

This bit must be DISABLED for End-of-pipe (Read) fences, PS_DEPTH_COUNT or TIMESTAMP queries. This bit is ignored if Depth Stall Enable is set. Further the render cache is not flushed even if Write Cache Flush Enable bit is set.

0 **Depth Cache Flush Enable**

Format:	Enable

Setting this bit enables flushing (i.e. writing back the dirty lines to memory and invalidating the tags) of depth related caches. This bit applies to HiZ cache, Stencil cache and depth cache.

Value	Name	Description
0h	Flush Disabled	Depth relates caches (HiZ, Stencil and Depth) are NOT flushed.
1h	Flush Enabled	Depth relates caches (HiZ, Stencil and Depth) are flushed.

Programming Notes

Ideally depth caches need to be flushed only when depth is required to be coherent in memory for later use as a texture, source or honoring CPU lock. This bit must be DISABLED for End-of-pipe (Read) fences, PS_DEPTH_COUNT or TIMESTAMP queries.

<table>
<tr><th colspan="3">PIPE_CONTROL</th></tr>
<tr><td rowspan="2">2</td><td>31:2</td><td>Address
<table><tr><td>Format:</td><td>GraphicsAddress[31:2]U32</td></tr></table>If Post Sync Operation is set to 1h LRI Post-Sync Operation must be clear): Bits 31:3 specify the QW address of where the Immediate Data following this DW in the packet to be stored. Bit 2 MBZ Ignored if "No Write" is the selected in Post-Sync Operation. If LRI Post-Sync Operation is set: Bits 22:2 (Bits 31:23 are reserved MBZ) specify the MMIO offset destination for the data in the Immediate Data Low (DW3) field. Only DW writes are valid.</td></tr>
<tr><td>1:0</td><td>Reserved
<table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table></td></tr>
<tr><td>3</td><td>31:0</td><td>Address High
<table><tr><td>Format:</td><td>GraphicsAddress[63:32]U32</td></tr></table>This field specifies the 4GB aligned base address of gfx 4GB virtual address space within the host's 64-bit virtual address space.This field is valid only if the post-sync operation is not 0 and the LRI Post-Sync Operation is clear.</td></tr>
<tr><td>4..5</td><td>63:0</td><td>Immediate Data
<table><tr><td>Format:</td><td>U64</td></tr></table>This field specifies the QWord value to be written to the targeted location. Only valid when Post-Sync Operation is 1h (Write Immediate Data) or LRI Post-Sync Operation is set. Ignored if Post-Sync Operation is "No write", "Write PS_DEPTH_COUNT" or "Write TIMESTAMP".</td></tr>
</table>

PIPELINE_SELECT

PIPELINE_SELECT

Source:	RenderCS, ComputeCS
Length Bias:	1

Description

The PIPELINE_SELECT command is used to specify which GPE pipeline is to be considered the 'current' active pipeline.

Issuing 3D-pipeline-specific commands when the GPGPU pipeline is selected, or vice versa, is UNDEFINED.

Programming common non pipeline commands (e.g., STATE_BASE_ADDRESS) is allowed in all pipeline modes.

Programming Notes

Software must ensure Render Cache, Depth Cache and HDC Pipeline flush are flushed through a stalling PIPE_CONTROL command prior to programming of PIPELINE_SELECT command transitioning Pipeline Select from 3D to GPGPU/Media.
Software must ensure HDC Pipeline flush and Generic Media State Clear is issued through a stalling PIPE_CONTROL command prior to programming of PIPELINE_SELECT command transitioning Pipeline Select from GPGPU/Media to 3D.
Example:

- Workload-3Dmode,
- PIPE_CONTROL (CS Stall, Depth Cache Flush Enable, Render Target Cache Flush Enable, HDC Pipeline Flush Enable) ,
- PIPELINE_SELECT (GPGPU),
- Workload-GPGPU mode,
- PIPE_CONTROL (CS Stall, HDC Pipeline Flush Enable, Generic Media State Clear),
- PIPELINE_SELECT (3D) ...

"Pipe Selection" must be never set to "3D" in PIPELINE_SELECT command programmed for workloads submitted to ComputeCS.

While GPU is operating in GPGPU mode of operation and when a Mid Thread Preemption (if enabled) occurs on a PIPELINE_SELECT command with Media Sampler DOP CG Enable reset along with Pipeline Select Mode set to 3D and on resubmission of this context on context restore Sampler DOP CG Enable will be reset. This would mean the GPGPU mid thread preempted threads restored will get executed with media sampler DOP clock not gated consuming media sampler DOP power until all GPGPU threads have retired.
Programming of the PIPELINE_SELECT can be modified to avoid the above inefficiency. This can be done by programming Pipeline Selection and Media Sampler DOP CG Enable fields in two different PIPELINE_SELECT commands instead of on single PIPELINE_SELECT command.
Example:
PIPELINE_SELECT (Pipeline Selection = 3D, Media Sampler DOP CG Enable = False)
To
PIPELINE_SELECT (Pipeline Selection = 3D)
PIPELINE_SELECT (Media Sampler DOP CG Enable = False)

<table>
<tr><th>DWord</th><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>31:29</td><td>Command Type
<table>
<tr><td>Default Value:</td><td>3h GFXPIPE</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>28:27</td><td>Command SubType
<table>
<tr><td>Default Value:</td><td>1h GFXPIPE_SINGLE_DW</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>26:24</td><td>3D Command Opcode
<table>
<tr><td>Default Value:</td><td>1h GFXPIPE_NONPIPELINED</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>23:16</td><td>3D Command Sub Opcode
<table>
<tr><td>Default Value:</td><td>04h PIPELINE_SELECT</td></tr>
<tr><td>Format:</td><td>OpCode</td></tr>
</table></td></tr>
<tr><td></td><td>15:8</td><td>Mask Bits
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>Must be set to modify corresponding bits in Bits 7:0. (For implemented bits)</td></tr>
</table></td></tr>
<tr><td></td><td>7</td><td>Reserved
<table>
<tr><td>Access:</td><td>RO</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>6</td><td>Media Sampler Power Clock Gate Disable
<table>
<tr><td>Format:</td><td>U1</td></tr>
</table>
By default, the media power clock gating is always ON.
When set, Command Streamer sends message to PM to disable media sampler power Clock Gating.
<table>
<tr><th>Programming Notes</th></tr>
<tr><td>Mask bit [14] has to be set for HW to look at this field when PIPELINE_SELECT command is parsed.
Each BB/workload is responsible to set this control. This can be only enabled/disabled at the frame level.</td></tr>
</table></td></tr>
<tr><td></td><td>5</td><td>Reserved
<table>
<tr><td>Access:</td><td>RO</td></tr>
<tr><td>Format:</td><td>MBZ</td></tr>
</table></td></tr>
<tr><td></td><td>4</td><td>Media Sampler DOP Clock Gate Enable
<table>
<tr><td>Format:</td><td>Enable</td></tr>
</table>
<table>
<tr><th>Value</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>Disabled</td><td>Command Streamer sends message to PM to disable sampler DOP Clock Gating.</td></tr>
<tr><td>1</td><td>Enabled</td><td>Command Streamer sends message to PM to enable media sampler DOP</td></tr>
</table></td></tr>
</table>

PIPELINE_SELECT

		Clock Gating.

Programming Notes
Mask bit [12] has to be set for HW to look at this field when PIPELINE_SELECT command is parsed.

3 **Render Sampler Power Gate Enable**

Format:	Enable

Value	Name	Description
0	Disabled	Command Streamer sends message to PM to disable render sampler Power Gating.
1	Enabled	Command Streamer sends message to PM to enable render sampler Power Gating.

Programming Notes
Mask bit [11] has to be set for HW to look at this field when PIPELINE_SELECT command is parsed.

2 **Render Slice common Power Gate Enable**

Format:	Enable

Value	Name	Description
0	Disabled	Command Streamer sends message to PM to disable render slice common Power Gating.
1	Enabled	Command Streamer sends message to PM to enable render slice common Power Gating.

Programming Notes
Mask bit [10] has to be set for HW to look at this field when PIPELINE_SELECT command is parsed.

1:0 **Pipeline Selection**

Value	Name	Description
0	3D	3D pipeline is selected
1	Media	Media pipeline is selected (Includes HD optical disc playback, HD video playback, and generic media workloads)
2	GPGPU	GPGPU pipeline is selected

Programming Notes
Mask bits [9:8] has to be set for HW to look at this field when PIPELINE_SELECT command is parsed. Setting only one of the mask bit [9] or [8] is illegal.

Read Surface Info MSD

MSD_RSI - Read Surface Info MSD

Source: EuSubFunctionReadOnlyDataPort

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHF Indicates that the message forbids a header.
	18	**Reserved** Access: RO Format: MBZ
	17:14	**Message Type** Default Value: 06h Format: Opcode Read Surface Info message
	13:8	**Reserved** Access: RO Format: MBZ
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

REP16 Render Target Write MSD

MSD_RTW_REP16 - REP16 Render Target Write MSD

Source: EuSubFunctionRenderDataPort

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Message Precision Subtype** Default Value: 0h Format: Opcode Full precision data message
	29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	**Per-Coarse Pixel PS outputs enable** Format: Enable This bit indicates the render target write is a coarse pixel write. **Programming Notes** This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.

MSD_RTW_REP16 - REP16 Render Target Write MSD

	17:14	**Message Type** Default Value: 0Ch Format: Opcode Render Target Write message
	13	**Per-Sample PS Enable** Format: Enable If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot. **Programming Notes** This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.
	12	**Last Render Target Select** Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.
	11	**Slot Group Select** Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 1h Format: Opcode SIMD16 Single source message with replicated data. Use slots [15:0] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:16] are referenced instead of [15:0].

MSD_RTW_REP16 - REP16 Render Target Write MSD		
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Reserved Instruction0

Reserved Instruction0		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:27	**Opcode 1** Default Value: 0x00000003 Format: Opcode
	26:24	**Opcode 2** Default Value: 0x00000000 Format: Opcode
	23:16	**Opcode 3** Default Value: 0x00000053 Format: Opcode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction1

Reserved Instruction1		
Length Bias:	1	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000000 Format: Opcode
	28:23	**Opcode 1** Default Value: 0x0000000E Format: Opcode
	22:0	**Reserved** Access: RO Format: MBZ
0..n	31:0	**Unknown Bitfield**

Reserved Instruction2

Reserved Instruction2		
Length Bias:	1	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000000 Format: Opcode
	28:23	**Opcode 1** Default Value: 0x0000000E Format: Opcode
	22:0	**Reserved** Access: RO Format: MBZ
0..n	31:0	**Unknown Bitfield**

Reserved Instruction3

Reserved Instruction3		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:16	**Opcode 1** Default Value: 0x00001608 Format: Opcode
	15:0	**Reserved** Access: RO Format: MBZ
0..n	31:0	**Unknown Bitfield**

Reserved Instruction4

Reserved Instruction4		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:16	**Opcode 1** Default Value: 0x00001600 Format: Opcode
	15:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction5

Reserved Instruction5		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:16	**Opcode 1** Default Value: 0x0000160A Format: Opcode
	15:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction6

Reserved Instruction6		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:16	**Opcode 1** Default Value: 0x00001609 Format: Opcode
	15:0	**Reserved** Access: RO Format: MBZ
0..n	31:0	**Unknown Bitfield**

Reserved Instruction7

Reserved Instruction7		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:27	**Opcode 1** Default Value: 0x00000002 Format: Opcode
	26:24	**Opcode 2** Default Value: 0x00000000 Format: Opcode
	23:21	**Opcode 3** Default Value: 0x00000000 Format: Opcode
	20:16	**Opcode 4** Default Value: 0x00000005 Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction8

Reserved Instruction8		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:27	**Opcode 1** Default Value: 0x00000002 Format: Opcode
	26:23	**Opcode 2** Default Value: 0x00000003 Format: Opcode
	22:16	**Opcode 3** Default Value: 0x00000007 Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction9

Reserved Instruction9		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:27	**Opcode 1** Default Value: 0x00000002 Format: Opcode
	26:23	**Opcode 2** Default Value: 0x00000003 Format: Opcode
	22:16	**Opcode 3** Default Value: 0x00000006 Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction10

Reserved Instruction10		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:27	**Opcode 1** Default Value: 0x00000002 Format: Opcode
	26:23	**Opcode 2** Default Value: 0x00000007 Format: Opcode
	22:16	**Opcode 3** Default Value: 0x00000007 Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction11

Reserved Instruction11		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:27	**Opcode 1** Default Value: 0x00000002 Format: Opcode
	26:23	**Opcode 2** Default Value: 0x00000007 Format: Opcode
	22:16	**Opcode 3** Default Value: 0x00000006 Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction12

Reserved Instruction12		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:27	**Opcode 1** Default Value: 0x00000002 Format: Opcode
	26:23	**Opcode 2** Default Value: 0x0000000B Format: Opcode
	22:16	**Opcode 3** Default Value: 0x00000007 Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction13

Reserved Instruction13		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:27	**Opcode 1** Default Value: 0x00000002 Format: Opcode
	26:23	**Opcode 2** Default Value: 0x0000000B Format: Opcode
	22:16	**Opcode 3** Default Value: 0x00000006 Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction14

Reserved Instruction14		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:27	**Opcode 1** Default Value: 0x00000002 Format: Opcode
	26:23	**Opcode 2** Default Value: 0x0000000B Format: Opcode
	22:16	**Opcode 3** Default Value: 0x00000003 Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction15

Reserved Instruction15		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:27	**Opcode 1** Default Value: 0x00000002 Format: Opcode
	26:23	**Opcode 2** Default Value: 0x0000000B Format: Opcode
	22:16	**Opcode 3** Default Value: 0x00000002 Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction16

Reserved Instruction16		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:27	**Opcode 1** Default Value: 0x00000002 Format: Opcode
	26:23	**Opcode 2** Default Value: 0x0000000B Format: Opcode
	22:16	**Opcode 3** Default Value: 0x00000001 Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction17

Reserved Instruction17		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:27	**Opcode 1** Default Value: 0x00000002 Format: Opcode
	26:23	**Opcode 2** Default Value: 0x0000000B Format: Opcode
	22:16	**Opcode 3** Default Value: 0x00000005 Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction18

Reserved Instruction18		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:27	**Opcode 1** Default Value: 0x00000002 Format: Opcode
	26:23	**Opcode 2** Default Value: 0x0000000B Format: Opcode
	22:16	**Opcode 3** Default Value: 0x00000000 Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction19

Reserved Instruction19		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:27	**Opcode 1** Default Value: 0x00000002 Format: Opcode
	26:23	**Opcode 2** Default Value: 0x0000000B Format: Opcode
	22:16	**Opcode 3** Default Value: 0x00000021 Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction20

Reserved Instruction20		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:27	**Opcode 1** Default Value: 0x00000002 Format: Opcode
	26:23	**Opcode 2** Default Value: 0x0000000B Format: Opcode
	22:16	**Opcode 3** Default Value: 0x00000020 Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction21

Reserved Instruction21		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:27	**Opcode 1** Default Value: 0x00000002 Format: Opcode
	26:23	**Opcode 2** Default Value: 0x0000000B Format: Opcode
	22:16	**Opcode 3** Default Value: 0x00000004 Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction22

Reserved Instruction22		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:27	**Opcode 1** Default Value: 0x00000002 Format: Opcode
	26:23	**Opcode 2** Default Value: 0x00000001 Format: Opcode
	22:21	**Opcode 3** Default Value: 0x00000000 Format: Opcode
	20:16	**Opcode 4** Default Value: 0x00000006 Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction23

Reserved Instruction23		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:27	**Opcode 1** Default Value: 0x00000002 Format: Opcode
	26:23	**Opcode 2** Default Value: 0x00000001 Format: Opcode
	22:21	**Opcode 3** Default Value: 0x00000000 Format: Opcode
	20:16	**Opcode 4** Default Value: 0x00000009 Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction24

Reserved Instruction24		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:27	**Opcode 1** Default Value: 0x00000002 Format: Opcode
	26:23	**Opcode 2** Default Value: 0x00000001 Format: Opcode
	22:21	**Opcode 3** Default Value: 0x00000000 Format: Opcode
	20:16	**Opcode 4** Default Value: 0x00000005 Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction25

Reserved Instruction25		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:27	**Opcode 1** Default Value: 0x00000002 Format: Opcode
	26:23	**Opcode 2** Default Value: 0x00000001 Format: Opcode
	22:21	**Opcode 3** Default Value: 0x00000000 Format: Opcode
	20:16	**Opcode 4** Default Value: 0x00000008 Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Reserved Instruction26

Reserved Instruction26		
Length Bias:	2	
DWord	**Bit**	**Description**
0	31:29	**Opcode 0** Default Value: 0x00000003 Format: Opcode
	28:27	**Opcode 1** Default Value: 0x00000002 Format: Opcode
	26:24	**Opcode 2** Default Value: 0x00000004 Format: Opcode
	23:21	**Opcode 3** Default Value: 0x00000000 Format: Opcode
	20:16	**Opcode 4** Default Value: 0x00000003 Format: Opcode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Count** Format: =n
0..n	31:0	**Unknown Bitfield**

Return

ret - Return

Source:	Eulsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

Return execution to the code sequence that called a subroutine. The ret instruction can be predicated or non-predicated. If non-predicated, all channels jump to the return IP in the first channel of src0 and restore CallMask from the second channel of src0. If predicated, the enabled channels jump to the return IP from the first channel of src0 and the corresponding bits in the CallMask are cleared to zero; if all CallMask bits are zero after the ret instruction, then execution jumps to the return IP from the first channel of src0. When SPF is on, the predication control must be scalar.

Format:

```
[(pred)] ret (exec_size) null src0
```

Restriction

This instruction cannot take accumulator as source.

Syntax

```
[(pred)] ret (exec_size) null reg
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         PcIP[n] = src0.chan[0];
         CallMask[n] = 0;
     } else {
         PcIP[n] = IP + 1;
     }
 }
 for ( n = exec_size; n < 32; n++ ) {
     PcIP[n] = IP + 1;
 }
 if ( CallMask[n:0] == 0) ) {  // all channels are zero
     Jump(src0.chan[0]);
     CallMask = src0.chan[1];
 }
```

DWord	Bit	Description
0..3	127:96	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ

ret - Return

	127:96	**JIP** Exists If: ([Src0.IsImm]==true) Format: S31 The byte-aligned jump distance if a jump is taken for the channel
	95:80	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	95:64	**Reserved** Exists If: ([Src0.IsImm]==true) Format: MBZ
	79:66	**Src0.Operand** Exists If: ([Src0.IsImm]==false) Format: **DirectOperand**
	65:64	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	63:50	**Dst.Operand** Format: **DirectOperand**
	49:47	**Reserved** Access: RO Format: MBZ
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — false; 1 — true
	45:34	**Reserved** Access: RO Format: MBZ
	33	**BranchCtrl** This field is used by*goto*,*if*, and*else*instructions to control branching. See the **goto** instruction description for more information about BranchCtrl.
	32	**AtomicCtrl** Format: **AtomicCtrl**
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 **Reserved**

29 **CmptCtrl**

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-

		registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Rotate Left

rol - Rotate Left

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	false
Source Modifier:	false

Perform component-wise logical rotate left operation of the bits in src0 by the rotate count indicated in src1, storing the result in dst. src0 and src1 are treated as unsigned numbers with only the bits within the specified datatype used during this operation. This operation does not produce sign or overflow conditions. Only the .e/.z or .ne/.nz conditional modifiers are supported. Extra precision bits available in accumulator are ignored during this operation and only the bits within the specified datatype are used.
src0 and dst must be of same datatype precision.

Format:

```
 [(pred)] rol[.cmod] (exec_size) dst src0 src1
```

Syntax

```
[(pred)] rol[.cmod] (exec_size) reg reg reg
 [(pred)] rol[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         RotateMask = sizeof(src0) * 8 - 1;
         dst.chan[n] = (src0.chan[n] « (src1.chan[n] & RotateMask)) |
                        src0.chan[n] » (-src1.chan[n] & RotateMask));
     }
 }
```

Src Types	Dst Types
UW, UD	UW, UD

DWord	Bit	Description
0..3	127:126	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	127:96	**Src1.ImmValue[31:0]** Exists If: ([Src1.IsImm]==true)
	125:122	**Reserved** Exists If: ([Src1.IsImm]==false)

	Format:	MBZ
121:120	**Src1.Mod**	
	Exists If:	([Src1.IsImm]==false)
	Format:	**SrcMod**
119:116	**Src1.VertStride**	
	Exists If:	([Src1.IsImm]==false)
	Format:	**VertStride**
115:113	**Src1.Width**	
	Exists If:	([Src1.IsImm]==false)
	Format:	**Width**
112	**Src1.AddrMode**	
	Exists If:	([Src1.IsImm]==false)
	Format:	**AddrMode**
111:98	**Src1.Operand**	
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)
	Format:	**IndirectOperand**
111:98	**Src1.Operand**	
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)
	Format:	**DirectOperand**
97:96	**Src1.HorzStride**	
	Exists If:	([Src1.IsImm]==false)
	Format:	**HorzStride**
95:92	**CondCtrl**	
	Format:	**FlagModifier**
91:88	**Src1.DataType**	
	Exists If:	([Src1.IsImm]==true)
	Format:	**ImmDataType**
91:88	**Src1.DataType**	
	Exists If:	([Src1.IsImm]==false)
	Format:	**RegDataType**
87:84	**Src0.VertStride**	
	Format:	**VertStride**
83:81	**Src0.Width**	
	Format:	**Width**
80	**Src0.AddrMode**	
	Format:	**AddrMode**

rol - Rotate Left

79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Direct) Format: **DirectOperand**
79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
65:64	**Src0.HorzStride** Format: **HorzStride**
63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
49:48	**Dst.HorzStride** Format: **HorzStride**
47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
45:44	**Src0.Mod** Format: **SrcMod**
43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: **RegDataType**
43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: **ImmDataType**
39:36	**Dst.DataType** Format: **RegDataType**

35	**Dst.AddrMode** Format: **AddrMode**
34	**Saturate** Format: **Saturate**
33	**AccWrCtrl** Format: **AccWrCtrl**
32	**AtomicCtrl** Format: **AtomicCtrl**
31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30	**Reserved**
29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the

		predication mask.
	27:24	**PredCtrl** Format: **PredCtrl** This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.
	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Rotate Right

ror - Rotate Right

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	false
Source Modifier:	false

Perform component-wise logical rotate right operation of the bits in src0 by the rotate count indicated in src1, storing the result in dst. src0 and src1 are treated as unsigned numbers with only the bits within the specified datatype used during this operation. This operation does not produce sign or overflow conditions. Only the .e/.z or .ne/.nz conditional modifiers are supported. Extra precision bits available in accumulator are ignored during this operation and only the bits within the specified datatype are used.
src0 and dst must be of same datatype precision.

Format:

```
[(pred)] ror[.cmod] (exec_size) dst src0 src1
```

Syntax

```
[(pred)] ror[.cmod] (exec_size) reg reg reg
[(pred)] ror[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
for ( n = 0; n < exec_size; n++ ) {
    if ( WrEn.chan[n] ) {
        RotateMask = sizeof(src0) * 8 - 1;
        dst.chan[n] = (src0.chan[n] » (src1.chan[n] & RotateMask)) |
                      (src0.chan[n] « (-src1.chan[n] & RotateMask));
    }
}
```

Src Types	Dst Types
UW, UD	UW, UD

DWord	Bit	Description
0..3	127:126	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	127:96	**Src1.ImmValue[31:0]** Exists If: ([Src1.IsImm]==true)
	125:122	**Reserved** Exists If: ([Src1.IsImm]==false)

		Format:	MBZ
	121:120	**Src1.Mod**	
		Exists If:	([Src1.IsImm]==false)
		Format:	**SrcMod**
	119:116	**Src1.VertStride**	
		Exists If:	([Src1.IsImm]==false)
		Format:	**VertStride**
	115:113	**Src1.Width**	
		Exists If:	([Src1.IsImm]==false)
		Format:	**Width**
	112	**Src1.AddrMode**	
		Exists If:	([Src1.IsImm]==false)
		Format:	**AddrMode**
	111:98	**Src1.Operand**	
		Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)
		Format:	**IndirectOperand**
	111:98	**Src1.Operand**	
		Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)
		Format:	**DirectOperand**
	97:96	**Src1.HorzStride**	
		Exists If:	([Src1.IsImm]==false)
		Format:	**HorzStride**
	95:92	**CondCtrl**	
		Format:	**FlagModifier**
	91:88	**Src1.DataType**	
		Exists If:	([Src1.IsImm]==true)
		Format:	**ImmDataType**
	91:88	**Src1.DataType**	
		Exists If:	([Src1.IsImm]==false)
		Format:	**RegDataType**
	87:84	**Src0.VertStride**	
		Format:	**VertStride**
	83:81	**Src0.Width**	
		Format:	**Width**
	80	**Src0.AddrMode**	
		Format:	**AddrMode**

ror - Rotate Right

79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Direct) Format: **DirectOperand**
79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
65:64	**Src0.HorzStride** Format: **HorzStride**
63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
49:48	**Dst.HorzStride** Format: **HorzStride**
47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value. 0 – false **[Default]** 1 – true
46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. 0 – false **[Default]** 1 – true
45:44	**Src0.Mod** Format: **SrcMod**
43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: **RegDataType**
43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: **ImmDataType**
39:36	**Dst.DataType** Format: **RegDataType**

35 **Dst.AddrMode**

Format:	AddrMode

34 **Saturate**

Format:	Saturate

33 **AccWrCtrl**

Format:	AccWrCtrl

32 **AtomicCtrl**

Format:	AtomicCtrl

31 **MaskCtrl**

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 **Reserved**

29 **CmptCtrl**

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the

		predication mask.
	27:24	**PredCtrl** Format: **PredCtrl** This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.
	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Round Down

rndd - Round Down

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	true
Source Modifier:	true

The rndd instruction takes component-wise floating point downward rounding (to the integral float number closer to negative infinity) of src0 and storing the rounded integral float results in dst. This is commonly referred to as the floor() function. Each result follows the rules in the following tables based on the floating-point mode.

Format:

```
[(pred)] rndd[.cmod] (exec_size) dst src0
```

Syntax

```
[(pred)] rndd[.cmod] (exec_size) reg reg
 [(pred)] rndd[.cmod] (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         dst.chan[n] = floor(src0.chan[n]);
     }
 }
```

Src Types	Dst Types
F	F

DWord	Bit	Description
0..3	127:96	**Src0.ImmValue[31:0]** Exists If: ([Src0.IsImm]==true)
	95:92	**CondCtrl** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **FlagModifier**
	95:64	**Src0.ImmValue[63:32]** Exists If: ([Src0.IsImm]==true) AND (([Src0.DataType]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df))

rndd - Round Down

	87:84	**Src0.VertStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **VertStride**
	83:81	**Src0.Width** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **Width**
	80	**Src0.AddrMode** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Reserved** Access: RO Format: MBZ
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value.

	Value	Name
	0	false **[Default]**
	1	true

45:44 Src0.Mod

Format:	SrcMod

43:40 Src0.DataType

Exists If:	([Src0.IsImm]==false)
Format:	RegDataType

43:40 Src0.DataType

Exists If:	([Src0.IsImm]==true)
Format:	ImmDataType

39:36 Dst.DataType

Format:	RegDataType

35 Dst.AddrMode

Format:	AddrMode

34 Saturate

Format:	Saturate

33 AccWrCtrl

Format:	AccWrCtrl

32 AtomicCtrl

Format:	AtomicCtrl

31 MaskCtrl

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 Reserved

29 CmptCtrl

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16 **ExecSize**

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

rndd - Round Down		
	15:0	**Header** Format: **Header**

Round to Nearest or Even

rnde - Round to Nearest or Even

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	true
Source Modifier:	true

The rnde instruction takes component-wise floating point round-to-even operation of src0 with results in two pieces - a downward rounded integral float results stored in dst and the round-to-even increments stored in the rounding increment bits. The round-to-even increment must be added to the results in dst to create the final round-to-even values to emulate the round-to-even operation, commonly known as the round() function. The final results are the one of the two integral float values that is nearer to the input values. If the neither possibility is nearer, the even alternative is chosen. Each result follows the rules in the following tables based on the floating-point mode.

Format:

```
[(pred)] rnde[.cmod] (exec_size) dst src0
```

Syntax

```
[(pred)] rnde[.cmod] (exec_size) reg reg
 [(pred)] rnde[.cmod] (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         if ( src0.chan[n] - floor(src0.chan[n]) > 0.5f ) {
             dst.chan[n] = floor(src0.chan[n]) + 1;
         } else if ( src0.chan[n] - floor(src0.chan[n]) < 0.5f ) {
             dst.chan[n] = floor(src0.chan[n]);
         } else {
             if ( floor(src0.chan[n]) is odd ) {
                 dst.chan[n] = floor(src0.chan[n]) + 1;
             } else {
                 dst.chan[n] = floor(src0.chan[n]);
             }
         }
     }
 }
```

Src Types	Dst Types
F	F

DWord	Bit	Description	
0..3	127:96	**Src0.ImmValue[31:0]**	
		Exists If:	([Src0.IsImm]==true)

rnde - Round to Nearest or Even

	95:92	**CondCtrl** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **FlagModifier**
	95:64	**Src0.ImmValue[63:32]** Exists If: ([Src0.IsImm]==true) AND (([Src0.DataType]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df))
	87:84	**Src0.VertStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **VertStride**
	83:81	**Src0.Width** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **Width**
	80	**Src0.AddrMode** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**

rnde - Round to Nearest or Even

49:48	**Dst.HorzStride** Format: **HorzStride**
47	**Reserved** Access: RO Format: MBZ
46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. **Value** / **Name** 0 – false **[Default]** 1 – true
45:44	**Src0.Mod** Format: **SrcMod**
43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: **RegDataType**
43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: **ImmDataType**
39:36	**Dst.DataType** Format: **RegDataType**
35	**Dst.AddrMode** Format: **AddrMode**
34	**Saturate** Format: **Saturate**
33	**AccWrCtrl** Format: **AccWrCtrl**
32	**AtomicCtrl** Format: **AtomicCtrl**
31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". **Value** / **Name** / **Description** 0 – Normal **[Default]** – Normal. Per channel write enable used for final write enable generation. 1 – NoMask – NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	**Reserved**

29 **CmptCtrl**

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

rnde - Round to Nearest or Even		
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Round to Zero

rndz - Round to Zero

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	true
Source Modifier:	true

The rndz instruction takes component-wise floating point round-to-zero operation of src0 with results in two pieces - a downward rounded integral float results stored in dst and the round-to-zero increments stored in the rounding increment bits. The round-to-zero increment must be added to the results in dst to create the final round-to-zero values to emulate the round-to-zero operation, commonly known as the truncate() function. The final results are the one of the two closest integral float values to the input values that is nearer to zero.

Format:

```
[(pred)] rndz[.cmod] (exec_size) dst src0
```

Syntax

```
[(pred)] rndz[.cmod] (exec_size) reg reg
 [(pred)] rndz[.cmod] (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         dst.chan[n] = floor(src0.chan[n]);
         if ( abs(src0.chan[n]) < abs(dst.chan[n]) ) {
             dst.chan[n] = floor(src0.chan[n]) + 1;
         } else {
             dst.chan[n] = floor(src0.chan[n]);
         }
     }
 }
```

Src Types	Dst Types
F	F

DWord	Bit	Description
0..3	127:96	**Src0.ImmValue[31:0]** Exists If: ([Src0.IsImm]==true)
	95:92	**CondCtrl** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))

		Format: **FlagModifier**
	95:64	**Src0.ImmValue[63:32]** Exists If: ([Src0.IsImm]==true) AND (([Src0.DataType]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df))
	87:84	**Src0.VertStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **VertStride**
	83:81	**Src0.Width** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **Width**
	80	**Src0.AddrMode** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**

rndz - Round to Zero

	47	**Reserved** Access: RO Format: MBZ
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — Name: false **[Default]** Value: 1 — Name: true
	45:44	**Src0.Mod** Format: **SrcMod**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: **RegDataType**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: **ImmDataType**
	39:36	**Dst.DataType** Format: **RegDataType**
	35	**Dst.AddrMode** Format: **AddrMode**
	34	**Saturate** Format: **Saturate**
	33	**AccWrCtrl** Format: **AccWrCtrl**
	32	**AtomicCtrl** Format: **AtomicCtrl**
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". Value: 0 — Name: Normal **[Default]** — Description: Normal. Per channel write enable used for final write enable generation. Value: 1 — Name: NoMask — Description: NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	30	**Reserved**

rndz - Round to Zero

29 CmptCtrl

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 PredInv

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 PredCtrl

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 FlagRegNum[0]

This field specifies bit[0] of the register number for a flag register operand.

22 FlagSubRegNum

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 ChanOff

		Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Round Up

rndu - Round Up

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	true
Source Modifier:	true

The rndu instruction takes component-wise floating point upward rounding (to the integral float number closer to positive infinity) of src0, commonly known as the ceiling() function. Each result follows the rules in the following tables based on the floating-point mode.

Format:

```
[(pred)] rndu[.cmod] (exec_size) dst src0
```

Syntax

```
[(pred)] rndu[.cmod] (exec_size) reg reg
 [(pred)] rndu[.cmod] (exec_size) reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         if ( src0.chan[n] - floor(src0.chan[n]) > 0.0f ) {
             dst.chan[n] = floor(src0.chan[n]) + 1;
         } else {
             dst.chan[n] = src0.chan[n];
         }
     }
 }
```

Src Types	Dst Types
F	F

DWord	Bit	Description
0..3	127:96	**Src0.ImmValue[31:0]** Exists If: ([Src0.IsImm]==true)
	95:92	**CondCtrl** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **FlagModifier**
	95:64	**Src0.ImmValue[63:32]** Exists If: ([Src0.IsImm]==true) AND (([Src0.DataType]==:q) OR ([Src0.DataType]==:uq) OR ([Src0.DataType]==:df))

rndu - Round Up

	87:84	**Src0.VertStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **VertStride**
	83:81	**Src0.Width** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **Width**
	80	**Src0.AddrMode** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: (([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df))) AND ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Exists If: ([Src0.IsImm]==false) OR (([Src0.DataType]!=:q) AND ([Src0.DataType]!=:uq) AND ([Src0.DataType]!=:df)) Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Reserved** Access: RO Format: MBZ
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value.

rndu - Round Up

Value	Name
0	false **[Default]**
1	true

45:44 **Src0.Mod**

Format:	**SrcMod**

43:40 **Src0.DataType**

Exists If:	([Src0.IsImm]==false)
Format:	**RegDataType**

43:40 **Src0.DataType**

Exists If:	([Src0.IsImm]==true)
Format:	**ImmDataType**

39:36 **Dst.DataType**

Format:	**RegDataType**

35 **Dst.AddrMode**

Format:	**AddrMode**

34 **Saturate**

Format:	**Saturate**

33 **AccWrCtrl**

Format:	**AccWrCtrl**

32 **AtomicCtrl**

Format:	**AtomicCtrl**

31 **MaskCtrl**

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 **Reserved**

29 **CmptCtrl**

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16 **ExecSize**

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

rndu - Round Up		
	15:0	**Header** Format: **Header**

Sampler Cache Media Block Read MSD

MSD_SC_MB - Sampler Cache Media Block Read MSD

Source: EuSubFunctionReadOnlyDataPort

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18	**Reserved** Access: RO Format: MBZ
	17:14	**Message Type** Default Value: 05h Format: Opcode Media Block Read Sampler Cache message
	13:11	**Reserved** Access: RO Format: MBZ
	10:8	**Vertical Line Stride Override** Format: MDC_VLSO If enabled, specifies the Vertical Line Stride and Vertical Line Stride Offset override fields.

MSD_SC_MB - Sampler Cache Media Block Read MSD

	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Sampler Cache Oword Aligned Block Read MSD

MSD_SC_OWAB - Sampler Cache Oword Aligned Block Read MSD

Source: EuSubFunctionReadOnlyDataPort

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18	**Reserved** Access: RO Format: MBZ
	17:14	**Message Type** Default Value: 04h Format: Opcode Oword Aligned Block Read Sampler Cache message
	13:11	**Reserved** Access: RO Format: MBZ
	10:8	**Data Elements** Format: MDC_DB_OW Specifies the number of contiguous Owords to be read

MSD_SC_OWAB - Sampler Cache Oword Aligned Block Read MSD		
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Scratch Block Read MSD

MSD0R_SB - Scratch Block Read MSD

Source: EuSubFunctionDataPort0

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18	**Scratch Block Message** Default Value: 1h Format: Opcode Scratch Block Message
	17	**Operation Type** Default Value: 0h Format: Opcode Scratch Block Read message
	16	**Reserved** Access: RO Format: MBZ
	15	**Invalidate After Read** Format: MDC_IAR Specifies if L3 cache lines accessed by the message should be invalidated after the read occurs

MSD0R_SB - Scratch Block Read MSD

	14	**Reserved** Access: RO Format: MBZ
	13:12	**Data Elements** Format: MDC_DB_HW Specifies the number of registers to be read or written
	11:0	**Address Offset** Format: GeneralStateOffset[16:5] HWORD (32 byte) based address offset to the BufferAddress in the Message Header.

Scratch Block Write MSD

MSD0W_SB - Scratch Block Write MSD

Source: EuSubFunctionDataPort0

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header.
	18	**Scratch Block Message** Default Value: 1h Format: Opcode Scratch Block Message
	17	**Operation Type** Default Value: 1h Format: Opcode Scratch Block Write message
	16:14	**Reserved** Access: RO Format: MBZ
	13:12	**Data Elements** Format: MDC_DB_HW Specifies the number of registers to be read or written

MSD0W_SB - Scratch Block Write MSD		
	11:0	**Address Offset** Format: GeneralStateOffset[17:6] HWORD (32 byte) based address offset to the BufferAddress in the Message Header.

Select

sel - Select

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	true
Source Modifier:	true

The sel instruction selectively moves the components in src0 or src1 into the channels of dst based on the predication. On a channel by channel basis, if the channel condition is true, data in src0 is moved into dst. Otherwise, data in src1 is moved into dst.
As the predication is used to select the two sources, it is not included in the evaluation of WrEn. The predicate clause is mandatory if cmod is omitted/0000b. If both predication and the conditional modifier are omitted, the results are undefined.
If the conditional modifier is specified (not 0000b, a compare is performed and the resulting condition flag is used for the sel instruction. Conditional modifiers .ge and .lt follow the cmpn rules, and all other conditional modifiers follow the cmp rules. Predication is not allowed in this mode.
A sel instruction with cmod .lt is used to emulate a MIN instruction.
A sel instruction with cmod .ge is used to emulate a MAX instruction.
For a sel instruction with a .lt or .ge conditional modifier, if one source is NaN and the other not NaN, the non-NaN source is the result. If both sources are NaNs, the result is NaN. For all other conditional modifiers, if either source is NaN then src1 is selected.
A sel instruction without a conditional modifier always copies a denorm source value to a denorm destination value (in the manner of a raw move). This applies even if the source modifies are set on the *sel* instruction sources.
The sel instruction uses any conditional modifier internally and does not update the flag register if a conditional modifier is used.

A sel instruction with cmod will flush denorm to zero, depending on the denorm mode bit.

Format:

```
(pred) sel[.cmod] (exec_size) dst src0 src1
```

Syntax

```
(pred) sel[.cmod] (exec_size) reg reg reg
 (pred) sel[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn, NoPMask);
 if (cmod == "0000") {  // no CMod Evaluate(PMask);
     for ( n = 0; n < exec_size; n++ ) {
         if ( WrEn.chan[n] ) {
             if ( PMask.channel[n] ) {
                 dst.chan[n] = src0.chan[n];
             } else {
                 dst.chan[n] = src1.chan[n];
```

sel - Select

```
            }
        }
    }
} else {  // with CMod Evaluate(CMod);
    for ( n = 0; n < exec_size; n++ ) {
        if ( WrEn.chan[n] ) {
            if ( CMod.chan[n] ) {
                dst.chan[n] = src0.chan[n];
            } else {
                dst.chan[n] = src1.chan[n];
            }
        }
    }
}
```

Src Types	Dst Types
*B,*W,*D	*B,*W,*D
F	F
HF	HF

DWord	Bit	Description
0..3	127:126	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	127:96	**Src1.ImmValue[31:0]** Exists If: ([Src1.IsImm]==true)
	125:122	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	121:120	**Src1.Mod** Exists If: ([Src1.IsImm]==false) Format: **SrcMod**
	119:116	**Src1.VertStride** Exists If: ([Src1.IsImm]==false) Format: **VertStride**
	115:113	**Src1.Width** Exists If: ([Src1.IsImm]==false) Format: **Width**
	112	**Src1.AddrMode** Exists If: ([Src1.IsImm]==false) Format: **AddrMode**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect)

sel - Select

	Format:	**IndirectOperand**
111:98	**Src1.Operand**	
	Exists If:	([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct)
	Format:	**DirectOperand**
97:96	**Src1.HorzStride**	
	Exists If:	([Src1.IsImm]==false)
	Format:	**HorzStride**
95:92	**CondCtrl**	
	Format:	**FlagModifier**
91:88	**Src1.DataType**	
	Exists If:	([Src1.IsImm]==true)
	Format:	**ImmDataType**
91:88	**Src1.DataType**	
	Exists If:	([Src1.IsImm]==false)
	Format:	**RegDataType**
87:84	**Src0.VertStride**	
	Format:	**VertStride**
83:81	**Src0.Width**	
	Format:	**Width**
80	**Src0.AddrMode**	
	Format:	**AddrMode**
79:66	**Src0.Operand**	
	Exists If:	([Src0.AddrMode]==Direct)
	Format:	**DirectOperand**
79:66	**Src0.Operand**	
	Exists If:	([Src0.AddrMode]==Indirect)
	Format:	**IndirectOperand**
65:64	**Src0.HorzStride**	
	Format:	**HorzStride**
63:50	**Dst.Operand**	
	Exists If:	([Dst.AddrMode]==Direct)
	Format:	**DirectOperand**
63:50	**Dst.Operand**	
	Exists If:	([Dst.AddrMode]==Indirect)
	Format:	**IndirectOperand**

sel - Select

	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value. Value: 0 — Name: false **[Default]** Value: 1 — Name: true
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — Name: false **[Default]** Value: 1 — Name: true
	45:44	**Src0.Mod** Format: **SrcMod**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: **RegDataType**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: **ImmDataType**
	39:36	**Dst.DataType** Format: **RegDataType**
	35	**Dst.AddrMode** Format: **AddrMode**
	34	**Saturate** Format: **Saturate**
	33	**AccWrCtrl** Format: **AccWrCtrl**
	32	**AtomicCtrl** Format: **AtomicCtrl**
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". Value: 0 — Name: Normal **[Default]** — Description: Normal. Per channel write enable used for final write enable generation. Value: 1 — Name: NoMask — Description: NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel,

		as described in the Evaluate Write Enable section.

30	**Reserved**
29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24	**PredCtrl** Format: **PredCtrl** This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.
23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

sel - Select		
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Send Message

send - Send Message

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false
Subfunctions:	SFID[95:92]

Description

The send instruction performs data communication between a thread and external function units, including shared functions (Data Port Read, Data Port Write, URB, and Message Gateway) and some fixed functions (e.g. Thread Spawner, who also have an unique Shared Function ID). The send instruction adds an entry to the EU's message request queue. The request message is stored in a split pair of contiguous GRF registers. Typically the header and addresses in one block and the data in another, but this is not strictly necessary and null may be passed as either parameter. The response message, if present, will be returned to a block of contiguous GRF registers. The return GRF writes may be in any order depending on the external function units. <src0> and <src1> are the lead GRF registers for the first and second block of the request respectively. <dest> is the lead GRF register for response. The message descriptor field <desc> contains the Message Length (the number of consecutive GRF registers corresponding to src0) and the Response Length (the number of consecutive GRF registers). It also contains the header present bit, and the function control signals. The extend message descriptor field <ex_desc> contains the target function ID, the Extended Message Length (the number of consecutive GRF registers corresponding to src1) and the extended function control signals. WrEn is forwarded to the target function in the message sideband. The send instruction is the only way to terminate a thread. When the EOT (End of Thread) bit of <ex_desc> is set, it indicates the end of thread to the EU, the Thread Dispatcher and, in most cases, the parent fixed function.

The send instruction performs data communication between a thread and external function units, including shared functions (Sampler, Data Port Read, Data Port Write, URB, and Message Gateway) and some fixed functions (e.g. Thread Spawner, who also have an unique Shared Function ID). The send instruction adds an entry to the EU's message request queue. The request message is stored in a split pair of contiguous GRF registers. Typically the header and addresses in one block and the data in another, but this is not strictly necessary and null may be passed as either parameter. The response message, if present, will be returned to a block of contiguous GRF registers. The return GRF writes may be in any order depending on the external function units. <src0> and <src1> are the lead GRF registers for the first and second block of the request respectively. <dest> is the lead GRF register for response. The message descriptor field <desc> contains the Message Length (the number of consecutive GRF registers corresponding to src0) and the Response Length (the number of consecutive GRF registers). It also contains the header present bit, and the function control signals. The extend message descriptor field <ex_desc> contains the target function ID, the Extended Message Length (the number of consecutive GRF registers corresponding to src1) and the extended function control signals. WrEn is forwarded to the target function in the message sideband. The send instruction is the only way to terminate a thread. When the EOT (End of Thread) bit of <ex_desc> is set, it indicates the end of thread to the EU, the Thread Dispatcher and, in most cases, the parent fixed function.

send - Send Message

Message descriptor field <desc> can be a 32-bit immediate, imm32, or a 32-bit scalar register, <reg32a>. The 32-bit scalar register <reg32a> must be the leading dword of the address register. It should be in the form of a0.0.

<src0> is a GRF register. It serves as the leading GRF register of the request.
<src1> is a GRF register or a null register. It serves as the leading GRF register for the second block of the request when it is not a null register. It is required that the second block of GRFs does not overlap with the first block. If it is a null register the Extended Message Length must be 0. The sum of Message Length and Extended Message Length must not be greater than 15.
<dest> serves for two purposes: to provide the leading GRF register location for the response message if present, and to provide parameters to form the channel-enable sideband signals.
<dest> signals whether there is a response to the message request. It can be either a null register or a GRF register.
If <dest> is null, there is no response to the request. Meanwhile, the Response Length field in <desc> must be 0. Certain types of message requests, such as memory write (store) through the Data Port, do not want response data from the function unit. If so, the posted destination operand can be null.
If <dest> is a GRF register, the register number is forwarded to the shared function. In this case, the target function unit must send one or more response message phases back to the requesting thread. The number of response message phases must match the Response Length field in <desc>, which of course cannot be zero. For some cases, it could be an empty return message. An empty return message is defined as a single phase message with all channel enables turned off.
The 16-bit channel enables of the message sideband are formed based on the WrEn. Interpretation of the channel-enable sideband signals is subject to the target external function. In general for a 'send' instruction with return messages, they are used as the destination dword write mask for the GRF registers starting at <dest>. For a message that has multiple return phases, the same set of channel enable signals applies to all the return phases.

Send a message stored in GRF locations starting at <src0> followed by <src1> to a shared function identified by <ex_desc> along with control from <desc> and <ex_desc> with a GRF writeback location at <dest>.

Format:

```
[(pred)] send.<sfid> (exec_size) <dest> <src0> <src1> <ex_desc> <desc> {[EOT]}
```

Programming Notes

Send instruction execution (reading GRFs and sending out) is guranteed to be in-order for a SharedFunction specified by SFID except for SLM. SLM SharedFunction is decoded as follows.
SLM = ((SFID==DC0) && (desc[18] == 0) && (desc[7:0]==0xFE)) ||
((SFID==DC1) && (desc[7:0]==0xFE)) ||
((SFID==DC2) && (desc[19] == 0)&&(desc[7]==0x1))

Restriction

Software must obey the following rules in signaling the end-of-thread using the send instruction: The posted destination operand must be null. No acknowledgement is allowed for the send instruction that signifies the end of thread. This is to avoid deadlock as the EU is expecting to free up the terminated thread's resource. A thread must terminate with a send instruction with message to a shared function on the output message bus; therefore, it cannot terminate with a send instruction with message to the NULL function. For example, a thread may terminate with a URB write message or a render cache write message. A root thread originated from the media (generic) pipeline must terminate with a send instruction with message to the Thread Spawner unit. A

send - Send Message

child thread should also terminate with a send to TS. Please refer to the Media Chapter for more detailed description.

Software must obey the following rules in signaling the end-of-thread using the send instruction: The posted destination operand must be null. No acknowledgement is allowed for the send instruction that signifies the end of thread. This is to avoid deadlock as the EU is expecting to free up the terminated thread's resource. A thread must terminate with a send instruction with message to a shared function on the output message bus; therefore, it cannot terminate with a send instruction with message to the sampler unit or the NULL function. For example, a thread may terminate with a URB write message or a render cache write message. A root thread originated from the media (generic) pipeline must terminate with a send instruction with message to the Thread Spawner unit. A child thread should also terminate with a send to TS. Please refer to the Media Chapter for more detailed description.

A send instruction cannot update accumulator registers.

EOT must NOT be set for the send instruction when using indirect register addressing mode.

An EOT send must use register space r112-r127 for sources. This is to enable loading of a new thread into the same slot while the message with EOT for current thread is pending dispatch.

Syntax

```
[(pred)] send.<sfid> (exec_size) reg  greg  reg  (imm32|reg32a) (imm32|reg32a) {[EOT]}
```

Pseudocode

```
Evaluate(WrEn);
 <MsgChEnable> = WrEn;
 <SourceReg0> = <src0>.RegNum;
 <SourceReg1> = <src1>.RegNum;
 MessageEnqueue(<MsgChEnable>, <ResponseReg>, <SourceReg0>, <SourceReg1>,<ex_desc>,
<dest>);
```

DWord	Bit	Description
0..3	127:124	**ExDesc[31:28]** Exists If: ([ExDesc.IsReg]==false) Format: **ExMsgDesc[31:28]**
	127:124	**Reserved** Exists If: ([ExDesc.IsReg]==true) Format: MBZ
	123:122	**Desc[31:30]** Exists If: ([Desc.IsReg]==false) Format: **MsgDesc[31:30]**
	123:113	**Reserved** Exists If: ([Desc.IsReg]==true) Format: MBZ
	121:113	**Desc[19:11]**

send - Send Message

	Exists If:	([Desc.IsReg]==false)
	Format:	**MsgDesc[19:11]**
112	**Reserved**	
	Access:	RO
	Format:	MBZ
111:104	**Src1.RegNum**	
	Format:	**DirectOperand[13:6]**
103:99	**Src1.Length**	
	Exists If:	([ExDesc.IsReg]==false)
	Format:	**ExMsgDesc[10:6]**
103:99	**Reserved**	
	Exists If:	([ExDesc.IsReg]==true)
	Format:	MBZ
98	**Src1.RegFile**	
	Format:	**DirectOperand[0]**
97:96	**Reserved**	
	Exists If:	([ExDesc.IsReg]==true)
	Format:	MBZ
97:96	**ExDesc[27:26]**	
	Exists If:	([ExDesc.IsReg]==false)
	Format:	**ExMsgDesc[27:26]**
95:92	**SFID**	
	Format:	**SFID**
91:81	**Reserved**	
	Exists If:	([Desc.IsReg]==true)
	Format:	MBZ
91:81	**Desc[10:0]**	
	Exists If:	([Desc.IsReg]==false)
	Format:	**MsgDesc[10:0]**
80	**Reserved**	
	Access:	RO
	Format:	MBZ
79:72	**Src0.RegNum**	
	Format:	**DirectOperand[13:6]**

send - Send Message

	71	**MsgDesc[29]** Exists If: ([Desc.IsReg]==false) Format: **MsgDesc[29:29]**
	71:67	**Reserved** Exists If: ([Desc.IsReg]==true) Format: MBZ
	70:67	**Src0.Length** Exists If: ([Desc.IsReg]==false) Format: **MsgDesc[28:25]**
	66	**Src0.RegFile** Format: **DirectOperand[0]**
	65:64	**Reserved** Exists If: ([ExDesc.IsReg]==true) Format: MBZ
	65:64	**ExDesc[25:24]** Exists If: ([ExDesc.IsReg]==false) Format: **ExMsgDesc[25:24]**
	63:56	**Dst.RegNum** Format: **DirectOperand[13:6]**
	55:51	**Reserved** Exists If: ([Desc.IsReg]==true) Format: MBZ
	55:51	**Dst.Length** Exists If: ([Desc.IsReg]==false) Format: **MsgDesc[24:20]**
	50	**Dst.RegFile** Format: **DirectOperand[0]**
	49	**ExDesc.IsReg** This field indicates that the extended message descriptor is provided by the address register, selected by the AddrSubRegNum[3:1]. Value: 0 — Name: false Value: 1 — Name: true
	48	**Desc.IsReg** This field indicates that the message descriptor is provided by the address subregister a0.0. Value: 0 — Name: false

	1	true

47:43 Reserved

Exists If:	([ExDesc.IsReg]==true)
Format:	MBZ

47:35 ExDesc[23:11]

Exists If:	([ExDesc.IsReg]==false)
Format:	**ExMsgDesc[23:11]**

42:40 AddrSubRegNum[3:1]

Exists If:	([ExDesc.IsReg]==true)
Format:	**AddrSubRegNum[3:1]**

38:36 Reserved

Exists If:	([ExDesc.IsReg]==true)
Format:	MBZ

35 Reserved

Exists If:	([ExDesc.IsReg]==true)
Format:	MBZ

34 EOT

This field controls the termination of the thread. For a send instruction, if this field is set, EU will terminate the thread and also set the EOT bit in the message sideband. This field only applies to the send instruction. It is not present for other instructions.

Value	Name
0	Thread is not terminated
1	EOT

33 FusionCtrl

This field provides explicit control for EU fusion lock-step execution. When this bit is set to 1b, the instruction is executed serially starting from the first EU to the last EU in the fused set.

Value	Name
0	Normal lockstep execution
1	Serialized execution

32 AtomicCtrl

Format:	**AtomicCtrl**

31 MaskCtrl

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel,

		as described in the Evaluate Write Enable section.

30 — **Reserved**

29 — **CmptCtrl**

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 — **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 — **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 — **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 — **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

send - Send Message		
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Send Message Conditional

sendc - Send Message Conditional

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false
Subfunctions:	SFID[95:92]

The sendc instruction has the same behavior as the sends instruction except the following. sendc first checks the dependent threads inside the Thread Dependency Register. There are up to 8 dependent threads in the TDR register. The sendc instruction executes only when all the dependent threads in the TDR register are retired.

Wait for dependencies in the TDR Register to clear, then send a message stored in GRF locations starting at <src0> followed by <src1> to a shared function identified by <ex_desc> along with control from <desc> and <ex_desc> with a GRF writeback location at <dest>.

Format:

```
[(pred)] sendc.<sfid> (exec_size) <dest> <src0> <src1> <ex_desc> <desc> {[EOT]}
```

Restriction

The sendc instruction has the same restrictions as the send instruction.

Syntax

```
[(pred)] sendc.<sfid> (exec_size) reg  greg  reg  (imm32|reg32a) (imm32|reg32a) {[EOT]}
```

Pseudocode

```
if (TDR[7] ... || TDR[2] || TDR[1] || TDR[0]) { wait; }
 Evaluate(WrEn);
 <MsgChEnable> = WrEn;
 MessageEnqueue(<MsgChEnable>, <ResponseReg>, <src0>.RegNum, <src1>.RegNum; ,<ex_desc>,
<dest>.RegNum);
```

DWord	Bit	Description
0..3	127:124	**ExDesc[31:28]** Exists If: ([ExDesc.IsReg]==false) Format: **ExMsgDesc[31:28]**
	127:124	**Reserved** Exists If: ([ExDesc.IsReg]==true) Format: MBZ

sendc - Send Message Conditional

	123:122	**Desc[31:30]** Exists If: ([Desc.IsReg]==false) Format: **MsgDesc[31:30]**
	123:113	**Reserved** Exists If: ([Desc.IsReg]==true) Format: MBZ
	121:113	**Desc[19:11]** Exists If: ([Desc.IsReg]==false) Format: **MsgDesc[19:11]**
	112	**Reserved** Access: RO Format: MBZ
	111:104	**Src1.RegNum** Format: **DirectOperand[13:6]**
	103:99	**Src1.Length** Exists If: ([ExDesc.IsReg]==false) Format: **ExMsgDesc[10:6]**
	103:99	**Reserved** Exists If: ([ExDesc.IsReg]==true) Format: MBZ
	98	**Src1.RegFile** Format: **DirectOperand[0]**
	97:96	**Reserved** Exists If: ([ExDesc.IsReg]==true) Format: MBZ
	97:96	**ExDesc[27:26]** Exists If: ([ExDesc.IsReg]==false) Format: **ExMsgDesc[27:26]**
	95:92	**SFID** Format: **SFID**
	91:81	**Reserved** Exists If: ([Desc.IsReg]==true) Format: MBZ
	91:81	**Desc[10:0]** Exists If: ([Desc.IsReg]==false) Format: **MsgDesc[10:0]**

sendc - Send Message Conditional

	80	**Reserved** Access: RO Format: MBZ
	79:72	**Src0.RegNum** Format: **DirectOperand[13:6]**
	71	**MsgDesc[29]** Exists If: ([Desc.IsReg]==false) Format: **MsgDesc[29:29]**
	71:67	**Reserved** Exists If: ([Desc.IsReg]==true) Format: MBZ
	70:67	**Src0.Length** Exists If: ([Desc.IsReg]==false) Format: **MsgDesc[28:25]**
	66	**Src0.RegFile** Format: **DirectOperand[0]**
	65:64	**Reserved** Exists If: ([ExDesc.IsReg]==true) Format: MBZ
	65:64	**ExDesc[25:24]** Exists If: ([ExDesc.IsReg]==false) Format: **ExMsgDesc[25:24]**
	63:56	**Dst.RegNum** Format: **DirectOperand[13:6]**
	55:51	**Reserved** Exists If: ([Desc.IsReg]==true) Format: MBZ
	55:51	**Dst.Length** Exists If: ([Desc.IsReg]==false) Format: **MsgDesc[24:20]**
	50	**Dst.RegFile** Format: **DirectOperand[0]**
	49	**ExDesc.IsReg** This field indicates that the extended message descriptor is provided by the address register, selected by the AddrSubRegNum[3:1].

sendc - Send Message Conditional

	Value	Name
	0	false
	1	true

48 Desc.IsReg

This field indicates that the message descriptor is provided by the address subregister a0.0.

Value	Name
0	false
1	true

47:43 Reserved

Exists If:	([ExDesc.IsReg]==true)
Format:	MBZ

47:35 ExDesc[23:11]

Exists If:	([ExDesc.IsReg]==false)
Format:	**ExMsgDesc[23:11]**

42:40 AddrSubRegNum[3:1]

Exists If:	([ExDesc.IsReg]==true)
Format:	**AddrSubRegNum[3:1]**

38:36 Reserved

Exists If:	([ExDesc.IsReg]==true)
Format:	MBZ

35 Reserved

Exists If:	([ExDesc.IsReg]==true)
Format:	MBZ

34 EOT

This field controls the termination of the thread. For a send instruction, if this field is set, EU will terminate the thread and also set the EOT bit in the message sideband. This field only applies to the send instruction. It is not present for other instructions.

Value	Name
0	Thread is not terminated
1	EOT

33 FusionCtrl

This field provides explicit control for EU fusion lock-step execution. When this bit is set to 1b, the instruction is executed serially starting from the first EU to the last EU in the fused set.

Value	Name
0	Normal lockstep execution
1	Serialized execution

sendc - Send Message Conditional

32	**AtomicCtrl** Format: AtomicCtrl
31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the the per channel write enables are used to generate the final write enable. This field should be normally "0". **Value** / **Name** / **Description** 0 / Normal **[Default]** / Normal. Per channel write enable used for final write enable generation. 1 / NoMask / NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
30	**Reserved**
29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. **Value** / **Name** / **Description** 0 / NoCompaction **[Default]** / No compaction. 128-bit native instruction supporting all instruction options. 1 / Compacted / Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields **Value** / **Name** / **Description** 0 / Positive **[Default]** / Positive polarity of predication. Use the predication mask produced by PredCtrl. 1 / Negative / Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.
27:24	**PredCtrl** Format: PredCtrl This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

SFC_AVS_CHROMA_Coeff_Table

SFC_AVS_CHROMA_Coeff_Table

Source: BSpec

Length Bias: 2

This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Format: OpCode Value: Ah — Name: Media Misc **[Default]** — Description: [] Media MFX/VEBOX+SFC Mode
	22:21	**SubOpcodeA** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcodeB** Default Value: 6h SFC_AVS CHROMA Coeff_Table Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 3Fh Excludes DWord (0,1) Format: =n Total Length - 2
1..64	2047:0	**AVS CHROMA Coefficient Table Body** Format: SFC_AVS_CHROMA_COEFF_TABLE_BODY[32]

SFC_AVS_LUMA_Coeff_Table

SFC_AVS_LUMA_Coeff_Table

Source: BSpec

Length Bias: 2

This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Format: OpCode Value: Ah — Name: Media Misc **[Default]** — Description: [] Media MFX/VEBOX+SFC Mode
	22:21	**SubOpcodeA** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcodeB** Default Value: 5h SFC_AVS LUMA Coeff_Table Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 7Fh Excludes DWord (0,1) Format: =n Total Length - 2
1..128	4095:0	**AVS LUMA Coefficient Table Body** Format: SFC_AVS_LUMA_COEFF_TABLE_BODY[32]

SFC_AVS_STATE

SFC_AVS_STATE

Source: BSpec

Length Bias: 2

This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Format: OpCode Value: Ah — Name: Media Misc **[Default]** — Description: [] Media MFX/VEBOX+SFC Modegen
	22:21	**SubOpcodeA** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcodeB** Default Value: 2h SFC_AVS_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 2h Excludes DWord (0,1) Format: =n Total Length - 2
1..3	95:0	**AVS State Body** Format: SFC_AVS_STATE_BODY

SFC_FRAME_START

SFC_FRAME_START

Source: BSpec

Length Bias: 2

This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Format: OpCode Value: Ah — Name: Media Misc **[Default]** — Description: [] Media MFX/VEBOX+SFC Mode
	22:21	**SubOpcodeA** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcodeB** Default Value: 4h SFC_FRAME_START Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n Total Length - 2
1	31:0	**Frame Start Body** Format: SFC_FRAME_START_BODY

SFC_IEF_STATE

SFC_IEF_STATE

Source: BSpec

Length Bias: 2

This command is sent from VDBOX/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Format: OpCode Value: Ah — Name: Media Misc **[Default]** — Description: [] Media MFX/VEBOX+SFC Mode
	22:21	**SubOpcodeA** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcodeB** Default Value: 3h SFC_IEF_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 16h Excludes DWord (0,1) Format: =n Total Length - 2
1..23	735:0	**SFC IEF State Body** Format: SFC_IEF_STATE_BODY

SFC_LOCK

SFC_LOCK

Source: BSpec

Length Bias: 2

Description
This command is used for VD/VE box to communicate with SFC before the start of any SFC workload. VD/VE uses this command to make sure that it has the ownership of SFC pipe before running workload with SFC since SFC is shared between VD/VE on a frame level. For VD(MFX)-SFC workload, only decoder mode is allowed. Encoder mode cannot use SFC.
For VD(HCP)-SFC workload, only decoder mode is allowed. Encoder mode cannot use SFC

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Format: OpCode Value: Ah — Name: Media Misc **[Default]** — Description: [] Media MFX/VEBOX+SFC Mode For VD(MFX)+SFC mode, only decoder mode is allowed. Encoder mode cannot use SFC
	22:21	**SubOpcodeA** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcodeB** Default Value: 0h SFC Lock Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n

SFC_LOCK		
		Total Length - 2
1	31:0	**SFC Lock Body** Format: SFC_LOCK_BODY

SFC_STATE

SFC_STATE

Source: BSpec

Length Bias: 2

This command is sent from VDBOX/HCP/VEBOX to SFC pipeline at the start of each frame once the lock request is granted.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Default Value: Ah Media MFX/VEBOX+SFC Mode Format: OpCode
	22:21	**SubOpcodeA** Default Value: 0h Common Format: OpCode
	20:16	**SubOpcodeB** Default Value: 1h SFC_State Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Default Value: 2Bh Excludes DWord (0,1) Format: =n Total Length - 2
1..49	1567:0	**SFC State Body** Format: SFC_STATE_BODY

Shift Left

shl - Shift Left

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	true
Source Modifier:	true

Perform component-wise logical left shift of the bits in src0 by the shift count indicated in src1, storing the results in dst, inserting zero bits in the number of LSBs indicated by the shift count. Hardware detects overflow properly and uses it to perform any saturation operation on the result, as long as the shifted result is within 33 bits. Otherwise, the result is undefined. Note: For word and DWord operands, the accumulators have 33 bits.

In QWord mode, the shift count is taken from the low six bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 63. Otherwise the shift count is taken from the low five bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 31. The operation uses QWord mode if src0 or dst has the Q or UQ type but not if src1 is the only operand with the Q or UQ type.

Format:

```
[(pred)] shl[.cmod] (exec_size) dst src0 src1
```

Restriction

Accumulator cannot be destination, implicit or explicit.

Syntax

```
[(pred)] shl[.cmod] (exec_size) reg reg reg
 [(pred)] shl[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         shiftCnt = src0 or dst has Q or UQ type ? src1.chan[n] & 0x3F : src1.chan[n] &
0x1F
         dst.chan[n] = src0.chan[n] « shiftCnt;
     }
 }
```

Src Types	Dst Types
*B,*W,*D	*B,*W,*D

DWord	Bit	Description
0..3	127:126	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ

shl - Shift Left

	127:96	**Src1.ImmValue[31:0]** Exists If: ([Src1.IsImm]==true)
	125:122	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	121:120	**Src1.Mod** Exists If: ([Src1.IsImm]==false) Format: **SrcMod**
	119:116	**Src1.VertStride** Exists If: ([Src1.IsImm]==false) Format: **VertStride**
	115:113	**Src1.Width** Exists If: ([Src1.IsImm]==false) Format: **Width**
	112	**Src1.AddrMode** Exists If: ([Src1.IsImm]==false) Format: **AddrMode**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect) Format: **IndirectOperand**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct) Format: **DirectOperand**
	97:96	**Src1.HorzStride** Exists If: ([Src1.IsImm]==false) Format: **HorzStride**
	95:92	**CondCtrl** Format: **FlagModifier**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==true) Format: **ImmDataType**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==false) Format: **RegDataType**
	87:84	**Src0.VertStride** Format: **VertStride**

shl - Shift Left

	83:81	**Src0.Width** Format: **Width**
	80	**Src0.AddrMode** Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
	45:44	**Src0.Mod** Format: **SrcMod**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: **RegDataType**

shl - Shift Left

	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: **ImmDataType**
	39:36	**Dst.DataType** Format: **RegDataType**
	35	**Dst.AddrMode** Format: **AddrMode**
	34	**Saturate** Format: **Saturate**
	33	**AccWrCtrl** Format: **AccWrCtrl**
	32	**AtomicCtrl** Format: **AtomicCtrl**
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". **Value** / **Name** / **Description** 0 / Normal **[Default]** / Normal. Per channel write enable used for final write enable generation. 1 / NoMask / NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	30	**Reserved**
	29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. **Value** / **Name** / **Description** 0 / NoCompaction **[Default]** / No compaction. 128-bit native instruction supporting all instruction options. 1 / Compacted / Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
	28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no

predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 **ChanOff**

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16 **ExecSize**

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

15:0 **Header**

Format:	Header

Shift Right

shr - Shift Right

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	true
Source Modifier:	true

Perform component-wise logical right shift with zero insertion of the bits in src0 by the shift count indicated in src1, storing the results in dst. Insert zero bits in the number of MSBs indicated by the shift count. When src0 is accumulator and/or source modifier is used with src0 the sign bit is inserted in the MSBs which come from the additional precision. **Note**: For Word and DWord operands, the accumulators have 33 bits.
Note: For unsigned src0 types, shr and asr produce the same result.

In QWord mode, the shift count is taken from the low six bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 63. Otherwise the shift count is taken from the low five bits of src1 regardless of the src1 type and treated as an unsigned integer in the range 0 to 31. The operation uses QWord mode if src0 or dst has the Q or UQ type but not if src1 is the only operand with the Q or UQ type.

Format:

```
[(pred)] shr[.cmod] (exec_size) dst src0 src1
```

Syntax

```
[(pred)] shr[.cmod] (exec_size) reg reg reg
 [(pred)] shr[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         shiftCnt = src0 or dst has Q or UQ type ? src1.chan[n] & 0x3F : src1.chan[n] &
0x1F
         dst.chan[n] = src0.chan[n] » shiftCnt;
     }
 }
```

Src Types	Dst Types
UB, UW, UD	UB, UW, UD

DWord	Bit	Description
0..3	127:126	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	127:96	**Src1.ImmValue[31:0]** Exists If: ([Src1.IsImm]==true)

shr - Shift Right

	125:122	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	121:120	**Src1.Mod** Exists If: ([Src1.IsImm]==false) Format: **SrcMod**
	119:116	**Src1.VertStride** Exists If: ([Src1.IsImm]==false) Format: **VertStride**
	115:113	**Src1.Width** Exists If: ([Src1.IsImm]==false) Format: **Width**
	112	**Src1.AddrMode** Exists If: ([Src1.IsImm]==false) Format: **AddrMode**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect) Format: **IndirectOperand**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct) Format: **DirectOperand**
	97:96	**Src1.HorzStride** Exists If: ([Src1.IsImm]==false) Format: **HorzStride**
	95:92	**CondCtrl** Format: **FlagModifier**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==true) Format: **ImmDataType**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==false) Format: **RegDataType**
	87:84	**Src0.VertStride** Format: **VertStride**
	83:81	**Src0.Width** Format: **Width**

shr - Shift Right

	80	**Src0.AddrMode** Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
	45:44	**Src0.Mod** Format: **SrcMod**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: **RegDataType**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: **ImmDataType**

	39:36	**Dst.DataType** Format: **RegDataType**
	35	**Dst.AddrMode** Format: **AddrMode**
	34	**Saturate** Format: **Saturate**
	33	**AccWrCtrl** Format: **AccWrCtrl**
	32	**AtomicCtrl** Format: **AtomicCtrl**
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0". **Value** / **Name** / **Description** 0 / Normal **[Default]** / Normal. Per channel write enable used for final write enable generation. 1 / NoMask / NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.
	30	**Reserved**
	29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information. **Value** / **Name** / **Description** 0 / NoCompaction **[Default]** / No compaction. 128-bit native instruction supporting all instruction options. 1 / Compacted / Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.
	28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields **Value** / **Name** / **Description** 0 / Positive / Positive polarity of predication. Use the predication mask produced

	[Default]	by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 PredCtrl

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 FlagRegNum[0]

This field specifies bit[0] of the register number for a flag register operand.

22 FlagSubRegNum

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.

21:19 ChanOff

Format:	ChanOff

This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.

18:16 ExecSize

Format:	ExecSize

This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.

15:0 Header

Format:	Header

Signal Event

MSD_SIGNAL_EVENT - Signal Event

Source: EuSubFunctionGateway

Length Bias: 1

Sends an event to all threads that are monitoring that Event ID.

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of GRF registers sent as the message payload. Value 1 — Name: One **[Default]** — Description: See MDP_EVENT Event Data payload definition.
	24:20	**Response Length** Default Value: 0 None Format: U5 Specifies the number of GRF registers expected as the message response payload.
	19:3	**Reserved** Access: RO Format: MBZ
	2:0	**Signal Event Subfunction** Default Value: 0x1 Format: OpCode

SIMD8 Render Target Read MSD

MSD_RTR_SIMD8 - SIMD8 Render Target Read MSD

Source: EuSubFunctionRenderDataPort

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Message Precision Subtype** Default Value: 0h Format: Opcode Full precision data message **Programming Notes** This field must be programmed to 0
	29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	**Reserved** Access: RO Format: MBZ
	17:14	**Message Type** Default Value: 0Dh Format: Opcode Render Target Read message

MSD_RTR_SIMD8 - SIMD8 Render Target Read MSD

	13	**Per-Sample PS Enable** Format: Enable If set, PS reads color phases on per sample basis for each slot. **Programming Notes** This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.
	12	**Reserved** Access: RO Format: MBZ
	11	**Slot Group Select** Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:9	**Reserved** Access: RO Format: MBZ
	8	**Render Target Message Subtype** Default Value: 1h Format: Opcode SIMD8 single source message. Use slots [7:0] for the pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [23:16] are referenced instead of [7:0].
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

SIMD8 Render Target Write MSD

MSD_RTW_SIMD8 - SIMD8 Render Target Write MSD

Source: EuSubFunctionRenderDataPort

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Message Precision Subtype** Default Value: 0h Format: Opcode Full precision data message
	29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	**Per-Coarse Pixel PS outputs enable** Format: Enable This bit indicates the render target write is a coarse pixel write. **Programming Notes** This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.

MSD_RTW_SIMD8 - SIMD8 Render Target Write MSD

	17:14	**Message Type** Default Value: 0Ch Format: Opcode Render Target Write message
	13	**Per-Sample PS Enable** Format: Enable If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot. **Programming Notes** This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.
	12	**Last Render Target Select** Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.
	11	**Slot Group Select** Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
	10:8	**Render Target Message Subtype** Default Value: 4h Format: Opcode SIMD8 single source message. Use slots [7:0] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [23:16] are referenced instead of [7:0].

MSD_RTW_SIMD8 - SIMD8 Render Target Write MSD		
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

SIMD16 Render Target Read MSD

MSD_RTR_SIMD16 - SIMD16 Render Target Read MSD

Source: EuSubFunctionRenderDataPort

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Message Precision Subtype** Default Value: 0h Format: Opcode Full precision data message **Programming Notes** This field must be programmed to 0
	29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	**Reserved** Access: RO Format: MBZ
	17:14	**Message Type** Default Value: 0Dh Format: Opcode Render Target Read message

MSD_RTR_SIMD16 - SIMD16 Render Target Read MSD

13	**Per-Sample PS Enable** Format: Enable If set, PS reads color phases on per sample basis for each slot. **Programming Notes** This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.
12	**Reserved** Access: RO Format: MBZ
11	**Slot Group Select** Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
10:9	**Reserved** Access: RO Format: MBZ
8	**Render Target Message Subtype** Default Value: 0h Format: Opcode SIMD16 single source message. Use slots [15:0] for the pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is selected, slots [31:16] are referenced instead of [15:0].
7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

SIMD16 Render Target Write MSD

MSD_RTW_SIMD16 - SIMD16 Render Target Write MSD

Source: EuSubFunctionRenderDataPort

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Message Precision Subtype** Default Value: 0h Format: Opcode Full precision data message
	29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the 2-register header.
	18	**Per-Coarse Pixel PS outputs enable** Format: Enable This bit indicates the render target write is a coarse pixel write. **Programming Notes** This bit must be DISABLED if a pixel shader thread is dispatched at pixel- or sample- rate. This bit may be DISABLED if a multi-rate pixel shader thread is dispatched at coarse rate. In such case, it indicates per-pixel or per-sample write (as determined by Per-Sample PS outputs enable) from a multi-rate shader, where the set of pixels is indicated by the Pixel shading phase field in Extended Message descriptor.When this bit is set and the message has oMask present,

MSD_RTW_SIMD16 - SIMD16 Render Target Write MSD

	oMask represents the pixel enables within the Coarse Pixel in the row major order.
17:14	**Message Type** Default Value: 0Ch Format: Opcode Render Target Write message
13	**Per-Sample PS Enable** Format: Enable If set, PS sends Render Target Write Message that outputs color, depth(optional) and stencil(optional) phases on per sample basis for each slot. **Programming Notes** This bit must be set when PS runs at sample-frequency i.e. pixel shader dispatch mode is PER_SAMPLE. When this bit is set and PS runs at pixel-frequency, i.e. pixel shader dispatch mode is PER_PIXEL, Render Target read and write messages interpret bits 9:6 in MCH_RT_C0 as sample index. In this mode, render target write message payload and render target read writeback payload contain color of a specific sample in all dispatched pixels. RT writes referring to out-of-bound samples have no effect. RT reads from out-of-bound samples return 0. When this bit is clear and PS runs at pixel-frequency, render target write messages contain color value for entire pixel (all samples). When this bit is clear and PS runs at pixel-frequency, render target reads are disallowed per API spec (RT read without specifying sample index forces sample-frequency dispatch). HW behavior is undefined in such case.
12	**Last Render Target Select** Format: Enable This bit must be set on the last render target write message sent for each group of pixels. For single render target pixel shaders, this bit is set on all render target write messages. For multiple render target pixel shaders, this bit is set only on messages sent to the last render target. This bit must be zero for SIMD8 Image Write message. In general, when threads are not launched by 3D FF, this bit must be zero.
11	**Slot Group Select** Format: MDC_RT_SGS This field selects whether slots 15:0 or slots 31:16 are used for bypassed data.
10:8	**Render Target Message Subtype** Default Value: 0h Format: Opcode SIMD16 Single source message. Use slots [15:0] for pixel enables, X/Y addresses, and oMask. **Programming Notes** The above slots indicated are within the 16 slots selected by Slot Group Select. If SLOTGRP_HI is

MSD_RTW_SIMD16 - SIMD16 Render Target Write MSD		
		selected, slots [31:16] are referenced instead of [15:0].
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

STATE_BASE_ADDRESS

STATE_BASE_ADDRESS

Source: BSpec

Length Bias: 2

The STATE_BASE_ADDRESS command sets the base pointers for subsequent state, instruction, and media indirect object accesses by the GPE.
For more information see the Base Address Utilization table in the Memory Access Indirection narrative topic.

Programming Notes

The following commands must be reissued following any change to the base addresses:

- 3DSTATE_CC_POINTERS
- 3DSTATE_BINDING_TABLE_POINTERS
- 3DSTATE_SAMPLER_STATE_POINTERS
- 3DSTATE_VIEWPORT_STATE_POINTERS

Execution of this command causes a full pipeline flush, thus its use should be minimized for higher performance.

If 3DSTATE_PS_EXTRA::Pixel Shader Is Per Coarse Pixel == 1, the 3DSTATE_CPS_POINTERS command must be reissued following any change to the dynamic state base address.

SW must always program PIPE_CONTROL with "CS Stall" and "Render Target Cache Flush Enable" set before programming STATE_BASE_ADDRESS command for GPGPU workloads i.e when pipeline select is GPGPU via PIPELINE_SELECT command. This is required to achieve better GPGPU preemption latencies in certain workload programming sequences. If programming PIPE_CONTROL has performance implications then preemption latencies can be traded off against performance by not implementing this programming note.

SW must always program PIPE_CONTROL with Command Cache Invalidate Enable following programming of STATE_BASE_ADDRESS command when State Cache redirect to CS Section enable bit is set in MMIO register SLICE_COMMON_ECO_CHCICKEN1 (0731Ch).

SW must always program PIPE_CONTROL command with HDC Pipleine FLush set prior to programming of STATE_BASE_ADDRESS command for GPGPU/Media workloads i.e when pipeline select is GPGPU or Media via PIPELINE_SELECT command. This is required to ensure the write data out of the prior thread group are flushed out prior to the state changes due to the programming of STATE_BASE_ADDRESS command take place.

DWord	Bit	Description	
0	31:29	**Command Type**	
		Default Value:	3h GFXPIPE
		Format:	OpCode
	28:27	**Command SubType**	
		Default Value:	0h GFXPIPE_COMMON
		Format:	OpCode

STATE_BASE_ADDRESS

26:24 3D Command Opcode

Default Value:	1h GFXPIPE_NONPIPELINED
Format:	OpCode

23:16 3D Command Sub Opcode

Default Value:	01h STATE_BASE_ADDRESS
Format:	OpCode

15:8 Reserved

Access:	RO
Format:	MBZ

7:0 DWord Length

Format:	=n

Value	Name	Description
14h	DWORD_COUNT_n **[Default]**	Excludes DWord (0,1)

1..2 **63:12 General State Base Address**

Format:	GraphicsAddress[63:12]

Description

Specifies the 4K-byte aligned base address for general state accesses. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].

Programming Notes

Bounds checking is performed on general state accesses by Data Port Shared Functions for stateless A32 messages.
Bounds checking is enabled when General State Base Address [46:12] + General State Buffer Size [31:12] is <= 2^47. This ensures that the General State Buffer does not straddle the canonical address boundary where GraphicsAddress [47] changes.

Restriction

General State Base Address [47:12] + General State Buffer Size [31:12] must be < 2^48. It is illegal programming for this to be >= 2^48.

When using stateless (A32) Data Port messages, General State Base Address [47:12] + Buffer Base Address [31:0] must be < 2^48. It is illegal for this to be >= 2^48.

11 Reserved

Access:	RO
Format:	MBZ

10:4 General State Memory Object Control State

Format:	MEMORY_OBJECT_CONTROL_STATE

Specifies the memory object control state for indirect state using the **General State Base**

Address, with the exception of the stateless data port accesses.

	3:1	**Reserved** Access: RO Format: MBZ
	0	**General State Base Address Modify Enable** Format: Enable The other fields in this DWord and the following DWord are updated only when this bit is set. 0h – Disable – Ignore the updated address. 1h – Enable – Modify the address.
3	31:26	**Reserved** Access: RO Format: MBZ
	25:23	**Reserved** Access: RO Format: MBZ
	22:16	**Stateless Data Port Access Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for stateless data port accesses.
	15:13	**Reserved** Access: RO Format: MBZ
	12:1	**Reserved** Access: RO Format: MBZ
	0	**Reserved** Access: RO Format: MBZ
4..5	63:12	**Surface State Base Address** Format: GraphicsAddress[63:12] Specifies the 4K-byte aligned base address for binding table and surface state accesses.GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].

STATE_BASE_ADDRESS

	11	**Reserved** Access: RO Format: MBZ
	10:4	**Surface State Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for indirect state using the **Surface State Base Address**.
	3:1	**Reserved** Access: RO Format: MBZ
	0	**Surface State Base Address Modify Enable** Format: Enable The other fields in this DWord and the following DWord are updated only when this bit is set.

Value	Name	Description
0h	Disable	Ignore the updated address.
1h	Enable	Modify the address.

Programming Notes
Setting this bit to 1 in a batch buffer causes the resource streamer to stop; for performance reasons the SW should only place commands with this bit set in the ring buffer.
Before programming the Surface State Base Address, the RS must be disabled. Within a batch buffer where the RS is enabled, RS may be disabled thru a MI_RS_CONTROL command with Resource Streamer Control cleared prior to the STATE_BASE_ADDRESS with Surface State Base Address Modify Enable set and then re-enabled with another MI_RS_CONTROL with Resource Streamer Control set.

6..7	63:12	**Dynamic State Base Address** Format: GraphicsAddress[63:12]

Description
Specifies the 4K-byte aligned base address for sampler and viewport state accesses. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].

	11	**Reserved** Access: RO Format: MBZ

	10:4	**Dynamic State Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for indirect state using the **Dynamic State Base Address**. Push constants defined in 3DSTATE_CONSTANT_(VS | GS | PS) commands do not use this control state, although they can use the corresponding base address. The memory object control state for push constants is defined within the command.
	3:1	**Reserved** Access: RO Format: MBZ
	0	**Dynamic State Base Address Modify Enable** Format: Enable The other fields in this DWord and the following DWord are updated only when this bit is set.

Value	Name	Description
0h	Disable	Ignore the updated address.
1h	Enable	Modify the address.

8..9	63:12	**Indirect Object Base Address** Format: GraphicsAddress[63:12] Specifies the 4K-byte aligned base address for indirect object load in MEDIA_OBJECT command.
	11	**Reserved** Access: RO Format: MBZ
	10:4	**Indirect Object Memory Object Control State** Format: MEMORY_OBJECT_CONTROL_STATE Specifies the memory object control state for indirect objects using the **Indirect Object Base Address**.
	3:1	**Reserved** Access: RO Format: MBZ
	0	**Indirect Object Base Address Modify Enable** Format: Enable The other fields in this DWord and the following DWord are updated only when this bit is set.

Value	Name	Description
0h	Disable	Ignore the updated address.
1h	Enable	Modify the address.

STATE_BASE_ADDRESS

10..11 | 63:12 — Instruction Base Address

Format:	GraphicsAddress[63:12]

Specifies the 4K-byte aligned base address for all EU instruction accesses.
GraphicsAddress[63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].

11 — Reserved

Access:	RO
Format:	MBZ

10:4 — Instruction Memory Object Control State

Format:	MEMORY_OBJECT_CONTROL_STATE

Specifies the memory object control state for EU instructions using the **Instruction Base Address**.

3:1 — Reserved

Access:	RO
Format:	MBZ

0 — Instruction Base Address Modify Enable

Format:	Enable

The other fields in this DWord and thefollowing DWord are updated only when this bit is set.

Value	Name	Description
0h	Disable	Ignore the updated address.
1h	Enable	Modify the address.

12 | 31:12 — General State Buffer Size

Format:	U20

FormatDesc
This field specifies the size of the buffer in 4K pages. Any access that straddles or goes past the end of the buffer returns 0. Note that BufferSize=0 indicates that there is no valid data in the buffer.

11:1 — Reserved

Access:	RO
Format:	MBZ

0 — General State Buffer Size Modify Enable

Format:	Enable

The fields in this DWord are updated only when this bit is set.

Value	Name	Description
0h	Disable	Ignore the updated bound.
1h	Enable	Modify the updated bound.

STATE_BASE_ADDRESS

13	31:12	**Dynamic State Buffer Size** Format: U20 FormatDesc This field specifies the size of the buffer in 4K pages. Any access that straddles or goes past the end of the buffer returns 0. Note that BufferSize=0 indicates that there is no valid data in the buffer.
	11:1	**Reserved** Access: RO Format: MBZ
	0	**Dynamic State Buffer Size Modify Enable** Format: Enable FormatDesc The fields in this DWord are updated only when this bit is set.

Value	Name	Description
0h	Disable	Ignore the updated bound.
1h	Enable	Modify the updated bound.

14	31:12	**Indirect Object Buffer Size** Format: U20 FormatDesc This field specifies the size of the buffer in 4K pages. Any access that straddles or goes past the end of the buffer returns 0. Note that BufferSize=0 indicates that there is no valid data in the buffer.
	11:1	**Reserved** Access: RO Format: MBZ
	0	**Indirect Object Buffer Size Modify Enable** Format: Enable FormatDesc The fields in this DWord are updated only when this bit is set.

Value	Name	Description
0h	Disable	Ignore the updated bound.
1h	Enable	Modify the updated bound.

STATE_BASE_ADDRESS

15 — 31:12 — Instruction Buffer Size

Format:	U20

FormatDesc
This field specifies the size of the buffer in 4K pages. Any access that straddles or goes past the end of the buffer returns 0. Note that BufferSize=0 indicates that there is no valid data in the buffer.

11:1 — Reserved

Access:	RO
Format:	MBZ

0 — Instruction Buffer size Modify Enable

Format:	Enable

FormatDesc
The fields in this DWord are updated only when this bit is set.

Value	Name	Description
0h	Disable	Ignore the updated bound.

16..17 — 63:12 — Bindless Surface State Base Address

Format:	GraphicsAddress[63:12]

Specifies the 4K-byte aligned base address for bindless surface state accesses.GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].

11 — Reserved

Access:	RO
Format:	MBZ

10:4 — Bindless Surface State Memory Object Control State

Format:	MEMORY_OBJECT_CONTROL_STATE

Specifies the memory object control state for indirect state using the **Bindless Surface State Base Address**.

3:1 — Reserved

Access:	RO
Format:	MBZ

0 — Bindless Surface State Base Address Modify Enable

Format:	Enable

Description
The other fields in this DWord and the following two DWords are updated only when this bit is

set.

Value	Name	Description
0h	Disable	Ignore the updated address
1h	Enable	Modify the address

18 | 31:12 | **Bindless Surface State Size**

Format:	U20

This field indicates the size-1 of the Bindless Surface State buffer in 64-Byte increments.Any SSO beyond this maximum size points to offset 0.
Example: If the buffer contains 512 surface states, then this field must be programmed to 0x1FF (511 decimal).

11:0 | **Reserved**

Access:	RO
Format:	MBZ

19..20 | 63:12 | **Bindless Sampler State Base Address**

Format:	GraphicsAddress[63:12]

Specifies the 4K-byte aligned base address for bindless sampler state accesses. GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47].

11 | **Reserved**

Access:	RO
Format:	MBZ

10:4 | **Bindless Sampler State Memory Object Control State**

Format:	MEMORY_OBJECT_CONTROL_STATE

Specifies the memory object control state for indirect state using the **Bindless Sampler State Base Address**.

3:1 | **Reserved**

Access:	RO
Format:	MBZ

0 | **Bindless Sampler State Base Address Modify Enable**

Format:	Enable

The other fields in this DWord and the following two DWords are updated only when this bit is set.

Value	Name	Description
0h	Disable	Ignore the updated address
1h	Enable	Modify the address

21	31:12	**Bindless Sampler State Buffer Size** Format: U20 This field specifies the size of the buffer in 4K pages. Any access that goes beyond the end of the buffer (as defined by the Sampler State Buffer Size) will use an offset of 0.
	11:0	**Reserved** Access: RO Format: MBZ

STATE_COMPUTE_MODE

STATE_COMPUTE_MODE

Source: RenderCS, ComputeCS

Length Bias: 2

This is a non-pipeline state command and is a general compute programming state that can be shared from the top to bottom of the pipeline.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 0h GFXPIPE_COMMON Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h GFXPIPE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 05h STATE_COMPUTE_MODE Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 0h Excludes DWord (0,1) Format: =n
1	31:16	**Mask** Format: Enable[16] This field is the mask bits for the state bits below. This is a bit wise mask where the bit number-16 is the value of the corresponding bit being masked in the same data word. For example, if you want to update state for bits 3:2 then bits 19:18 must be set.
	15	**Reserved** Access: RO Format: MBZ
	14	**Reserved** Format: MBZ

STATE_COMPUTE_MODE

13 **Disable L1 Invalidate for non-L1-cacheable Writes**

Format:	Disable

When this bit is set, HDC global memory write requests that are marked "non-L1-cacheable" (either due to MOCS setting or L1-cache-disable mode bits set in this register) will not send "Invalidation" request to the SamplerL1 cache. The implication of this bit being set is that HDC will not maintain RAW and WAR ordering between L1-cacheable and non-L1-cacheable requests to the same address. URB writes never sends Invalidate commands to Sampler L1 cache (regardless of this bit).

Value	Name	Description
0b	Enable **[Default]**	HDC Non-L1-cacheable writes to Global memory will send Invalidate command to Sampler L1 cache.
1b	Disable	HDCNon-L1-cacheable writes to Global memory will NOT send Invalidate command to Sampler L1 cache.

12:11 **Reserved**

Access:	RO
Format:	MBZ

10 **Binding Table Alignment**

When this bit is set, the format of the compute context's binding table is SW binding table format whether Binding Table Pool is enabled or disabled.

Value	Name	Description
0	Legacy **[Default]**	Binding table pointer 15:5 Maps to 15:5 for INTERFACE_DESCRIPTOR DATA. Binding table pointer 15:5 Maps to 15:5 for 3DSTATE_BINDING_TABLE_POINTER_* if Binding Table Pool is disabled. Binding table pointer 15:5 Maps to 16:6 for 3DSTATE_BINDING_TABLE_POINTER_* if Binding Table Pool is enabled.

Programming Notes
Compute dispatches from the RenderCS context use GT_MODE[10] and not this bit. This bit is only used by ComputeCS contexts.

9:5 **Reserved**

Access:	RO
Format:	MBZ

4:3 **Force Non-Coherent**

Format:	U2

Force all Data Cache Data Port access to be Non-Coherent (virtual addresses) and non-faultable regardless of the surface state or binding table index.

Value	Name	Description
2h	Force GPU Non-Coherent	GPU accesses are forced as non-coherent with other GPU, as well as with the CPU.

STATE_COMPUTE_MODE

		Programming Notes Only change this mode after a pipeflush and cache flush (all threads and their all accesses completed). CPU-GPU or GPU-GPU coherency is not supported. Hence the driver must set this field to 0x2 (Force GPU non-coherent), if the data-port message has coherency enabled via BTI or surface state.
	2:0	**Reserved** Access: RO Format: MBZ

STATE_SIP

STATE_SIP

Source: BSpec

Length Bias: 2

The STATE_SIP command specifies the starting instruction location of the System Routine that is shared by all threads in execution.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h GFXPIPE Format: OpCode
	28:27	**Command SubType** Default Value: 0h GFXPIPE_COMMON Format: OpCode
	26:24	**3D Command Opcode** Default Value: 1h GFXPIPE_NONPIPELINED Format: OpCode
	23:16	**3D Command Sub Opcode** Default Value: 02h STATE_SIP Format: OpCode
	15:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Format: =n Value / Name / Description 1h / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)
1..2	63:4	**System Instruction Pointer** Format: InstructionBaseOffset[63:4] Specifies the instruction address of the system routine associated with the current context as a 128-bit granular offset from the Instruction Base Address. SIP is shared by all threads in execution. The address specifies the double quadword aligned instruction location.GraphicsAddress [63:48] are ignored by the HW and assumed to be in correct canonical form [63:48] == [47]. **Programming Notes** This portion of the command is not context save/restored. The context image may restore this command as a 2 dword command rather than a 3 dword command.

STATE_SIP		
	3:0	**Reserved** Access: RO Format: MBZ

Subtraction with Borrow

subb - Subtraction with Borrow

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	true
Saturation:	true
Source Modifier:	false

The subb instruction performs component-wise subtraction of src0 and src1 and stores the results in dst, it also stores the borrow into acc. If the operation produces a borrow (src0 < src1), write 0x00000001 to acc, else write 0x00000000 to acc.

Format:

```
[(pred)] subb[.cmod] (exec_size) dst src0 src1
```

Programming Notes

The accumulator is an implicit destination and thus cannot be an explicit destination operand.

Restriction

AccWrEn is required.

Syntax

```
[(pred)] subb[.cmod] (exec_size) reg reg reg
 [(pred)] subb[.cmod] (exec_size) reg reg imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < exec_size; n++ ) {
     if ( WrEn.chan[n] ) {
         dst.chan[n] = src0.chan[n] - src1.chan[n];
         acc.chan[n] = borrow(src.chan[n] - src1.chan[n]);
     }
 }
```

Src Types	Dst Types
UD	UD

DWord	Bit	Description
0..3	127:126	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	127:96	**Src1.ImmValue[31:0]** Exists If: ([Src1.IsImm]==true)

subb - Subtraction with Borrow

	125:122	**Reserved** Exists If: ([Src1.IsImm]==false) Format: MBZ
	121:120	**Src1.Mod** Exists If: ([Src1.IsImm]==false) Format: **SrcMod**
	119:116	**Src1.VertStride** Exists If: ([Src1.IsImm]==false) Format: **VertStride**
	115:113	**Src1.Width** Exists If: ([Src1.IsImm]==false) Format: **Width**
	112	**Src1.AddrMode** Exists If: ([Src1.IsImm]==false) Format: **AddrMode**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Indirect) Format: **IndirectOperand**
	111:98	**Src1.Operand** Exists If: ([Src1.IsImm]==false) AND ([Src1.AddrMode]==Direct) Format: **DirectOperand**
	97:96	**Src1.HorzStride** Exists If: ([Src1.IsImm]==false) Format: **HorzStride**
	95:92	**CondCtrl** Format: **FlagModifier**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==true) Format: **ImmDataType**
	91:88	**Src1.DataType** Exists If: ([Src1.IsImm]==false) Format: **RegDataType**
	87:84	**Src0.VertStride** Format: **VertStride**
	83:81	**Src0.Width** Format: **Width**

subb - Subtraction with Borrow

	80	**Src0.AddrMode** Format: **AddrMode**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Direct) Format: **DirectOperand**
	79:66	**Src0.Operand** Exists If: ([Src0.AddrMode]==Indirect) Format: **IndirectOperand**
	65:64	**Src0.HorzStride** Format: **HorzStride**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Direct) Format: **DirectOperand**
	63:50	**Dst.Operand** Exists If: ([Dst.AddrMode]==Indirect) Format: **IndirectOperand**
	49:48	**Dst.HorzStride** Format: **HorzStride**
	47	**Src1.IsImm** This field indicate that Source 1 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — false **[Default]** Value: 1 — true
	45:44	**Src0.Mod** Format: **SrcMod**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==false) Format: **RegDataType**
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: **ImmDataType**

subb - Subtraction with Borrow

	39:36	**Dst.DataType** Format: RegDataType
	35	**Dst.AddrMode** Format: AddrMode
	34	**Saturate** Format: Saturate
	33	**AccWrCtrl** Format: AccWrCtrl
	32	**AtomicCtrl** Format: AtomicCtrl
	31	**MaskCtrl** Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

	30	**Reserved**
	29	**CmptCtrl** Format: MBZ Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

	28	**PredInv** This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive	Positive polarity of predication. Use the predication mask produced

subb - Subtraction with Borrow

	[Default]	by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24	**PredCtrl** Format: **PredCtrl** This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.
23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
15:0	**Header** Format: **Header**

Synchronize

sync - Synchronize

Source:	EuIsa
Length Bias:	4
Predication:	false
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false
Subfunctions:	SyncFC[95:92]

Wait on Dependency performs various operations related to synchronization such as waiting on registers (barriers registers) or for software scoreboarding (SWSB), which is used to specify pipeline hazards to the EU. The instruction has several sub-operations (function controls), including:

- **nop** (0000b): no operation (encoded SWSB information available to every instruction is still honored). This might be used if an instruction depends on two different out-of-order sources. The consumer can only specify a dependency on one, hence an extra instruction must be added for this.
- Reserved (0001b): reserved for future expansion.
- **allrd** (0010b): blocks until all out-of-order sources are read (e.g. input arguments to a send or math op).
- **allwr** (0011b): blocks until all out-of-order destinations are written back (e.g. writes from a send or math op).
- Reserved (0100-1100b): reserved for future expansion
- **fence** (1101b): blocks on the notification register for fence response. When fence response is received from message gateway, bit 0 of n0.2 notification register is set. Instruction sync.fence blocks until the bit is set and clears before progressing to the next instruction.
- **bar** (1110b): blocks on the notification register for barriers response. When barrier response is received from message gatewaybits corresponding to the barrier id are set in the notification register n0. Instruction sync.bar(barrier id) blocks until the bit corresponding to the barrier id is set, and clears it before progressing to the next instruction.
- **host** (1111b): blocks on the notification register for host interaction. When host notification is received, the bit 0 of n1 notification register is set. Instruction sync.host blocks until the bit is set and clears it before progressing to the next instruction.

See the SyncFC BXML enum for more information.

Format: sync.[sync_fc] src0

Programming Notes

The format is that of a basic instruction. The immediate operand is encoded as src0 and may explicitly be null if not used. Src1 and dst must be null.

Syntax

```
sync.nop null [instopts]
sync.allrd (null | imm32) [instopts]
```

sync - Synchronize

sync.allwr (null | imm32) [instopts]
sync.bar null[instopts]
sync.host null [instopts]

Pseudocode

```
     Evaluate(WrEn);
          switch (func) {
          case nop:
             // regular SWSB dep check from instruction options executes
             break;
          case allrd:
             for (sbid = 0; sbid < MAX_SBIDS; sbid++) {
                if (Src0.IsImm) {
                  // wait until selected OOO reads are finished
                  if(Src0.ImmValue[sbid]) wait_on_sbid_read_access(sbid); // transition to
wait_dst or idle
                } else {
                   // wait until all OOO reads are finished
                   wait_on_sbid_read_access(sbid); // transition to wait_dst or idle
                }
             }
             break;
          case allwr:
             for (sbid = 0; sbid < MAX_SBIDS; sbid++) {
               if (Src0.IsImm) {
                  // wait until selected OOO writes are finished
                   if(Src0.ImmValue[sbid]) wait_on_sbid_write_access(sbid); // transition
to idle
               } else {
                 // wait until selected OOO writes are finished
                   wait_on_sbid_write_access(sbid); // transition to idle
               }
             }
             break;
          case bar:
             wait_on_barrier_notification(1 « Src0);
             if (Src0.IsImm) {
                wait_on_barrier_notification(1 « Src0.ImmValue[4:0]) // waits until the
corresponding barrier bit is set
             else if (Src0.RegFile == ARF) {
                wait_on_barrier_notification(1 « Src0[4:0]) // waits until the
corresponding barrier bit is set
             else {
                wait_on_barrier_notification(1) // waits until the barrier bit 0 is set
             }
             break;
          case host:
             wait_on_host_notification(); // waits until the host signals the host barrier
             break;
          }
```

Src Types
*B,*W,*D,*Q, HF, F, DF

DWord	Bit	Description
0..3	127:96	**Reserved**

		Exists If: ([Src0.IsImm]==false) Format: MBZ
	127:96	**Src0.ImmValue[31:0]** Exists If: ([Src0.IsImm]==true)
	95:92	**SyncCtrl** Format: SyncFC
	91:88	**Reserved** Format: MBZ
	87	**Reserved** Format: MBZ
	86:80	**Reserved** Format: MBZ
	79:66	**Reserved** Format: MBZ
	65:50	**Reserved** Format: MBZ
	49:48	**Dst.HorzStride** Value: 01b — Name: 1 elements **[Default]** Value: Others — Name: Reserved
	47	**Reserved** Format: MBZ
	46	**Src0.IsImm** This field indicate that Source 0 operand is carrying an immediate value. Value: 0 — Name: false Value: 1 — Name: true
	45:44	**Reserved** Format: MBZ
	43:40	**Src0.DataType** Exists If: ([Src0.IsImm]==true) Format: ImmDataType
	43:40	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ

39:33 **Reserved**

Format:	MBZ

32 **AtomicCtrl**

Format:	AtomicCtrl

31 **MaskCtrl**

Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".

Value	Name	Description
0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 **Reserved**

29 **CmptCtrl**

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the

		content of the selected flag register.
	23	**FlagRegNum[0]** This field specifies bit[0] of the register number for a flag register operand.
	22	**FlagSubRegNum** This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Typed Surface Read MSD

MSD1R_TS - Typed Surface Read MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.

MSD1R_TS - Typed Surface Read MSD

	18:14	**Message Type** Default Value: 05h Format: Opcode Typed Surface Read message
	13:12	**Slot Group** Format: MDC_SG3 Specifies the Slot Group mode of the message (which slots are processed)
	11:8	**Channel Mask** Format: MDC_CMASK Specifies which RGBA channels are included in the message payload.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Typed Surface Write MSD

MSD1W_TS - Typed Surface Write MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.

MSD1W_TS - Typed Surface Write MSD

	18:14	**Message Type** Default Value: 0Dh Format: Opcode Typed Surface Write message
	13:12	**Slot Group** Format: MDC_SG3 Specifies the Slot Group mode of the message (which slots are processed)
	11:8	**Channel Mask** Format: MDC_CMASK Specifies which RGBA channels are included in the message payload.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Untyped Surface Read MSD

MSD1R_US - Untyped Surface Read MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.

MSD1R_US - Untyped Surface Read MSD

	18:14	**Message Type** Default Value: 01h Format: Opcode Untyped Surface Read message
	13:12	**SIMD Mode** Format: MDC_SM3 Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Channel Mask** Format: MDC_CMASK Specifies which RGBA channels are included in the message payload.
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Untyped Surface Write MSD

MSD1W_US - Untyped Surface Write MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.

MSD1W_US - Untyped Surface Write MSD

	18:14	**Message Type** Default Value: 09h Format: Opcode Untyped Surface Write message
	13:12	**SIMD Mode** Format: MDC_SM3 Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Channel Mask** Format: MDC_UW_CMASK Specifies which RGBA channels are included in the message payload.
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

URB Dword Read MSD

MSDUR_DWS - URB Dword Read MSD

Source: EuSubFunctionReadOnlyDataPort

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload.Valid value ranges are 0 to 16.
	19	**Header Present** Default Value: 1 Present Format: Opcode Indicates that the message requires a header.
	18	**Reserved** Access: RO Format: MBZ
	17	**Per Slot Offset Present** Format: Enable Specifies if per-slot offset message payload is present.
	16	**Reserved** Access: RO Format: MBZ
	15	**Channel Mask Present** Default Value: 0 Not Present Format: Opcode Must be clear on read messages, indicating the Channel Mask Message phase is not present.

MSDUR_DWS - URB Dword Read MSD

14:4 Global Offset

Format:	U11

Specifies the offset, in units of Oword elements, from the start of the URB handle for the access. If **Per Slot Offset Present** is set, this global offset is added to each of the slot offsets to form the overall offset.

Value	Name
[0-2047]	

3:0 URB Opcode

Format:	Opcode

Value	Name	Description
8	URB_SIMD8_READ **[Default]**	SIMD8 URB Dword Read message. Reads 1..8 Dwords, based on RLEN.

URB Dword Write MSD

MSDUW_DWS - URB Dword Write MSD

Source: EuSubFunctionReadOnlyDataPort

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Default Value: 1 Present Format: Opcode Indicates that the message requires a header.
	18	**Reserved** Access: RO Format: MBZ
	17	**Per Slot Offset Present** Format: Enable Specifies if per-slot offset message payload is present. If present, it will be added to the **Global Offset**.
	16	**Reserved** Access: RO Format: MBZ
	15	**Channel Mask Present** Default Value: 0 Not Present Format: Opcode Indicates the channel Mask Message phase is not present.

MSDUW_DWS - URB Dword Write MSD

14:4 Global Offset

Format:	U11

Specifies the offset, in units of Oword elements, from the start of the URB handle for the access. If **Per Slot Offset** is set, the global offset is added to those offsets to form the overall offset.

Value	Name
[0-2047]	

3:0 URB Opcode

Format:	Opcode

Value	Name	Description
7	URB_SIMD8_WRITE **[Default]**	SIMD8 URB Dword Write message. Writes1..8 Dwords, based on RLEN and Channel Mask.

URB Masked Dword Write MSD

MSDUW_MDWS - URB Masked Dword Write MSD

Source: EuSubFunctionReadOnlyDataPort

Length Bias: 1

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Default Value: 1 Present Format: Opcode Indicates that the message requires a header.
	18	**Reserved** Access: RO Format: MBZ
	17	**Per Slot Offset Present** Format: Enable Specifies if per-slot offset message payload is present. If present, it will be added to the **Global Offset**.
	16	**Reserved** Access: RO Format: MBZ
	15	**Channel Mask Present** Default Value: 1 Present Format: Opcode Indicates the Channel Mask Message phase is present and will be used to mask which data elements written.

MSDUW_MDWS - URB Masked Dword Write MSD

	14:4	**Global Offset** Format: U11 Specifies the offset, in units of Oword elements, from the start of the URB handle for the access. If **Per Slot Offset** is set, the global offset is added to those offsets to form the overall offset. **Value** / **Name** [0-2047]
	3:0	**URB Opcode** Format: Opcode **Value** / **Name** / **Description** 7 / URB_SIMD8_WRITE **[Default]** / SIMD8 URB Dword Write message. Writes1..8 Dwords, based on RLEN and Channel Mask.

VD_CONTROL_STATE

VD_CONTROL_STATE

Source: VideoCS

Length Bias: 2

This command can be used in HCPpipe.
For HCP, it is selected with the **Media Instruction Opcode "7h".**
Each command has assigned a media instruction command as defined in DWord 0, BitField 22:16. It will be different between HCP.

This command is used to modify the control of HCP pipe. It can be inserted anywhere within a frame. It can be inserted multiple times within a frame as well.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 7h Codec/Engine Name for HCP Format: OpCode Codec/Engine Name = HCP = 7h
	22:16	**Media Instruction Command** Default Value: Ah VD_CONTROL_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 2h — Name:
1..2	63:0	**VD Control State Body** Format: VD_CONTROL_STATE_BODY

VD_PIPELINE_FLUSH

VD_PIPELINE_FLUSH

Source: VideoCS

Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:23	**Media Command Opcode** Default Value: Fh Extended command Format: OpCode
	22:21	**SubOpcodeA** Default Value: 0h Format: OpCode
	20:16	**SubOpcodeB** Default Value: 0h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWORD_COUNT_n** Default Value: 0h Excludes DWord (0) Format: =n Total Length - 2
1	31:23	**Reserved** Access: RO Format: MBZ
	22	**Reserved** Access: RO Format: MBZ
	21	**AVP pipeline command flush** Format: U1

VD_PIPELINE_FLUSH		
	20	**HuC Pipeline command flush** Format: U1
	19	**MFX pipeline command flush** Format: U1
	18	**Reserved**
	17	**VD-ENC pipeline command flush** Format: U1
	16	**HEVC pipeline command flush** Format: U1
	15:8	**Reserved** Access: RO Format: MBZ
	7	**Reserved** Access: RO Format: MBZ
	6	**AVP pipeline Done** Format: U1
	5	**HuC pipeline Done** Format: U1
	4	**VD command/message parser Done** Format: U1
	3	**MFX pipeline Done** Format: U1
	2	**Reserved**
	1	**VD-ENC pipeline Done** Format: U1
	0	**HEVC pipeline Done** Format: U1

VDENC_CONTROL_STATE

VDENC_CONTROL_STATE

Source: BSpec
Length Bias: 2

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline Type** Default Value: 2h Format: OpCode
	26:23	**Media Instruction Opcode** Default Value: 1h Codec/Engine Name for VDENC Format: OpCode Codec/Engine Name = VDNEC = 1h;
	22:16	**Media Instruction Command** Default Value: Bh VD_CONTROL_STATE for VDNEC Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**Dword Length** Format: =n (Excludes Dwords 0, 1). Value: 0h — Name:
1	31:2	**Reserved** Format: MBZ
	1	**VDenc Initialization** This bit, when set, clears internal states for VDenc Pipe. This bit should be set once at the beginning of a frame.
	0	**Reserved** Format: MBZ

VDENC_DS_REF_SURFACE_STATE

VDENC_DS_REF_SURFACE_STATE

Source: VideoCS

Length Bias: 2

This command specifies the surface state parameters for the downscaled reference surfaces.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_COMMON Format: OpCode
	26:23	**Opcode** Default Value: 1h VDENC_PIPE Format: OpCode
	22:21	**SubOpA** Default Value: 0h Format: OpCode
	20:16	**SubOpB** Default Value: 3h VDENC_DS_REF_SURFACE_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 Value: 4h — Name: DWORD_COUNT_n **[Default]** — Description: Excludes DWord (0,1)
1	31:0	**Reserved** Format: MBZ
2..5	127:0	**Dwords 2..5** Format: VDENC_Surface_State_Fields Four DWords that define the bit fields used by the three surface state commands.

VDENC_PIPE_BUF_ADDR_STATE

VDENC_PIPE_BUF_ADDR_STATE

Source: VideoCS

Length Bias: 2

This state command provides the memory base addresses for all row stores, Streamin/StreamOut, DMV buffer along with the uncompressed source, reference pictures and downscaled reference pictures required by the VDENC pipeline. All reference pixel surfaces in the Encoder are programmed with the same surface state(NV12 and TileY format), except each has its own frame buffer base address. Same holds true for the down-scaled reference pictures too. In the tile format, there is no need to provide buffer offset for each slice; since from each MB address, the hardware can calculated the corresponding memory location within the frame buffer directly. VDEnc supports 3 Downscaled reference frames (2 fwd, 1 bwd) and 4 normal reference frames (3 fwd, 1 bwd).The driver will sort out the base address from the DPB table and populate the base addresses that map to the corresponding reference index for both DS references and normal reference frames.
Each of the individual DS ref/ Normal ref frames have their own MOCS DW that corresponds to the respective base address. The only thing that is different in the MOCS DW amongst the DS reference frames is the MMCD controls (specified in bits [10:9] of the MOCS DW). Driver needs to ensure that the other bits need to be the same across the different DS ref frames. The same is applicable for the normal reference frames.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_COMMON Format: OpCode
	26:23	**Opcode** Default Value: 1h VDENC_PIPE Format: OpCode
	22:21	**SubOpA** Default Value: 0h Format: OpCode
	20:16	**SubOpB** Default Value: 4h VDENC_PIPE_BUF_ADDR_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2

VDENC_PIPE_BUF_ADDR_STATE

		Value / **Name** / **Description** 41h / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)
1..3	95:0	**DS FWD REF0** Format: VDENC_Down_Scaled_Reference_Picture
4..6	95:0	**DS FWD REF1** Format: VDENC_Down_Scaled_Reference_Picture **Description** This field is used for pointing to DSFWD reference picture for L0 Refidx 1 used for Stage 1 of Motion Estimation. AVC Mode: This field must be programmed to have the same base address as DS FWD REF0 [List 0, refidx 0] when VDEncPerfMode is enabled in the VDENC_IMAGE_STATE. In Normal Mode, This field points to DS FWD REF1 [List 0, refidx 1] In HEVC / VP9 Mode: This field points to Stage 1 (8X Downscaled) base address[List 0, refidx 1] for LDB frame type in HEVC / P - frame in VP9. HEVC Mode: This field points to Stage1 (8X DS BWD REF 0) [List 1, refidx 0] for RA frame type in HEVC.
7..9	95:0	**Reserved** Format: MBZ
10..12	95:0	**Original Uncompressed Picture** Format: VDENC_Original_Uncompressed_Picture
13..15	95:0	**Reserved** Format: MBZ
16..18	95:0	**Row Store Scratch Buffer** Format: VDENC_Row_Store_Scratch_Buffer_Picture **Programming Notes** This surface can point to internal buffer or external memory. When mapped to external memory and scalability is enabled, care should be taken to keep the base addresses for multiple pipes apart to ensure row store write from one pipe doesn't overwrite the other. Example: Pipe1: Base address : X and tile width in pixels is W. Pipe 2 Base address should be >= X + 2 * W
19..21	95:0	**Colocated MV Read Buffer** Format: VDENC_Colocated_MV_Picture

VDENC_PIPE_BUF_ADDR_STATE

		Programming Notes This is ignored by HW since B-Frames are not supported.
22..24	95:0	**FWD REF0** Format: VDENC_Reference_Picture **Programming Notes** The reference base address and cache control values programmed here would have to match reference picture address seen by PAK after HCP_REF_IDX translation. Example: Reference Picture 7 is used as reference for current frame. VDENC FWD REF0 should be programmed with [7] HCP_REF_IDX (L0) should map to reference picture 7 for reference index 0 in Reference picture list 0. This reference address points to current frame pre-deblocked reconstructed if NumRefIdxL0_minus1 equals 0 and TBC is enabled.
25..27	95:0	**FWD REF1** Format: VDENC_Reference_Picture **Programming Notes** The reference base address and cache control values programmed here would have to match reference picture address seen by PAK after HCP_REF_IDX translation. Example: Reference Picture6 is used as reference for current frame. VDENC FWD REF1 should be programmed with [6] HCP_REF_IDX (L0) should map to reference picture6 for reference index1 in Reference picture list 0. This reference address points to current frame pre-deblocked reconstructed if NumRefIdxL0_minus1 equals 1 and TBC is enabled.
28..30	95:0	**FWD REF2** Format: VDENC_Reference_Picture **Programming Notes** The reference base address and cache control values programmed here would have to match reference picture address seen by PAK after HCP_REF_IDX translation. Example: Reference Picture0 is used as reference for current frame. VDENC FWD REF1 should be programmed with [0] HCP_REF_IDX (L0) should map to reference picture0 for reference index2 in Reference picture list 0. This reference address points to current frame pre-deblocked reconstructed if NumRefIdxL0_minus1 equals2 and TBC is enabled.

VDENC_PIPE_BUF_ADDR_STATE

31..33	95:0	**BWD REF0** Format: VDENC_Reference_Picture **Programming Notes** The reference base address and cache control values programmed here would have to match reference picture address seen by PAK after HCP_REF_IDX translation. Example: Reference Picture1 is used as only backward reference. VDENC BWD REF0 should be programmed with [1] HCP_REF_IDX (L1) should map to reference picture1 for reference index0 in Reference picture list 1. This reference address points to current frame pre-deblocked reconstructed if NumRefIdxL0_minus1 equals3 and TBC is enabled.
34..36	95:0	**Reserved** Format: MBZ
37..39	95:0	**DS FWD REF0 4X** Format: VDENC_Down_Scaled_Reference_Picture For HEVC / VP9 VDEnc, this field indicates the 4X DS surface base address used in the stage2 ME for Reference frame0.
40..42	95:0	**DS FWD REF1 4X** Format: VDENC_Down_Scaled_Reference_Picture **Description** HEVC Mode: In RA pictures, this field points to DS BWD REF0 (List 1 refidx 0)used in the stage2 HME. HEVC Mode / VP9 Mode: In LDB picture of HEVC and VP9 Mode, this field points to DS FWD REF1, Down scaled Forward Reference 1(List 0 refidx 1), used in stage 2 HME
43..45	95:0	**Reserved** Format: MBZ
62..64	95:0	**VDENC Tile Row store Buffer** Format: VDENC_Row_Store_Scratch_Buffer_Picture Specifies 64 byte aligned Tile Row store buffer. This surface is a linear surface. The total size of the surface to be allocated is 32*(W+31)/32 * 4 Bytes, Where W is width of the picture in pixels. Each 32x32 needs one CL for row store. To support tile replay we need tile write rowstore and tile read rowstore.
65..67	95:0	**Reserved** Format: MBZ
86..88	95:0	**Reserved** Format: MBZ

VDENC_PIPE_MODE_SELECT

VDENC_PIPE_MODE_SELECT

Source: VideoCS

Length Bias: 2

Specifies which codec and hardware module is being used to encode/decode the video data, on a per-frame basis.
The **VDENC_PIPE_MODE_SELECT** command specifies which codec and hardware module is being used to encode/decode the video data, on a per-frame basis. It also configures the hardware pipeline according to the active encoder/decoder operating mode for encoding/decoding the current picture.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_COMMON Format: OpCode
	26:23	**Opcode** Default Value: 1h VDENC_PIPE Format: OpCode
	22:21	**SubOpA** Default Value: 0h Format: OpCode
	20:16	**SubOpB** Default Value: 0h VDENC_PIPE_MODE_SELECT Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 Value: 4 — Name:
1	31	**Reserved** Format: MBZ
	30:28	**Reserved** Format: MBZ

VDENC_PIPE_MODE_SELECT

	27:26	**Reserved**
	25	**Reserved**
	24:23	**Reserved**
	22:21	**Reserved**
	20	**Reserved**
	19	**Reserved** Format: MBZ
	18	**isRandomAccess bit** This bit needs to be set to 1 in B-frames.
	17	**Reserved**
	16:15	**PAK chroma sub-sampling type** Format: U2 This field is applicable only in HEVC and VP9. In AVC, this field is ignored. **Value** / **Name** / **Description** 0 / Reserved / 1 / 4:2:0 **[Default]** / Used for Main8 and Main10 HEVC, VP9 profile0. 3 / 4:4:4 / HEVC RExt 444, VP9 444 profiles.
	14:10	**Reserved** Format: MBZ
	9	**Reserved**
	8	**Reserved**
	7	**Reserved**
	6	**Reserved** Format: MBZ
	5	**Reserved**
	4	**Scalability Mode** Format: Enable When this is set, frame is encoded using multiple VDENC pipes. This bit is not used by HW.
	3:0	**Standard Select** Format: U4 **Value** / **Name** 2 / AVC [4-15] / Reserved

VDENC_PIPE_MODE_SELECT		
2	31:28	**Reserved** Access: RO Format: MBZ
	27:24	**Reserved**
	23:20	**Reserved**
	19:16	**Reserved**
	15:12	**Reserved** Format: MBZ
	11:8	**Reserved**
	7:4	**Reserved**
	3	**Reserved**
	2:1	**Reserved**
	0	**Reserved**
3	31:28	**Reserved** Format: MBZ
	27:24	**Reserved**
	23:20	**Reserved**
	19:16	**Reserved**
	15:12	**Reserved** Format: MBZ
	11:8	**Reserved**
	7:4	**Reserved**
	3:2	**Reserved** Format: MBZ
	1	**Reserved**
	0	**Reserved**
4	31:0	**Reserved** Format: MBZ
5	31:0	**Reserved** Format: MBZ

VDENC_REF_SURFACE_STATE

VDENC_REF_SURFACE_STATE

Source: VideoCS

Length Bias: 2

This command specifies the surface state parameters for the normal reference surfaces.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_COMMON Format: OpCode
	26:23	**Opcode** Default Value: 1h VDENC_PIPE Format: OpCode
	22:21	**SubOpA** Default Value: 0h Format: OpCode
	20:16	**SubOpB** Default Value: 2h VDENC_REF_SURFACE_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 Value: 4h — Name: DWORD_COUNT_n **[Default]** — Description: Excludes DWord (0,1)
1	31:0	**Reserved** Format: MBZ
2..5	127:0	**Dwords 2..5** Format: VDENC_Surface_State_Fields Four DWords that define the bit fields used by the three surface state commands. **Programming Notes** The height and width fields in surface state commands are not used by HW.

VDENC_SRC_SURFACE_STATE

VDENC_SRC_SURFACE_STATE

Source: VideoCS

Length Bias: 2

This command specifies the uncompressed original input picture to be encoded.
The actual base address is defined in the VDENC_PIPE_BUF_ADDR_STATE. Pitch can be wider than the Picture Width in pixels and garbage will be there at the end of each line. T
For non pixel data, such as row stores, DMV and streamin/out, a linear buffer is employed. For row stores, the H/W is designed to guarantee legal memory accesses (read and write). For the remaining cases, indirect object base address, indirect object address upper bound, object data start address (offset) and object data length are used to fully specified their corresponding buffer. This mechanism is chosen over the pixel surface type because of their variable record sizes.
All row store surfaces are linear surface. Their addresses are programmed in VDEnc_Pipe_Buf_Base_State.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_COMMON Format: OpCode
	26:23	**Opcode** Default Value: 1h VDENC_PIPE Format: OpCode
	22:21	**SubOpA** Default Value: 0h Format: OpCode
	20:16	**SubOpB** Default Value: 1h VDENC_SRC_SURFACE_STATE Format: OpCode
	15:12	**Reserved** Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 Value / Name / Description 4h / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1)
1	31:0	**Reserved** Format: MBZ

VDENC_SRC_SURFACE_STATE		
2..5	127:0	**Dwords 2..5** Format: VDENC_Surface_State_Fields Four DWords that define the bit fields used by the three surface state commands. **Programming Notes** The Width field programmed in DW 2 bit 17:4 should match frame width in pixel, ignoring the comment in Width field

VDENC_WALKER_STATE

VDENC_WALKER_STATE

Source: VideoCS

Length Bias: 2

This command provides the macroblock start location for the VDEnc walker. Current programming to always have this command at the frame level, hence the macroblock X, Y location need to be programmed to 0,0 to always start at frame origin. Once the hardware receives this command packet, it internally starts the VDEnc pipeline. This should be the last command that is programmed for the VDEnc pipeline.

This command is programmed per super-slice. The X location always needs to be programmed to 0. The Y location needs to be programmed to the starting point of the current super-slice. The programming needs to ensure that all super-slices are contiguous. It is illegal to have gaps between the super-slices.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h MFX_COMMON Format: OpCode
	26:23	**Opcode** Default Value: 1h VDENC_PIPE Format: OpCode
	22:21	**SubOpA** Default Value: 0h Format: OpCode
	20:16	**SubOpB** Default Value: 7h VDENC_WALKER_STATE Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Format: =n Total Length - 2 Value / Name / Description 0h / DWORD_COUNT_n **[Default]** / Excludes DWord (0,1) Fh / DWORD_COUNT_n(HEVC_VP9_SCC) /

VDENC_WALKER_STATE

1	31:29	**Reserved** Access: RO Format: MBZ
	28	**First Super Slice** This bit is set for only the first walker command of a tile. Subsequent walker commands in the same tile wouldn't have this bit set.
	27:25	**Reserved** Access: RO Format: MBZ
	24:16	**MB/LCU Start X Position** Format: U9 The MB walker is programmed to start at the beginning of the super-slice. But the X position is still 0 since the super-slices need to be a multiple of MB Rows.
	15:9	**Reserved** Access: RO Format: MBZ
	8:0	**MB/LCU Start Y Position** Format: U9 The MB walker is programmed to start at the beginning of the super-slice.The MB Start Y position is no longer programmed to be always zero. It points to the MB Y offset for the first MB in the super-slice.
2	31:26	**Reserved** Access: RO Format: MBZ
	25:16	**NextSlice MB/LCU Start X Position** This field indicates the next slice starting MB X position. This needs to be programmed to zero since super-slices are MB Row aligned. Programming this value to non-zero is illegal for frames that have more than one slice (multi-slice). **Programming Notes** Format: U10. This field is not used by HW.
	15:10	**Reserved** Access: RO Format: MBZ
	9:0	**NextSlice MB Start Y Position** This field indicates the start Y position of the MB walker when running VDEnc in the walker mode.

VDENC_WALKER_STATE

		For the very last super-slice in the frame, this parameter is set to the **FrameHeightinMBs**. **Programming Notes** Format: U10
3	31:24	**Tile number**
	23:17	**Reserved** Access: RO Format: MBZ
	16	**Tile Row store Select** This bit selects Tile row store offset for write and read when TILE_REPLAY feature is enabled. Tile Width: TW, Adjusted Tile Width ATW = 32 * (TW+31)/32 Tile start position: X Valid Value: 0 — Tile Row store read offset in CL: X/32 * 2 — Tile Row store write offset in CL: X/32 * 2 + (ATW) /32 Valid Value: 1 — Tile Row store read offset in CL: X/32 * 2 + (ATW) /32 — Tile Row store write offset in CL: X/32 * 2
	15:11	**Reserved** Access: RO Format: MBZ
	10:9	**NUM_PAR_ENGINE**
	8:7	**Reserved** Access: RO Format: MBZ
	6:4	**Reserved** Access: RO Format: MBZ
	3	**Reserved** Access: RO Format: MBZ
	2:0	**Log 2 Weight Denom Luma** It is the base 2 logarithm of the denominator for all Luma weighting factors. It is set to the value of the syntax element read from the Slice Header Pred_Weight_Table(). In AVC VDENC, this is applied to VIME.
4	31:16	**Tile Start CTB-X** Format: U16 This parameter indicates the Tile starting X location. It is currently a place-holder for Tiling support. [Restriction]: This must be programmed to zero if the entire frame is one tile. If there are multiple tiles in the frame and more than one super-slice in the current tile, this parameter must be the same across all super-slice commands within the current tile. Note: Slices cannot cross tile boundaries.

VDENC_WALKER_STATE

	15:0	**Tile Start CTB-Y** Format: U16 This parameter indicates the Tile starting Y location. It is currently a place-holder for Tiling support. [Restriction]: This must be programmed to zero if the entire frame is one tile. If there are multiple tiles in the frame and more than one super-slice in the current tile, this parameter must be the same across all super-slice commands within the current tile. Note: Slices cannot cross tile boundaries.
5	31:16	**Tile Height** Format: U16 This parameter indicates the Height of the current Tile in pixels (Zero based). **Programming Notes** Tile height programmed should be n*8 +7, where n is integer greater than or equal to 15. In other words it tile height is greater than or equal to 128 and it should be multiple of 8 pixels. Maximum height of a tile is 8191
	15:0	**Tile Width** Format: U16 This parameter indicates the width of the current Tile in pixels (Zero based). When tiling is disabled, this field should be width of the frame in pixels (zero based) **Programming Notes** Tile width programmed should be n*8 +7, where n is integer greater than or equal to 31. In other words, it tile width is equal to or greater than 256 pixels and it should be multiple of 8 pixels. When this field is less than 319, IBC Control can't be set to 3.
6	31:6	**Tile Streamin Offset** Format: U26 **Programming Notes** Tile offset in Cache Lines for reading streamIn surface for the current tile.
	5:1	**Reserved** Access: RO Format: MBZ
	0	**Streamin Offset enable** **Programming Notes** This bit should be set for the first tile of a Tile column.

VDENC_WALKER_STATE

7	31:6	**Tile Rowstore Offset** Format: U26 Tile offset in Cache Lines for Row store buffer address where row store data for the current tile is written.
	5:1	**Reserved** Access: RO Format: MBZ
	0	**Row store Offset enable** **Programming Notes** This bit should be set for the first tile of a Tile column.
8	31:6	**Tile streamout offset** Format: U26 Offset in Cache Lines from Streamout buffer address from where the current tiles statistics are written out.
	5:1	**Reserved** Access: RO Format: MBZ
	0	**Tile streamout offset enable** **Programming Notes** This bit should be set for all Tiles.
9	31:6	**Tile LCU stream out offset** This field gives tile offset for writing LCU PAK_OBJ commands.
	5:1	**Reserved** Access: RO Format: MBZ
	0	**LCU stream out offset enable** **Programming Notes** This bit should be set for tiles with start y = 0 and start x != 0.
10	31:0	**Reserved** Access: RO Format: MBZ
11	31:0	**Reserved** Access: RO Format: MBZ

VDENC_WALKER_STATE

12	31:0	**Reserved** Access: RO Format: MBZ
13	31:0	**Reserved** Access: RO Format: MBZ
14	31	**Max Escape Bins check enable** Format: Enable When "Max Escape Bins Check enable" is set, Max Escape Bins by 8 field would be used to limit number of Escape chars in CU. This field is added to get predictable PAK performance.
	30:24	**Max Escape Bins by 8** Default Value: 63 Format: U7 Defined per 8x8 CU and scaled linearly for 16x16 CU and 32x32 CU. After a given CU exceeds this number of bins spent on escapes within a CU, no more escapes for that CU will be mapped even if the best valid color identified exceeds ESCT. **Programming Notes** Validation commit is to have this field same in all walker commands of a frame.
	23	**Reserved** Access: RO Format: MBZ
	22:21	**HashGuardBand 16x16** Format: U2 Each 16x16 CU has at least this many new colors reserved for it (while each 32x32 has limited to 32 new colors total). **Value** / **Name** [0-3] 2 **[Default]** **Programming Notes** Validation commit is have this field same in all walker commands of a frame.
	20:16	**HashThreshold Class 1 Length** Format: U5 No more than HT1L Class1 Hashcolors can be valid for mapping. If this is programmed to zero, then no HT1L Class1 Hashcolors will be valid for mapping. **Value** / **Name** [0-31]

VDENC_WALKER_STATE

		Programming Notes Validation commit is have this field same in all walker commands of a frame.
	15:14	**Reserved** Access: RO Format: MBZ
	13:8	**HashThreshold Class 0 Count** Format: U6 A Class0 Hashcolor must occur at least this many times to be valid for mapping. **Programming Notes** Validation commit is have this field same in all walker commands of a frame.
	7:6	**Reserved** Access: RO Format: MBZ
	5:0	**HashThreshold Class 1 Count** Format: U6 A Class1 Hashcolor must occur at least this many times to be valid for mapping. **Programming Notes** Validation commit is have this field same in all walker commands of a frame.
15	31:0	**Reserved** Format: MBZ
16	31:0	**Reserved** Format: MBZ
17	31:6	**Cumulative CU Tile offset** Format: U26 **Programming Notes** Tile offset in Cache Lines for cumulative CU count.
	5:1	**Reserved** Access: RO Format: MBZ
	0	**Cumulative CU Tile Offset enable**
18	31:0	**Reserved** Format: MBZ
19	31:0	**Reserved** Format: MBZ

VDENC_WALKER_STATE		
20	31:0	**Reserved** Format: MBZ
21	31:0	**Reserved** Format: MBZ
22	31:0	**Reserved** Format: MBZ
23	31:0	**Reserved** Format: MBZ
24	31:0	**Reserved** Format: MBZ
25	31:0	**Reserved** Format: MBZ
26	31:0	**Reserved** Format: MBZ

VEBOX_STATE

VEBOX_STATE

Source: VideoEnhancementCS

Length Bias: 2

This command controls the internal functions of the VEBOX. This command has a set of indirect state buffers:

- DN/DI state
- IECP general state
- IECP Gamut Expansion/Compression state
- IECP Gamut Vertex Table state
- Capture Pipe state

Adds the LACE LUT Table as an indirect state buffer.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Command OpCode** Default Value: 4h VEBOX Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 2h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Format: =n Value: 11h, Name: , Description: (Excludes DWords 0, 1)

VEBOX_STATE

1	31:25	**State Surface Control Bits** All Indirect state buffers use state surface control bits, only exception being 3D LUT state buffer for which the state surface control bits are tied to 0. See definition under "VEB_DI_IECP_COMMAND_SURFACE_CONTROL_BITS" Bits[6:0] is only used.
	24	**FP16 mode enable**
	23	**Reserved**
	22	**Gamut Expansion Position** If Gamut Expansion is enabled, it can be configured either in front or backend of the IECP pipe using this bit. **Value** / **Name** 0b – Gamut Expansion at the Backend of IECP pipe 1b – Gamut Expansion at the Front of IECP pipe
	21	**Forward Gamma Correction Enable** Format: Enable **Programming Notes** Single Pipe IECP Enable must also be set if this is enabled. When enabled the forward gamma will always be in front of the IECP pipe. In case disabled it will be always configured as Gamut expansion. Gamut Expansion, HDR and Forward Gamma Correction are mutually exclusive.
	20	**Scalar Mode** When Scalar Mode is enabled, all other VEBOX functions must be disabled (DN/DI/DM/IECP/Chroma upsampling).
	19	**Single Pipe Enable** Indicates that the Capture Pipe features that only exist in a single pipe can be enabled. This bit must be set if any of the following features are enabled: Demosaic Denoise with one of the RGBA input formats IECP only mode with Forward Gamma Correction enabled with RGB input formats (All other modes are not supported in single pipe) **Value** / **Name** 1 – Enable 0 – Default **[Default]** **Programming Notes** Note that the pixel throughput is 1/2 when this mode is selected. The **Global IECP Enable** must also be set.
	18	**Disable Temporal Denoise Filter** If set this bit will force the denoise filter to only use the spatial filter. This will eliminate the read of the previous denoise surface and STMM/Denoise History surface and the write of the current denoised surface and STMM/Denoise History surface.

		Programming Notes **The Global IECP Enable** or **Demosaic Enable** must be set along with this bit. This bit must be set if the input to Denoise is RGB. This bit must not be set if the Deinterlacer is enabled. This bit must be clear if both **DN Enable**=0 and **Hot Pixel Filtering Enable**=0This bit must be set if **Hot Pixel Filtering Enable**=1 and both DN and DI are disabled
	17	**Disable Encoder Statistics** If set this bit will disable writing the per block Encoder statistics. The memory format is not changed, so the area set aside for these statistics will still be there.
	16	**LACE Correction Enable** This bit enables the correction of the image according to the local ACE LUT tables. This is independent from the enable for the collection of LACE histograms. **Programming Notes** LACE correction is only enabled if both this bit and the **Global IECP Enable** are set. The **ACE Enable** bit should also be set if this bit is set, since ACE correction can be used for part of the luma range instead of LACE.
	15:14	**Reserved** Access: RO Format: MBZ
	13	**Hot Pixel Filtering Enable** Enables hot pixel detection/filtering.
	12	**Alpha Plane Enable** Enables the reading of an independent Alpha plane. Mutually exclusive with Vignette Enable. If **Alpha from State Select** is set it overrides this bit. **Programming Notes** IECP must also be enabled and output format must have alpha if this bit is enabled. Should be 0 if Alpha from State Select is 1.
	11	**Vignette Enable** Enables Vignette Correction surface read and correction in IECP. Mutually exclusive with Alpha Plane Enable. **Programming Notes** Demosaic must also be enabled if this bit is enabled.
	10	**Demosaic Enable** The Demosaic will be used, and White balance statistics will be gathered. The Capture Pipe State Table will be read. This bit is mutually exclusive with **DI Enable.** **Programming Notes** IECP must also be enabled if this bit is enabled.
	9:8	**DI Output Frames** Indicates which frames to output in DI mode. **Value** / **Name** 00b / Output Both Frames

01b	Output Previous Frame Only
10b	Output Current Frame Only

Programming Notes
Field is ignored if DI Enable = 0.If Previous Frame Only or Current Frame Only are selected, then the **LACE Single Histogram Set** must not try to collect a histogram from the disabled frame.
Field must be programmed to 10 (Output Current Frame Only) for DI First Frame.

7:6 **Reserved**

Access:	RO
Format:	MBZ

5 **DN/DI First Frame**

Format:	Enable

Indicates that this is the first frame of the stream, so previous clean is not available.

Value	Name
0	Not first field; previous clean surface state is valid
1	First field; previous clean surface state is invalid

Programming Notes
If both DN and DI are disabled, this bit must be 0.

4 **DI Enable**

Format:	Enable

Deinterlacer is bypassed if this is disabled: the output is the same as the input (same as a 2:2 cadence). FMD and STMM are not calculated and the values in the response message are 0.

Value	Name
0	Do not calculate DI
1	Calculate DI

3 **DN Enable**

Format:	Enable

Denoise is bypassed if this is low - BNE is still calculated and output, but the denoised fields are not. VDI does not read in the denoised previous frame but uses the pointer for the original previous frame.

Value	Name
0	Do not denoise frame
1	Denoise frame

Programming Notes
If DN and/or Hotpixel are the only functions enabled then the only output is the Denoised Output which is the same surface format as the input. To get a format conversion with DN only, enable the Global IECP bit, but disable all the individual functions. The IECP output uses the

		output surface format. If DN is used with RGB then the **Global IECP** Enable must also be
	2	**Global IECP Enable** Indicates if any of the IECP features is enabled. If this is disabled then no state will be read from any of the state pointers. If set then the IECP state will be read.
	1	**Color Gamut Compression Enable** Indicates if the Gamut Compression feature is enabled. If set then the Gamut State will be read. VEB_VERTEXTABLE_STATE is only needed if this bit is set.
	0	**Color Gamut Expansion Enable** Indicates if the Gamut Expansion feature is enabled. If set then the Gamut State will be read. This can be enabled only if Single pipe enable is disabled.
2	31:12	**DN/DI State Pointer Low** Format: GraphicsAddress[31:12] Bits 31:12 of the starting address of the DN/DI State buffer. This points to a buffer containing the 10 Dwords of the DN/DI state. When Scalar mode is enabled this pointer is used for Scalar state table.
	11:0	**Reserved** Access: RO Format: MBZ
3	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**DN/DI State Pointer High** Format: GraphicsAddress[47:32] Bits 47:32 of the starting address of the DN/DI State Buffer. When Scalar mode is enabled this pointer is used for Scalar state table.
4	31:12	**IECP State Pointer Low** Format: GraphicsAddress[31:12] Bits 31:12 of the starting address of the IECP State buffer. This points to a buffer containing the 64 Dwords of IECP state.
	11:0	**Reserved** Access: RO Format: MBZ
5	31:16	**Reserved** Access: RO Format: MBZ

VEBOX_STATE

	15:0	**IECP State Pointer High** Format: GraphicsAddress[47:32] Bits 47:32 of the starting address of the IECP State Buffer Table.
6	31:12	**Gamut/HDR State Pointer Low** Format: GraphicsAddress[31:12] Bits 31:12 of the starting address of the State Buffer. If Gamut Expansion is enabled, this points to a buffer containing the Gamut Expansion Gamma Correction state. If HDR is enabled, this points to a buffer containing the HDR state.
	11:0	**Reserved** Access: RO Format: MBZ
7	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Gamut/HDR State Pointer High** Format: GraphicsAddress[47:32] Bits 47:32 of the starting address of the Gamut/HDR State Buffer.
8	31:12	**Vertex Table State Pointer Low** Format: GraphicsAddress[31:12] Bits 31:12 of the starting address of the Vertex Table. This points to a buffer containing the 512 Dwords of the Gamut Compression Vertex Table.
	11:0	**Reserved** Access: RO Format: MBZ
9	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Vertex Table State Pointer High** Format: GraphicsAddress[47:32] Bits 47:32 of the starting address of the Vertex State Buffer.
10	31:12	**Capture Pipe State Pointer Low** Format: GraphicsAddress[31:12] Bits 31:12 of the starting address of the Capture Pipe State Table. This points to a buffer containing the X Dwords of the Capture Pipe State.
	11:0	**Reserved** Access: RO Format: MBZ

VEBOX_STATE

11	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Capture Pipe State Pointer High** Format: GraphicsAddress[47:32] Bits 47:32 of the starting address of the Capture Pipe State Table.
12	31:12	**LACE LUT Table State Pointer Low** Format: GraphicsAddress[31:12] Bits [31:12] of the starting address of the LACE Look-up Tables.
	11:0	**Reserved** Access: RO Format: MBZ
13	31:30	**Arbitration Priority Control - For LACE LUT** Format: U2 This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. Value / Name: 0 – Highest priority 1 – Second highest priority 2 – Third highest priority 3 – Lowest priority
	29:16	**Reserved** Access: RO Format: MBZ
	15:0	**LACE LUT Table State Pointer High** Format: GraphicsAddress[47:32] Bits [47:32] of the starting address of the LACE Look-up Tables.
14..15	63:12	**Gamma Correction Values Address** Format: VIRTUAL_ADDR[63:12] Specifies the 4K byte aligned address reading the Gamma Correction Values in case enabled.
	11:0	**Reserved** Access: RO Format: MBZ

16	31:12	**3D LUT State Pointer Low** Format: GraphicsAddress[31:12] Bits [31:12] of the starting address of the 3D LUT.
	11:0	**Reserved** Access: RO Format: MBZ
17	31:30	**Arbitration Priority Control - For 3D LUT** Format: U2 This field controls the priority of arbitration used in the GAC/GAM pipeline for this surface. Value 0: Highest Priority Value 1: Second highest priority Value 2: Third highest priority Value 3: Lowest priority
	29	**Reserved**
	28:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:16	**3D LUT MOCS table** These are surface control bits for VEBOX 3DLUT data requests to GAV
	15:0	**3D LUT State Pointer High** Format: GraphicsAddress[47:32] Bits [47:32] of the starting address of the 3D LUT.
18	31	**3D LUT Enable** Default Value: 0 Format: Enable **3D LUT is required only if this is enabled.** Restriction: The frame height needs to be multiple of 8 when enabling 3dlut in VEBOX dual pipe mode.
	30:29	**3D LUT Size** Format: U2 Value 00b: 33x33x33

	01b	17x17x17
	10b	65x65x65

28:23 **Reserved**

Access:	RO
Format:	MBZ

22:16 **Reserved**

Access:	RO
Format:	MBZ

15:14 **Reserved**

Access:	RO
Format:	MBZ

13:12 **Frame statistics ID**

Format:	U2

This field specifies the Statistics Surface ID number to the VEBOX to writeout the frame statistics.

11 **Bypass Chroma Downsampling**

Format:	U1

When enabled will drop chroma samples at odd position and not use the co-sited offsets.

10 **Bypass Chroma Upsampling**

Format:	U1

When enabled will replicate chroma samples at odd position and not use the co-sited offsets.

9:7 **Chroma Downsampling Co-Sited Vertical Offset**

Format:	U3

Value	Name
0	**[Default]**
[0,2]	Valid Range

6:5 **Chroma Downsampling Co-Sited Horizontal Offset**

Format:	U2

Value	Name
0	**[Default]**
[0,2]	Valid Range

VEBOX_STATE

	4:2	**Chroma Upsampling Co-Sited Vertical Offset** Format: U3 **Value** / **Name** 0 / **[Default]** [0,4] / Valid Range
	1:0	**Chroma Upsampling Co-Sited Horizontal Offset** Format: U2 **Value** / **Name** 0 / **[Default]** [0,1] / Valid Range

VEBOX_SURFACE_STATE

VEBOX_SURFACE_STATE	
Source:	VideoEnhancementCS
Length Bias:	2

The input and output data containers accessed are called "surfaces". Surface state is sent to VEBOX via an inline state command rather than using binding tables. SURFACE_STATE contains the parameters defining each surface to be accessed, including its size, format, and offsets to its subsurfaces. The surface's base address is in the execution command. Despite having multiple input and output surfaces, we limit the number of surface states to one for input surfaces and one for output surfaces. The other surfaces are derived from the input/output surface states.
The Current Frame Input surface uses the Input SURFACE_STATE
The Previous Denoised Input surface uses the Input SURFACE_STATE. (For 16-bit Bayer pattern inputs this will be 16-bit.)
The Current Denoised Output surface uses the Input SURFACE_STATE. (For 16-bit Bayer pattern inputs this will be 16-bit.)
The STMM/Noise History Input surface uses the Input SURFACE_STATE with Tile-Y and Width/Height a multiple of 4.
The STMM/Noise History Output surface uses the Input SURFACE_STATE with Tile-Y and Width/Height a multiple of 4.
The Current Deinterlaced/IECP Frame Output surface uses the Output SURFACE_STATE.
The Previous Deinterlaced/IECP Frame Output surface uses the Output SURFACE_STATE.
The FMD per block output / per Frame Output surface uses the Linear SURFACE_STATE (see note below).
The Alpha surface uses the Linear A8 SURFACE_STATE with Width/Height equal to Input Surface. Pitch is width rounded to next 64.
The Skin Score surface uses the Output SURFACE_STATE.
The STMM height is the same as the Input Surface height except when the input **Surface Format** is Bayer Pattern and the **Bayer Pattern Offset** is 10 or 11, in which case the height is the input height + 4. For Bayer pattern inputs when the **Bayer Pattern Offset** is 10 or 11, the Current Denoised Output/Previous Denoised Input will also have a height which is the input height + 4.For Bayer pattern inputs only the Current Denoised Output/Previous Denoised Input are in Tile-Y.
The linear surface for FMD statistics is linear (not tiled). The height of the per block statistics is (Input Height +3)/4 - the Input Surface height in pixels is rounded up to the next even 4 and divided by 4. The width of the per block section in bytes is equal to the width of the Input Surface in pixels rounded up to the next 16 bytes. The pitch of the per block section in bytes is equal to the width of the Input Surface in pixels rounded up to the next 64 bytes.
The STMM surfaces must be identical to the Input surface except for the tiling mode must be Tile-Y and the pitch is specified in DW7. The pitch for the Current Denoised Output/Previous Denoised Input is specified in DW7. The width and height must be a multiple of 4 rounded up from the input height.

VEBOX_SURFACE_STATE

The Vignette Correction surface uses the Linear 16-bit SURFACE_STATE with :
Width=(Ceil(Image Width / 4) +1) * 4
Height= Ceil(Image Height / 4) +1
Pitch in bytes is (vignette width *2) rounded to the next 64

Programming Notes
VEBOX may write to memory between the surface width and the surface pitch for output surfaces.
VEBOX can support a frame level X/Y offset which allows processing of 2 side-by-side frames for certain 3D video formats.
The X/Y Offset for Frame state applies only to the Current Frame Input and the Current Deinterlaced/IECP Frame Output and Previous Deinterlaced/IECP Frame Output. The statistics surfaces, the denoise feedback surfaces and the alpha/vignette surfaces have no X/Y offsets.
For 8bit Alpha input, when converted to 16bit output, the 8 bit alpha value is replicated to both the upper and lower 8 bits to form the 16 bit alpha value.
Skin Score Output Surface uses the same tiling format as the Output surface.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Media Command Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Media Command OpCode** Default Value: 4h VEBOX Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h VEBOX Format: OpCode
	20:16	**SubOpcode B** Default Value: 0h VEBOX Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Format: =n

VEBOX_SURFACE_STATE

		Value	Name	Description
		7h	DWORD_COUNT_n **[Default]**	(Excludes DWords 0, 1)
1	31:1	**Reserved**		
		Access:		RO
		Format:		MBZ
	0	**Surface Identification** Specifies which set of surfaces this command refers to:		
		Value	**Name**	
		1	Output surface (all except the Denoised Current output surface)	
		0	Input surface and Denoised Current Output Surface	
2	31:18	**Height**		
		Format:		U14
		This field specifies the height of the surface in units of pixels. For PLANAR surface formats, this field indicates the height of the Y (luma) plane.		
		Value	**Name**	**Description** / **Exists If**
		[15, 16383]		representing heights [16,16384]
		[15, 8191]		//Scalar Enabled - For Input surface only
		[63, 2047]		//Scalar + SFC Enabled - For Input surface only
		Programming Notes		
		Height (field value + 1) must be a multiple of 2 for PLANAR_420 surfaces. **Height** (field value +1) must be a multiple of 2 when the deinterlace function is enabled (field mode) or when the denoise function is enabled with **Progressive DN** = 0. It must be a multiple of 4 when interleaved deinterlace/denoise and PLANAR_420 are both being used. **VEBOX** supports a minimum height of 16.		
		Height (field value + 1) must be a multiple of 2 for Bayer surfaces.		
	17:4	**Width**		
		Format:		U14
		This field specifies the width of the surface in units of pixels. For PLANAR surface formats, this field indicates the width of the Y (luma) plane.		
		Value	**Name**	**Description** / **Exists If**
		[63,16383]		representing widths [64,16384]
		[63,8191]		//Scalar Enabled - For Input surface only
		[63,2047]		//Scalar and SFC Enabled - For Input Surface only

		Programming Notes The Width specified by this field multiplied by the pixel size in bytes must be less than or equal to the surface pitch (specified in bytes via the **Surface Pitch** field). **Width** (field value + 1) must be a multiple of 2 for PLANAR_420, PLANAR_422, and all YCRCB_* surfaces, and must be a multiple of 4 for PLANAR_411 surfaces. **VEBOX** supports a minimum width of 64
	3:0	**Reserved** Access: RO Format: MBZ
3	31:27	**Surface Format** Format: U5 Specifies the format of the surface. All of the Y and G channels will use table 0 and all of the Cr/Cb/R/B channels will use table 1.

Value	Name	Description
0	YCRCB_NORMAL	
1	YCRCB_SWAPUVY	
2	YCRCB_SWAPUV	
3	YCRCB_SWAPY	
4	PLANAR_420_8	NV12 with Interleave Chroma set
5	PACKED_444A_8	
6	PACKED_422_16	
7	R10G10B10A2_UNORM / R10G10B10A2_UNORM_SRGB	
8	R8G8B8A8_UNORM / R8G8B8A8_UNORM_SRGB	
9	PACKED_444_16	
10	PLANAR_422_16	
11	Y8_UNORM	
12	PLANAR_420_16	
13	R16G16B16A16	
14	Bayer pattern	
15	Y16_UNORM	

26:25 **Bayer Pattern Offset**

Specifies the starting pixel offset for the Bayer pattern used for Capture Pipe.

Value	Name
00b	Pixel at X=0, Y=0 is Blue
01b	Pixel at X=0, Y=0 is Red
10b	Pixel at X=0, Y=0 is Green, Pixel at X=1, Y=0 is Red
11b	Pixel at X=0, Y=0 is Green, Pixel at X=1, Y=0 is Blue

VEBOX_SURFACE_STATE

24 **Bayer Pattern Format**

Specifies the format of the Bayer Pattern:

Value	Name
0b	8-bit input at a 8-bit stride
1b	16-bit input at a 16-bit stride

23:22 **Bayer Input Alignment**

Format:	U2

Value	Name
00b	MSB aligned data **[Default]**
01b	10bit LSB aligned data
10b	12bit LSB aligned data
11b	14bit LSB aligned data

Programming Notes
Valid only Bayer Pattern Format is 16bit input

21 **Reserved**

Access:	RO
Format:	MBZ

20 **Interleave Chroma**

Format:	Enable

This field indicates that the chroma fields are interleaved in a single plane rather than stored as two separate planes. This field is only used for PLANAR surface formats.

19:3 **Surface Pitch**

Format:	U17

This field specifies the surface pitch in (#Bytes - 1):

Value	Name	Description
[63, 131071]	For other linear surfaces	[64B, 128KB]
[511, 131071]	For X-tiled surface	[512B, 128KB] = [1tile, 256 tiles]
[127, 131071]	For Y-tiled surfaces	[128B,128KB] = [1 tile, 1024 tiles]

Programming Notes
For tiled surfaces, the pitch must be a multiple of the tile width. For linear surfaces, the pitch must be a multiple of 64.If Half Pitch for Chroma is set, this field must be a multiple of two tile widths for tiled surfaces, or a multiple of 2 bytes for linear surfaces.

2 **Half Pitch for Chroma**

Format:	Enable

This field indicates that the chroma plane(s) will use a pitch equal to half the value specified in the **Surface Pitch** field. This field is only used for PLANAR surface formats.

		Programming Notes Must be programmed to Zero always as this field is not used
	1	**Tiled Surface** Format: Boolean This field specifies whether the surface is tiled. 1 – True – Tiled 0 – False – Linear **Programming Notes** Linear surfaces can be mapped to Main Memory (uncached) or System Memory (cacheable, snooped). Tiled surfaces can only be mapped to Main Memory. The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit.
	0	**Tile Walk** Format: Boolean This field specifies the type of memory tiling (XMajor or YMajor) employed to tile this surface. See *Memory Interface Functions* for details on memory tiling and restrictions. This field is ignored when the surface is linear. 0 – TILEWALK_XMAJOR 1 – TILEWALK_YMAJOR **Programming Notes** The corresponding cache(s) must be invalidated before a previously accessed surface is accessed again with an altered state of this bit.
4	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**X Offset for U** Format: U13 This field must be zero for the VEBOX surface formats
	15	**Reserved** Access: RO Format: MBZ
	14:0	**Y Offset for U** Format: U15 This field specifies the vertical offset in rows from the start (origin) or the Luma(Y) plane to the start (origin) of the U(Cb) plane or the interleaved UV plane if **Interleave Chroma** is enabled.

This field is only used for PLANAR surface formats.

Programming Notes

This field must indicate an even number (bit 0 = 0).This field must be evenly divisible by 4 for Tile-Y surfaces (so the offset points to the start of a cache line) For Planar formats, if the surface is in YS or YF tile modes, the Y Offset for U should be an integral multiple of the Tile height of the Luma plane

5	31:29	**Reserved** Access: RO Format: MBZ
	28:16	**X Offset for V** Format: U13 This field must be zero for the VEBOX surface formats.
	15	**Reserved** Access: RO Format: MBZ
	14:0	**Y Offset for V** Format: U15 This field specifies the vertical offset in rows from the start (origin) of the Luma(Y) plane?to the start (origin) of the V(Cr) plane. This field is only used for PLANAR surface formats with **Interleave Chroma** disabled. **Programming Notes** This field must indicate an even number (bit 0 = 0).This field must be evenly divisible by 4 for Tile-Y surfaces (so the offset points to the start of a cache line). For Planar formats, if the surface is in YS or YF tile modes, the Y Offset for V should be an integral multiple of the Tile height of the Luma plane
6	31	**Reserved** Access: RO Format: MBZ
	30:16	**X Offset for Frame** Format: U15 This is an offset in X from the Surface Base Address in pixels for all planes using this surface. For U/V planes this is added to the X Offset for U/V. After converting to bytes this must be an integer multiple of cache lines in the tiling mode. This specifies the edge of the frame, so adjacent pixels needed for Denoise/Deinterlace/Demosaic are replicated or mirrored. **Programming Notes** If **Y Offset for Frame** >0 the X Offset must be 0. If memory compression is enabled then this must be an even number of cache lines.

	15	**Reserved** Access: RO Format: MBZ
	14:0	**Y Offset for Frame** Format: U15 This is an offset in Y from the Surface Base Address in pixels for all planes using this surface. For U/V planes this is added to the Y Offset for U/V. After converting to bytes this must be an integer multiple of cache lines in the tiling mode. This specifies the edge of the frame, so adjacent pixels needed for Denoise/Deinterlace/Demosaic are replicated or mirrored. **Programming Notes** If **X Offset for Frame** >0 the Y Offset must be 0. For Planar formats, if the surface is in YS or YF tile modes, the Y Offset for Frame should be an integral multiple of the Tile height.
7	31:27	**Reserved** Access: RO Format: MBZ
	26:17	**Reserved** Access: RO Format: MBZ
	16:0	**Derived Surface Pitch** Format: U17 This field specifies the surface pitch in (#Bytes - 1) for the derived surfaces:STMM/Denoise statistic surface is described when the **Surface Identification** bit is 0 (Input Surface). The (Current Denoise Output)/(Previous Denoise Input) surfaces are described when the bit is 1 (Output Surface).

Value	Name	Description	Exists If
[63, 131071]		[64B, 128KB]	[Tiled Surface] == 0
[511, 131071]		[512B, 128KB] = [1tile, 256 tiles]	([Tiled Surface] == 1) AND ([Tile Walk] == 0)
[127, 131071]		[128B,128KB] = [1 tile, 1024 tiles]	([Tiled Surface] == 1) AND ([Tile Walk] == 1)

Programming Notes

In DN Only mode, the pitch for the (Current Denoise Output)/(Previous Denoise Input) and the Surface Pitch must be programed the same.

The pitch must be a multiple of the tile width.

8	31:17	**Reserved** Access: RO Format: MBZ

16:0 **Surface Pitch for Skin Score Output Surfaces**

Format:	U17

This field specifies the surface pitch in (#Bytes - 1) for the Skin Score Output surface if enabled; This is present only in the output surface format and reserved for Input surface format. The height and width are the same as in the Output surface mentioned above.

Value	Name	Description	Exists If
[63, 131071]		[64B, 128KB]	[Tiled Surface] == 0
[511, 131071]		[512B, 128KB] = [1tile, 256 tiles]	([Tiled Surface] == 1) AND ([Tile Walk] == 0)
[127, 131071]		[128B,128KB] = [1 tile, 1024 tiles]	([Tiled Surface] == 1) AND ([Tile Walk] == 1)

Programming Notes
The pitch must be a multiple of the tile width.

VEBOX_TILING_CONVERT

VEBOX_TILING_CONVERT

Source: VideoEnhancementCS

Length Bias: 2

This command takes the input surface and writes directly to the output surface at high speed. The surface format and width/height of the input and output must be the same, only the tiling mode and pitch can change.

DWord	Bit	Description
0	31:29	**Command Type** Default Value: 3h PARALLEL_VIDEO_PIPE Format: OpCode
	28:27	**Pipeline** Default Value: 2h Media Format: OpCode
	26:24	**Command OpCode** Default Value: 4h VEBOX Format: OpCode
	23:21	**SubOpcode A** Default Value: 0h Format: OpCode
	20:16	**SubOpcode B** Default Value: 1h Format: OpCode
	15:12	**Reserved** Access: RO Format: MBZ
	11:0	**DWord Length** Format: =n Value | Name | Description 3h | | (Excludes DWords 0, 1)
1..2	63:12	**Input Address** Format: VIRTUAL_ADDR[63:12] Specifies bits 47:12 of the 4Kbyte-aligned frame buffer address for reading the current frame.
	11	**Reserved** Access: RO

VEBOX_TILING_CONVERT		
		Format: MBZ
	10:0	**Input Surface Control Bits** Format: VEB_DI_IECP_COMMAND_SURFACE_CONTROL_BITS
3..4 **Programming Notes:** Output address must be different from input address	63:12	**Output Address** Format: VIRTUAL_ADDR[63:12] Specifies bits 47:12 of the 4Kbyte-aligned frame buffer address for writing the current frame.
	11	**Reserved** Access: RO Format: MBZ
	10:0	**Output Surface Control Bits** Format: VEB_DI_IECP_COMMAND_SURFACE_CONTROL_BITS

Wait for Event

MSD_WAIT_FOR_EVENT - Wait for Event

Source: EuSubFunctionGateway

Length Bias: 1

Send a writeback if Event ID occured after MonitorEvent.

DWord	Bit	Description
0	31:29	**Reserved** Access: RO Format: MBZ
	28:25	**Message Length** Format: U4 Specifies the number of GRF registers sent as the message payload. Value 1 — Name: One **[Default]** — Description: See MDP_TIMEOUT Timeout Data Payload definition.
	24:20	**Response Length** Format: U5 Specifies the number of GRF registers expected as the message response payload. Value 1 — Name: One **[Default]** — Description: Ignore the data in the register. The update of the register is used as the event completion notification
	19:3	**Reserved** Access: RO Format: MBZ
	2:0	**Wait for Event Subfunction** Default Value: 0x6 Format: OpCode

while - While

Source:	EuIsa
Length Bias:	4
Predication:	true
Conditional Modifier:	false
Saturation:	false
Source Modifier:	false

The while instruction marks the end of a do-while block. The instruction first evaluates the loop termination condition for each channel based on the current channel enables and the predication flags specified in the instruction. If any channel has not terminated, a branch is taken to a destination address specified in the instruction, and the loop continues for those channels. Otherwise, execution continues to the next instruction.ld point to the first instruction with the do label of the do-while block of code. It should be a negative number for the backward referencing. If SPF is ON, none of the PcIP are updated.

Format:

```
[(pred)] while (exec_size) JIP
```

Restriction

The execution size must be the same for the while instruction and any break and cont instructions of the same code block.

Syntax

```
[(pred)] while (exec_size) imm32
```

Pseudocode

```
Evaluate(WrEn);
 for ( n = 0; n < 32; n++ ) {
     if (WrEn.chan[n] ) {
         PcIP[n] = IP + JIP;
     } else {
         PcIP[n] = IP + 1;
     }
 }
 if ( | PMask == 1 ) {  // any enabled channel true
     Jump(IP + JIP);
 }
```

DWord	Bit	Description
0..3	127:96	**Reserved** Exists If: ([Src0.IsImm]==false) Format: MBZ
	127:96	**JIP** Exists If: ([Src0.IsImm]==true)

while - While

	Format:	S31
	The byte-aligned jump distance if a jump is taken for the channel	
95:80	**Reserved**	
	Exists If:	([Src0.IsImm]==false)
	Format:	MBZ
95:64	**Reserved**	
	Exists If:	([Src0.IsImm]==true)
	Format:	MBZ
79:66	**Src0.Operand**	
	Exists If:	([Src0.IsImm]==false)
	Format:	**DirectOperand**
65:64	**Reserved**	
	Exists If:	([Src0.IsImm]==false)
	Format:	MBZ
63:50	**Dst.Operand**	
	Format:	**DirectOperand**
49:47	**Reserved**	
	Access:	RO
	Format:	MBZ
46	**Src0.IsImm**	
	This field indicate that Source 0 operand is carrying an immediate value.	
	Value	**Name**
	0	false
	1	true
45:34	**Reserved**	
	Access:	RO
	Format:	MBZ
33	**BranchCtrl**	
	This field is used by *goto*, *if*, and *else* instructions to control branching. See the **goto** instruction description for more information about BranchCtrl.	
32	**AtomicCtrl**	
	Format:	**AtomicCtrl**
31	**MaskCtrl**	
	Mask Control (formerly Write Enable Control). This field determines if the per channel write enables are used to generate the final write enable. This field should be normally "0".	
	Value	**Name** / **Description**

0	Normal **[Default]**	Normal. Per channel write enable used for final write enable generation.
1	NoMask	NoMask.Skips the check for PcIP[n] == ExIP before enabling a channel, as described in the Evaluate Write Enable section.

30 **Reserved**

29 **CmptCtrl**

Format:	MBZ

Compaction Control Indicates whether the instruction is compacted to the 64-bit compact instruction format. When this bit is set, the 64-bit compact instruction format is used. The EU decodes the compact format using lookup tables internal to the hardware, but documented for use by software tools. Only some instruction variations can be compacted, the variations supported by those lookup tables and the compact format. See EU Compact Instruction Format for more information.

Value	Name	Description
0	NoCompaction **[Default]**	No compaction. 128-bit native instruction supporting all instruction options.
1	Compacted	Compaction is enabled. 64-bit compact instruction supporting only some instruction variations.

28 **PredInv**

This field, together with PredCtrl, enables and controls the generation of the predication mask for the instruction. When it is set, the predication uses the inverse of the predication bits generated according to setting of Predicate Control. In other words, effect of PredInv happens after PredCtrl. This field is ignored by hardware if Predicate Control is set to 0000 - there is no predication. PMask is the final predication mask produced by the effects of both fields

Value	Name	Description
0	Positive **[Default]**	Positive polarity of predication. Use the predication mask produced by PredCtrl.
1	Negative	Negative polarity of predication. If PredCtrl is nonzero, invert the predication mask.

27:24 **PredCtrl**

Format:	PredCtrl

This field, together with PredInv, enables and controls the generation of the predication mask for the instruction. It allows per-channel conditional execution of the instruction based on the content of the selected flag register.

23 **FlagRegNum[0]**

This field specifies bit[0] of the register number for a flag register operand.

22 **FlagSubRegNum**

This field specifies the sub-register number for a flag register operand. There are two sub-registers in the flag register. Each sub-register contains 16 flag bits. The selected flag sub-register is the source for predication if predication is enabled for the instruction. It is the destination to store conditional flag bits if conditional modifier is enabled for the instruction. The same flag sub-register can be both the predication source and conditional destination, if

		both predication and conditional modifier are enabled.
	21:19	**ChanOff** Format: **ChanOff** This field provides offset information for ARF selection. The can be thought of as a starting channel offset for the execution mask and other ARF registers implicitly accessed.
	18:16	**ExecSize** Format: **ExecSize** This field determines the number of channels operating in parallel for this instruction. The size cannot exceed the maximum number of channels allowed for the given data type.
	15:0	**Header** Format: **Header**

Word Atomic Counter with Return Data Operation MSD

MSD1R_WAC - Word Atomic Counter with Return Data Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header

MSD1R_WAC - Word Atomic Counter with Return Data Operation MSD

	18:14	**Message Type** Default Value: 0Ch Format: Opcode Atomic Half Counter Operation message
	13	**Return Data Control** Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**Reserved** Access: RO Format: MBZ
	11:8	**Atomic Integer Operation** Format: MDC_AOP Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Word Atomic Counter Write Only Operation MSD

MSD1W_WAC - Word Atomic Counter Write Only Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHR Indicates that the message requires a header

MSD1W_WAC - Word Atomic Counter Write Only Operation MSD

	18:14	**Message Type** Default Value: 0Ch Format: Opcode Atomic Half Counter Operation message
	13	**Return Data Control** Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Reserved** Access: RO Format: MBZ
	11:8	**Atomic Integer Operation** Format: MDC_AOP Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Word Typed Atomic Integer with Return Data Operation MSD

MSD1R_WTAI - Word Typed Atomic Integer with Return Data Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.

MSD1R_WTAI - Word Typed Atomic Integer with Return Data Operation MSD

	18:14	**Message Type** Default Value: 07h Format: Opcode Typed Atomic Half Integer Operation message
	13	**Return Data Control** Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**Reserved** Access: RO Format: MBZ
	11:8	**Atomic Integer Operation** Format: MDC_AOP Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Word Typed Atomic Integer Write Only Operation MSD

MSD1W_WTAI - Word Typed Atomic Integer Write Only Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.

MSD1W_WTAI - Word Typed Atomic Integer Write Only Operation MSD

	18:14	**Message Type** Default Value: 07h Format: Opcode Typed Atomic Half Integer Operation message
	13	**Return Data Control** Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	**Reserved** Access: RO Format: MBZ
	11:8	**Atomic Integer Operation** Format: MDC_AOP Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS Specifies the Binding Table Index for the message

Word Untyped Atomic Float with Return Data Operation MSD

MSD1R_WAF - Word Untyped Atomic Float with Return Data Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header). Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.

MSD1R_WAF - Word Untyped Atomic Float with Return Data Operation MSD

	18:14	**Message Type** Default Value: 1Ch Format: Opcode Untyped Atomic Half Float Operation message
	13	**Return Data Control** Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2R Specifies the SIMD mode of the message (number of slots processed)
	11	**Reserved** Access: RO Format: MBZ
	10:8	**Atomic Float Operation** Format: MDC_FOP Specifies the atomic float operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Word Untyped Atomic Float Write Only Operation MSD

MSD1W_WAF - Word Untyped Atomic Float Write Only Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.

MSD1W_WAF - Word Untyped Atomic Float Write Only Operation MSD

	18:14	**Message Type** Default Value: 1Ch Format: Opcode Untyped Atomic Half Float Operation message
	13	**Return Data Control** Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2R Specifies the SIMD mode of the message (number of slots processed)
	11	**Reserved** Access: RO Format: MBZ
	10:8	**Atomic Float Operation** Format: MDC_FOP Specifies the atomic float operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Word Untyped Atomic Integer with Return Data Operation MSD

MSD1R_WAI - Word Untyped Atomic Integer with Return Data Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.

MSD1R_WAI - Word Untyped Atomic Integer with Return Data Operation MSD

	18:14	**Message Type** Default Value: 03h Format: Opcode Untyped Atomic Half Integer Operation message
	13	**Return Data Control** Default Value: 1h Format: Opcode Specifies that return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2R Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

Word Untyped Atomic Integer Write Only Operation MSD

MSD1W_WAI - Word Untyped Atomic Integer Write Only Operation MSD

Source: EuSubFunctionDataPort1

Length Bias: 1

DWord	Bit	Description
0	31	**Reserved** Access: RO Format: MBZ
	30	**Packed Data Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD data item in the source and writeback data payload is stored in GRF as a 32-bit value (8 per GRF), and smaller data items are zero padded to 32 bits. When set, each SIMD data item in the source and writeback data payload is stored in GRF as a 16-bit value (16 per GRF). **Restriction** Only 32-bit data packing is supported at this time.
	29	**Packed Address Payload** Default Value: 0 32 bit Format: Enable When clear, each SIMD address in the address payload is stored in GRF as a 32-bit value (8 per GRF). When set, each SIMD address in the address payload is stored in GRF as a 16-bit value (16 per GRF). Addresses in message header payloads are always stored as 32-bit values.
	28:25	**Message Length** Format: U4 Specifies the number of 256-bit GRF registers sent as the message payload (including the header).Valid value ranges are 1 to 15.
	24:20	**Response Length** Format: U5 Specifies the number of 256-bit GRF registers expected as the message response payload. Valid value ranges are 0 to 16.
	19	**Header Present** Format: MDC_MHP If set, indicates that the message includes the header.

MSD1W_WAI - Word Untyped Atomic Integer Write Only Operation MSD

	18:14	**Message Type** Default Value: 03h Format: Opcode Untyped Atomic Half Integer Operation message
	13	**Return Data Control** Default Value: 0h Format: Opcode Specifies that no return data is sent back to the thread.
	12	**SIMD Mode** Format: MDC_SM2R Specifies the SIMD mode of the message (number of slots processed)
	11:8	**Atomic Integer Operation** Format: MDC_AOP Specifies the atomic integer operation to be performed.
	7:0	**Binding Table Index** Format: MDC_BTS_SLM_A32 Specifies the Binding Table Index for the message

XY_BLOCK_COPY_BLT

XY_BLOCK_COPY_BLT

Source: BlitterCS

Length Bias: 2

Description

XY_BLOCK_COPY_BLT instruction performs a color source copy where the only operands involved are a color source and destination of the same bit width. The source and destination surfaces CAN overlap, the hardware handles this internally. This new blit command will happen in large numbers, consecutively, possibly an entire batch will comprise only new blit commands. Legacy commands and new blit command will not be interspersed. If they are, they will be separated by implied HW flush: Whenever there is a transition between this new Fast Blit command and the Legacy Blit commands (2D BLT instructions other than XY_BLOCK_COPY_BLT, XY_FAST_COPY_BLT, XY_FAST_COLOR_BLT), the HW will impose an automatic flush BEFORE the execution (at the beginning) of the next blitter command. The starting pixel of the blit operation for both source and destination should be on a pixel boundary.
Note that when two sequential block copy blits have different source surfaces, but their destinations refer to the same destination surfaces and therefore destinations overlap it is imperative that a Flush be inserted between the two blits.

DWord	Bit	Description
0	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Format: Opcode Value: 41h — Name: INSTRUCTION_TARGET_XY_BLOCK_COPY_BLT **[Default]**
	21:19	**Color Depth** This field actually programs bits per pixel value for each pixel of the surface. Reprogramming of these bits require explicit flushing of Copy Engine. 000b: 8 bit color **[Default]** 001b: 16 bit color 010b: 32 bit color 011b: 64 bit color 100b: 96 bit color (only linear case is supported) 101b: 128 bit color 110b: RESERVED 111b: RESERVED

		Programming Notes Color depth programming for 96 bit color is invalid unless both the source and destination surfaces are linear.
	18:14	**Reserved** Access: RO Format: MBZ
	13:9	**Reserved** Access: RO Format: MBZ
	8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Format: =n **Description** n = 10 This field indicates length of the instruction in DWORD. **Value** 10 — **Name** Excludes DWORD 0,1 **[Default]**
1	31:30	**Destination Tiling** These bits indicate destination tiling method. 00b — LINEAR **[Default]** — Linear mode (no tiling) 01b — YMAJOR — Y major tiling
	29:28	**Reserved** Access: RO Format: MBZ
	27:21	**Destination MOCS value** MOCS (Memory Object State Control) for destination operand.
	20:18	**Reserved** Access: RO Format: MBZ
	17:0	**Destination Pitch** Format: U18-1 For **Linear Surfaces**, the pitch must be multiple of pixel width in bytes. For Tiled surfaces, the pitch has to be a **multiple of the Tile Width** (X direction width of the Tile) expressed in dwords

		(4 bytes).
2	31:16	**Destination Y1 Coordinate (Top)** 16-bit signed number. The destination start line (inclusive) for Block Copy blit.
	15:0	**Destination X1 Coordinate (Left)** 16-bit signed number. The destination start pixel (inclusive) for Block Copy blit.
3	31:16	**Destination Y2 Coordinate (Bottom)** 16-bit signed number. The destination end line (exclusive) for Block Copy blit.
	15:0	**Destination X2 Coordinate (Right)** 16-bit signed number. The destination end pixel (exclusive) for Block Copy blit.
4..5	63:0	**Destination Base Address** Format: GraphicsAddress[63:0] **Description** This bitfield contains the base address of the destination surface. When Tiling is enabled this address is cacheline (64Byte) aligned. When Destination Tiling is disabled, this address is byte aligned.
6	31	**Reserved** Access: RO Format: MBZ
	30	**Reserved** Access: RO Format: MBZ
	29:16	**Destination Y offset** **Description** Format is U14. This field specifies the vertical offset in lines from the Surface Base Address to the start (origin) of the surface. **Programming Notes** For Linear surface Y offset must be 0. For Tiled surface Y offset must be greater than or equal to 0 and less than 16K in lines.
	15:14	**Reserved** Access: RO Format: MBZ
	13:0	**Destination X offset** **Description** Format is U14.

This field specifies the horizontal offset in pixels from the surface base address to the start (origin) of the surface.

Programming Notes
For Linear surface X offset must be 0. For Tiled surface X offset must be greater than or equal to 0 and less than 16K in pixels.

Source Y1 Coordinate (Top)
Format is S16.
The source start line (inclusive) for the Block Copy blit.

Source X1 Coordinate (Left)
Format is S16.
Source start pixel (inclusive) for Block Copy blit.

Source Tiling

Description
These bits indicate source tiling method.

Value	Name	Description
00b	LINEAR **[Default]**	Linear Tiling (tiking disabled)
01b	YMAJOR	Y major tiling

Reserved

Access:	RO
Format:	MBZ

Source MOCS

Description
MOCS (Memory Object State Control) value for source operand.

Reserved

Access:	RO
Format:	MBZ

Source Pitch

Format:	U18-1

For **Linear surfaces**, the pitch must be multiple of pixel width in bytes. For tiled surfaces, the pitch has to be multiple of the Tile width (X direction width of the tile) expressed in dwords (4 byte).

Source Base Address

Format:	GraphicsAddress[63:0]

This bitfield contains the base address of the source surface. When Tiling is enabled this

		address is cacheline (64Byte) aligned. When Source Tiling is disabled, this address is byte aligned.
11	31	**Reserved** Access: RO Format: MBZ
	30	**Reserved** Access: RO Format: MBZ
	29:16	**Source Y offset** Format U14. This field specifies the vertical offset in lines from the Surface Base Address to the start (origin) of the surface. **Programming Notes** For Linear surface Y offset must be 0. For Tiled surface Y offset must be greater than or equal to 0 and less than 16K in lines.
	15:14	**Reserved** Access: RO Format: MBZ
	13:0	**Source X offset** Format U14. This field specifies the horizontal offset in pixels from Surface Base Address to the start (origin) of the surface. **Programming Notes** For Linear surface X offset must be 0. For Tiled surface X offset must be greater than or equal to 0 and less than 16K in pixels.

XY_COLOR_BLT

XY_COLOR_BLT

Source: BlitterCS

Length Bias: 2

COLOR_BLT is the simplest BLT operation. It performs a color fill to the destination (with a possible ROP). The only operand is the destination operand which is written dependent on the raster operation. The solid pattern color is stored in the pattern background register.
This instruction is optimized to run at the maximum memory write bandwidth.
The typical (and fastest) Raster operation code = F0 which performs a copy of the pattern background register to the destination.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 50h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. Value / Name: 1xb – Write Alpha Channel x1b – Write RGB Channel
	19:12	**Reserved** Access: RO Format: MBZ
	11	**Tiling Enable** Value / Name / Description: 0b – Tiling Disabled (Linear Blit) 1b – Tiling Enabled – Tile-X or Tile-Y
	10:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 05h Format: =n
1 BR13	31	**Reserved** Access: RO

XY_COLOR_BLT

		Format: MBZ
	30	**Clipping Enabled** Value: 0b — Name: Disabled Value: 1b — Name: Enabled
	29:26	**Reserved** Access: RO Format: MBZ
	25:24	**Color Depth** 00b — 8 Bit Color 01b — 16 Bit Color(565) 10b — 16 Bit Color(1555) 11b — 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4..5	63:0	**Destination Base Address** Format: VIRTUAL_ADDR[63:0] This address must be Cache Line (64Byte) aligned for all surface types (linear or tiled).
6 BR16	31:0	**Solid Pattern Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]

XY_FAST_COLOR_BLT

XY_FAST_COLOR_BLT

Source: BlitterCS

Length Bias: 2

Description

XY_FAST_COLOR_BLT instruction performs a color blit where the only operands involved are an in-line color source and destination surface of the same bit width. This new blit command will happen in large numbers, consecutively, possibly an entire batch will comprise only new blit commands Legacy commands and new blit command will not be interspersed. If they are, they will be separated by implied HW flush: Whenever there is a transition between this new Fast Blit commands and the Legacy Blit commands (2D BLT instructions other than XY_FAST_COPY_BLT, XY_BLOCK_COPY_BLT, XY_FAST_COLOR_BLT), the HW will impose an automatic flush BEFORE the execution (at the beginning) of the next blitter command. The starting pixel of Fast Color blit for destination should be on pixel boundary.

Note that when two sequential fast copy blits have different source surfaces, but their destinations refer to the same destination surfaces and therefore destinations overlap it is imperative that a Flush be inserted between the two blits.

DWord	Bit	Description
0	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Format: Opcode Value 44h: XY_FAST_COLOR_BLT **[Default]**
	21:19	**Color Depth** Description: This field actually programs bits per pixel value for each pixel of the surface. Reprogramming these bits require explicit flush of Copy Engine.

Value	Name
000b	8 bit color **[Default]**
001b	16 bit color
010b	32 bit color
011b	64 bit color
100b	96 bit color (only supported for linear case)
101b	128 bit color
110b	RESERVED

XY_FAST_COLOR_BLT

		111b RESERVED
	18:14	**Reserved** Access: RO Format: MBZ
	13:9	**Reserved** Access: RO Format: MBZ
	8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Format: =n Total Length - 2 **Value** / **Name** 09h: Excludes DWORD 0,1 **[Default]**
1	31:30	**Destination Tiling** These bits indicate destination tiling method. **Value** / **Name** / **Description** 01b / YMAJOR / Y major tiling
	29:28	**Reserved** Access: RO Format: MBZ
	27:21	**Destination MOCS value** MOCS (Memory Object State Control) for destination operand.
	20:18	**Reserved** Access: RO Format: MBZ
	17:0	**Destination Pitch** Format: U18-1 For **Linear Surfaces**, the pitch must be multiple of pixel width in bytes. For Tiled surfaces, the pitch has to be a **multiple of the Tile Width** (X direction width of the Tile) expressed in dwords (4 bytes).
2	31:16	**Destination Y1 Coordinate (Top)** The destination start line (inclusive) for Fast Color blit. Format is 16-bit signed number.
	15:0	**Destination X1 Coordinate (Left)** The destination start pixel (inclusive) for Fast Color blit.

		Format is 16-bit signed number.
3	31:16	**Destination Y2 Coordinate (Bottom)** The destination end line (exclusive) for Fast Color blit. Format is 16-bit signed number.
	15:0	**Destination X2 Coordinate (Right)** The destination end pixel (exclusive) for Fast Color blit. Format is 16-bit signed number.
4..5	63:0	**Destination Base Address** Format: GraphicsAddress[63:0] **Description** This bitfield contains the base address of the destination surface. When Tiling is enabled this address is cacheline (64Byte) aligned. When Destination Tiling is disabled, this address is byte aligned.
6	31	**Reserved** Access: RO Format: MBZ
	30	**Reserved** Access: RO Format: MBZ
	29:16	**Destination Y offset** **Description** Format is U14. This field specifies the vertical offset in lines from the Surface Base Address to the start (origin) of the surface. **Programming Notes** For Linear surface Y offset must be 0. For Tiled surface Y offset must be greater than or equal to 0 and less than 16K in lines.
	15:14	**Reserved** Access: RO Format: MBZ
	13:0	**Destination X offset** **Description** Format is U14. This field specifies the horizontal offset in pixels from the Surface Base Address to the start (origin) of the surface.

XY_FAST_COLOR_BLT		
		Programming Notes For Linear surface X offset must be 0. For Tiled surface X offset must be greater than or equal to 0 and less than 16K in pixels.
7..10	127:0	**Fill Color** **Description** The Dwords contains Color data to use for the fill operation. Format depends on the color depth selected and is always packed little endian way starting with DW[7][0]. 8 bit Color Format: DW[7][7:0] 16 bit Color Format: DW[7][15:0] 32 bit Color Format: DW[7][31:0] 64 bit Color Format: DW[8], DW[7] 96 bit Color Format: DW[9], DW[8], DW[7] 128 bit Color Format: DW[10], DW[9], DW[8], DW[7]

XY_FAST_COPY_BLT

XY_FAST_COPY_BLT

Source: BlitterCS

Length Bias: 2

Description

This BLT instruction performs a color source copy where the only operands involved are a color source and destination of the same bit width. Note that this command does not support Clipping operations. This new blit command will happen in large numbers, consecutively, possibly an entire batch will comprise only new blit commands Legacy commands and new blit command will not be interspersed. If they are, they will be separated by implied HW flush: Whenever there is a transition between this new Fast Blit command and the Legacy Blit commands (2D BLT instructions other than XY_BLOCK_COPY_BLT, XY_FAST_COPY_BLT and XY_FAST_COLOR_BLT), the HW will impose an automatic flush BEFORE the execution (at the beginning) of the next blitter command. New blit command can use any combination of memory surface type - linear, tiledX, tiledY, and the tiling information is conveyed as part of the new Fast Copy command. The Fast Copy Blit supports the new 64KB Tiling. The starting pixel of Fast Copy blit for both source and destination should be on an OWord boundary.

Note that when two sequential fast copy blits have different source surfaces, but their destinations refer to the same destination surfaces and therefore destinations overlap it is imperative that a Flush be inserted between the two blits.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 42h Format: Opcode
	21:20	**Source Tiling Method** SW is required to flush the HW before changing the polarity of these bits for subsequent blits. Value / Name / Description 00b / Linear (Tiling Disabled) / 10b / YMAJOR / Choosing between 'Legacy Tile-Y' or the 'Tile4' can be done in DWord 1, Bit[31]. 11b / Tile64 /
	19:15	**Reserved** Access: RO Format: MBZ
	14:13	**Destination Tiling Method**

SW is required to flush the HW before changing the polarity of these bits for subsequent blits.

Value	Name	Description
00b	Linear (Tiling Disabled)	
10b	YMAJOR	Choosing between 'Legacy Tile-Y' or the 'Tile4' can be done in DWord 1, Bit[30].
11b	Tile64	

12:8 Reserved

Access:	RO
Format:	MBZ

7:0 DWord Length

Default Value:	08h Excludes DWORD 0,1
Format:	=n

08h

1 BR13 — 31 Tile Y Type for Source

Source being Tile-Y can be selected in DWord 0, Bit[21:20].

Value	Name
0b	Legacy Tile-Y
1b	Tile4

30 Tile Y Type for Destination

Destination being Tile-Y can be selected in DWord 0, Bit[14:13].

Value	Name
0b	Legacy Tile-Y
1b	Tile4

29:28 Reserved

Access:	RO
Format:	MBZ

27 Reserved

Access:	RO
Format:	MBZ

26:24 Color Depth

Value	Name	Programming Notes
000b	8 bit color	
001b	16 bit color (565)	
010b	RESERVED	Programming of 010b is not supported.
011b	32 bit color	

		100b: 64 bit color (for 64KB Tiling) 101b: 128 bit color (for 64KB Tiling)
	23:16	**Reserved**
	15:0	**Destination Pitch** Format: U16
2 BR22	31:16	**Destination Y1 Coordinate (Top)** Format: S15 Destination start line (inclusive) for Fast Copy blit.
	15:0	**Destination X1 Coordinate (Left)** Format: S15 Destination start pixel (inclusive) for Fast Copy blit. It should be on an OWord boundary.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** Format: S15 Destination end line (exclusive) for Fast Copy blit.
	15:0	**Destination X2 Coordinate (Right)** Format: S15 Destination end pixel (exclusive) for Fast Copy blit.
4..5	63:0	**Destination Base Address** Format: VIRTUAL_ADDR[63:0] This address must be Cache Line (64Byte) aligned for all surface types (linear or tiled).
6 BR26	31:16	**Source Y1 Coordinate (Top)** Format: S15 Source start line (inclusive) for Fast Copy blit.
	15:0	**Source X1 Coordinate (Left)** Source start pixel (inclusive) for Fast Copy blit. It should be on an OWord boundary.
7 BR11	31:16	**Reserved** Format: MBZ
	15:0	**Source Pitch** Format: U16
8..9	63:0	**Source Base Address** Format: VIRTUAL_ADDR[63:0] This address must be Cache Line (64Byte) aligned for all surface types (linear or tiled).

XY_FULL_BLT

XY_FULL_BLT

Source: BlitterCS

Length Bias: 2

The full BLT is the most comprehensive BLT instruction. It provides the ability to specify all 3 operands: destination, source, and pattern. The source and pattern operands are the same bit width as the destination operand.

The source and destination operands may overlap, which means that the X and Y directions can be either forward or backwards. The BLT Engine takes care of all situations. The base addresses plus the X and Y coordinates determine if there is an overlap between the source and destination operands. If the base addresses of the source and destination are the same and the Source X1 is less than Destination X1, then the BLT Engine performs the accesses in the X-backwards access pattern. There is no need to look for an actual overlap. If the base addresses are the same and Source Y1 is less than Destination Y1, then the scan line accesses start at Destination Y2 with the corresponding source scan line and the strides are subtracted for every scan line access. All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.

The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 55h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. Value / Name: 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:16	**Reserved** Access: RO Format: MBZ
	15	**Src Tiling Enable** Value / Name / Description: 0b Tiling Disabled (Linear Blit) 1b Tiling Enabled — Tile-X or Tile-Y.
	14:12	**Pattern Horizontal Seed**

		Pixel of the scan line to start on corresponding to DST X=0.
	11	**Dest Tiling Enable** Value / Name / Description 0b / Tiling Disabled (Linear Blit) / 1b / Tiling Enabled / Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Starting scan line of the 8x8 pattern corresponding to DST Y=0.
	7:0	**DWord Length** Default Value: 0Ah
1 BR13	31	**Reserved** Access: RO Format: MBZ
	30	**Clipping Enabled** Value / Name 0b / Disabled 1b / Enabled
	29:26	**Reserved** Access: RO Format: MBZ
	25:24	**Color Depth** Value / Name 00b / 8 Bit Color 01b / 16 Bit Color(565) 10b / 16 Bit Color(1555) 11b / 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4..5	63:0	**Destination Base Address**

		Format: VIRTUAL_ADDR[63:0] Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address must be CL (64byte) aligned.
6 BR11	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Source Pitch (double word aligned and signed) and in DWords** 2's complement. For Tiled Src (bit 15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be up to 128Kbytes (or 32KDwords).
7 BR26	31:16	**Source Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Source X1 Coordinate (Left)** 16 bit signed number.
8..9	63:0	**Source Address** Format: VIRTUAL_ADDR[63:0] Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address must be CL (64byte) aligned.
10..11	63:0	**Pattern Base Address** Format: VIRTUAL_ADDR[63:0] Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address must be CL (64byte) aligned.

XY_FULL_IMMEDIATE_PATTERN_BLT

XY_FULL_IMMEDIATE_PATTERN_BLT

Source: BlitterCS

Length Bias: 2

The full BLT is the most comprehensive BLT instruction. It provides the ability to specify all 3 operands: destination, source, and pattern. The source and immediate pattern operands are the same bit width as the destination operand. The immediate data sizes are 64 bytes (16 DWs), 128 bytes (32 DWs), or 256 (64 DWs) for 8, 16, and 32 bpp color patterns. DWL indicates the total number of Dwords of immediate data.

The source and destination operands may overlap, which means that the X and Y directions can be either forward or backwards. The BLT Engine takes care of all situations. The base addresses plus the X and Y coordinates determine if there is an overlap between the source and destination operands. If the base addresses of the source and destination are the same and the Source X1 is less than Destination X1, then the BLT Engine performs the accesses in the X-backwards access pattern. There is no need to look for an actual overlap. If the base addresses are the same and Source Y1 is less than Destination Y1, then the scan line accesses start at Destination Y2 with the corresponding source scan line and the strides are subtracted for every scan line access.

All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.

The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 74h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:16	**Reserved** Access: RO Format: MBZ
	15	**Src Tiling Enable** 0b Tiling Disabled (Linear) 1b Tiling Enabled — Tile-X or Tile-Y.

XY_FULL_IMMEDIATE_PATTERN_BLT

	14:12	**Pattern Horizontal Seed** (pixel of the scan line to start on corresponding to DST X=0)
	11	**Dest Tiling Enable** **Value** / **Name** / **Description** 0b / Tiling Disabled (Linear Blit) / 1b / Tiling Enabled / Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Starting scan line of the 8x8 pattern corresponding to DST Y=0.
	7:0	**DWord Length** Format: =n n = 08 + DWL = (where 'DWL' is Number of Immediate data in terms of double words). Immediate data size is 16 DW for 8 BPP, 32 DW for 16 BPP and 64 DW for 32 BPP. **Value** / **Name** [24,72] / Excludes DWORD 0,1
1 BR13	31	**Reserved** Access: RO Format: MBZ
	30	**Clipping Enabled** **Value** / **Name** 0b / Disabled 1b / Enabled
	29:26	**Reserved** Access: RO Format: MBZ
	25:24	**Color Depth** **Value** / **Name** 00b / 8 Bit Color 01b / 16 Bit Color(565) 10b / 16 Bit Color(1555) 11b / 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.

XY_FULL_IMMEDIATE_PATTERN_BLT

3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4..5	63:0	**Destination Base Address** Format: VIRTUAL_ADDR[63:0] Base address of the destination surface: X=0, Y=0. When Src Tiling is enabled (Bit_15 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
6 BR11	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Source Pitch (double word aligned and signed) and in DWords** 2's complement. For Tiled Src (bit 15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
7 BR26	31:16	**Source Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Source X1 Coordinate (Left)** 16 bit signed number.
8..9	63:0	**Source Address** Format: VIRTUAL_ADDR[63:0] Base address of the source surface: X=0, Y=0. When Src Tiling is enabled (Bit_15 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
10..n	31:0	**Immediate Data 0**

XY_FULL_MONO_PATTERN_BLT

XY_FULL_MONO_PATTERN_BLT

Source: BlitterCS

Length Bias: 2

The full BLT is the most comprehensive BLT instruction. It provides the ability to specify all 3 operands: destination, source, and pattern. The pattern operand is monochrome and the source operand is the same bit width as the destination operand.
The source and destination operands may overlap, which means that the X and Y directions can be either forward or backwards. The BLT Engine takes care of all situations. The base addresses plus the X and Y coordinates determine if there is an overlap between the source and destination operands. If the base addresses of the source and destination are the same and the Source X1 is less than Destination X1, then the BLT Engine performs the accesses in the X-backwards access pattern. There is no need to look for an actual overlap. If the base addresses are the same and Source Y1 is less than Destination Y1, then the scan line accesses start at Destination Y2 with the corresponding source scan line and the strides are subtracted for every scan line access.
The monochrome pattern transparency mode indicates whether to use the pattern background color or de-assert the write enables when the bit in the source is 0. When the source bit is 1, then the pattern foreground color is used in the ROP operation.
All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.
The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.
Setting both Solid Pattern Select =1 and Mono Pattern Transparency = 1 is mutually exclusive. The device implementation results in NO PIXELs DRAWN.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 57h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. Value / Name: 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:16	**Reserved** Access: RO Format: MBZ

XY_FULL_MONO_PATTERN_BLT

	15	**Src Tiling Enable** **Value** / **Name** / **Description** 0b / Tiling Disabled (Linear Blit) / 1b / Tiling Enabled / Tile-X or Tile-Y.
	14:12	**Pattern Horizontal Seed** (pixel of the scan line to start on corresponding to DST X=0)
	11	**Dest Tiling Enable** **Value** / **Name** / **Description** 0b / Tiling Disabled (Linear Blit) / 1b / Tiling Enabled / Tile-X or Tile-Y.
	10:8	**Pattern Vectical Seed** Starting scan line of the 8x8 pattern corresponding to DST Y=0.
	7:0	**DWord Length** **Value** / **Name** 0Ch /
1 BR13	31	**Solid Pattern Select** **Value** / **Name** 0 / No Solid Pattern 1 / Solid Pattern
	30	**Clipping Enabled** **Value** / **Name** 0b / Disabled 1b / Enabled
	29	**Reserved** Access: / RO Format: / MBZ
	28:27	**Mono Source Transparency Mode** **Value** / **Name** 0 / Use Background 1 / Transparency Enabled
	26	**Reserved** Access: / RO Format: / MBZ
	25:24	**Color Depth** **Value** / **Name** 00b / 8 Bit Color

XY_FULL_MONO_PATTERN_BLT

		01b: 16 Bit Color(565) 10b: 16 Bit Color(1555) 11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4..5	63:0	**Destination Base Address** Format: VIRTUAL_ADDR[63:0] Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_15 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
6 BR11	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Source Pitch (double word aligned and signed) and in DWords** 2's complement. For Tiled Src (bit 15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
7 BR26	31:16	**Source Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Source X1 Coordinate (Left)** 16 bit signed number.
8..9	63:0	**Source Address** Format: VIRTUAL_ADDR[63:0] Base address of the source surface: X=0, Y=0. When Src Tiling is enabled (Bit_15 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
10 BR16	31:0	**Pattern Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
11 BR17	31:0	**Pattern Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]

XY_FULL_MONO_PATTERN_BLT		
12 BR20	31:0	**Pattern Data 0** (least significant DW)
13 BR21	31:0	**Pattern Data 1** (most significant DW)

XY_FULL_MONO_PATTERN_MONO_SRC_BLT

XY_FULL_MONO_PATTERN_MONO_SRC_BLT

Source: BlitterCS

Length Bias: 2

The full BLT provides the ability to specify all 3 operands: destination, source, and pattern. The pattern and source operands are monochrome.
The monochrome source transparency mode indicates whether to use the source background color or de-assert the write enables when the bit in the source is 0. When the source bit is 1, then the source foreground color is used in the ROP operation.
All non-text monochrome sources are word aligned. At the end of a scan line the monochrome source, the remaining bits until the next word boundary are ignored. The Monochrome source data bit position field [2:0] indicates which bit position within the first byte should be used as the first source pixel which corresponds to the destination X1 coordinate.
The monochrome pattern transparency mode indicates whether to use the pattern background color or de-assert the write enables when the bit in the pattern is 0. When the source bit is 1, then the pattern foreground color is used in the ROP operation. The monochrome source transparency mode works identical to the pattern transparency mode.
All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.
The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.
Setting both Solid Pattern Select =1 and Mono Pattern Transparency = 1 is mutually exclusive. The device implementation results in NO PIXELs DRAWN.
Negative Stride (= Pitch) is NOT ALLOWED.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 58h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:17	**Monochrome source data bit position of the first pixel within a byte per scan line.**
	16:15	**Reserved**

XY_FULL_MONO_PATTERN_MONO_SRC_BLT

		Access: RO Format: MBZ
	14:12	**Pattern Horizontal Seed** (pixel of the scan line to start on corresponding to DST X=0)
	11	**Tiling Enable** Value: 0b — Name: Tiling Disabled (Linear Blit) Value: 1b — Name: Tiling Enabled — Description: Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Starting scan line of the 8x8 pattern corresponding to DST Y = 0.
	7:0	**DWord Length** Value: 0Ch
1 BR13	31	**Solid Pattern Select** Value: 0 — Name: No Solid Pattern Value: 1 — Name: Solid Pattern
	30	**Clipping Enabled** Value: 0b — Name: Disabled Value: 1b — Name: Enabled
	29	**Mono Source Transparency Mode** Value: 0 — Name: Use Background Value: 1 — Name: Transparency Enabled
	28	**Mono Pattern Transparency Mode** Value: 0 — Name: Use Background Value: 1 — Name: Transparency Enabled
	27:26	**Reserved** Access: RO Format: MBZ
	25:24	**Color Depth** Value: 00b — Name: 8 Bit Color

XY_FULL_MONO_PATTERN_MONO_SRC_BLT

		01b: 16 Bit Color(565) 10b: 16 Bit Color(1555) 11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4..5 This bitfield contains the base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Destination Base Address** Format: VIRTUAL_ADDR[63:0]
6..7 Address corresponds to DST X1, Y1. Note no NPO2 change here. The Mono Source Base Address must always be Cache Line (64byte) aligned.	63:0	**Mono Source Address** Format: VIRTUAL_ADDR[63:0]
8 BR18	31:0	**Source Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
9 BR19	31:0	**Source Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
10 BR16	31:0	**Pattern Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
11 BR17	31:0	**Pattern Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
12 BR20	31:0	**Pattern Data 0** (least significant DW)
13 BR21	31:0	**Pattern Data 1** (most significant DW)

XY_FULL_MONO_SRC_BLT

XY_FULL_MONO_SRC_BLT

Source: BlitterCS

Length Bias: 2

The full BLT is the most comprehensive BLT instruction. It provides the ability to specify all 3 operands: destination, source, and pattern. The source operand is monochrome and the pattern operand is the same bit width as the destination.
The monochrome source transparency mode indicates whether to use the source background color or de-assert the write enables when the bit in the source is 0. When the source bit is 1, then the source foreground color is used in the ROP operation.
All non-text and non-immediate monochrome sources are word aligned. At the end of a scan line the monochrome source, the remaining bits until the next word boundary are ignored. The Monochrome source data bit position field [2:0] indicates which bit position within the first byte should be used as the first source pixel which corresponds to the Destination X1 coordinate.
All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.
The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.
Negative Stride (= Pitch) is NOT ALLOWED

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 56h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:17	**Monochrome source data bit position of the first pixel within a byte per scan line.**
	16:15	**Reserved** Access: RO Format: MBZ
	14:12	**Pattern Horizontal Seed** (pixel of the scan line to start on corresponding to DST

XY_FULL_MONO_SRC_BLT

		X=0)
	11	**Tiling Enable** Value / Name / Description 0b / Tiling Disabled (Linear Blit) / 1b / Tiling Enabled / Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Starting scan line of the 8x8 pattern corresponding to DST Y = 0.
	7:0	**DWord Length** Value / Name 0Ah /
1 BR13	31	**Reserved** Access: RO Format: MBZ
	30	**Clipping Enabled** Value / Name 0b / Disabled 1b / Enabled
	29	**Mono Source Transparency Mode** Value / Name 0 / Use Background 1 / Transparency Enabled
	28:26	**Reserved** Access: RO Format: MBZ
	25:24	**Color Depth** Value / Name 00b / 8 Bit Color 01b / 16 Bit Color(565) 10b / 16 Bit Color(1555) 11b / 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2	31:16	**Destination Y1 Coordinate (Top)**

XY_FULL_MONO_SRC_BLT		
BR22		16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address must be CL (64byte) aligned.	63:0	**Destination Base Address** Format: VIRTUAL_ADDR[63:0]
6..7 Address corresponds to DST X1, Y1. Note no NPO2 change here. The Mono Source Base Address must always be Cache Line (64byte) aligned.	63:0	**Mono Source Address** Format: VIRTUAL_ADDR[63:0]
8 BR18	31:0	**Source Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
9 BR19	31:0	**Source Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
10..11 28:06 are implemented. Note no NPO2 change here. The pattern data must be located in linear memory. The programmed Pattern Base Address must always be Cache Line (64byte) aligned.	63:0	**Pattern Base Address** Format: VIRTUAL_ADDR[63:0]

XY_FULL_MONO_SRC_IMMEDIATE_PATTERN_BLT

XY_FULL_MONO_SRC_IMMEDIATE_PATTERN_BLT

Source: BlitterCS

Length Bias: 2

The full BLT is the most comprehensive BLT instruction. It provides the ability to specify all 3 operands: destination, source, and pattern. The source operand is a monochrome and the immediate pattern operand is the same bit width as the destination. The immediate data sizes are 64 bytes (16 DWs), 128 bytes (32 DWs), or 256 (64DWs) for 8, 16, and 32 bpp color patterns. The monochrome source transparency mode indicates whether to use the source background color or de-assert the write enables when the bit in the source is 0. When the source bit is 1, then the source foreground color is used in the ROP operation. All non-text monochrome sources are word aligned. At the end of a scan line the monochrome source, the remaining bits until the next word boundary are ignored. The Monochrome source data bit position field [2:0] indicates which bit position within the first byte should be used as the first source pixel which corresponds to the destination X1 coordinate. All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation. The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8. Negative Stride (= Pitch) is NOT ALLOWED.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 75h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:17	**Monochrome source data bit position of the first pixel within a byte per scan line.**
	16:15	**Reserved** Access: RO Format: MBZ
	14:12	**Pattern Horizontal Seed** (pixel of the scan line to start on corresponding to DST X=0)

XY_FULL_MONO_SRC_IMMEDIATE_PATTERN_BLT

	11	**Tiling Enable** Value / Name / Description: 0b – Tiling Disabled (Linear Blit) 1b – Tiling Enabled – Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Starting scan line of the 8x8 pattern corresponding to DST Y=0.
	7:0	**DWord Length** Format: =n n = 08 + DWL , (where 'DWL' is Number of Immediate data in terms of double words). Immediate data size is 16 DW for 8 BPP, 32 DW for 16 BPP and 64 DW for 32 BPP. Value / Name: [24,72] – Excludes DWORD 0,1
1 BR13	31	**Reserved** Access: RO Format: MBZ
	30	**Clipping Enabled** Value / Name: 0b – Disabled 1b – Enabled
	29	**Mono Source Transparency Mode** Value / Name: 0 – Use Background 1 – Transparency Enabled
	28:26	**Reserved** Access: RO Format: MBZ
	25:24	**Color Depth** Value / Name: 00b – 8 Bit Color 01b – 16 Bit Color(565) 10b – 16 Bit Color(1555) 11b – 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity

XY_FULL_MONO_SRC_IMMEDIATE_PATTERN_BLT		
		for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Destination Base Address** Format: VIRTUAL_ADDR[63:0]
6..7 Address corresponds to DST X1, Y1. Note no NPO2 change here. The Mono Source Base Address must always be Cache Line (64byte) aligned.	63:0	**Mono Source Address** Format: VIRTUAL_ADDR[63:0]
8 BR18	31:0	**Source Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
9 BR19	31:0	**Source Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
10..n	31:0	**Immediate Data**

XY_MONO_PAT_BLT

XY_MONO_PAT_BLT

Source: BlitterCS

Length Bias: 2

MONO_PAT_BLT is used when we have no source and the monochrome pattern is not trivial (is not a solid color only). The monochrome pattern is loaded from the instruction stream.
All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.
The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.
The monochrome pattern transparency mode indicates whether to use the pattern background color or de-assert the write enables when the bit in the pattern is 0. When the pattern bit is 1, then the pattern foreground color is used in the ROP operation.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 52h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:15	**Reserved** Access: RO Format: MBZ
	14:12	**Pattern Horizontal Seed** Pixel of the scan line to start on corresponding to DST X=0.
	11	**Tiling Enable** 0b Tiling Disabled (Linear Blit) 1b Tiling Enabled — Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed**

XY_MONO_PAT_BLT

		Scan line of the 8x8 pattern to start on corresponding to DST Y=0.
	7:0	**DWord Length** Value: 08h Name:
1 BR13	31	**Reserved** Access: RO Format: MBZ
	30	**Clipping Enabled** 0b: Disabled 1b: Enabled
	29	**Reserved** Access: RO Format: MBZ
	28	**Mono Pattern Transparency Mode** 0: Use Background 1: Transparency Enabled
	27:26	**Reserved** Access: RO Format: MBZ
	25:24	**Color Depth** 00b: 8 Bit Color 01b: 16 Bit Color(565) 10b: 16 Bit Color(1555) 11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.

XY_MONO_PAT_BLT		
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Destination Base Address** Format: VIRTUAL_ADDR[63:0]
6 BR16	31:0	**Pattern Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
7 BR17	31:0	**Pattern Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
8 BR20	31:0	**Pattern Data 0**
9 BR21	31:0	**Pattern Data 1**

XY_MONO_PAT_FIXED_BLT

XY_MONO_PAT_FIXED_BLT

Source: BlitterCS

Length Bias: 2

MONO_PAT_FIXED_BLT is used when we have no source and the monochrome pattern is not trivial (is not a solid color only). The monochrome pattern is one of 10 fixed patterns described below. The pattern seeds can still be used with the fixed patterns, creating even more fixed patterns. This eliminates 2 doublewords compared to the XY_MONO_PAT_BLT command packet.

All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.

The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.

The monochrome pattern transparency mode indicates whether to use the pattern background color or de-assert the write enables when the bit in the pattern is 0. When the pattern bit is 1, then the pattern foreground color is used in the ROP operation.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 59h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. Value / Name: 00b – **[Default]** 1xb – Write Alpha Channel x1b – Write RGB Channel
	19	**Reserved** Access: RO Format: MBZ
	18:15	**Fixed Pattern** Value / Name: 0000b – HS_HORIZONTAL 0001b – HS_VERTICAL 0010b – HS_FDIAGONAL 0011b – HS_BDIAGONAL

		0100b: HS_CROSS 0101b: HS_DIAGCROSS 0110b: Reserved 0111b: Reserved 1000b: Screen Door 1001b: SD Wide 1010b: Walking Bit (one) 1011b: Walking Zero 1100b: Reserved 1101b: Reserved 1110b: Reserved 1111b: Reserved
	14:12	**Pattern Horizontal Seed** Pixel of the scan line to start on corresponding to DST X=0.
	11	**Tiling Enable** Value: 0b — Name: Tiling Disabled (Linear Blit) Value: 1b — Name: Tiling Enabled — Description: Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Scan line of the 8x8 pattern to start on corresponding to DST Y=0.
	7:0	**DWord Length** Format: =n Value: 06h — Name:
1 BR13	31	**Reserved** Access: RO Format: MBZ
	30	**Clipping Enabled** Value: 0b — Name: Disabled Value: 1b — Name: Enabled
	29	**Reserved** Access: RO Format: MBZ
	28	**Mono Pattern Transparency Mode** Value — Name

XY_MONO_PAT_FIXED_BLT

		0 — Use Background 1 — Transparency Enabled
	27:26	**Reserved** Access: RO Format: MBZ
	25:24	**Color Depth** 00b — 8 Bit Color 01b — 16 Bit Color(565) 10b — 16 Bit Color(1555) 11b — 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4..5	63:0	**Destination Base Address** Format: VIRTUAL_ADDR[63:0] Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address should be CL (64byte) aligned.
6 BR16	31:0	**Pattern Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
7 BR17	31:0	**Pattern Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]

XY_MONO_SRC_COPY_BLT

XY_MONO_SRC_COPY_BLT

Source: BlitterCS

Length Bias: 2

This BLT instruction performs a monochrome source copy where the only operands involved is a monochrome source and destination. The source and destination operands cannot overlap therefore the X and Y directions are always forward.

All non-text monochrome sources are word aligned. At the end of a scan line of monochrome source, all bits until the next word boundary are ignored. The monochrome source data bit position field [2:0] indicates the bit position within the first byte of the scan line that should be used as the first source pixel which corresponds to the destination X1 coordinate.

The monochrome source transparency mode indicates whether to use the source background color or de-assert the write enables when the bit in the source is 0. When the source bit is 1, then the source foreground color is used in the ROP operation. The ROP value chosen must involve source and no pattern data in the ROP operation. Negative Stride (= Pitch) is NOT ALLOWED.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 54h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:17	**Monochrome source data bit position of the first pixel within a byte per scan line.**
	16:12	**Reserved** Access: RO Format: MBZ
	11	**Tiling Enable** 0b Tiling Disabled (Linear Blit) 1b Tiling Enabled — Tile-X or Tile-Y.
	10:8	**Reserved**

XY_MONO_SRC_COPY_BLT

		Access: RO Format: MBZ
	7:0	**DWord Length** Value: 08h
1 BR13	31	**Reserved** Access: RO Format: MBZ
	30	**Clipping Enabled** 0b: Disabled 1b: Enabled
	29	**Mono Source Transparency Mode** 0: Use Background 1: Transparency Enabled
	28:26	**Reserved** Access: RO Format: MBZ
	25:24	**Color Depth** 00b: 8 Bit Color 01b: 16 Bit Color(565) 10b: 16 Bit Color(1555) 11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)**

XY_MONO_SRC_COPY_BLT		
		16 bit signed number.
4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Destination Base Address** Format: VIRTUAL_ADDR[63:0]
6..7 Address corresponds to DST X1, Y1. Note no NPO2 change here. The Mono Source Base Address must always be Cache Line (64byte) aligned.	63:0	**Mono Source Address** Format: VIRTUAL_ADDR[63:0]
8 BR18	31:0	**Source Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
9 BR19	31:0	**Source Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]

XY_MONO_SRC_COPY_IMMEDIATE_BLT

XY_MONO_SRC_COPY_IMMEDIATE_BLT

Source: BlitterCS

Length Bias: 2

This instruction allows the Driver to send monochrome data through the instruction stream, eliminating the read latency of the source during command execution.
The IMMEDIATE_BLT data MUST transfer an even number of doublewords and the exact number of quadwords. DWL indicates the total number of Dwords of immediate data.
All non-text monochrome sources are word aligned. At the end of a scan line of monochrome source, all bits until the next word boundary are ignored. The Monochrome source data bit position field [2:0] indicates the bit position within the first byte of the scan line that should be used as the first source pixel which corresponds to the destination X1 coordinate.
The monochrome source transparency mode indicates whether to use the source background color or de-assert the write enables when the bit in the source is 0. When the source bit is 1, then the source foreground color is used in the ROP operation. The ROP value chosen must involve source and no pattern data in the ROP operation.
The monochrome source data supplied corresponds to the Destination X1 and Y1 coordinates.
Negative Stride (= Pitch) is NOT ALLOWED.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 71h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. Value / Name: 00b [Default] 1xb Write Alpha Channel x1b Write RGB Channel
	19:17	**Monochrome source data bit position of the first pixel within a byte per scan line.**
	16:12	**Reserved** Access: RO Format: MBZ
	11	**Tiling Enable** Value / Name / Description: 0b Tiling Disabled (Linear) 1b Tiling Enabled Tile-X or Tile-Y.

XY_MONO_SRC_COPY_IMMEDIATE_BLT

	10:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Format: =n n = 06 + DWL Where DWL is number of immediate data in terms of dwords. Immediate data size is 16 DW for 8 BPP, 32 DW for 16 BPP and 64 DW for 32 BPP. [22,70] Excludes DWORD 0,1
1 BR13	31	**Reserved** Access: RO Format: MBZ
	30	**Clipping Enabled** 0b Disabled 1b Enabled
	29	**Mono Source Transparency Mode** 0b Transparency Enabled 1b Use Background
	28:26	**Reserved** Access: RO Format: MBZ
	25:24	**Color Depth** 00b 8 Bit Color 01b 16 Bit Color(565) 10b 16 Bit Color(1555) 11b 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.

XY_MONO_SRC_COPY_IMMEDIATE_BLT		
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Destination Base Address** Format: VIRTUAL_ADDR[63:0]
6 BR18	31:0	**Source Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
7 BR19	31:0	**Source Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0]
8..n	31:0	**Immediate Data**

XY_PAT_BLT_IMMEDIATE

XY_PAT_BLT_IMMEDIATE

Source: BlitterCS

Length Bias: 2

PAT_BLT_IMMEDIATE is used when there is no source and the color pattern is not trivial (is not a solid color only) and the pattern is pulled through the command stream. The immediate data sizes are 64 bytes (16 DWs), 128 bytes (32 DWs), or 256 (64DWs) for 8, 16, and 32 bpp color patterns.
DWL indicates the total number of Dwords of immediate data. All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.
The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 72h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:15	**Reserved** Access: RO Format: MBZ
	14:12	**Pattern Horizontal Seed** Pixel of the scan line to start on corresponding to DST X=0.
	11	**Tiling Enable** 0b Tiling Disabled (Linear Blit) 1b Tiling Enabled — Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Scan line of the 8x8 pattern to start on corresponding to DST Y=0.

XY_PAT_BLT_IMMEDIATE

	7:0	**DWord Length** Default Value: [20,68] Excludes DWORD 0,1 Format: =n n = 04 + DWL Where DWL indicates number of immediate data in terms of dwords.
1 BR13	31	**Reserved** Access: RO Format: MBZ
	30	**Clipping Enabled** 0b Disabled 1b Enabled
	29:26	**Reserved** Access: RO Format: MBZ
	25:24	**Color Depth** 00b 8 Bit Color 01b 16 Bit Color(565) 10b 16 Bit Color(1555) 11b 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement (Negative Pitch Not allowed for Pixel nor Text) For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4..5 Base address of the destination surface:	63:0	**Destination Base Address** Format: VIRTUAL_ADDR[63:0]

XY_PAT_BLT_IMMEDIATE		
X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.		
6..n	31:0	**Immediate Data**

XY_PAT_BLT

XY_PAT_BLT

Source: BlitterCS

Length Bias: 2

PAT_BLT is used when there is no source and the color pattern is not trivial (is not a solid color only).
If clipping is enabled, all scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation. The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 51h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:15	**Reserved** Access: RO Format: MBZ
	14:12	**Pattern Horizontal Seed** Pixel of the scan line to start on corresponding to DST X=0.
	11	**Tiling Enable** 0b Tiling Disabled (Linear Blit) 1b Tiling Enabled — Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Scan line of the 8x8 pattern to start on corresponding to DST Y=0.
	7:0	**DWord Length** Default Value: 06h
1	31	**Reserved**

XY_PAT_BLT

BR13		Access: RO Format: MBZ
	30	**Clipping Enabled** Value: 0b – Disabled Value: 1b – Enabled
	29:26	**Reserved** Access: RO Format: MBZ
	25:24	**Color Depth** Value: 00b – 8 Bit Color Value: 01b – 16 Bit Color(565) Value: 10b – 16 Bit Color(1555) Value: 11b – 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement (Negative Pitch Not allowed for Pixel nor Text) For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Destination Base Address** Format: VIRTUAL_ADDR[63:0]
6..7 28:06 are implemented. Note no NPO2 change here. The pattern data must be located in linear memory. The programmed Pattern Base Address must	63:0	**Pattern Base Address** Format: VIRTUAL_ADDR[63:0]

XY_PAT_BLT		
always be Cache Line (64byte) aligned.		

XY_PAT_CHROMA_BLT_IMMEDIATE

XY_PAT_CHROMA_BLT_IMMEDIATE

Source: BlitterCS

Length Bias: 2

PAT_BLT_IMMEDIATE is used when there is no source and the color pattern is not trivial (is not a solid color only) and the pattern is pulled through the command stream. The immediate data sizes are 64 bytes (16 DWs), 128 bytes (32 DWs), or 256 (64DWs) for 8, 16, and 32 bpp color patterns.
DWL indicates the total number of Dwords of immediate data. All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.
The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 77h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:17	**Transparency Range Mode** (chroma-key) - Dst Chroma-key modes ONLY (SRC ILLEGAL)
	16:15	**Reserved** Access: RO Format: MBZ
	14:12	**Pattern Horizontal Seed** Pixel of the scan line to start on corresponding to DST X=0.
	11	**Tiling Enable** 0b Tiling Disabled (Linear Blit)

XY_PAT_CHROMA_BLT_IMMEDIATE

		1b: Tiling Enabled — Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Scan line of the 8x8 pattern to start on corresponding to DST Y=0.
	7:0	**DWord Length** Default Value: [22,70] Excludes DWORD 0,1 Format: =n n = 06 + DWL Where DWL is immediate data pattern size in dwords.
1 BR13	31	**Reserved** Access: RO Format: MBZ
	30	**Clipping Enabled** 0b: Disabled 1b: Enabled
	29:26	**Reserved** Access: RO Format: MBZ
	25:24	**Color Depth** 00b: 8 Bit Color 01b: 16 Bit Color(565) 10b: 16 Bit Color(1555) 11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.

XY_PAT_CHROMA_BLT_IMMEDIATE		
4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Destination Base Address** Format: VIRTUAL_ADDR[63:0]
6 BR18	31:0	**Transparency Color Low** (Chroma-key Low = Pixel Greater or Equal)
7 BR19	31:0	**Transparency Color High** (Chroma-key High = Pixel Less or Equal)
8..n	31:0	**Immediate Data**

XY_PAT_CHROMA_BLT

XY_PAT_CHROMA_BLT

Source: BlitterCS

Length Bias: 2

PAT_BLT is used when there is no source and the color pattern is not trivial (is not a solid color only).
All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.
The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 76h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:17	**Transparency Range Mode** (chroma-key) - Dst Chroma-key modes ONLY (SRC ILLEGAL)
	16:15	**Reserved** Access: RO Format: MBZ
	14:12	**Pattern Horizontal Seed** Pixel of the scan line to start on corresponding to DST X=0.
	11	**Tiling Enable** 0b Tiling Disabled (Linear Blit) 1b Tiling Enabled — Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Scan line of the 8x8 pattern to start on corresponding to DST Y=0.
	7:0	**DWord Length** Default Value: 08h Excludes DWORD 0,1

XY_PAT_CHROMA_BLT

1 BR13	31	**Reserved** Access: RO Format: MBZ
	30	**Clipping Enabled** 0b: Disabled 1b: Enabled
	29:26	**Reserved** Access: RO Format: MBZ
	25:24	**Color Depth** 00b: 8 Bit Color 01b: 16 Bit Color(565) 10b: 16 Bit Color(1555) 11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement (Negative Pitch Not allowed for Pixel nor Text) For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4..5	63:0	**Destination Base Address** Format: VIRTUAL_ADDR[63:0] This address must be Cache Line (64Byte) aligned for all surface types (linear or tiled).
6..7	63:0	**Pattern Base Address** Format: VIRTUAL_ADDR[63:0] (28:06 are implemented) (Note no NPO2 change here). The pattern data must be located in linear memory. The Pattern Base Address must always be Cache Line (64byte) aligned.
8	31:0	**Transparency Color Low**

XY_PAT_CHROMA_BLT		
BR18		(Chroma-key Low = Pixel Greater or Equal)
9 BR19	31:0	**Transparency Color High** (Chroma-key High = Pixel Less or Equal)

XY_PIXEL_BLT

XY_PIXEL_BLT

Source: BlitterCS

Length Bias: 2

The Destination X coordinate and Destination Y coordinate is compared with the ClipRect registers. If it is within all 4 comparisons, then the pixel supplied in the XY_SETUP_BLT instruction is written with the raster operation to (Destination Y Address + (Destination Y coordinate * Destination pitch) + (Destination X coordinate * bytes per pixel)).
ROP field must specify pattern or fill with 0's or 1's. There is no source operand.
Negative Stride (= Pitch) specified in the Setup command is Not Allowed

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 24h Format: Opcode
	21:12	**Reserved** Access: RO Format: MBZ
	11	**Tiling Enable** Value / Name / Description 0b / Tiling Disabled (Linear Blit) / 1b / Tiling Enabled / Tile-X or Tile-Y.
	10:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 00h
1 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.

XY_SCANLINES_BLT

XY_SCANLINES_BLT

Source: BlitterCS

Length Bias: 2

All scan lines and pixels that fall within the ClipRect Y and X coordinates are written. Only pixels within the ClipRectX coordinates and the Destination X coordinates are written using the raster operation.
The Pattern Seeds correspond to Destination X = 0 (horizontal) and Y = 0 (vertical). The alignment is relative to the destination coordinates. The pixel of the pattern used / scan line is the (destination X coordinate + horizontal seed) modulo 8. The scan line of the pattern used is the (destination Y coordinate + vertical seed) modulo 8.
Solid pattern should use the XY_SETUP_MONO_PATTERN_SL_BLT instruction.
ROP field must specify pattern or fill with 0's or 1's. There is no source operand.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 25h Format: Opcode
	21:15	**Reserved** Access: RO Format: MBZ
	14:12	**Pattern Horizontal Seed** Pixel of the scan line to start on corresponding to DST X=0.
	11	**Tiling Enable** Value / Name / Description 0b / Tiling Disabled (Linear Blit) / 1b / Tiling Enabled / Tile-X or Tile-Y.
	10:8	**Pattern Vertical Seed** Scan line of the 8x8 pattern to start on corresponding to DST Y=0.
	7:0	**DWord Length** Default Value: 01h
1 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
2 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.

XY_SETUP_BLT

XY_SETUP_BLT

Source: BlitterCS

Length Bias: 2

This setup instruction supplies common setup information including clipping coordinates used by the XY commands: XY_PIXEL_BLT, XY_SCANLINE_BLT, XY_TEXT_BLT, and XY_TEXT_BLT_IMMEDIATE.
These are the only instructions that require that state be saved between instructions other than the Clipping parameters. There are 5 dedicated registers to contain the state for the 3 setup BLT instructions (XY_SETUP_BLT, XY_SETUP_MONO_PATTERN_SL_BLT, and XY_SETUP_CLIP_BLT. All other BLTs use a temporary version of these. The 5 double word registers are: DW1 (Setup Control), DW6 (Setup Foreground color), DW5 (Setup Background color), DW7 (Setup Pattern address), and DW4 (Setup Destination Base Address).

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 01h Format: Opcode
	21:20	**32 bpp Byte Mask** Value / Name 1xb: Write Alpha Channel x1b: Write RGB Channel
	19:12	**Reserved** Access: RO Format: MBZ
	11	**Tiling Enable** Value / Name 0b: Tiling Disabled (Linear Blit) 1b: Tiling Enabled (Tile-X or Tile-Y)
	10:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 08h
1 BR01	31	**Reserved** Access: RO Format: MBZ

XY_SETUP_BLT

	30	**Clipping Enabled** Value: 0b — Disabled Value: 1b — Enabled
	29	**Mono Source Transparency Mode** Value: 0b — Use Background Value: 1b — Transparency Enabled
	28:26	**Reserved** Access: RO Format: MBZ
	25:24	**Color Depth** Value: 00b — 8 Bit Color Value: 01b — 16 Bit Color(565) Value: 10b — 16 Bit Color(1555) Value: 11b — 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement (Negative Pitch Not allowed for Pixel nor Text) For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR24	31:16	**ClipRect Y1 Coordinate (Top)** (30:16 = 15 bit positive number)
	15:0	**ClipRect X1 Coordinate (Left)** (14:00 = 15 bit positive number)
3 BR25	31:16	**ClipRect Y2 Coordinate (Bottom)** (30:16 = 15 bit positive number)
	15:0	**ClipRect X2 Coordinate (Right)** (14:00 = 15 bit positive number)
4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Destination Base Address** Format: VIRTUAL_ADDR[63:0]
6 BR05	31:0	**Setup Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0] All

XY_SETUP_BLT		
7 BR06	31:0	**Setup Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0] (SLB and TB only)
8..9 28:06 are implemented. Note no NPO2 change here. The pattern data must be located in linear memory. The Setup Pattern Base Address for Color Pattern must always be Cache Line (64byte) aligned.	63:0	**Setup Pattern Base Address for Color Pattern** Format: VIRTUAL_ADDR[63:0]

XY_SETUP_CLIP_BLT

XY_SETUP_CLIP_BLT

Source: BlitterCS

Length Bias: 2

This command is used to only change the clip coordinate registers. These are the same clipping registers as the Setup clipping registers above.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 03h Format: Opcode
	21:12	**Reserved** Access: RO Format: MBZ
	11	**Tiling Enable** Value: 0b – Name: Tiling Disabled (Linear Blit) Value: 1b – Name: Tiling Enabled (Tile-X or Tile-Y
	10:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 01h
1 BR24	31:16	**ClipRect Y1 Coordinate (Top)** (30:16 = 15 bit positive number)
	15:0	**ClipRect X1 Coordinate (Left)** (14:00 = 15 bit positive number)
2 BR25	31:16	**ClipRect Y2 Coordinate (Bottom)** (30:16 = 15 bit positive number)
	15:0	**ClipRect X2 Coordinate (Right)** (14:00 = 15 bit positive number)

XY_SETUP_MONO_PATTERN_SL_BLT

XY_SETUP_MONO_PATTERN_SL_BLT

Source: BlitterCS

Length Bias: 2

This setup instruction supplies common setup information including clipping coordinates used exclusively with the following instruction: XY_SCANLINE_BLT (SLB) - 1 scan line of monochrome pattern and destination are the only operands allowed.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 11h Format: Opcode
	21:20	**32 bpp Byte Mask** 1xb: Write Alpha Channel x1b: Write RGB Channel
	19:12	**Reserved** Access: RO Format: MBZ
	11	**Tiling Enable** 0b: Tiling Disabled (Linear Blit) 1b: Tiling Enabled (Tile-X or Tile-Y
	10:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Default Value: 08h
1 BR01	31	**Solid Pattern Select** (SLB and Pixel only) 0: No Solid Pattern 1: Solid Pattern
	30	**Clipping Enabled**

XY_SETUP_MONO_PATTERN_SL_BLT

		<table><tr><th>Value</th><th>Name</th></tr><tr><td>0b</td><td>Disabled</td></tr><tr><td>1b</td><td>Enabled</td></tr></table>
	29	**Reserved** <table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>
	28	**Mono Pattern Transparency Mode** <table><tr><th>Value</th><th>Name</th></tr><tr><td>0b</td><td>Use Background</td></tr><tr><td>1b</td><td>Transparency Enabled</td></tr></table>
	27:26	**Reserved** <table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>
	25:24	**Color Depth** <table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>8 Bit Color</td></tr><tr><td>01b</td><td>16 Bit Color(565)</td></tr><tr><td>10b</td><td>16 Bit Color(1555)</td></tr><tr><td>11b</td><td>32 Bit Color</td></tr></table>
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement (Negative Pitch Not allowed for Pixel nor Text) For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR24	31:16	**ClipRect Y1 Coordinate (Top)** (30:16 = 15 bit positive number)
	15:0	**ClipRect X1 Coordinate (Left)** (14:00 = 15 bit positive number)
3 BR25	31:16	**ClipRect Y2 Coordinate (Bottom)** (30:16 = 15 bit positive number)
	15:0	**ClipRect X2 Coordinate (Right)** (14:00 = 15 bit positive number)
4..5 This bitfield contains the base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL	63:0	**Setup Destination Base Address** <table><tr><td>Format:</td><td>VIRTUAL_ADDR[63:0]</td></tr></table>

XY_SETUP_MONO_PATTERN_SL_BLT		
(64byte) aligned.		
6 BR05	31:0	**Setup Background Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0] All
7 BR06	31:0	**Setup Foreground Color** 8 bit = [7:0], 16 bit = [15:0], 32 bit = [31:0] (SLB and TB only)
8 BR20	31:0	**DW0 (least significant) for a Monochrome Pattern**
9 BR21	31:0	**DW1 (most significant) for a Monochrome Pattern**

XY_SRC_COPY_BLT

Source: BlitterCS

Length Bias: 2

This BLT instruction performs a color source copy where the only operands involved is a color source and destination of the same bit width.
The source and destination operands may overlap, which means that the X and Y directions can be either forward or backwards. The BLT Engine takes care of all situations. The base addresses plus the X and Y coordinates determine if there is an overlap between the source and destination operands. If the base addresses of the source and destination are the same and the Source X1 is less than Destination X1, then the BLT Engine performs the accesses in the X-backwards access pattern. There is no need to look for an actual overlap. If the base addresses are the same and Source Y1 is less than Destination Y1, then the scan line accesses start at Destination Y2 with the corresponding source scan line and the strides are subtracted for every scan line access. The ROP value chosen must involve source and no pattern data in the ROP operation.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 53h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:16	**Reserved** Access: RO Format: MBZ
	15	**Src Tiling Enable** 0b Tiling Disabled (Linear) 1b Tiling Enabled — Tile-X or Tile-Y.
	14:12	**Reserved** Access: RO Format: MBZ
	11	**Dest Tiling Enable**

XY_SRC_COPY_BLT

		<table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0b</td><td>Tiling Disabled (Linear Blit)</td><td></td></tr><tr><td>1b</td><td>Tiling Enabled</td><td>Tile-X or Tile-Y.</td></tr></table>
	10:8	**Reserved** <table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>
	7:0	**DWord Length** <table><tr><td>Format:</td><td>=n</td></tr></table> <table><tr><th>Value</th><th>Name</th></tr><tr><td>08h</td><td></td></tr></table>
1 BR13	31	**Reserved** <table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>
	30	**Clipping Enabled** <table><tr><th>Value</th><th>Name</th></tr><tr><td>0b</td><td>Disabled</td></tr><tr><td>1b</td><td>Enabled</td></tr></table>
	29:26	**Reserved** <table><tr><td>Access:</td><td>RO</td></tr><tr><td>Format:</td><td>MBZ</td></tr></table>
	25:24	**Color Depth** <table><tr><th>Value</th><th>Name</th></tr><tr><td>00b</td><td>8 Bit Color</td></tr><tr><td>01b</td><td>16 Bit Color(565)</td></tr><tr><td>10b</td><td>16 Bit Color(1555)</td></tr><tr><td>11b</td><td>32 Bit Color</td></tr></table>
	23:16	**Raster Operation** It identifies the bit-wise operations that needs to be performed. Details of bit-wise operations can be found **here**.
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)**

XY_SRC_COPY_BLT		
		16 bit signed number.
3 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Destination Base Address** Format: VIRTUAL_ADDR[63:0]
6 BR26	31:16	**Source Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Source X1 Coordinate (Left)** 16 bit signed number.
7 BR11	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Source Pitch (double word aligned) and in DWords** 2's complement. For Tiled Src (bit 15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Yand can be upto 128Kbytes (or 32KDwords).
8..9 Base address of the source surface: X=0, Y=0. When Src Tiling is enabled (Bit_15 enabled), this address must be 4KB-aligned. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Source Base Address** Format: VIRTUAL_ADDR[63:0]

XY_SRC_COPY_CHROMA_BLT

XY_SRC_COPY_CHROMA_BLT

Source: BlitterCS

Length Bias: 2

This BLT instruction performs a color source copy with chroma-keying where the only operands involved is a color source and destination of the same bit width.
The source and destination operands may overlap, which means that the X and Y directions can be either forward or backwards. The BLT Engine takes care of all situations. The base addresses plus the X and Y coordinates determine if there is an overlap between the source and destination operands. If the base addresses of the source and destination are the same and the Source X1 is less than Destination X1, then the BLT Engine performs the accesses in the X-backwards access pattern. There is no need to look for an actual overlap. If the base addresses are the same and Source Y1 is less than Destination Y1, then the scan line accesses start at Destination Y2 with the corresponding source scan line and the strides are subtracted for every scan line access. The ROP value chosen must involve source and no pattern data in the ROP operation.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 73h Format: Opcode
	21:20	**32bpp Byte Mask** This field is only used for 32bpp. 00b **[Default]** 1xb Write Alpha Channel x1b Write RGB Channel
	19:17	**Transparency Range Mode** (chroma-key)
	16	**Reserved** Access: RO Format: MBZ
	15	**Src Tiling Enable** 0b Tiling Disabled (Linear) 1b Tiling Enabled — Tile-X or Tile-Y.
	14:12	**Reserved** Access: RO

XY_SRC_COPY_CHROMA_BLT

		Format: MBZ
	11	**Dest Tiling Enable** 0b: Tiling Disabled (Linear Blit) 1b: Tiling Enabled — Tile-X or Tile-Y.
	10:8	**Reserved** Access: RO Format: MBZ
	7:0	**DWord Length** Value 0Ah: Name
1 BR13	31	**Reserved** Access: RO Format: MBZ
	30	**Clipping Enabled** 0b: Disabled 1b: Enabled
	29:26	**Reserved** Access: RO Format: MBZ
	25:24	**Color Depth** 00b: 8 Bit Color 01b: 16 Bit Color(565) 10b: 16 Bit Color(1555) 11b: 32 Bit Color
	23:16	**Raster Operation**
	15:0	**Destination Pitch in DWords** 2's complement For Tiled surfaces (bit_11 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Y and can be upto 128Kbytes (or 32KDwords).
2 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
3	31:16	**Destination Y2 Coordinate (Bottom)**

XY_SRC_COPY_CHROMA_BLT		
BR23		16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
4..5 Base address of the destination surface: X=0, Y=0. When Tiling is enabled (Bit_11 enabled), this address is limited to 4Kbytes. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Destination Base Address** Format: VIRTUAL_ADDR[63:0]
6 BR26	31:16	**Source Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Source X1 Coordinate (Left)** 16 bit signed number.
7 BR11	31:16	**Reserved** Access: RO Format: MBZ
	15:0	**Source Pitch (double word aligned) and in DWords** 2's complement. For Tiled Src (bit 15 enabled) this pitch is of 512Byte granularity for Tile-X, 128B granularity for Tile-Yand can be upto 128Kbytes (or 32KDwords).
8..9 Base address of the source surface: X=0, Y=0. When Src Tiling is enabled (Bit_15 enabled), this address must be 4KB-aligned. When Tiling is not enabled, this address is to be CL (64byte) aligned.	63:0	**Source Base Address** Format: VIRTUAL_ADDR[63:0]
10 BR18	31:0	**Transparency Color Low** (Chroma-key Low = Pixel Greater or Equal)
11 BR19	31:0	**Transparency Color High** (Chroma-key High = Pixel Less or Equal)

XY_TEXT_BLT

XY_TEXT_BLT

Source: BlitterCS

Length Bias: 2

All source scan lines and pixels that fall within the ClipRect Y and X coordinates are written. The source address corresponds to Destination X1 and Y1 coordinate.
Text is either bit or byte packed. Bit packed means that the next scan line starts 1 pixel after the end of the current scan line with no bit padding. Byte packed means that the next scan line starts on the first bit of the next byte boundary after the last bit of the current line.
Source expansion color registers are always in the SETUP_BLT.
Negative Stride (= Pitch) is NOT ALLOWED.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 26h Format: Opcode
	21:17	**Reserved** Access: RO Format: MBZ
	16	**Bit / Byte Packed** Byte packed is for the NT driver. Value: 0 — Name: Bit Value: 1 — Name: Byte
	15:12	**Reserved** Access: RO Format: MBZ
	11	**Tiling Enable** Value: 0b — Name: Tiling Disabled (Linear Blit) Value: 1b — Name: Tiling Enabled — Description: Tile-X or Tile-Y.
	10:8	**Reserved** Access: RO Format: MBZ

XY_TEXT_BLT		
	7:0	**DWord Length** Default Value: 03h
1 BR22	31:16	**Destination Y1 Coordinate (Top)** 16 bit signed number.
	15:0	**Destination X1 Coordinate (Left)** 16 bit signed number.
2 BR23	31:16	**Destination Y2 Coordinate (Bottom)** 16 bit signed number.
	15:0	**Destination X2 Coordinate (Right)** 16 bit signed number.
3..4 Address of the first byte on a scan line corresponding to source X1, Y1. Note no NPO2 change here. The source address must always be Cache Line (64byte) aligned.	63:0	**Source Address** Format: VIRTUAL_ADDR[63:0]

XY_TEXT_IMMEDIATE_BLT

XY_TEXT_IMMEDIATE_BLT

Source: BlitterCS

Length Bias: 2

This instruction allows the Driver to send data through the instruction stream that eliminates the read latency of reading a source from memory.
If an operand is in system cacheable memory and either small or only accessed once, it can be copied directly to the instruction stream versus to graphics accessible memory. The IMMEDIATE_BLT data MUST transfer an even number of doublewords.
The BLT engine will hang if it does not get an even number of doublewords. All source scan lines and pixels that fall within the ClipRect X and Y coordinates are written. The source data corresponds to Destination X1 and Y1 coordinate.
Source expansion color registers are always in the SETUP_BLT. NEGATIVE STRIDE (= PITCH) IS NOT ALLOWED.

DWord	Bit	Description
0 BR00	31:29	**Client** Default Value: 02h 2D Processor Format: Opcode
	28:22	**Instruction Target(Opcode)** Default Value: 31h Format: Opcode
	21:17	**Reserved** Access: RO Format: MBZ
	16	**Bit / Byte Packed** Byte packed is for the NT driver. Value: 0 — Name: Bit Value: 1 — Name: Byte
	15:12	**Reserved** Access: RO Format: MBZ
	11	**Tiling Enable** Value: 0b — Name: Tiling Disabled (Linear Blit) Value: 1b — Name: Tiling Enabled — Description: Tile-X or Tile-Y.
	10:8	**Reserved** Access: RO Format: MBZ

<table>
<tr><th colspan="3">XY_TEXT_IMMEDIATE_BLT</th></tr>
<tr><td></td><td>7:0</td><td>DWord Length<table><tr><td>Default Value:</td><td>[17,65] Excludes DWORD 0,1</td></tr><tr><td>Format:</td><td>=n</td></tr></table>n = 01 + DWL
Where DWL indicates size of indirect data in dwords.</td></tr>
<tr><td rowspan="2">1
BR22</td><td>31:16</td><td>Destination Y1 Coordinate (Top)
16 bit signed number.</td></tr>
<tr><td>15:0</td><td>Destination X1 Coordinate (Left)
16 bit signed number.</td></tr>
<tr><td rowspan="2">2
BR23</td><td>31:16</td><td>Destination Y2 Coordinate (Bottom)
16 bit signed number.</td></tr>
<tr><td>15:0</td><td>Destination X2 Coordinate (Right)
16 bit signed number.</td></tr>
<tr><td>3..n</td><td>31:0</td><td>Immediate Data</td></tr>
</table>

Intel® Iris® Xe and UHD Graphics Open Source

Programmer's Reference Manual

For the 2020-2021 11th Generation Intel Xeon®, Core™, Celeron®, Pentium® Gold Processors based on the "Tiger Lake" Platform

Volume 6: Memory Views

May 2023, Revision 2.0

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Revision History

Revision	Description
1.0	Initial Release
2.0	The following sections were added: • Memory Types and Cache Interface • Memory Object Control State (MOCS) • L3 Control Registers • Memory Interface Control Registers • Required PAT & MOCS Tables • LNCFCMOCSx • GLOB_MOCS_LECC_x

Table of Contents

Memory Views **1**
Introduction 1
Graphics Virtual Memory 3
Graphics Translation Tables 4
Memory Types and Cache Interface 12
Memory Object Control State (MOCS) 13
L3 Control Registers 15
Memory Interface Control Registers 16
Required PAT & MOCS Tables 20
Virtual Addressed TR Translation Tables **22**
Walk with 64KB Page 27
Walk with 2MB Page 28
Pointer to PML4 table 30
PML4E: Pointer to PDP Table 30
PDPE: Pointer to PD Table 31
PDPE for PD 31
PDPE for 1GB Page 32
PD: Pointer to Page Table 33
PDE for Page Table 33
PDE for 2MB Page 34
PTE: Page Table Entry for 64KB Page 35
PTE: Page Table Entry for 4KB Page 36
LNCFCMOCSx 38
GLOB_MOCS_LECC_x 41

Memory Views

Introduction

A modern GPU consists of multiple "engines", including Compute, Render (including Fixed Functions), Media Encode/Decode, Media Enhancement, Blitter/Copy, etc. Engines can operate concurrently and independently using different virtual address spaces.

All engines rely heavily on access to and from memory resources to perform their various functions. The memory subsystem connects engines to the memory resources, and provides services such as address translation, compression, caching, and HW virtualization. The memory subsystem is the heart of the GPU.

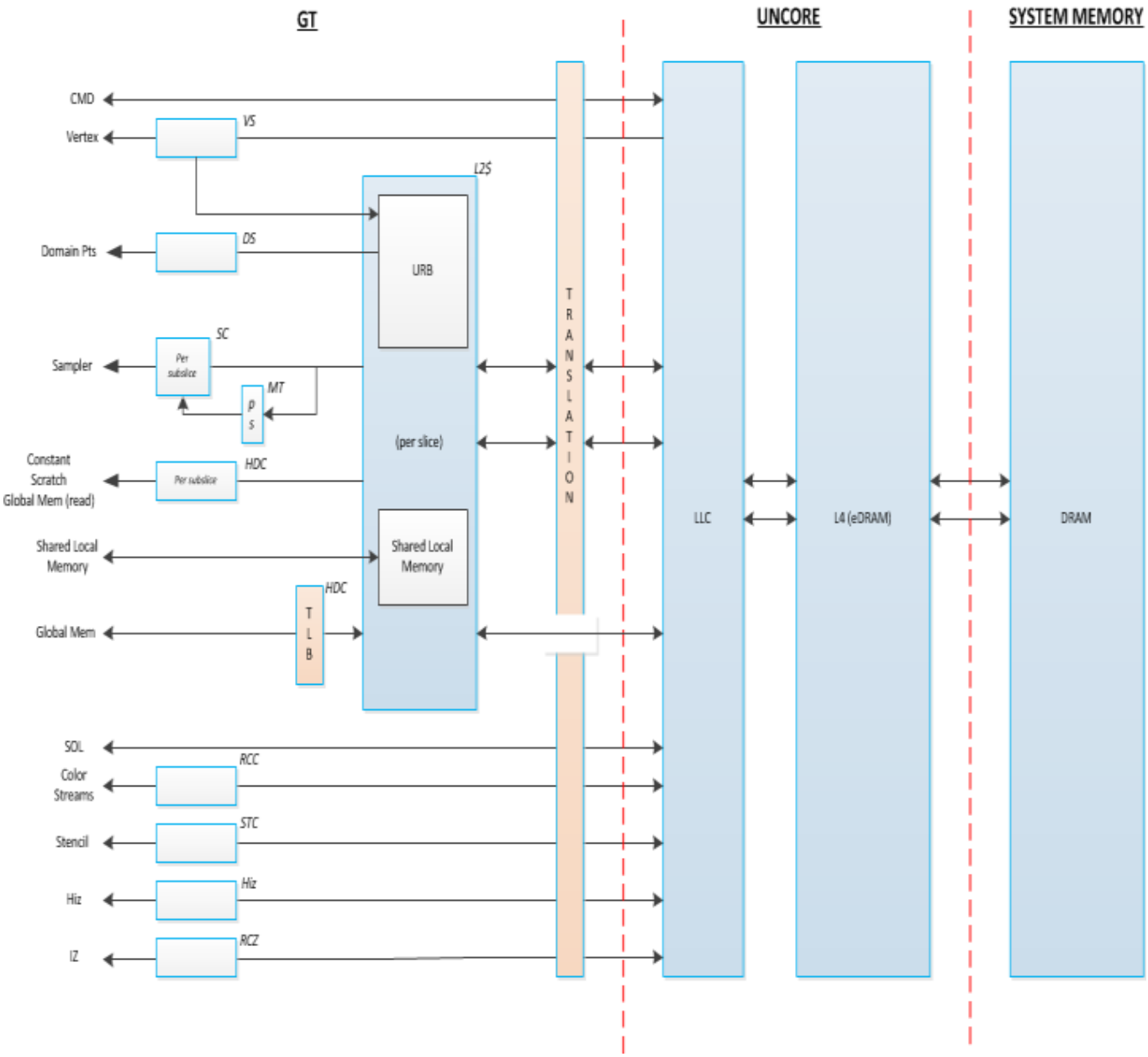
Key components of the memory subsystem include:

- Translation Services : Translation Lookaside Buffer (TLB) that translates virtual addresses associated with a context (process) running on an engine, to the physical address space of the GPU or System, and performs page table walks if a request misses in the TLB
- L3 Cache : Level 3 cache that is shared by all engines within the "GT" domain. Engines may have their own Level 1 and Level 2 caches that are not shared.
- LLC : Last Level Cache of the CPU host (only relevant for Integrated GPU)
- L4 Cache : Optional cache between L3 and Memory
- Compression : Handles lossless compression and decompression of memory objects
- System Memory : Memory that is physically attached to the CPU and managed entirely by the Operating System or Hypervisor
- Device Memory : Memory that is physically attached to a discrete GPU, or "stolen" from System Memory for an Integrated GPU, and managed entirely by the GPU device driver

The memory subsystem supports flows that are coherent with CPU memory (including CPU caches), as well as those that are not coherent with CPU memory. In general, non-coherent flows provide higher bandwidth more efficiently than coherent flows, but may require special handling by SW.

The following diagram provides an overview of a typical memory subsystem an Integrated GPU.

Cache and Memory Hierarchy



Graphics Virtual Memory

The GPU uses a virtual memory address space, where the graphics virtual address is mapped through a Page Table to a physical memory address. Normally, this mapping is set up by the graphics device driver and is private to the GPU context. However, in some cases the graphics virtual address is shared with the CPU - see for more information.

The range of valid graphics virtual addresses, and the types of page tables supported for address translation, varies with the GPU configuration. See the section for a summary the ranges and features supported by a specific graphics device.

Although the range of supported graphics virtual addresses varies, most GPU commands and GPU instructions use a common 64 bit definition for a graphics virtual address. Addresses outside of the supported range are reserved for future address space expansion. See the **GraphicsAddress** structure definition for specific details.

Some GPU devices support an extended graphics virtual memory address mapping called Tiled Resources. When enabled, the Tiled Resources Translation Table (TR-TT) pre-processes graphics virtual addresses. TR-TT maps a graphics virtual memory address either to a new graphics virtual memory address or to a Null Tile. Null Tiles return zero on reads and drop writes. For translations that are not Null Tiles, the new graphics virtual memory address is then used for the graphics virtual address and translated through the normal Page Table to generate a physical memory address.

Graphics Translation Tables

The GPU supports standard virtual memory models as defined by the IA programmer's guide. This section describes the different paging models, their behaviors, and the page table formats.

The Graphics Translation Tables (GTT) are memory-resident page tables containing an array of Page Translation Entries (PTEs) used in mapping graphics virtual addresses to physical memory addresses. There are two types of page tables: Global GTT and Per-Process GTT.

The base address of the GGTT and the PPGTT are programmed via the PGTBL_CTL and PGTBL_CTL2 MI registers, respectively. The translation table base addresses must be 4KB aligned. The GGTT size is 8MB, to cover 4GB of Global Virtual Address space, and is physically contiguous (ie, "flat"). The global GTT should only be programmed via the MMIO range within the GTTMMADR BAR. The PPGTT is programmed directly in memory and is multi-level. The page tables are further described in later sections.

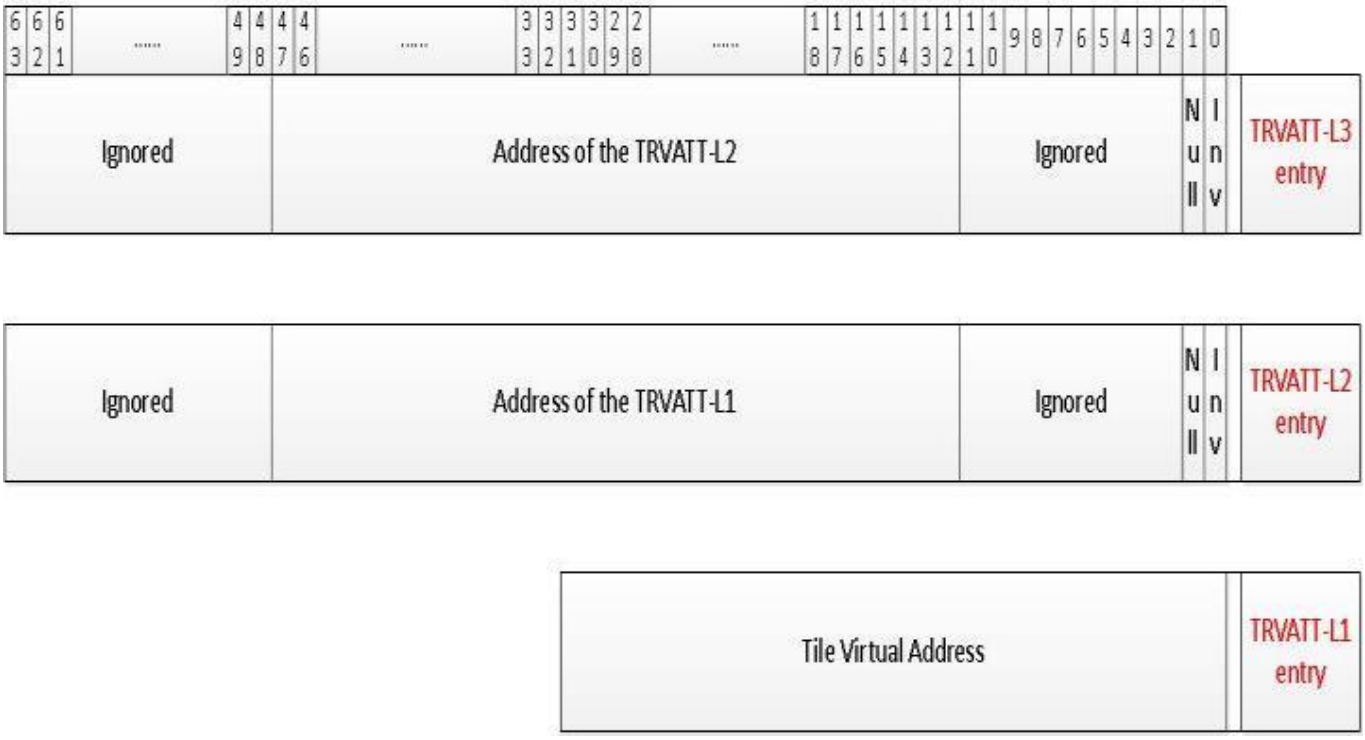
GFX Page Tables

This section describes the different types of address translation tables used by the GPU.

Tiled Resources Translation Tables

Sparse Tiled Resources can be thought of as a kind of application-controlled virtual memory scheme. The application allocates a resource in a virtual address space. Then the application tells the driver to map specified 64KB tiles within the surface to memory, within resources called Tile Pools. Tiles that are not mapped to a Tile Pool are null tiles.

Tiled Resource Translation Table (TRTT) is constructed as a 3 level tile Table. Each tile is 64KB in size which leaves behind 44-16=28 address bits. 28bits are partitioned as 9+9+10 which corresponds to TRVATT L3, L2 and L1 respectively. This is where TRVATT L3 has 512 entries, L2 has 512 entries and L1 has 1024 entries where each level is contained within a 4KB page hence L3 and L2 is composed of 64b entries and L1 is composed of 32b entries.



The contents of the TRVATT tables are as listed above where L3 and L2 points to the address of the next level which is a 4KB page and L1 contains the 32b VA address pointer needed to map the TR tile to virtual address space.

L1 Entry:

Bits	Field	Description
31:0	ADDR: Address	GFX virtual address of 64KB tile is referenced by this entry. This field is treated as GFX Virtual Address (GVA) when translated and maps to 47:16.

L2 Entry:

Bits	Field	Description
63:48	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
47:12	ADDR: Address	GFX virtual address or Guest Physical Address of 4KB base address pointing to TR-TT L1. *TR-TT table entries for L2 and L3 can be in GFX virtual address mode or Guest Physical address mode chosen by GFX software.*
11:2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	Null	Null Tile where reads to this tile returns zero with a Null indicator and writes are dropped.
0	Invalid	Invalid Tile where reads to this tile returns zero and writes are dropped. Additional interrupt is generated to GFX software when an invalid tile is accessed.

L3 Entry:

Bits	Field	Description
63:48	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
47:12	ADDR: Address	GFX virtual address or Guest Physical Address of 4KB base address pointing to TR-TT L2. *TR-TT table entries for L2 and L3 can be in GFX virtual address mode or Guest Physical address mode chosen by GFX software.*
11:2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	Null	Null Tile where reads to this tile returns zero with a Null indicator and writes are dropped.
0	Invalid	Invalid Tile where reads to this tile returns zero and writes are dropped. Additional interrupt is generated to GFX software when an invalid tile is accessed.

Programming Note	
Context:	Tiled ResourceTranslation Tables in Gfx Page Tables
GFX Driver has to disable the TR-TT bypass mode before using tiled resources translation tables. Details of the registers are given in "registers for TR-TT management."	

Programming Note	
Context:	Tiled ResourceTranslation Tables in Gfx Page Tables
GFX Driver is not allowed to put TR-TT entries into TR-VA space.	

Programming Note	
Context:	Tiled ResourceTranslation Tables in Gfx Page Tables
Usage model for TR translations are restricted to GFX Render Engine (& POSH pipeline).	

Programming Note	
Context:	Tiled ResourceTranslation Tables in Gfx Page Tables
TRTT is only for PPGTT64 (Advanced or Legacy PPGTT64). Enabling TRTT in Legacy PPGTT32 context or GGTT context is considered as invalid programming.	

Programming Note	
Context:	Tiled ResourceTranslation Tables in Gfx Page Tables
When partitioned address space based Dual Context is enabled, bit[47] of the virtual address in L3, L2 and L1 entries must be 0.	

Registers for TR-TT Management

Following register is a global mechanism to disable the bypass mode which is considered to be default for h/w. GFX driver has to set this bit to disable bypass mode before using TR-TTs.

Following registers shall be part of the h/w context.

Tiled Resources VA Translation Table L3 Pointer		
Register Space:		MMIO: 0/2/0
DWord	**Bit**	**Description**
1	63:48	**Reserved** **Access:** RO Reserved.
	47:32	**Tiled Resource - VA translation Table L3 Pointer (Upper Address)** **Default Value:** 0000h **Access:** R/W Upper address bits for tiled resource VA to virtual address translation L3 table. For physical memory option, address bits [47:39] has to be programmed to "0" as it is defined the limit of physical memory allocation.
0	31:16	**Tiled Resource - VA translation Table L3 Pointer (Lower Address)** **Default Value:** 0000h **Access:** R/W Lower address bits for tiled resource VA to virtual address translation L3 table.
	15:0	**Reserved** **Access:** RO Reserved.

Tiled Resources Null Tile Detection Register		
Register Space:		MMIO: 0/2/0
DWord	**Bit**	**Description**
	31:0	**Null Tile Detection Value** **Default Value:** 00000000h **Access:** R/W A 32bit value programmed to enable h/w to perform a match of TR-VA TT entries to detect Null Tiles. Hardware will flag each entry and space behind it as Null Tile for matched entries.

Tiled Resources Invalid Tile Detection Register		
Register Space:		MMIO: 0/2/0
DWord	**Bit**	**Description**
	31:0	**Invalid Tile Detection Value** **Default Value:** 00000000h **Access:** R/W A 32bit value programmed to enable h/w to perform a match of TR-VA TT entries to detect Invalid Tiles. Hardware will flag each entry and space behind it as Invalid Tile for matched entries.

Tiled Resources Virtual Address Detection Registers (TRVADR)		
Register Space:		MMIO: 0/2/0
DWord	**Bit**	**Description**
0	31:8	**Reserved** **Access:** RO Reserved.
	7:4	**TRVA Mask Value (TRVAMV)** **Default Value:** 0000b **Access:** R/W 4bit MASK value that is mapped to incoming address bits[47:44]. MASK bits are used to identify which address bits need to be considered for compare. If particular mask bit is "1", mapping address bit needs to be compared to DATA value provided. If "0", corresponding address bit is masked which makes it don't care for compare (*this field defaults to "0000" to disable detection*)

Tiled Resources Virtual Address Detection Registers (TRVADR)		
		Note that h/w supports two possible values for MASK: "0000" which is disabled case and "1111" where 44 bit TR-VA space is carved out.
	3:0	**TRVA Data Value (TRVADV)** **Default Value:** 0b **Access:** R/W 4bit DATA value that is mapped to incoming address bits[47:44]. Data bits are used to compare address values that are not filtered by the TRVAMV for match.

Tiled Resources Translation Table Control Register (TRTTE)		
Register Space:		MMIO: 0/2/0
DWord	**Bit**	**Description**
0	31:2	**Reserved** **Access:** RO Reserved.
	1	**TR-VA Translation Table Memory Location** **Default Value:** 0b **Access:** R/W This fields specifies whether the translation tables for TR-VA to VA are in virtual address space vs physical (GPA) address space. 0: Tables are in Physical (GPA) Space 1: Tables are in Virtual Address Space **Tiled Resource Translation Tables in GPA space is not supported in any generations. This mode should never be set as GPA mode (always set to '1). HW will set TRTT tables in Virtual address space mode only.**
	0	**TR-TT Enable** **Default Value:** 0b **Access:** R/W TR translation tables are disabled as default. This field needs to be enabled via s/w to get TR translation active.

The following register (0x4DFC[0]) has enable and disable control of the bypass path across TR translations. By default, bypass is enabled, and bypass needs to be disabled (by setting 0x4DFC[0] = '1) for TR translations to function. Disabling the bypass should be done before render power gating is enabled.

Detection and Treatment of Null and Invalid Tiles

Two types of definition that need to be extracted from TR-VA walk in addition to reaching the GFX virtual address.

1. **Null Tiles**: Null tiles provide the applications the of capability to preventing OS mapping the entire surface. When a memory access hits a Null tile, the access is terminated and zero's are returned to the originator of the memory access for loads along with a null indicator and for stores the access is dropped at the page walker level.
2. **Invalid Tiles**: This is the case where GFX software did not update the value of the mapping properly for hardware to separate resident vs null tiles. The Invalid Tile treatment is exactly same however additionally a unique interrupt is generated in h/w

Both detections are done by GPU:

- For L2/L3 entries, Null and Invalid tile information is already embedded in the TR-TT entries
- For L1 entries, the contents (32bits) are compared in hardware to pre-programmed values by GFX software (*values are provided in GFX MMIO space*). For the match values, two separate 32b registers are defined, one for Null Tile detection and one for Invalid Tile detection.

Hardware walking matching the value or detecting L2/L3 would terminate the walk (i.e. rest of the tables are not valid) and define the access as either Null or Invalid.

Programming Note	
Context:	Detection and treatment of null and invalid tiles.
The software is not allowed to program both Null and Invalid values to be the same.	

Programming Note	
Context:	TileX Surfaces and Null Tiles
NULL or Invalid Tiles are not supported on TileX surfaces.	

GPU implements a counter mechanism to roll-up the Null tile accesses detected. The counter value is exposed to GFX software via GFX MMIO.

In implementation, when the TR translation tables are in virtual address domain, the pages faults encountered while walking the IA32e pages are not reported back to the TR walkers or TLBs. These faults are handled as fault & halt, making these faults transparent to the TR walkers. However, when such a fault is not fixed (unsuccessful fault response) or when a non-recoverable fault encountered, main page walker HW converts the cycle to an invalid cycle. Thus, in this case, TR walker or TR TLBs will get incorrect read return data without any notification of the non-recoverable fault condition. Thus, TR walker/TLBs will continue with the TR-walk with incorrect data. This can lead to spurious cycles being generated. However, a Gfx reset/FLR is expected as a result of the non-recoverable fault.

TR-TT Modes

The L3 table pointer along with TRTTL3e/TRTTL2e is projected to support two modes of address space. Original intent was to have the contents to be in Virtual Address space (OS managed) and have them to be translated to GPA to HPA before getting accessed. Such mechanism will incur high latency penalties due to nested page translations. GPU shall have an additional mode where tiled-resources translation tables are in physical address space (GPA) and eliminate the need to have nested translations to reduce the potentially high miss latencies.

TR-TT walker shall have both modes supported. The Mode bit will be part of the same Register that provides TR-VA TT L3 pointer.

Memory Types and Cache Interface

This section has additional information on the types of memory which are accessible via the various GT mechanisms. It includes discussion on how the various paging models are used and accessed. See the Graphics Translation Tables for more detailed discussions on paging models.

This section also includes descriptions of how different surface types (MOCS) can be cached in the L3 and the different behaviors which can be enabled.

Memory Object Control State (MOCS)

The memory object control state defines the behavior of memory accesses beyond the graphics core, including encryption, graphics data types that allow selective flushing of data from outer caches, and controlling cacheability in the outer caches.

This control uses several mechanisms. Control state for all memory accesses can be defined page by page in the GTT entries. Memory objects that are defined by state per surface generally have additional memory object control state in the state structure that defines the other surface attributes. Memory objects without state defining them have memory object state control defined per class in the STATE_BASE_ADDRESS command, with class divisions the same as the base addresses. Finally, some memory objects only have the GTT entry mechanism for defining this control. The table below enumerates the memory objects and the location of the control state for each:

Memory Object	Location of Control State
surfaces defined by SURFACE_STATE: sampling engine surfaces, render targets, media surfaces, pull constant buffers, streamed vertex buffers	SURFACE_STATE
depth, stencil, and hierarchical depth buffers	corresponding state command that defined the buffer attributes
stateless buffers accessed by data port	STATE_BASE_ADDRESS
indirect state objects	STATE_BASE_ADDRESS
kernel instructions	STATE_BASE_ADDRESS
push constant buffers	3DSTATE_CONSTANT_(VS | GS | PS)
index buffers	3DSTATE_INDEX_BUFFER
vertex buffers	3DSTATE_VERTEX_BUFFERS
indirect media object	STATE_BASE_ADDRESS
generic state prefetch	GTT control only
ring/batch buffers	GTT control only
context save buffers	GTT control only
store DWord	GTT control only

MOCS Registers

These registers provide the detailed format of the MOCS table entries, that need to be programmed to define each surface state.

MEMORY_OBJECT_CONTROL_STATE

Size (in bits): 7

Default Value: 0x00000000

DWord	Bit	Description
0	6:1	**Index to MOCS Tables** The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime. **Programming Notes** When an access is made through Data Port and the index to MOCS[6:1] = [48,59] (decimal), that surface or stateless memory access can be cached in HDC L1 cache. Accesses made through Data Port with MOCS[6:1] < 48 or > 59 will bypass the HDC L1 cache. This bypass is useful when software wants to ensure that Data Port access are coherent with the L3 memory.
	0	**Reserved**

L3 Control Registers

64x16b control registers are defined within L3 space to interpret MOCS indexing and map it to cache events.

The incoming MOCS value is used to index into one of these registers which hardware uses as control parameters for a given surface. It allows 64 concurrent surface definitions with unique control values for L3 caching.

Also attached are the default settings for each 64 locations if driver chooses to use as is.

Following 16b defines per selection definition:

Register#64 (MOCS value 63) is reserved for h/w use and should not be used by s/w.

In L3 Node: B020-B09F (128 Bytes). Please refer to the register section for default values.

Bits	Description
16:6	Reserved.
5:4	**L3 Cacheability Control (L3CC).** Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls. 00b: Use binding table index for direct EU accesses - for rest it is reserved. 01b: Uncacheable (UC) - non-cacheable. 10b: Reserved 11b: Writeback (WB).
3:1	**Skip Caching Control (SCC).** Defines the bit values to enable caching. Outcome overrides the L3caching for the surface. If "0" - than corresponding address bit value is don't care. Bit[1]=1: Address bit[9] needs to be "0" to cache in target. Bit[2]=1: Address bit[10] needs to be "0" to cache in target. Bit[3]=1: Address bit[11] needs to be "0" to cache in target.
0	**Enable Skip Caching (ESC).** Enable for the Skip cache mechanism. 0: Not enabled. 1: Enabled for L3.

Memory Interface Control Registers

64x32b control registers are defined within the page walker where control parameters for LLC/eDRAM caching are defined. Incoming memory control object state index is used to do a look up into the table where the corresponding control parameters are picked for a given surface. These control values are used to control LLC/eDRAM caching.

For EU surfaces where binding table index is used, we also pass two bits of information in the hardware.

Following 32b defines per selection definition:

All MOCS registers are global and need to be saved/restored on RC6 entry/exit.

Certain MOCS indices are used by hardware and carry special meanings. These indices are restricted from being used for regular surfaces. In the following list, the indices are 0 based (i.e., MOCS index 0 is the first register and MOCS index 63 is the last register.)

- Index 'd63 is used for two purposes.
  - It is used by the L3 for all its evictions. The programming of the index 'd63 is expected to allow LLC cacheability to enable coherent flows to be maintained.
  - It is also used by hardware to force L3 uncacheable cycles. The programming of the index #63 is expected to make the surface L3 uncacheable.
- Index 'd62
  - This index is used for Tiled-Resources page walker accesses in previous projects.
  - This index is used for AuxTT Nodes for current projects.
- Index 'd61 is used for displayable surfaces. The programming of the index 'd61 is expected to disallow LLC cacheability for the surface to be displayable.
- Index 'd60 is reserved for use in the 3D CCS accesses.

Bits	Description
31:19	**Reserved**
18:17	**Self Snoop Enable** 00: Default value. Self snoop attribute sent to the uncore is as normal - determined by MIDI unit logic 01: Override the self snoop bit generated by MIDI with 0. No self snoops are sent to the uncore for any transactions from this surface 11: Override the self snoop bit generated by MIDI with 1. Self snoops are always sent to the uncore for any transactions from this surface
16:15	**Class of Service** This field controls the Class of Service sent to the LLC to determine which sub-set of Ways the surface will be stored in. The allocation of certain LLC ways to different class of service settings is a project dependent decision and listed in the Bspec. 00: Value from Private PAT Registers(40E0/40E4/40E8/40EC) 01: Class 1 10: Class 2

Bits	Description
	11: Class 3
14	**Snoop Control Field (SCF)**: Enables s/w to have GFX h/w to be able to consume IA generated buffers that are tagged as WB. Driver can mark these buffers as WB when generating them from IA. In LP-SOCs, the fabric is not forced to be coherent all the time. IA-core generated WB buffers can only be consumed by GPU if that buffer is tagged as snoop-able in GPUs buffer definitions (or via GPU Page tables). 1: Non-Coherent Write/Read 0: Coherent Access
13:11	**Page Faulting Mode** This fields controls the page faulting mode that will be used in the memory interface block for the given request coming from this surface: 000: Use the global page faulting mode from context descriptor (default) 001-111: Reserved
10:8	**Skip Caching Control** Defines the bit values to enable caching. Outcome overrides the LLC caching for the surface. If "0" - than corresponding address bit value is do not care Bit[8]=1: address bit[9] needs to be "0" to cache in target Bit[9]=1: address bit[10] needs to be "0" to cache in target Bit[10]=1: address bit[11] needs to be "0" to cache in target The default value of this field is '000. **For coherent surfaces, skip caching should not be enabled, as not caching in LLC breaks the coherency.**
7	**Enable Reverse Skip Caching** Enable for the Skip cache mechanism 0: Not enabled 1: Enabled for LLC
6	**Don't Allocate on miss** Controls defined for RO surfaces in mind, where if the target cache is missed - do not bring the line (applicable to LLC/eDRAM). 0: Allocate on MISS (normal cache behavior) 1: Do NOT allocate on MISS
5:4	**LRU (Cache Replacement) Management (LRUM).** This field allows the selection of AGE parameter for a given surface in LLC or eLLC. If a particular allocation is done at youngest age ("3") it tends to stay longer in the cache as compared to older age allocations ("2", "1",

Bits	Description
	or "0"). This option is given to driver to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches. 00: Take the age value from Uncore CRs. 01: Assign the age of "0" 10: Don't change the age on a hit. 11: Assign the age of "3"
3:2	**Target Cache (TC).** This field allows the choice of LLC vs eLLC for caching. 00b: Use TC/LRU controls from page table 01b: LLC Only. 10b: LLC/eLLC Allowed. 11b: LLC/eLLC Allowed. **For coherent surfaces ensure that LLC caching is enabled - even when using target cache controls from page table.**
1:0	**LLC/eDRAM Cacheability Control (LeCC).** Memory type information used in LLC/eDRAM. 00b: Use Cacheability Controls from page table / UC with Fence (if coherent cycle). 01b: Uncacheable (UC) - non-cacheable. 10b: Writethrough (WT). 11b: Writeback (WB). **Note:** In case of SVM (advanced context), LLC/eDRAM memory type is used based on the page table controls and cannot be managed via MOCS index.

Defaults Table

Default	LeCC	TC	LRUM	AOM	ESC	SCC	PFM	Default	LeCC	TC	LRUM	AOM	ESC	SCC	PFM
000000	00	00	11	0	0	00	000	000000	00	00	11	0	0	00	000
000001	00	01	11	0	0	00	000	000001	00	01	11	0	0	00	000
000010	00	10	11	0	0	00	000	000010	00	10	11	0	0	00	000
000011	01	00	11	0	0	00	000	000011	01	00	11	0	0	00	000
000100	10	00	11	0	0	00	000	000100	10	00	11	0	0	00	000
000101	10	01	11	0	0	00	000	000101	10	01	11	0	0	00	000
000110	10	10	11	0	0	00	000	000110	10	10	11	0	0	00	000
000111	11	00	11	0	0	00	000	000111	11	00	11	0	0	00	000
001000	11	01	11	0	0	00	000	001000	11	01	11	0	0	00	000
001001	11	10	11	0	0	00	000	001001	11	10	11	0	0	00	000
001010	10	00	11	0	0	00	000	001010	10	00	11	0	0	00	000
001011	10	01	11	0	0	00	000	001011	10	01	11	0	0	00	000
001100	10	10	11	0	0	00	000	001100	10	10	11	0	0	00	000
001101	11	00	11	0	0	00	000	001101	11	00	11	0	0	00	000
001110	11	01	11	0	0	00	000	001110	11	01	11	0	0	00	000
001111	11	10	11	0	0	00	000	001111	11	10	11	0	0	00	000
010000	00	00	11	0	0	00	000	010000	00	00	11	0	0	00	000
010001	00	01	11	0	0	00	000	010001	00	01	11	0	0	00	000
010010	00	10	11	0	0	00	000	010010	00	10	11	0	0	00	000
010011	01	00	11	0	0	00	000	010011	01	00	11	0	0	00	000
010100	10	00	11	0	0	00	000	010100	10	00	11	0	0	00	000
010101	10	01	11	0	0	00	000	010101	10	01	11	0	0	00	000
010110	10	10	11	0	0	00	000	010110	10	10	11	0	0	00	000
010111	11	00	11	0	0	00	000	010111	11	00	11	0	0	00	000
011000	11	01	11	0	0	00	000	011000	11	01	11	0	0	00	000
011001	11	10	11	0	0	00	000	011001	11	10	11	0	0	00	000
011010	10	00	11	0	0	00	000	011010	10	00	11	0	0	00	000
011011	10	01	11	0	0	00	000	011011	10	01	11	0	0	00	000
011100	10	10	11	0	0	00	000	011100	10	10	11	0	0	00	000
011101	11	00	11	0	0	00	000	011101	11	00	11	0	0	00	000
011110	11	01	11	0	0	00	000	011110	11	01	11	0	0	00	000
011111	11	10	11	0	0	00	000	011111	11	10	11	0	0	00	000

Required PAT & MOCS Tables

Rather than hard-wire the available PAT and MOCS Table entries, the tables are software programmable. But to reduce virtualization overhead, Intel requires that all drivers use the table values specified in this section. This minimizes virtualization overhead while still allowing the flexibility of post-silicon/post-launch improvements from new performance findings and updated tables.

To ensure software backward and forward compatibility, table entries are version-numbered, where

- An entry version indicates the version of the table in which it was introduced.
- Each platform starts with version 1 entries.
- Entries are only ever added (never modified or removed—except to correct documentation/etc. errors, if the change is backward-compatible).
- Added entries use previously highest version number, incremented by one.
- Multiple entries added at same time share same version number.
- Spec'ed entries never change versions—Version increases are only for addition of new slots.

Should table versions > 1 be necessary on a given platform, graphics drivers will be able to query to determine which table version is available on their current (potentially virtualized) platform. When new drivers find themselves on a platform with older tables, they should remap their new/unavailable entries back to the most appropriate matches in the available tables.

PAT:

Version	Group	Use	PAT Index	PAT_INDEX MEM_TYPE
1	LLC	WB	0	3
1		WC	1	1
1		WT	2	2
1	UC	UC	3	0

MOCS:

Version	Group	Use	MOCS Index	LNCFCMOCSx			GLOB_MOCS_LECC_x									
				ESC	SCC	L3CC	LeCC	TC	LRUM	DAoM	ERSC	SCC	PFM	SCF	CoS	SSE
1	Base	Error (Reserved for Non-Use)	0	0	0	3	3	1	3	0	0	0	0	0	0	0
1		Reserved	1													
1		L3 + LLC	2	0	0	3	3	1	3	0	0	0	0	0	0	0
1		Uncached	3	0	0	1	1	1	0	0	0	0	0	0	0	0
1		L3 (Read-Only*)	4	0	0	3	1	1	0	0	0	0	0	0	0	0
1		LLC	5	0	0	1	3	1	3	0	0	0	0	0	0	0
1	Age 0	LLC (Age 0)	6	0	0	1	3	1	1	0	0	0	0	0	0	0
1		L3 + LLC (Age 0)	7	0	0	3	3	1	1	0	0	0	0	0	0	0
1	Age: Don't Chg.	LLC (Age:DC)	8	0	0	1	3	1	2	0	0	0	0	0	0	0
1		L3 + LLC (Age:DC)	9	0	0	3	3	1	2	0	0	0	0	0	0	0
1	No AOM	LLC (No AOM)	10	0	0	1	3	1	3	1	0	0	0	0	0	0
1		L3 + LLC (No AOM)	11	0	0	3	3	1	3	1	0	0	0	0	0	0
1		LLC (No AOM; Age 0)	12	0	0	1	3	1	1	1	0	0	0	0	0	0
1		L3 + LLC (No AOM; Age 0)	13	0	0	3	3	1	1	1	0	0	0	0	0	0
1		LLC (No AOM; Age:DC)	14	0	0	1	3	1	2	1	0	0	0	0	0	0
1		L3 + LLC (No AOM; Age:DC)	15	0	0	3	3	1	2	1	0	0	0	0	0	0
1	Reserved	Reserved	16													
1		Reserved	17													
1	Self-Snoop	L3 + LLC (Self Snoop)	18	0	0	3	3	1	3	0	0	0	0	0	0	3
1	Skip Caching	L3 + LLC(12.5%)	19	0	0	3	3	1	3	0	0	7	0	0	0	0
1		L3 + LLC(25%)	20	0	0	3	3	1	3	0	0	3	0	0	0	0
1		L3 + LLC(50%)	21	0	0	3	3	1	3	0	0	1	0	0	0	0
1		L3 + LLC(75%)	22	0	0	3	3	1	3	0	1	3	0	0	0	0
1		L3 + LLC(87.5%)	23	0	0	3	3	1	3	0	1	7	0	0	0	0
1	Reserved	Reserved	24													
1		Reserved	25													
1	Special Indexes	HDC:L1 + L3 + LLC	**48**	0	0	3	3	1	3	0	0	0	0	0	0	0
1		HDC:L1 + L3	**49**	0	0	3	1	1	0	0	0	0	0	0	0	0
1		HDC:L1 + LLC	**50**	0	0	1	3	1	3	0	0	0	0	0	0	0
1		HDC:L1	**51**	0	0	1	1	1	0	0	0	0	0	0	0	0
1		Special Case (CCS)	**60**	0	0	1	3	1	3	0	0	0	0	0	0	0

1		Special Case (Displayable**)	**61**	0	0	3	1	1	0	0	0	0	0	0	0	0
1		HW Reserved—SW program but never use.	**62**	0	0	1	3	1	3	0	0	0	0	0	0	0
1		HW Reserved—SW program but never use.	**63**	0	0	1	3	1	3	0	0	0	0	0	0	0

HDC:L1 should only be used in read-only use-cases.

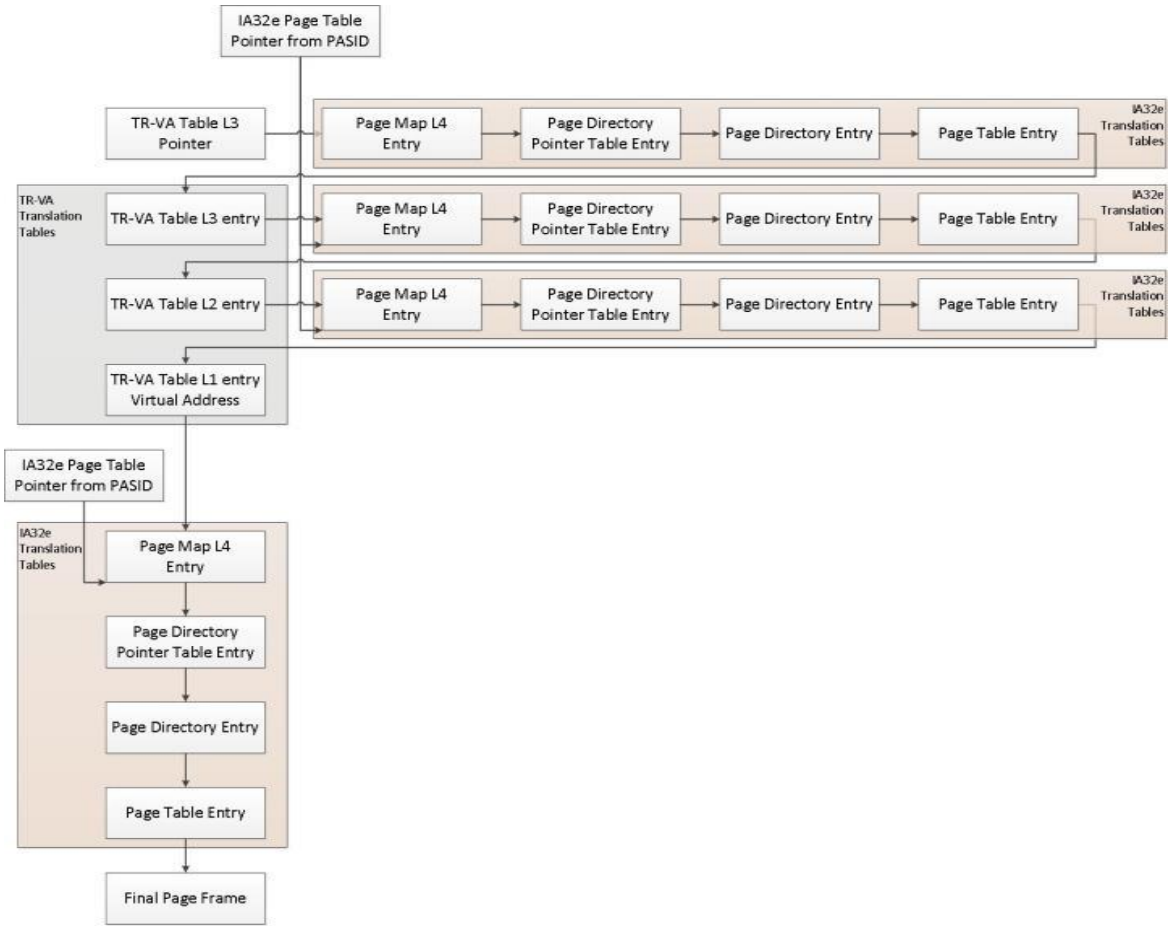
* Suitable solely for Read-Only use, since (written/dirty) L3 evictions essentially always go to LLC (via hard-wiring to MOCS#63), except for the MOCS#61 special-case.

** MOCS#61 is generally said to be for "Displayable" memory objects—though more precisely, it is for any memory object desiring writable L3 caching without LLC caching.

Virtual Addressed TR Translation Tables

Having sparse tiled resource translation tables in GFX virtual space requires the h/w TR-TT walker to walk thru the 1st level tile tables for table accesses to reach to Physical address at the L1 TR translation tables.

The following diagrams provide the view of the walk TR-VA translation tables are in physical memory and no 2nd Level (VTd) translations enabled.



Once 2nd level translations are enabled each level of 1st level walk needs to be further walked through VTd page tables.

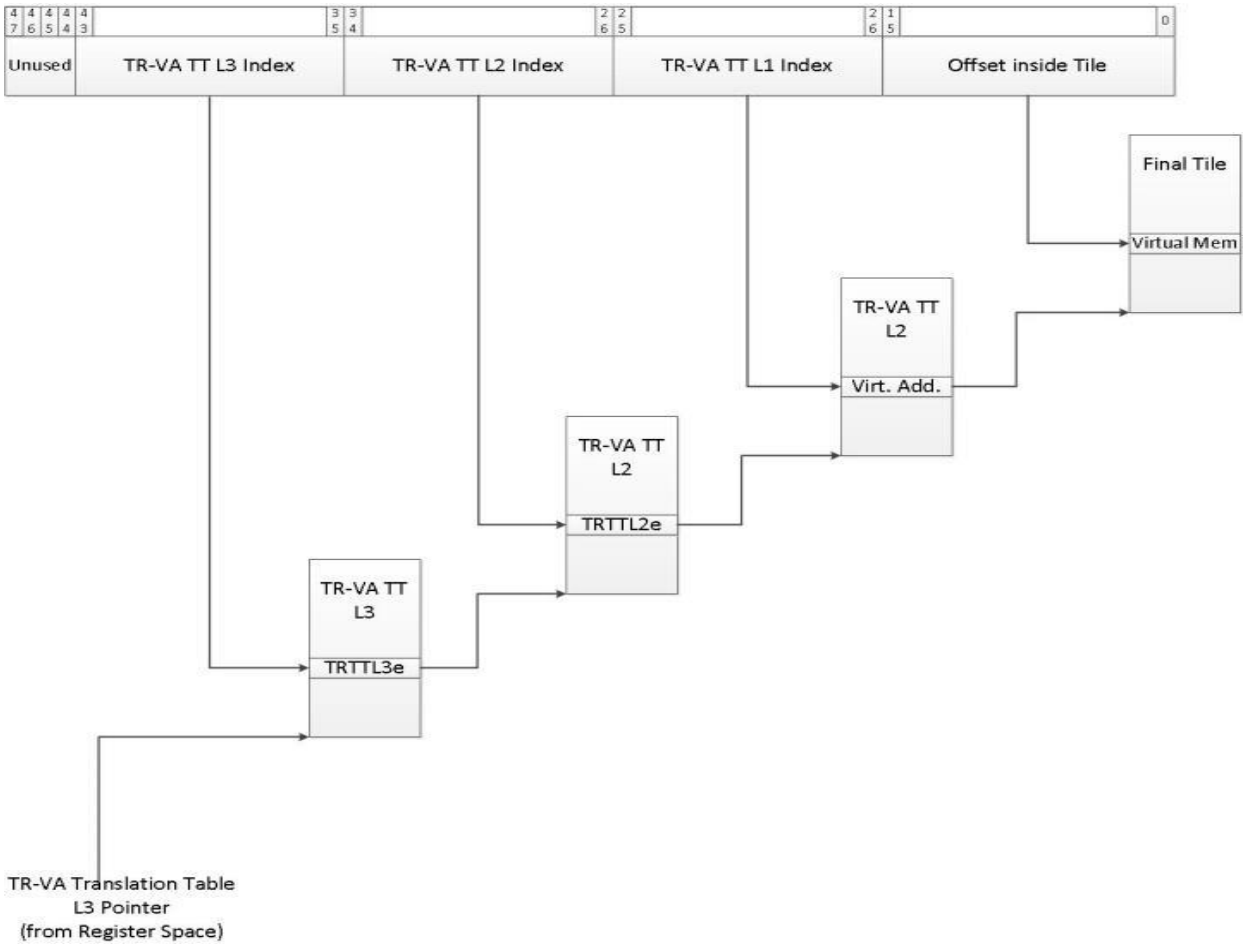
The level of nested walks does not change the structure of the TR-VA walker; it just defines the recursive nature of the translations.

TR-TT Page Walk

Sparse Tiled Resources translation tables are separated into 3-levels. The pointer to L3 table is going to be set up in GFX MMIO space as part of the context, this pointer be would be available to page walker ahead of any TR-VA memory accesses.

TR-TT L3 walk will be consistent of calculating the 64b of interest based on the L3 table pointer and using the 9 bit index (address bits[43:35]). L2 will use TR-TT L3 entry as the table pointer and use the next set of 9 address bits ([34:26]) to locate the L2 entry which is a pointer to L1 table. Final L1 table is located with L2 entry and indexed by remaining 10 address bits (25:16) to index where 32b virtual address is extracted.

Post TR-TT walk 32b entry from L1 is mapped to final virtual address 47:16 and remaining 15:0 is passed from the original TR-VA access as is given all tiles in TR-VA space are 64KB in size.



Page Table Modes

GFX Aperture and Display accesses are mapped thru Global GTT to keep the walk simple (i.e. 1-level) and latency sensitive. GPU accesses to memory can be mapped via Global GTT and/or ppGTT with various addressing modes.

Supported walk modes are listed as following:

1. **Global GTT with 32b virtual addressing**: Global GTT usage is similar to previous generations with extended capability of increasing virtual address (VA) up to 4GB (from 2GB) and use a standard 64b PTE format. The breakdown of the PTE for global GTT is given in later sections and allows 1-level page walk where the 20b index is used to select the 64b PTE from memory.
2. **Legacy 48b VA with ppGTT**: GFX address expansion beyond 4GB is added to address 48b virtual address space. 48b VA requires 36b indexing (4x9b) translating into 4-levels of page walk. To reduce the overhead of 4 level walk, GPU will cache the entire content of PML4 (4kB) to limit the on-demand walks to 3 levels. The caching happens as part of the initial demand where no further replacements required.

Global GTT

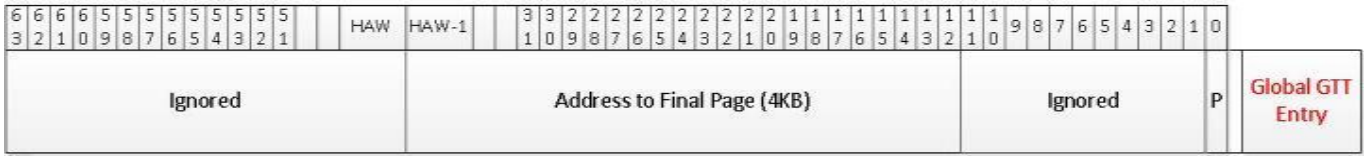
The GPU supports a Global Virtual Address space that is used for resources that are accessible to privileged (ie kernel-mode) processes, and not tied to a specific user-level process. For example, the Graphics micro-Controller (GuC) and Display Engine (if present) utilize Global address space. The Global VA space is 4GB in size (32-bit VA).

The Global GTT (GGTT) translates from the Global virtual address to a physical address that can be accessed by HW, using a page size of 4KB. The GGTT is a flat, single-level table. Each page table entry (PTE) in the GGTT is 8B in size, so the overall size of the GGTT 8MB.

The Global GGTT entries can be updated by the graphics driver, via the GTTMMADR MMIO range of the device, and also via the Command Streamer command MI_UPDATE_GTT.

Page Table Entry

The following page table entry will be used for Global GTT:



Bits	Field	Description
63:57	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
51:HAW	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	Address	Physical address of 4KB memory page referenced by this entry.
11:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
4:2	Function	The Number of the Function to which this page has been assigned. This field is

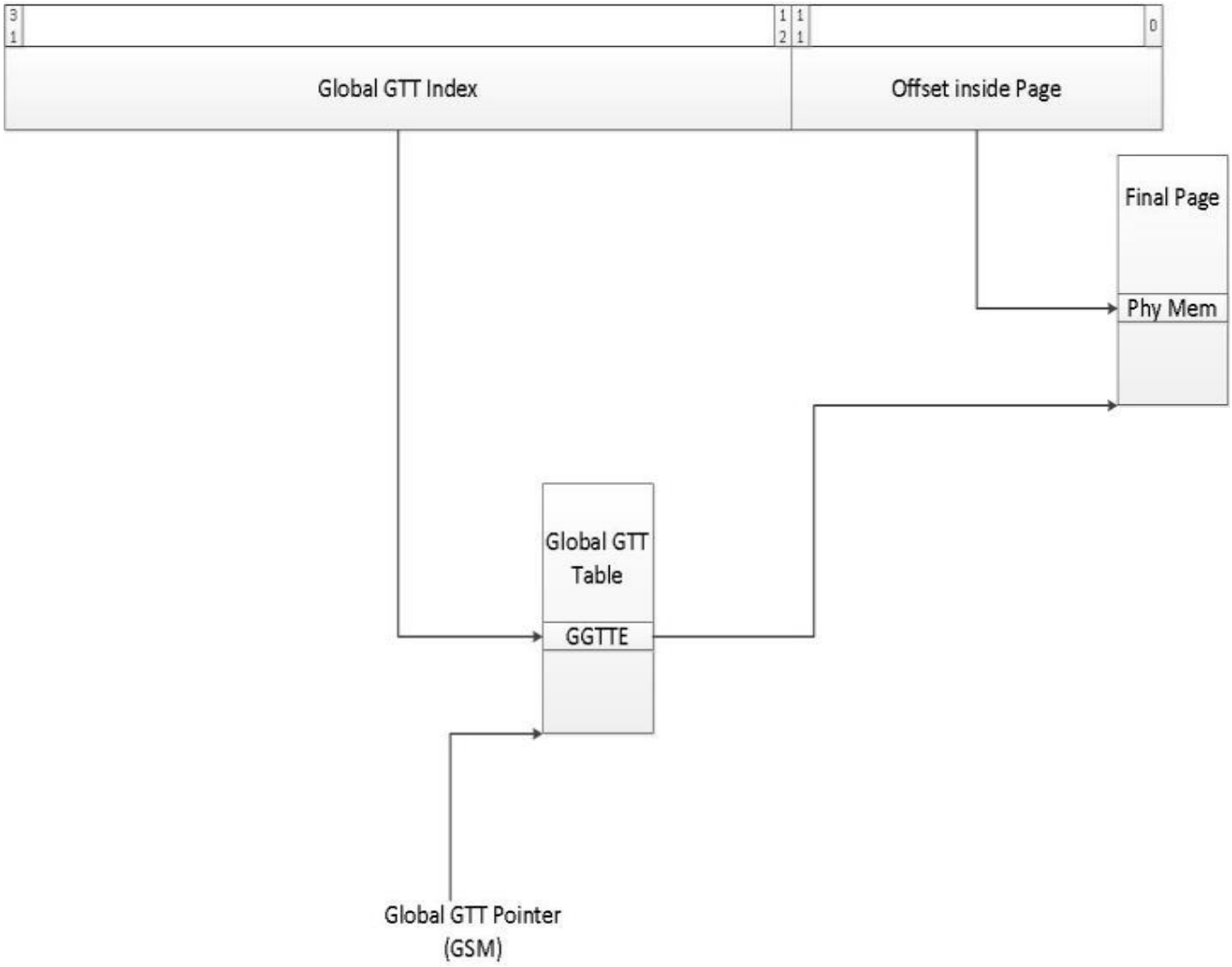
Bits	Field	Description
	Number	ignored.
1	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
0	Present	When set to 1, indicates that this Page Table Entry is Valid, and the corresponding page is Present in physical memory

* HAW = 39 for client, and 46 for server.

The GPU accesses GGTT table entries as uncacheable.

Page Walk

The global GTT page walk is identical to what it was prior. The only difference would be that each entry is 8B (instead of 4B) hence the entry selection needs to be updated once the corresponding Page Table miss read is returned.



Per-Process GTT with 48b VA

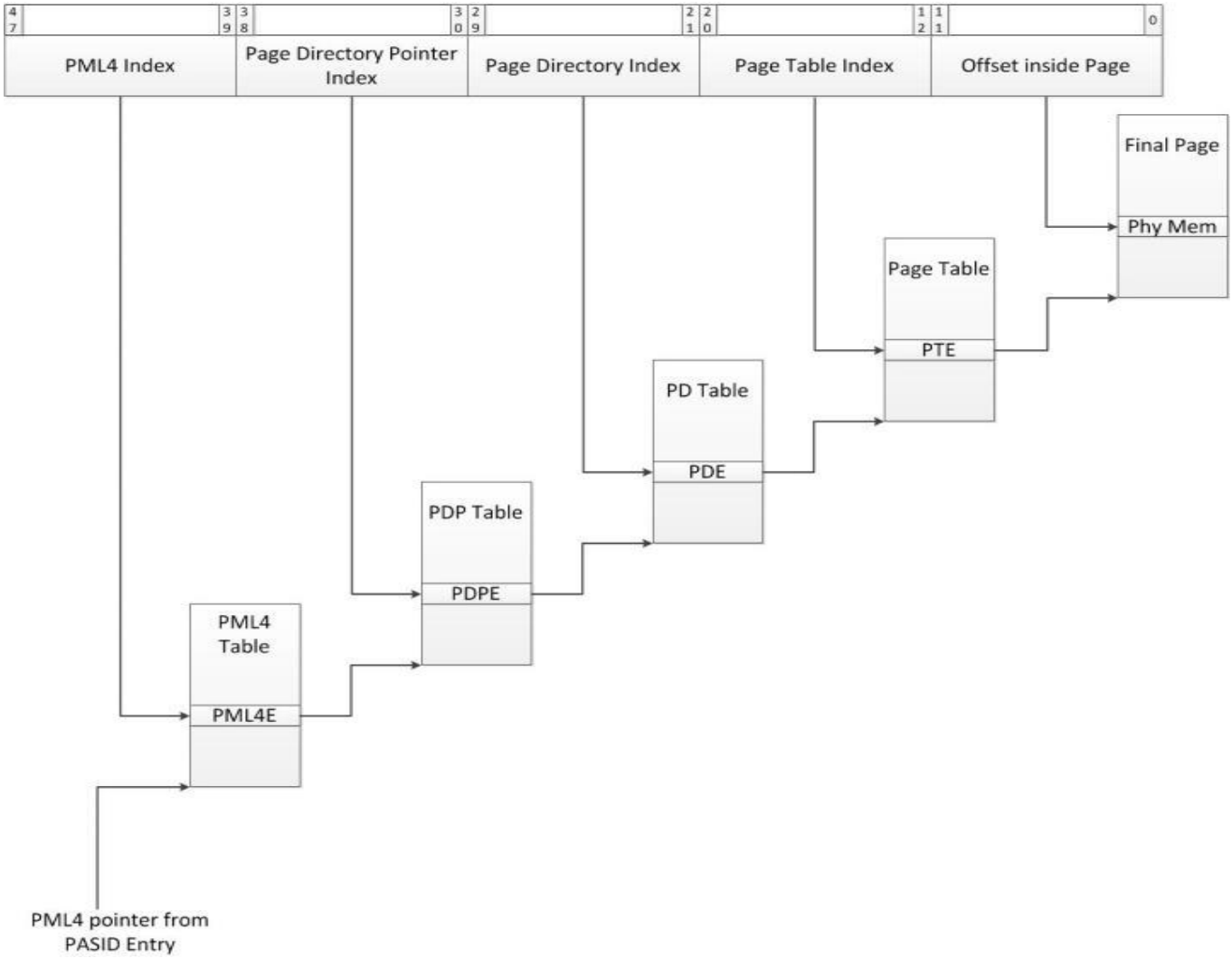
The GPU typically operates on behalf of user-level processes (applications), each of which has it's own "Per-Process" virtual address space. The size of this space is 256TB (48b address width).

The Per-Process GTT (PPGTT) translates these virtual addresses to physical addresses that are used by HW. The PPGTT is a multi-level table very similar to the IA32e table utilized by Intel CPUs. Each entry in the PPGTT is 8 Bytes.

Page Walk in Legacy 48b Mode

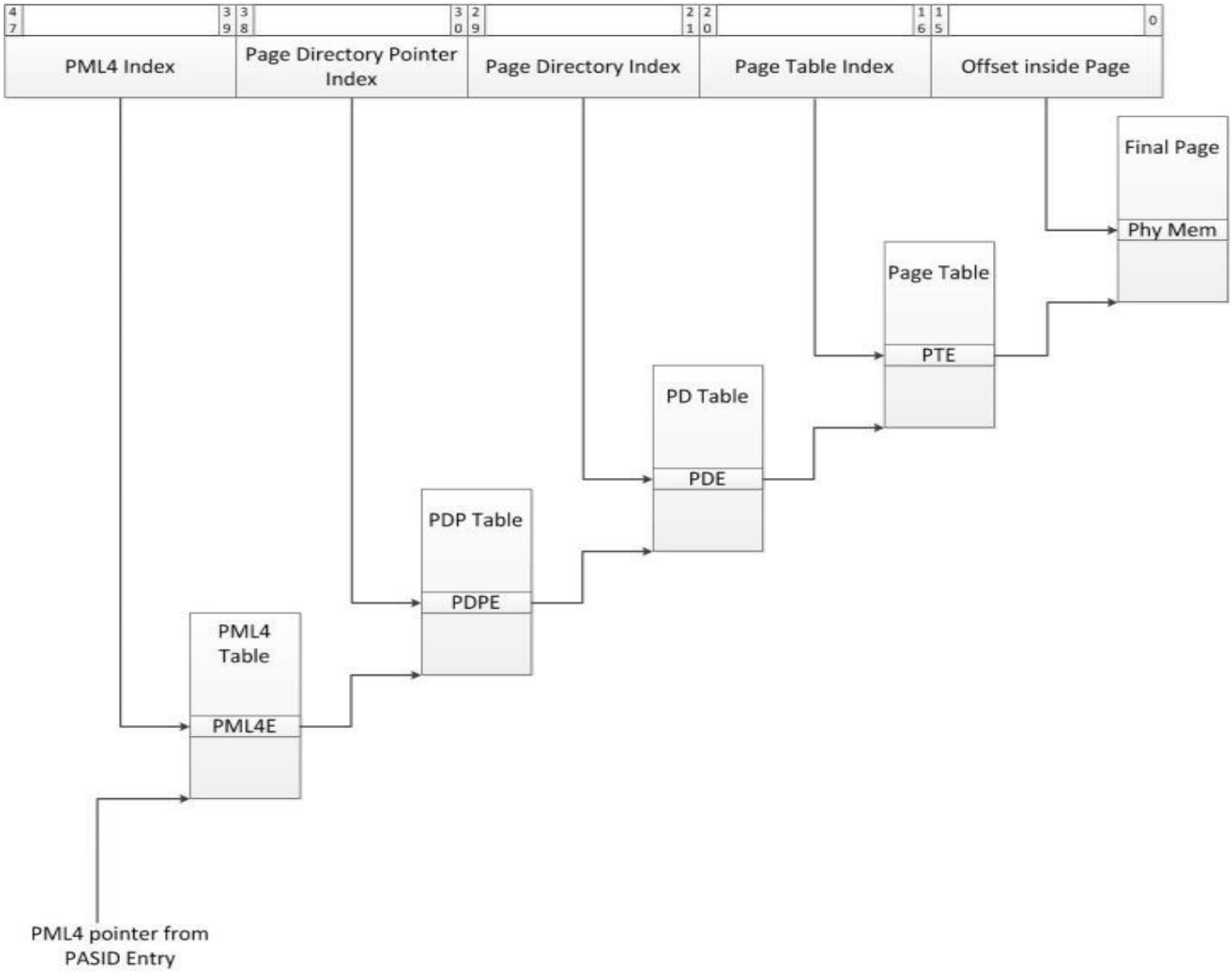
Translation of a 48b VA requires 4 levels of page table. assuming each table level is 4KB and each entry is 8B. The top-most level of the PPGTT is located via the PML4 table pointer associated with the process, and the 48b VA as used to index into consecutive levels of page table.

The following diagram shows the page walk that is needed for a 4KB page.



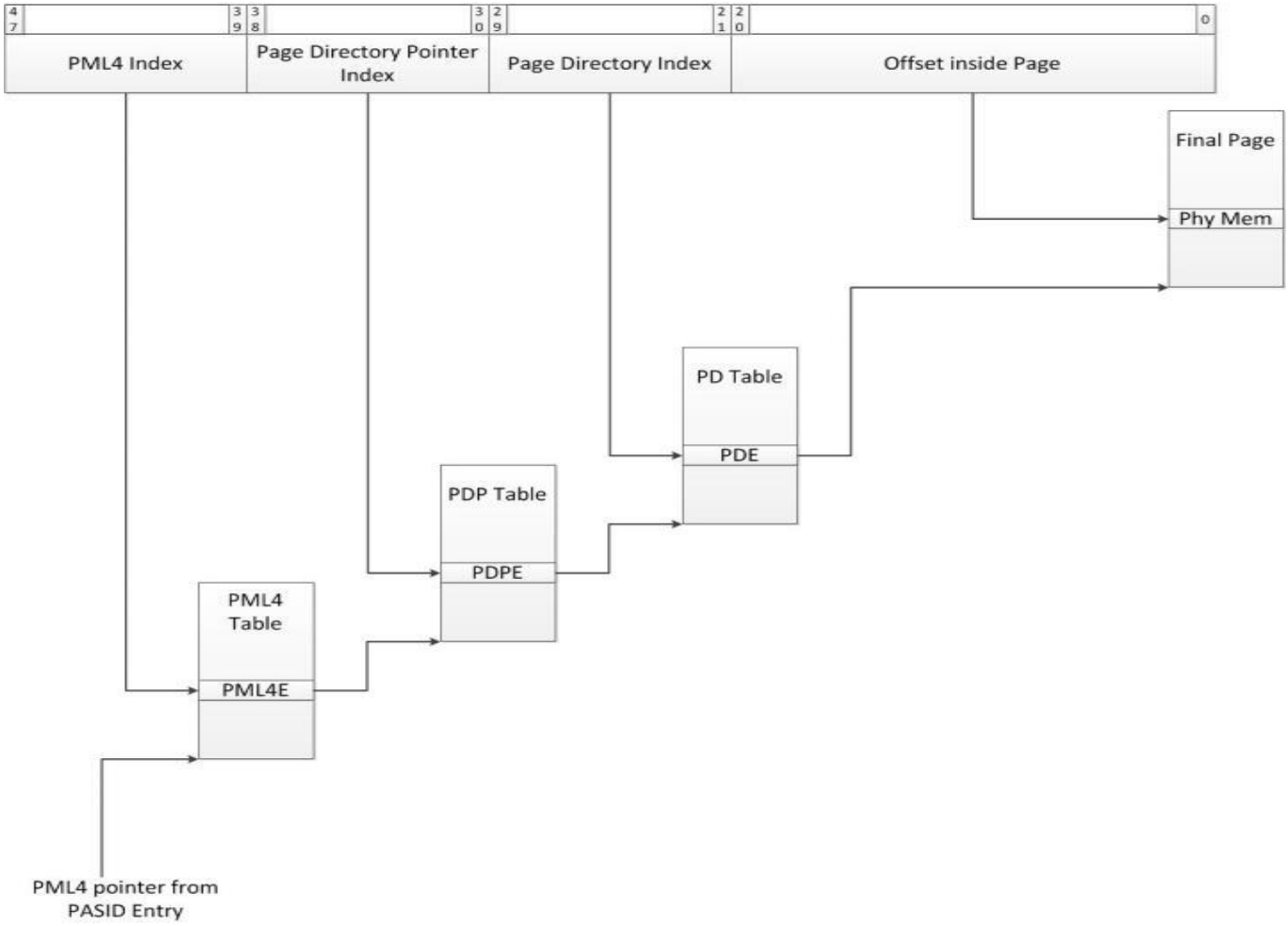
Walk with 64KB Page

64KB Page size has a slightly different usage for how PTEs are selected for the corresponding 64KB page. In page table every 16th entry (PTE#0, PTE#16, PTE#32....PTE#496) should be used to index. This is calculated using address [20:16]& "0000". Note that hardware should not make any assumptions for any other PTEs. 64K paging in the PTE is indicated by [11] of PDE. When PDE[11] = '1, every 16th PTE entry is read (by masking Adr[15:12] bits).



Walk with 2MB Page

With the 2MB Page walk, last level of the page walk is skipped where the PD entry points to the final page.



Walk with 1GB Page

1GB pages are supported by completing address translation at the PDP level, similar to how 2MB page translation is complete at PDE level.

Page Tables Entry PTE Formats

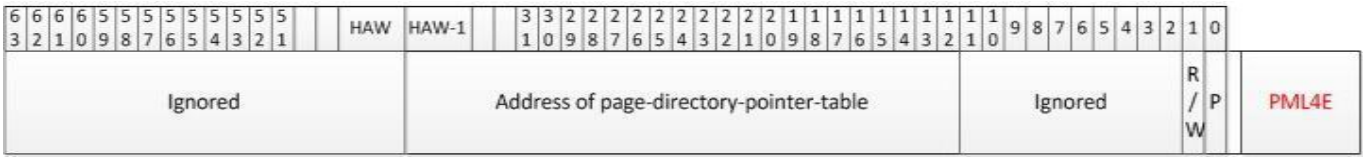
Page Table Entry (PTE) formats will follow the IA32e layout as given below:

Each table entry is further broken down along with the required functions. GFX has a 4 level page table which is pointed out by context descriptor starting with the PML4. The next levels have slightly different formats depending on the size of the page supported. 1GB and 2MB page formats are required for support.

Pointer to PML4 table

The pointer to PML4 table is provided via the context descriptor.

PML4E: Pointer to PDP Table



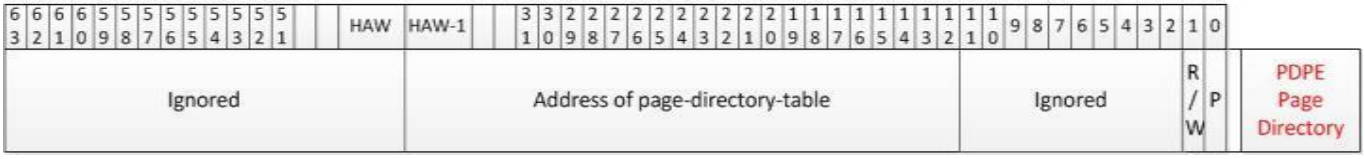
Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page-directory-pointer table referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11:2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	PML4 Entry is present. It must be "1" to point to a page directory pointer table

* HAW = 39 for client, and 46 for server.

PDPE: Pointer to PD Table

PDP entry is used to locate the page directory. IA32e supports 1GB pages, the PDPE has a mechanism to identify a way to say whether this PDPE represents a pointer to page directory or to a contiguous 1GB physical memory.

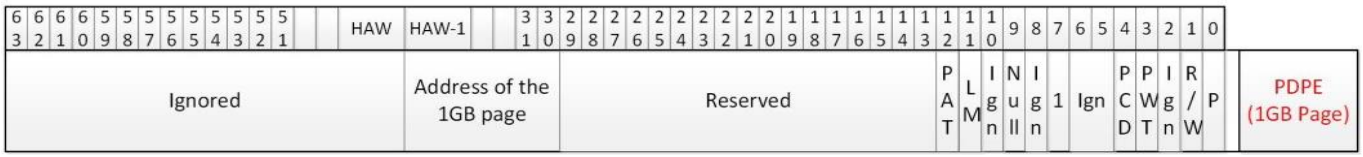
PDPE for PD



Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page-directory table referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11:2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. Access rights are described later. *GPU does not support Supervisor mode contexts.*
0	P: Present	PDP Entry is present. It must be "1" to point to a page directory pointer table

* HAW = 39 for client, and 46 for server.

PDPE for 1GB Page



Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):30	ADDR: Address	Physical address of 1GB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
29:12	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
11	Local Memory	Physical Page is located in Local Memory instead of System Memory. Only applicable for device configurations with local device memory that is managed by the Device Driver instead of the OS. For other configurations it is ignored
10	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1st Level) translation tables. If Null=1, the h/w will avoid the memory access and return all zero's for the read access with a null completion, write accesses are dropped.
8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	This bit must be "1" to point to a valid Page.

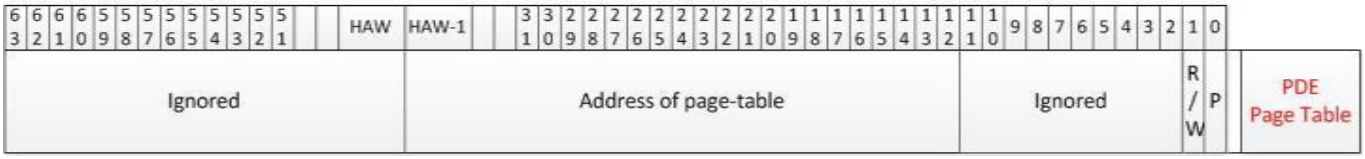
* HAW = 39 for client, and 46 for server.

PD: Pointer to Page Table

This section describes the following:

- PDE for Page Table
- PDE for 2 MB Page

PDE for Page Table



Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page- table referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11:2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	PDP Entry is present. The value must be "1" to point to a page directory pointer table.

* HAW = 39 for client, and 46 for server.

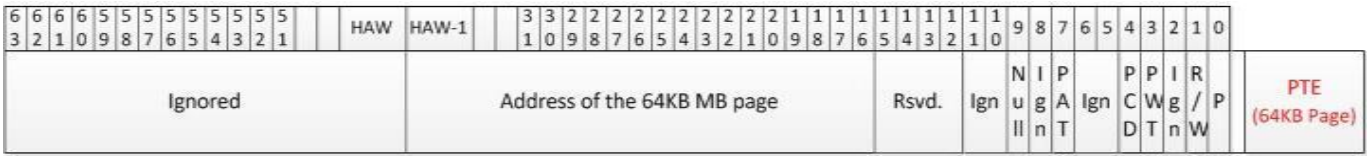
PDE for 2MB Page

Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):21	ADDR: Address	Physical address of 1GB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
20:12	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
11	Local Memory	Physical Page is located in Local Memory instead of System Memory. Only applicable for device configurations with local device memory that is managed by the Device Driver instead of the OS. For other configurations it is ignored
10	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1st Level) translation tables. If Null=1, the h/w will avoid the memory access and return all zero's for the read access with a null completion, write accesses are dropped.
8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant*

Bits	Field	Description
		extended-context-entry) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	It must be "1" to point to a 1GB Page.

* HAW = 39 for client, and 46 for server.

PTE: Page Table Entry for 64KB Page

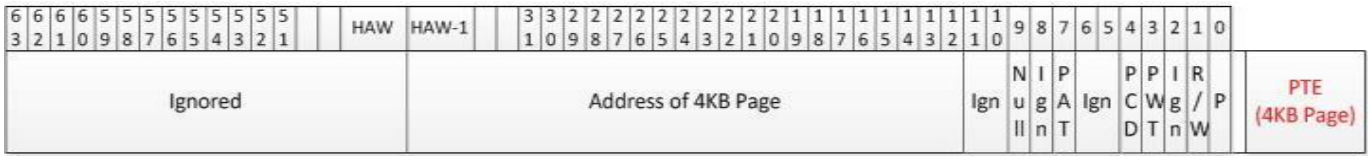


Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):16	ADDR: Address	Physical address of 64KB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
15:12	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
11	Local Memory	Physical Page is located in Local Memory instead of System Memory. Only applicable for device configurations with local device memory that is managed by the Device Driver instead of the OS.
10	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1st Level) translation tables. If Null=1, the h/w will avoid the memory access and return all zero's for the read access with a null completion, write accesses are dropped.
8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table

Bits	Field	Description
		referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	It must be "1" to point to a 64KB Page.

* HAW = 39 for client, and 46 for server.

PTE: Page Table Entry for 4KB Page



Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	ADDR: Address	Physical address of 64KB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11:10	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1st Level) translation tables. If Null=1, the h/w will avoid the memory access and return all zero's for the read access with a null completion, write accesses are dropped.
8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.

Bits	Field	Description
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry) to the memory region controlled by this entry. See a later section for access rights. GPU does not support Supervisor mode contexts.
0	P: Present	It must be "1" to point to a 4KB Page.

* HAW = 39 for client, and 46 for server.

LNCFCMOSx

LNCFCMOCS0 - LNCF MOCS Register 0	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B020h

Programming Notes
WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	UPPER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC **[Default]**	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

DWord	Bit	Description
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

LNCFCMOCS0 - LNCF MOCS Register 0

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.

If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	10 ^ 16
1	11 ^ 17
2	12 ^ 18

16 **Upper MOCS Index - Enable Skip Caching**

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 **Lower MOCS Index Mask Bit**

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 **Reserved**

Access:	RO
Format:	MBZ

7:6 **Reserved**

Access:	RO
Format:	MBZ

5:4 **Lower MOCS Index - L3 Cacheability Control**

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.

For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT **[Default]**	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC	Uncacheable
2h	LOWER_RESERVED	Reserved

LNCFCMOCS0 - LNCF MOCS Register 0		
	3h	LOWER_WB	Writeback
3:1	**Lower MOCS Index - Skip Caching Control**	
	Default Value:	0h
	Access:	R/W Lock
	Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.	
	Programming Notes	
	If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.	
	Bit Offset	**Corresponding Address Bit**
	0	10 ^ 16
	1	11 ^ 17
	2	12 ^18
0	**Lower MOCS Index - Enable Skip Caching**	
	Access:	R/W Lock
	Enable the skip cache mechanism	
	Value	**Name**
	0h	LOWER_ESC_DISABLE **[Default]**
	1h	LOWER_ESC_ENABLE

GLOB_MOCS_LECC_x

GLOB_MOCS_LECC_00_TC_00 - Global MOCS LECC 00 TC 00 Register	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
_Custom_GTIIsContextSaved:	true
Address:	04000h
Name:	Global MOCS 0
ShortName:	GLOB_MOCS_0
Address:	04040h
Name:	Global MOCS 16
ShortName:	GLOB_MOCS_16
Address:	04080h
Name:	Global MOCS 32
ShortName:	GLOB_MOCS_32
Address:	040C0h
Name:	Global MOCS 48
ShortName:	GLOB_MOCS_48
MOCS register	

DWord	Bit	Description
0	31:19	**Reserved** Access: RO Format: MBZ
	18:17	**Self Snoop Enable** Access: R/W 00: Default value. Self snoop attribute sent to the uncore is as today - determined by MIDI unit logic 01: Override the self snoop bit generated by MIDI with 0. No self snoops are sent to the uncore for any transactions from this surface 11: Override the self snoop bit generated by MIDI with 1. Self snoops are always sent to the uncore for any transactions from this surface

Value	Name	Description
11b	Generated by MIDI with 1	Override the self snoop bit generated by MIDI with 1. Self snoops are always sent to the uncore for any transactions from this surface
01b	Generated by MIDI with 0	Override the self snoop bit generated by MIDI with 0. No self snoops are sent to the uncore for any transactions from this surface.

GLOB_MOCS_LECC_00_TC_00 - Global MOCS LECC 00 TC 00 Register		
	00b	Determined by MIDI unit logic **[Default]**	Default value. Self snoop attribute sent to the uncore is as today - determined by MIDI unit logic

16:15 **Class of Service**

Access:	R/W

Description
Class of Service sent to LLC to determine subset of ways the memory object will be stored in. 00: Class 0 01: Class 1 10: Class 2 11: Class 3 *Max* QoS:* Class 0 *Relative* QoS:* 0 > 1 > 2 ≥ 3** * Max/Relative statements above based on default/non-firmware-overridden GT QoS masks. ** CLOS2 = CLOS3 equivalence only on 4-way LLC SKUs.

Value	Name
00b	Class 0 **[Default]**
01b	Class 1
10b	Class 2
11b	Class 3

14 **Snoop Control Field**

Access:	R/W

Enables s/w to have GFX h/w to be able to consume IA generated buffers that are tagged as WB. Driver can mark these buffers as WB when generating them from IA

In SOCs, the fabric is not forced to be coherent all the time. IA-core generated WB buffers can only be consumed by GPU if that buffer is tagged as snoop-able in GPUs buffer definitions (or via GPU Page tables)

1: Non-Coherent Write/Read

0: Coherent Access

Note: There is a performance and power penalty in accessing surfaces that are tagged as snooped

Value	Name
0b	Coherent Access **[Default]**

	1b	Non-Coherent Write/Read

13:11 Page Faulting Mode

Default Value:	000b Global page faulting mode
Access:	R/W

This fields controls the page faulting mode that will be used in the memory interface block for the given request coming from this surface:

000: Use the global page faulting mode from context descriptor (default)

001-111: Reserved

10:8 Skip Caching control

Default Value:	000b Value is do not care
Access:	R/W
Format:	Enable[3]

Defines the bit values to enable caching. Outcome overrides the LLC caching for the surface.

If "0" - than corresponding address bit value is do not care

Bit[8]=1: address bit[9] needs to be "0" to cache in target

Bit[9]=1: address bit[10] needs to be "0" to cache in target

Bit[10]=1: address bit[11] needs to be "0" to cache in target

7 Enable Reverse Skip Caching

Access:	R/W
Format:	Enable

Enable for the Skip cache mechanism

0: Not enabled

1: Enabled for LLC

Value	Name
0b	Not Enabled **[Default]**
1b	Enabled for LLC

6 Dont allocate on miss

Access:	R/W

Controls defined for RO surfaces in mind, where if the target cache is missed - do not bring the line (applicable to LLC/eDRAM).

0: Allocate on MISS (normal cache behavior)

1: Do NOT allocate on MISS

Received confirmation from Altug on 03/13/13 that nothing needs to be done on this bit

GLOB_MOCS_LECC_00_TC_00 - Global MOCS LECC 00 TC 00 Register		
		Value / **Name** 0b — Allocate on MISS **[Default]** 1b — Not allocate on MISS
	5:4	**LRU management** Access: R/W **Description** This field allows the selection of AGE parameter for a given surface in LLC or eLLC. . If a particular allocation is done at youngest age ("3") it tends to stay longer in the cache as compared to older age allocations ("2", "1", or "0"). This option is given to driver to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches. LRU Age value is assigned as follows: 11: Assign the age of "3" 10: do not change the age on a hit. 01: Assign the age of "0" 00: Take the age value from Uncore CRs **Value** / **Name** 11b — Assign age 3 **[Default]** 10b — Do not change age 01b — Assign age 0 00b — Age value from Uncore
	3:2	**Target Cache** Access: R/W **Description** This field allows the choice of LLC vs eLLC for caching 00: eLLC Only 01: LLC Only 10: LLC/eLLC Allowed 11: LLC/eLLC Allowed **Value** / **Name** 00b — eLLC Only **[Default]** 01b — LLC Only

GLOB_MOCS_LECC_00_TC_00 - Global MOCS LECC 00 TC 00 Register		
		10b: LLC/eLLc Allowed 11b: LLC/eLLc Allowed 2
	1:0	**LLC/eDRAM cacheability control** Access: R/W **Description** Memory type information used in LLC/eDRAM. 00: Uncacheable (UC) 01: Uncacheable (WC) 10: Writethrough (WT) 11: Writeback (WB) Note: Final memory type is based on memory type resolution using MOCS and Page Table memory types and legacy/advanced mode Note: Binding table index based memory typing cannot be used for LLC/eDRAM memory type. Instead page table based controls have to be used Note: In case of SVM (advanced context), LLC/eDRAM memory type is used based on the page table controls and cannot be managed via MOCS index **Value** / **Name** 00b: Uncacheable (UC) 01b: Uncacheable (WC) 10b: Writethrough (WT) 11b: Writeback (WB) **[Default]**